ALGORITMA DAN PEMROGRAMAN 2

Penyusun :

Joko Riyanto

Fitri Yanti

Bambang Santoso

Ari Syaripudin

UNPAM PRESS
Lembaga Penerbit & Publikasi

Jl. Surya Kencana No. 1 Pamulang

Gd. A, Ruang 212 Universitas Pamulang

Tangerang Selatan – Banten

ALGORITMA DAN PEMROGRAMAN 2

Penulis :
Joko Riyanto, Fitri Yanti, Bambang Santoso, Ari Syaripudin

ISBN : 978-623-5437-29-3

Editor :
Jaka Sutresna
Farida Nurlaila

Desain Sampul:
Putut Said Permana

Tata Letak:
Ramdani Putra

Penerbit:
Unpam Press

Redaksi:
Jl. Surya Kencana No. 1

R. 212, Gd. A Universitas Pamulang Pamulang | Tangerang Selatan | Banten
Tlp/Fax: 021. 741 2566 – 7470 9855 Ext: 1073
Email: unpampress@unpam.ac.id
Anggota IKAPI

Cetakan pertama, 28 November 2022
BA265-28112022-01

KATA PENGANTAR

Segala puji dan syukur saya panjatkan kepada Allah SWT, karena atas berkat rahmat-Nya lah saya dapat menyelesaikan buku ajar Algoritma dan Pemrograman 2 ini dengan baik. Diharapkan melalui buku ajar ini dapat memberikan pengetahuan mengenai alur pikiran dalam menyelesaikan masalah yang merupakan gabungan antara seni dan sains.

Buku ajar ini memberikan pemahaman tentang seni karena memerlukan kreatifitas dan imajinasi yang jenius. Sains karena membuat program adalah suatu pekerjaan yang bersifat membuat rancang bangun yang memerlukan teknik, tool dan metodologi yang sistimatis.

Algoritma dan pemrograman 2 ini merupakan pengembangan dari algoritma dan pemrograman 1. Bahasa pemrograman yang di pake menggunakan bahasa c++. Bahasa disini hanya digunakan sebagai alat menulis algoritma. Yang merupakan rangkaian instruksi-instruksi algoritma sehingga terbentuk program yang benar dan efisien.

Mata Kuliah ini memberikan pemahaman bahwa algoritma hanya terkait pada masalah yang akan dipecahkan. Contoh masalah, misalkan bagaimana mencetak nim mahasiswa secara berurutan mulai dari yang terkecil sampai nim mahasiswa terbesar. Ada banyak algoritma yang dapat kita rancang sendiri untuk menyelesaikan masalah ini, mulai dari algoritma yang dianggap sederhana sampai yang dianggap jenius, bahkan

mungkin dianggap aneh, rumit atau berbelit-belit. Tapi bila yang dicetak urut adalah satusan bahkan ribuan nim, maka diperlukan algoritma yang sudah mengikuti teknik pemrograman atau pola tertentu.

Akhirnya kami sampaikan salam, semoga buku ajar yang kami susun dapat memberikan kebermanfaatan untuk pembaca.

Tangerang Selatan, Agustus 2022

Penulis

DAFTAR ISI

BAB I

POINTER

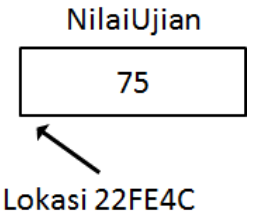
A. CAPAIAN PEMBELAJARAN

Mahasiswa mengerti definisi pointer dan dapat menggunakan dalam program sederhana.

B. URAIAN MATERI

1. Pengertian Pointer

Tiap variabel yang didefinisikan di program akan diberikan satu tempat di memori. Nilai variabel disimpan di lokasi tersebut.



***Gambar** 1.1 Variabel Nilai Ujian disimpan di salah satu lokasi di memori*

Untuk melihat di lokasi mana nilai ini disimpan, dapat dipakai operator & atau *reference*, yaitu mendapatkan alamat dari variabel tersebut.

```
int NilaiUjian = 75;

cout << &NilaiUjian;
```

Keluaran dari potongan program di atas adalah alamat dari memori tempat sistem operasi menyimpan variabel NilaiUjian. Contoh adalah semacam berikut.

```
0x22fe4c
```

Catatan:

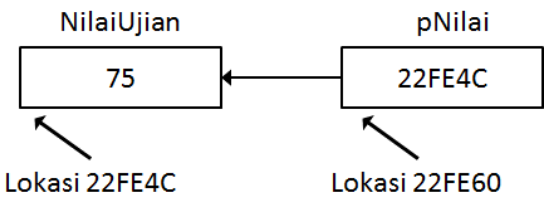
Keluaran program anda dapat berbeda dengan contoh di atas karena alamat memori tergantung perangkat keras, sistem operasi, software yang sedang berjalan, dan banyak faktor lain.

Kita dapat menyimpan alamat ini dalam suatu variabel lain. Variabel tempat kita menyimpan alamat variabel lain disebut pointer. Dengan kata lain, pointer adalah suatu variable yang berisi alamat memori dari suatu variable lain.

Contoh Program 1

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4          int NilaiUjian = 75;
5          int *pNilai;
6          pNilai=&NilaiUjian;
7          cout << &NilaiUjian<<endl;
8          cout << &pNilai<<endl;
9          cout << pNilai<<endl;
10         cout << *pNilai<<endl;
11 }
```

Program di atas dapat digambarkan sebagai berikut.



Gambar 1.2 *Variabel NilaiUjian dan pointer pNilai*

Variabel pNilai merupakan variabel pointer. Penulisannya dilakukan dengan menambahkan tanda

bintang (*) di depan nama variabel. Variabel pNilai diisi dengan alamat dari variabel NilaiUjian dengan perintah pNilai = &NilaiUjian. Sedang variabel NilaiUjian berisi angka 75.

Keluaran dari program di atas adalah seperti di bawah.

0x22fe4c

0x22fe60

0x22fe4c

75

Angka pertama dan kedua adalah alamat dari NilaiUjian dan alamat dari pNilai, berturut-turut. Angka ketiga adalah nilai yang disimpan di variabel pNilai yaitu alamat NilaiUjian, sehingga tampilan angka pertama dan ketiga nilainya sama. Angka terakhir adalah nilai dari Nilai Ujian, yakni 75.

2. Deklarasi Variabel Pointer

Bentuk umum deklarasi variabel pointer adalah sebagai berikut.

```
tipe_data* namaVariabel
```

Contoh :

```
int* pNilai;
```

Ada tiga cara penulisan deklarasi pointer. Perbedaan ada pada penulisan spasi.

```
int* pNilai;

int *pNilai;

int * pNilai;
```

Ketiga cara penulisan di atas benar menurut C++.

Agar pointer menunjuk ke suatu variabel maka perlu diisi dengan alamat dari variabel yang bersangkutan. Perlu diperhatikan bahwa variabel dan pointernya harus mempunyai tipe data yang sama.

Contoh :

```
int Usia;

int* pUsia;

pUsia = &Usia;
```

Tanda * menunjukkan bahwa variabel dideklarasikan sebagai pointer. Variabel pointer pUsia mempunyai tipe data integer dan menunjuk ke sebuah obyek (Usia) yang juga memiliki tipe integer.

Tanda & menandakan operator alamat, merupakan operator *unary* yang mengembalikan alamat dari operand.

Catatan:

Operator *unary* adalah operator yang hanya mempunyai satu operand. Contoh adalah operator – (minus) seperti pada `-A`. Bila `A` bernilai positif, maka `-A` adalah negatif. Sebaliknya, jika `A` negatif maka nilai `-A` adalah positif.

Operator unary **&** memberikan alamat variabel operand. Perintah &A akan memberikan alamat dari variabel A (operand) di dalam memori. Variabel A harus sudah dideklarasikan terlebih dahulu. Operator & sering disebut sebagai *reference*.

Sebaliknya, operator ***** memberikan nilai dari alamat yang ditunjuk oleh operand di mana operand di sini adalah variabel pointer. Perintah `*pA` memberikan nilai dari alamat yang disimpan di pointer pA. Operator `*` sering disebut sebagai *dereference*.

Contoh Program 2

```
1  #include <iostream>
2  using namespace std;
3  int main(void)
4  {
5    int A=25;       // deklarasi integer
6
```

```
7   int* pA;                    // deklarasi pointer ke integer
8   pA=&A;                      // pointer ke alamat A
9   cout<<"A = "<<A<<endl;      // tampilkan nilai A
10  cout<<"&A = "<<&A<<endl;    // tampilkan alamat A
11  cout<<"pA = "<<pA<<endl;    //tampilkan nilai pA
12  cout<<"&pA = "<<&pA<<endl;//tampilkan alamat pA
    cout<<"*pA = "<<*pA<<endl;//nilai yang ditunjuk pA
    }
```

Hasil tampilan adalah sebagai berikut.

```
A = 25
&A = 0x22fe3c
pA = 0x22fe3c
&pA = 0x22fe30
*pA = 25
```

Di baris ke 7 dalam program di atas, variabel pointer pA diisi alamat A dengan perintah pA = &A. Sehingga pA jika ditampilkan akan berisi alamat A.

Sedang *pA adalah nilai yang ditunjuk oleh pointer pA, yaitu nilai A.

Peringatan:

Perhatikan bahwa tampilan `pA`, `&pA`, maupun `*pA` menampilkan nilai yang berbeda-beda. Contoh di atas menggambarkan hal ini dengan jelas. Ini harus dimengerti benar agar pemakaian pointer dapat tepat dilakukan.

Contoh Program 3

```
1  #include <iostream>
2  using namespace std;
3  int main(void)
4  {
5    int A=25;       // deklarasi integer
6    int* pA;                // deklarasi pointer ke
7  integer
8    pA=&A;                  // mengarahkan pointer ke
9  alamat A
10   cout<<"A="<<A<<endl;        // tampilkan nilai A
11   cout<<"*pA="<<*pA<<endl; //nilai yang ditunjuk
   pA
```

```
12   A=35;            // nilai A diubah menjadi 35
13   cout<<"A="<<A<<endl;        // tampilkan nilai A
14   cout<<"*pA="<<*pA<<endl; //nilai yang ditunjuk
15 pA
16   *pA+=10; // nilai A ditambah 10
     cout<<"A="<<A<<endl;        // tampilkan nilai A
     cout<<"*pA="<<*pA<<endl; //nilai yang ditunjuk
     pA
     }
```

Hasil keluaran dari program di atas adalah:

```
A=25
*pA=25
A=35
*pA=35
A=45
*pA=45
```

Di baris 7, pointer pA diisi dengan alamat dari variabel A. Maka A dan *pA ketika ditampilkan akan bernilai sama yaitu 25.

Di baris 10, variabel A diubah nilainya menjadi 35. Maka nilai *pA juga akan berubah nilainya menjadi 35. Lihat tampilan untuk baris 11 (nilai A) dan baris 12 (nilai *pA).

Di baris 13 nilai *pA ditambah 10 sehingga menjadi 45. Maka nilai A juga berubah menjadi 45. Perhatikan hasil tampilan untuk baris 14 (nilai A) dan baris 15 (nilai *pA).

Dari program di atas dapat dilihat bahwa mengubah nilai A dapat dilakukan lewat variabel A maupun lewat variabel pointer pA. Ini karena pA menunjuk ke alamat memori yang sama dengan A.

Contoh Program 4

```
1  #include <iostream>
2  using namespace std;
3  int main(void)
4  {
5    string makanan = "Bakso"; // Tipe string
6    string* ptr = &makanan;   // variabel pointer
7    cout << makanan << endl;  // tampilkan nilai
8    cout << *ptr << endl;     // tampilkan nilai
9    cout << &makanan << endl; // tampilkan alamat
10   cout << ptr << endl;      // tampilkan alamat
11 }
```

Tampilan dari program di atas adalah sebagai berikut.

Bakso

Bakso

0x22fe20

0x22fe20

Program membuat variabel pointer dengan nama ptr, yang menunjuk ke variabel bertipe string. Perhatikan bahwa tipe data variabel pointer harus sama dengan tipe data variabel yang akan ditunjuk.

Program memakai operator & untuk mengambil alamat dari variabel makanan dan menyimpannya di variabel pointer ptr. Selanjutnya, ptr sudah memiliki alamat variabel makanan. Selanjutnya dapat ditebak bahwa tampilan makanan dan *ptr akan sama yaitu nilai yang tersimpan di variabel makanan. Dan tampilan &makanan dan ptr akan sama yaitu alamat dari variabel makanan.

Latihan

Soal 1

```
int panjang=25, lebar=11, A=10;
int* pA;
pA = &panjang;
cout<<"Nilai: "<<*pA<<endl;
```

```
pA = &lebar;
cout<<"Nilai: "<<*pA<<endl;
pA = &A;
cout<<"Nilai: "<<*pA<<endl;
A = panjang * lebar;
cout<<"Nilai: "<<*pA<<endl;
```

Apa tampilan yang dihasilkan oleh potongan program di atas? Jelaskan mengapa 4 perintah yang sama cout<<"Nilai: "<<*pA<<endl; dapat menghasilkan tampilan yang berbeda-beda.

Soal 2

```
double panjang=25;
int *pA;
pA = &panjang;
```

Potongan program di atas menghasilkan kesalahan saat kompilasi di baris ke 3. Mengapa?

Soal 3

```
int panjang=25;
int* pA, pB;
pA = &panjang;
pB = &panjang;
```

Potongan program di atas menghasilkan kesalahan saat kompilasi di baris ke 4. Jelaskan.

3. Pointer dan Larik 1 Dimensi

Pointer dapat digunakan untuk menunjuk ke larik (*array*) dan selanjutnya pointer dapat dipakai untuk mengakses elemen-elemen larik. Contoh, dideklarasikan variabel larik integer dengan 3 anggota dan pointer sebagai berikut.

```
int A[]={14,16,19};
int *pA;
```

Pemberian nilai (*assignment*) pointer ke larik 1 dimensi dapat dilakukan dengan dua cara.

```
pA = A;
```

atau

```
pA = &A[0];
```

Untuk mengarahkan pointer ke anggota pertama dari suatu larik, dapat dilakukan dengan memberikan variabel larik tanpa indeks, sebagaimana contoh di atas, yaitu perintah pA=A. Dapat juga dilakukan dengan memberi alamat anggota pertama dari larik, yaitu anggota dengan indeks 0. Contoh adalah pA=&A[0]. Kedua perintah tersebut diperlakukan sama oleh C++.

Catatan:

Nama larik adalah constant pointer ke elemen pertama dari larik. Karenanya jika dideklarasikan `int A[10]`, maka A adalah sama dengan &A[0]. Yaitu alamat dari anggota pertama dari larik. Misal:

```
int A[10]={1,2,3,4,5,6,7,8,9,10};
cout<<A<<endl;
```

Maka akan ditampilkan alamat elemen pertama A, yaitu

```
0x22fe30
```

(Alamat bisa berbeda, tergantung komputer)

Perhatikan yang ditampilkan bukan nilai A[0], tapi alamat dari A[0].

Penunjukan alamat oleh variabel pointer dapat digambarkan sebagai berikut.

Gambar 1.3 *Pointer ke anggota pertama dari larik*

Alamat dari A[0] adalah 0x23fe20. Dengan perintah pA=A atau pA=&A[0] maka pA akan berisi alamat dari A[0].

Catatan:

Membuat larik tidak harus menyebutkan jumlah anggota. C++ akan menghitung jumlah anggota yang dituliskan, dan membuat larik sesuai dengan jumlah anggota yang dirinci di perintah tersebut. Contoh perintah `int A[]={14,16,19}` akan membuat larik dengan anggota 3 buah integer karena ada tiga anggota yang dituliskan yaitu 14, 16, dan 19.

Contoh Program 1

```
1  #include <iostream>
2  using namespace std;
3  int main(void) {
4    int *pTgl;       // deklarasi pointer
5    int tglLahir[]= {19,4,1980}; // deklarasi larik
6    pTgl=tglLahir;          // pemberian nilai
7    //Menampilkan isi array dengan pointer
8    for(int i=1; i<=3; i++) {          // looping
9      cout<<*pTgl<<endl; //tampilkan nilai
```

```
10      pTgl++;                 // increment pointer
11    }
12    cout << endl;             // satu baris kosong
13    pTgl=&tglLahir[0]; // pemberian nilai
14    cout<<*pTgl<<endl;              // tampilkan nilai
15    cout<<*(pTgl+1)<<endl;      // alamat tambah 1
16    cout<<*(pTgl+2)<<endl;      // alamat tambah 2
      }
```

Tampilan program di atas:

```
19
4
1980

19
4
1980
```

Di baris ke 6 pTgl diberi nilai alamat tglLahir dengan perintah pTgl=tglLahir. Ini identik dengan memberi alamat tglLahir[0]. Dalam pengulangan ditampilkan nilai yang ditunjuk pointer pTgl. Pertama kali ini akan menampilkan tglLahir[0] yaitu 19. Perintah increment di baris 10 akan

membuat pTgl berisi alamat tglLahir[1] sehingga perintah cout berikutnya akan menampilkan angka 4. Increment sekali lagi membuat pTgl menunjuk ke alamat tglLahir[2]. Perintah cout berikutnya menampilkan angka 1980.

Baris 13 mengembalikan pointer ke alamat tglLahir[0]. Perintah *(pTgl+1) akan menampilkan nilai larik tglLahir di indeks 1. Perintah *(pTgl+2) akan menampilkan nilai larik indeks 2.

Contoh Program 2

```
1  #include<iostream>
2  using namespace std;
3  int main(void){
4    int angka[5];
5    int * p;
6    p = angka; *p = 10;
7    p++; *p = 20;
8    p = &angka[2]; *p = 30;
9    p = angka + 3; *p = 40;
10   p = angka; *(p+4) = 50;
11   for (int n=0; n<5; n++)
12     cout << angka[n] << " ";
13 }
```

Program di atas membuat deklarasi angka sebagai larik integer (bilangan bulat) dengan 5 anggota. Kemudian diperlihatkan beberapa macam cara untuk mengakses larik.

Keluaran dari program di atas adalah seperti di bawah.

10 20 30 40 50

Terlihat nilai dari larik angka dapat berubah dengan cara mengakses nilainya dengan memakai pointer.

C. LATIHAN SOAL

Apa tampilan potongan program berikut? (Buatlah program lengkap untuk dapat menjalankan program-program berikut)

Soal 1.

```
int *P;
int A[5]={2,1,7,0,5};
P=A;
cout<<*P<<endl;
P++;
cout<<*P<<endl;
```

Soal 2.

```
int *P;
int A[5]={2,1,7,0,5};
P=A;
cout<<*P++<<endl;
cout<<*P<<endl;
```

Soal 3.

```
int *P;
int A[5]={2,1,7,0,5};
P=A;
cout<<(*P)++<<endl;
cout<<*P<<endl;
```

Soal 4.

```
int *P;
int A[5]={2,1,7,0,5};
P=A;
cout<<*++P<<endl;
cout<<*P<<endl;
```

Soal 5.

```
int *P;

int A[5]={2,1,7,0,5};

P=A;

cout<<++*P<<endl;

cout<<*P<<endl;
```

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan Java*. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

BAB II

POINTER DAN LARIK 2 DIMENSI

A. CAPAIAN PEMBELAJARAN

Mahasiswa dapat mengerti dan dapat menggunakan pointer dalam pengalamatan larik 2 dimensi.

B. URAIAN MATERI

1. Memetakan Larik 2 Dimensi Dalam Memori Komputer

Dalam C++, larik (array) selalu diperlakukan sebagai larik 1 dimensi. Ini mengikuti pengalamatan dalam komputer yang hanya 1 dimensi. Perhatikan program berikut.

Contoh Program 1

```
1  #include<iostream>
2  using namespace std;
3  int main() {
4     char A[3][5]= {
5        'A','B','C','D',
```

```
6       'E','F','G','H',
7       'I','J','K','L'
8     };
9     char *P;
10    P = &A[0][0];
11    cout << *P;
12    P = &A[1][3];
13    cout << *P;
14  }
```

Dalam program di atas, dideklarasikan variabel A sebagai larik dengan tipe data char 2 dimensi. Ukuran larik adalah 3 x 5, yaitu 3 baris dan 5 kolom. Dalam 2 dimensi, penggambaran variabel A adalah sebagai berikut.

	0	1	2	3
0	A	B	C	D
1	E	F	G	H
2	I	J	K	L

Anggota larik dinamakan dengan nama variabel diikuti dua angka dalam kurung siku sebagai indeks. Misal baris kedua kolom ketiga disebut A[1][2] dengan nilai ‘G’.

Untuk menampilkan nilai larik, dapat dilakukan dengan perintah cout << A[1][2]. Untuk mengganti nilai ‘G’ dengan ‘W’ dapat dilakukan dengan perintah A[1][2]=’W’.

Catatan:

Indeks dalam C++ selalu dimulai dari 0. Larik A[m][n] akan membuat larik 2 dimensi m x n. Indeks baris terbesar adalah m-1 dan indeks kolom terbesar adalah n-1.

Sebagai contoh, deklarasi variabel `int A[4][6]` akan membuat larik dengan jumlah baris 4 dan jumlah kolom 6. Indeks baris adalah dari 0 sampai dengan 3 dan indeks kolom dari 0 sampai dengan 5.

Dalam memori komputer, larik tersebut disimpan dalam larik 1 dimensi berukuran 12. Angka ini didapat dari perkalian 2 dimensi A, yaitu 3 x 4. Penggambaran variabel A di dalam memori adalah seperti gambar berikut.

0,0	0,1	0,2	0,3	1,0	1,1	1,2	1,3	2,0	2,1	2,2	2,3
A	B	C	D	E	F	G	H	I	J	K	L
baris-0				baris-1				baris-2			

Perintah P=&A[0][0] akan menyebabkan variabel P berisi alamat A[0][0]. Sehingga perintah cout<<*P akan menampilkan ‘A’. Perintah berikut mengubah isi variabel P.

```
P = &A[1][3];

cout << *P;
```

Jelas bahwa dalam perintah cout << *P di sini, yang akan ditampilkan bukan lagi ‘A’ tetapi ‘H’. Maka keluaran dari program di atas adalah:

```
AH
```

Catatan:

Perintah P=&A[0][0] dapat dituliskan sebagai P=A[0]. Kedua perintah ini identik, yaitu mengisi variabel P dengan anggota pertama dari larik A. Sedang perintah P=&A[1][0] sama dengan perintah P=A[1], yaitu akan mengisi variabel P dengan alamat dari A[1][0].

Larik 2 dimensi 3 x 5 diperlakukan sebagai kumpulan 3 larik 1 dimensi A[0], A[1], dan A[2], masing-masing mempunyai 5 elemen.

Contoh Program 2

```
1  #include<iostream>
2  #include<iomanip>
3  using namespace std;
4  int main() {
5     char A[3][5]= {
6        'A','B','C','D','E',
7        'F','G','H','I','J',
8        'K','L','M','N','O'
9     };
10    char *P;
11    P=A[0];
12    for(int i=0; i<15; i++)
13       cout<<setw(2)<<*P++;
14 }
```

Keluaran dari program di atas adalah:

A B C D E F G H I J K L M N O

Sekali lagi, larik 2 dimensi dapat diperlakukan sebagai larik 1 dimensi. Perintah *P++ adalah

menaikkan (*increment*) pointer satu indeks. Larik dengan 3 baris x 5 kolom diperlakukan sebagai larik 1 dimensi dengan 15 anggota.

Demikian juga:

- Larik 2 dimensi 4 x 8 sama dengan larik 1 dimensi dengan 32 anggota.
- Larik 3 dimensi 3 x 4 x 5 sama dengan larik 1 dimensi dengan 60 anggota.
- Larik 4 dimensi 3 x 4 x 5 x 6 identik dengan larik 1 dimensi 360 anggota.

Contoh Program 3

Ubahlah program 2 di atas sehingga tampilannya menjadi:

A B C D E

F G H I J

K L M N O

Jawab:

```
1  #include<iostream>
2  #include<iomanip>
3  using namespace std;
4  int main() {
5     char A[3][5]= {
6        'A','B','C','D','E',
```

```
7       'F','G','H','I','J',
8       'K','L','M','N','O'
9     };
10    char *P;
11    P=A[0];
12    for(int i=0; i<3; i++) {
13       for (int j=0; j<5; j++)
14          cout<<setw(2)<<*P++;
15       cout<<endl;
16    }
17 }
```

Dalam program di atas, kita dapat dua variabel sementara yaitu i dan j untuk menampilkan larik. Variabel i akan berjalan dari 0 sampai 2, dan variabel j akan berjalan dari 0 sampai 4. Setiap kali variabel j sudah melakukan pengulangan 0 sampai 4 yaitu sudah mencetak 5 angka, akan dicetak endl (baris 15) yang berarti berganti baris. Baru kemudian diulangi mencetak lima angka lagi.

Contoh Program 4

Buatlah larik int 2 dimensi berukuran 3 x 4. Tuliskan semua alamat dari anggota larik.

Jawab:

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4    int arr[3][4];
5    int *pA=arr[0];
6    for (int i=0; i<3; i++)
7      for (int j=0; j<4; j++)
8        cout<<"arr["<<i<<"]["<<j<<"]                    =
9  "<<pA++<<endl;
   }
```

Kita mendeklarasikan larik 2 dimensi berukuran 3 x 4 di baris 4. Di baris 5 pointer pA dideklarasikan dan diberi nilai alamat dari larik 2-dimensi tersebut. Maka nilai pA adalah alamat dari arr[0][0]. Selanjutnya di pengulangan, pA akan ditampilkan kemudian ditambah satu (*increment*) dengan perintah pA++.

Keluaran dari program adalah seperti di bawah.

arr[0][0] = 0x22fe00

arr[0][1] = 0x22fe04

arr[0][2] = 0x22fe08

arr[0][3] = 0x22fe0c

arr[1][0] = 0x22fe10

arr[1][1] = 0x22fe14

arr[1][2] = 0x22fe18

arr[1][3] = 0x22fe1c

arr[2][0] = 0x22fe20

arr[2][1] = 0x22fe24

arr[2][2] = 0x22fe28

arr[2][3] = 0x22fe2c

Tampilan di komputer anda akan sedikit berbeda nilai. Tapi dapat dilihat bahwa selisih dari tiap anggota ke anggota berikutnya adalah 4 byte, yaitu ukuran dari tipe data int.

2. Latihan

Dideklarasi larik 2-dimensi ukuran 5x6 dengan tipe data int. Mintalah masukan dari pengguna 2 angka untuk indeks baris dan kolom. Kemudian konversikan indeks 2-dimensi ini ke indeks 1-dimensi. Tampilkan isi larik memakai pointer ke anggota pertama larik ditambah indeks 1-dimensi.

Misal: masukan dari pengguna adalah 2 untuk baris dan 1 untuk kolom. Maka indeks 1-dimensi adalah 13. Tampilan adalah angka 13.

	0	1	2	3	4	5
0	0	1	2	3	4	5
1	6	7	8	9	10	11
2	12	13	14	15	16	17
3	18	19	20	21	22	23
4	24	25	26	27	28	29

3. Pointer Dan String

String dalam C++ dapat dideklarasikan sebagai larik dari tipe data char. Contoh adalah sebagai berikut.

```
char Nama[30];
```

Catatan:

Di dalam C++, string memerlukan satu karakter tambahan sebagai penanda bahwa string sudah berakhir. Karakter ini adalah '\0' atau `null`. Karenanya untuk deklarasi string, jumlahnya harus minimal satu karakter lebih banyak dari yang diperlukan.

Misal kita ingin membuat string berisi "BUKU" maka deklarasi variabel minimal harus dengan 5 anggota.

```
char item[5]={"BUKU"};
```

Karakter terakhir secara otomatis akan berisi `null`. Kita juga dapat mendeklarasikan string tanpa jumlah anggota tapi dengan menyebutkan nilai string, seperti:

```
char item[]={"BUKU"};
```

Maka C++ otomatis akan mengalokasikan 5 karakter untuk variabel `item`. Karakter terakhir akan berisi `null`.

Karena bentuk string adalah larik, kita dapat memakai pointer untuk memprosesnya. Berikut adalah contoh program memakai pointer untuk memproses string.

Contoh Program 1

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5    char kalimat[]=
6    {"Universitas Pamulang di Tangerang
7  Selatan"};
8    char *p=kalimat;
9    for (int i=0; i<sizeof(kalimat); i++){
10     if (i>=12 && i<=27) cout << *p;
11     p++;
12   }
   }
```

Baris 5 dan 6 mendeklarasikan kalimat sebagai larik char. Baris 7 deklarasi pointer ke char dan inisiasi pointer menunjuk ke alamat kalimat. Baris 8 sampai 11 pengulangan untuk seluruh char dalam larik. Tapi hanya ditampilkan huruf yang berada di posisi 12 sampai dengan 27 (baris 9). Tampilan dari program ini adalah:

Pamulang di Tang

Cobalah bermain dengan kalimat yang lain. Juga diubah posisi yang akan ditampilkan. Dapatkah anda membalik urutan huruf dalam kalimat tersebut?

Contoh Program 2

Program ini menghitung jumlah huruf besar dan huruf kecil dalam sebuah kaliimat.

```
1  #include<iostream>
2  #include<iomanip>
3  using namespace std;
4  int main()
5  {
6    // mendefinisikan string
7    char kalimat[]={"Selamat Belajar Bahasa
8  C++"};
9    char *pKarakter; // pointer ke char
10   int LowerCase=0; // penghitung huruf kecil
11   int UpperCase=0; // penghitung huruf besar
12   pKarakter=kalimat;  //pointer diisi alamat string
13   while(*pKarakter) {   // looping
14     char kar=*pKarakter;        // huruf di posisi ini
15     if(kar>='a' && kar<='z')     // cek huruf kecil
16       LowerCase++; //increment penghitung
```

```
17      if(kar>='A' && kar<='Z')    // cek huruf besar
18        UpperCase++; //increment penghitung
19      pKarakter++; // maju satu posisi
20    }
21    // tampilkan hasil
22    cout<<"Jumlah Huruf Kecil =
23  "<<LowerCase<<endl;
    cout<<"Jumlah Huruf Besar =
  "<<UpperCase<<endl;
  }
```

Baris 7 adalah deklarasi string kalimat berupa larik char. Dengan deklarasi ini, C++ akan menghitung jumlah huruf, kemudian dialokasikan memori sejumlah huruf tersebut dengan ditambah satu sebagai tempat null.

Baris 8 deklarasi pointer untuk char. Baris 11 adalah pengisian nilai untuk pointer, yaitu dengan perintah pKarakter=kalimat. Perintah ini dapat diganti dengan perintah pKarakter=&kalimat[0] karena larik sendiri merupakan pointer ke anggota pertama, yaitu indeks ke 0.

Dalam pengulangan while baris 12 sampai 19, diperiksa karakter satu per satu. Jika huruf kecil, penghitung huruf kecil LowerCase ditambah satu. Jika huruf besar, penghitung huruf besar UpperCase ditambah satu. Perintah pKarakter++ membuat indeks

bertambah satu, yaitu pointer menunjuk ke posisi huruf berikutnya.

Akhir program akan ditampilkan hasil penghitungan. Di bawah ini hasil tampilan.

Jumlah Huruf Kecil = 17

Jumlah Huruf Besar = 4

Cocokkan penghitungan di atas dengan huruf kecil dan besar di kalimat “Selamat Belajar Bahasa C++”. Cobalah mengganti kalimat dan lihat serta periksa lagi hasil penghitungan.

Contoh Program 3

Membuat enkripsi sederhana dengan menggeser setiap huruf besar dan kecil ke 3 posisi berikutnya dalam alfabet. Untuk huruf besar, jika digeser 3 posisi nilai huruf melewati ‘Z’ maka *wrap around* (berputar ke depan) menjadi ‘A’ kembali. Untuk huruf kecil, jika digeser 3 posisi nilai huruf melewati ‘z’ maka berputar ke depan menjadi ‘a’.

```
1  #include<iostream>
2  #include<iomanip>
3  using namespace std;
4  int main() {
5     // deklarasi kalimat
```

```
6   char kalimat[]="Jakarta Kota Metropolitan
7   XYZ";
8   char *P, kar; // variabel pointer dan char
9   P=kalimat; // pointer diisi alamat kalimat
10  cout<<"Kalimat: "<<kalimat<<endl; //kalimat
    asal
11  for(int i=0; i<sizeof(kalimat); i++) { // loop
12    kar=*P;     // ambil satu huruf
13    if (kar>='a' && kar<='z') { // huruf kecil
14      kar+=3;     // geser tiga posisi
15      if (kar>'z') // lebih dari 'z'
16        kar=kar-'z'+'a'-1;        // berputar ke 'a'
17    }
18        if (kar>='A' && kar<='Z') {      //      huruf
    besar
19      kar+=3;     // geser tiga posisi
20      if (kar>'Z') // lebih dari 'Z'
21        kar=kar-'Z'+'A'-1;        // berputar ke 'A'
22    }
23        *P++=kar; // letakkan ke kalimat
24  }
25  cout << "Enkript: "<< kalimat; //tampilkan hasil
```

	}

Program di atas memakai pointer untuk memproses huruf satu per satu. Tiap huruf digeser tiga posisi ke belakang dalam alfabet dengan perintah kar+=3 (baris 13 dan baris 18). Jika asalnya ‘A’ akan diubah menjadi ‘D’. Huruf kecil ‘c’ akan diubah menjadi ‘f’.

Tapi ketika digeser tiga posisi huruf melewati ‘z’ untuk huruf kecil, atau ‘Z’ untuk huruf besar, maka diputar (*wrap around*) ke huruf ‘a’ atau ‘A’ lagi. Misal huruf asal ‘Y’, ketika ditambah 3 akan lebih dari ‘Z’. Maka diubah dengan perintah kar=kar-’Z’+’A’-1 (baris 20). Maka ‘Y’ akan diubah menjadi ‘B’.

Untuk huruf kecil, jika huruf asal ‘x’, ketika ditambah 3 akan lebih besar dari ‘z’. Diubah dengan perintah kar=kar-’z’+’a’-1 (baris 15). Maka ‘x’ akan menjadi ‘a’.

Tampilan dari program di atas adalah seperti di bawah.

Kalimat: Jakarta Kota Metropolitan XYZ

Enkript: Mdnduwd Nrwd Phwursrolwdq ABC

Cobalah mengganti kalimat dengan kalimat anda sendiri kemudian di-compile serta dijalankan lagi.

C. LATIHAN SOAL

Soal 1

Ubahlah contoh program 2 di atas dengan menghitung

- jumlah angka dan
- karakter selain angka dan huruf.
- total huruf

Misal untuk kalimat “Jl. Candi Mulyo 1 No. 25” maka keluaran adalah:

Jumlah Huruf Kecil = 10

Jumlah Huruf Besar = 4

Jumlah Angka = 3

Jumlah Selainnya = 7

Total Huruf = 24

Soal 2

Buatlah program yang mencari satu karakter dalam string. Karakter yang dicari diminta masukan dari pengguna. Ditampilkan “Huruf yang dicari ada di posisi X” dengan X sebagai indeks huruf tersebut dalam kalimat. Jika huruf tidak ditemukan, ditampilkan “Huruf tidak ada dalam kalimat”.

Soal 3

Buatlah program yang mengganti sejumlah huruf dalam kalimat menjadi “X”. Misal kalimat “Password anda tangsel555, jangan diberikan ke orang lain”. Diubah menjadi “Password anda XXXXXXXXXX, jangan diberikan ke orang lain”. Posisi awal dan akhir diminta masukan dari pengguna.

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan Java*. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

BAB III

LARIK POINTER DAN POINTER KE POINTER

A. CAPAIAN PEMBELAJARAN

Mahasiswa dapat mengerti dan dapat menggunakan pointer dalam pengalamatan larik 2 dimensi.

B. URAIAN MATERI

1. Larik dari Pointer

Pointer dapat juga dibuat larik, yaitu banyak pointer dengan nama yang sama. Ini memudahkan dalam melakukan pengulangan pada larik.

Contoh Program 1

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     char Hobi[4][10]=
5        {"Renang","Basket","Lukis","Musik"};
6     char *pH[4];
```

```
7     for (int i=0; i<4; i++)
8        pH[i]=Hobi[i];
9     for (int i=0; i<4; i++)
10       printf("%s\n", pH[i]);
11 }
```

Di atas diperlihatkan larik dari string Hobi. Kemudian dibuat larik pointer char *pH[4]. Pointer diisi dengan alamat dari Hobi (baris 7 dan 8). Di baris 9 dan 10 ditampilkan larik string dengan memakai pointernya.

Contoh Program 2

```
1  #include <iostream>
2  #define JML_MHS 10  //Banyaknya Mahasiswa
3  #define JML_MK 5    //Jumlah Mata Kuliah
4  using namespace std;
5  int main(void) {
6     // deklarasi variabel dan pointernya
7     // nama variabel dengan tanda * adalah pointer
8     int i,k;
9     char *pNIM[JML_MHS], NIM[JML_MHS][15];
10
```

```
11    int
12  *pNilai[JML_MHS],NILAI[JML_MHS][JML_MK];
13    FILE *pF;  // pointer untuk file
14    // Mengisi pointer dengan alamat NIM dan
    NILAI
15    for(i=0; i<JML_MHS; i++) {
16       pNIM[i]=NIM[i];
17       pNilai[i]=NILAI[i];
18    }
19    // membuka file
20    if((pF=fopen("D:\\cpp\\data.txt","r")) ==NULL) {
21       printf("File tidak dapat dibuka\n");
22       return(0);
23    }
24    // membaca file
25    for(i=0; i<JML_MHS; i++) {
26       fscanf(pF, "%s", NIM[i]);
27       for(k=0; k<JML_MK; k++)
28          fscanf(pF,"%d",&NILAI[i][k]);
29    }
30    // menampilkan isi file dengan pointer
31    for(i=0; i<JML_MHS; i++) {
```

```
32      printf("%10s ", pNIM[i]);
33      for(k=0; k<JML_MK; k++)
34         printf("%02d ",*pNilai[i]++);
35      printf("\n");
36    }
37    // menutup file
      fclose(pF);
    }
```

Ada 10 mahasiswa yang datanya sudah ditulis dalam sebuah file. Tiap mahasiswa mengikuti 5 mata kuliah. Program di atas akan membaca satu file data.txt yang berisi data sebagai berikut.

1901501234 84 65 94 61 58

1901501235 65 49 83 72 59

1901501236 49 85 76 82 81

1901501237 79 84 86 92 76

1901501238 85 46 95 76 81

1901501239 78 56 98 45 26

1901501240 25 56 49 52 56

1901501241 89 56 89 45 86

1901501242 72 56 98 65 26

1901501243 83 45 98 56 36

Pendeklarasian untuk NIM adalah char NIM[JML_MHS][15]. Ini merupakan larik 2 dimensi dengan 10 mahasiswa, tiap mahasiswa mempunyai NIM sebanyak 15 huruf. Pendekarasian pointer untuk NIM adalah char *pNIM[JML_MHS]. Di baris 14 diberikan alamat dari NIM sebagai nilai dari pointer pNIM.

Pendeklarasian Nilai adalah int NILAI[JML_MHS][JML_MK] di baris 10. Pendeklarasian untuk pointer adalah int *pNilai[JML_MHS]. Di baris 15 diberikan alamat dari NILAI sebagai isi dari pointer pNilai.

Baris 23 sampai 27 adalah pembacaan isi file dan disimpan di variabel NIM dan NILAI. Di baris 29 sampai 34 program menampilkan isi file dengan memakai pointer.

2. Latihan

a. Ubahlah program di atas dengan menghitung nilai rata-rata setiap mahasiswa. Kemudian ditampilkan sebagai kolom terakhir dari nilai.
b. Ubahlah program di atas dengan dengan menuliskan NIM, semua nilai termasuk nilai rata-rata ke dalam satu file output.

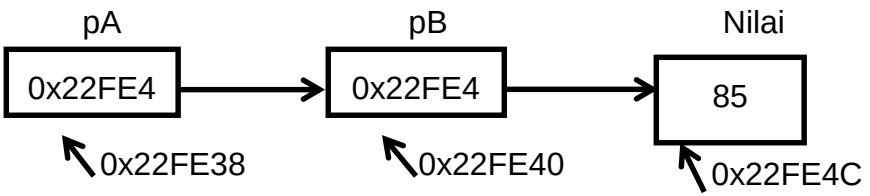
3. Pointer ke Pointer

Pointer ke pointer adalah bentuk referensi ganda atau rantai pointer. Biasanya, pointer mengandung alamat dari satu variabel. Ketika kita mendeklarasikan pointer ke pointer, pointer pertama mengandung alamat dari pointer kedua. Pointer kedua mengandung alamat dari variabel sebenarnya.

Contoh, perhatikan potongan program berikut.

```
int Nilai = 85;
int *pB;
int **pA;
pA = &pB;
pB = &Nilai;
```

Di bawah ini penggambaran potongan program di atas.



pA adalah pointer ke pointer. pB adalah pointer yang menunjuk ke tipe int. Nilai adalah variabel bertipe int.

Contoh Program 1

```
1  #include<iostream>
2  using namespace std;
3  int main() {
4          int Nilai = 85;
5          int *pB;
6          int **pA;
7          pB = &Nilai;
8          pA = &pB;
9          cout<<"Isi Nilai via pB = "<<*pB<<endl;
10         cout<<"Isi Nilai via pA = "<<**pA<<endl;
11 }
```

Contoh Program 2

```
1  #include<iostream>
2  using namespace std;
3  int main(int argc, char **argv) {
4    cout<<"Jumlah parameter: "<<argc<<endl;
5    char **argv2;
6    argv2=argv;
```

```
7    for (int i=0; i<argc; i++)
8        cout<<"Par"<<i<<": "<<*argv2++<<endl;
9  }
```

Di dalam program C++, metode main dapat menerima masukan parameter (*argument*) berupa:

- argc: int
  Argument Counter. Jumlah parameter yang diberikan pengguna.
- argv: pointer ke pointer ke char.
  Argument Value. Daftar parameter yang diberikan oleh pengguna. Parameter yang pertama adalah cara kita memanggil program.

Di program, *argv++ akan memberikan nilai argument, kemudian pointer akan dinaikkan satu. Pengulangan (*looping*) dengan perintah for di baris 7 dan 8 akan menampilkan parameter satu per satu.

Contoh di bawah adalah bagaimana menjalankan program di atas dengan Command Prompt dengan beberapa parameter yang berubah-ubah. Nama program adalah program2.exe.

D:\cpp>program2

Jumlah parameter: 1

Par0: program2

D:\cpp>program2 satu

Jumlah parameter: 2

Par0: program2

Par1: satu

D:\cpp>program2 satu dua

Jumlah parameter: 3

Par0: program2

Par1: satu

Par2: dua

D:\cpp>program2 satu dua tiga

Jumlah parameter: 4

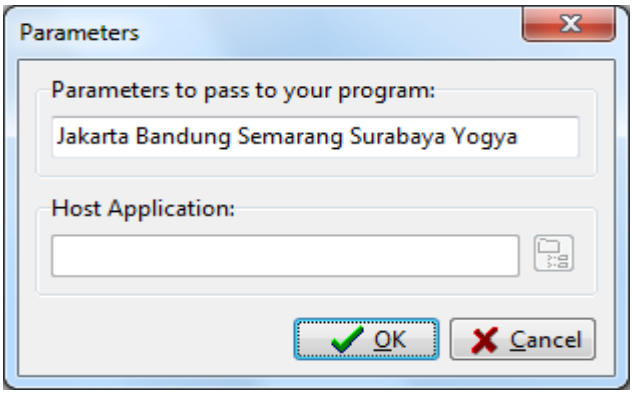
Par0: program2

Par1: satu

Par2: dua

Par3: tiga

Di dalam Dev-C++, parameter dapat diatur dengan memilih menu Execute → Parameters. Berikut contoh pengaturan.



Dengan lima parameter di atas, program akan menerima argc = 6 atau enam parameter. Parameter pertama adalah nama program yang secara implisit akan diberikan oleh sistem. Tampilan program dengan parameter di atas adalah sebagai berikut:

Jumlah parameter: 6

Par0: D:\cpp\Program2.exe

Par1: Jakarta

Par2: Bandung

Par3: Semarang

Par4: Surabaya

Par5: Yogya

Contoh Program 3

```
1  #include<iostream>
2  using namespace std;
3  int main(int argc, char **argv) {
4     cout<<"Jumlah parameter: "<<argc<<endl;
5     for (int i=0; i<argc; i++)
6        cout<<"Par"<<i<<": "<<argv[i]<< endl;
7  }
```

Program 3 sama fungsinya dengan program 2 sebelumnya, yaitu menampilkan jumlah parameter dan nilai parameter yang diberikan pengguna. Hanya saja, cara menampilkannya berbeda. Program 3 memperlakukan pointer ke pointer sebagai larik. Dan memakai notasi larik argv[i] untuk mengakses nilai parameter. Bandingkan penulisan ini dengan program 2.

4. Tugas

Soal 1

Buatlah program menghitung luas segi empat dengan parameter panjang dan lebar diberikan saat menjalankan program dengan memakai parameter di fungsi main().

Soal 2

Buatlah program menampilkan angka dari m sampai n di mana parameter m dan n diberikan saat menjalankan program dengan memakai 2 parameter di fungsi main().

5. Pengalokasian Memori Secara Dinamis

Dengan menggunakan pointer dapat dilakukan pengalokasian memori secara dinamis, yaitu pointer menunjuk ke lokasi memori yang dibuat ketika program sedang berjalan. Memori yang dialokasikan dapat dihapus kembali (dealokasi memori) kalau sudah diperlukan lagi.

Lokasi memori tersebut dikenal dengan nama: heap.

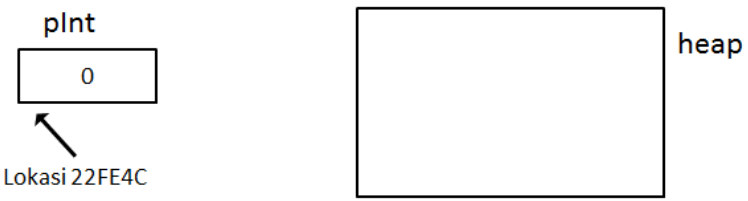
Operator yang digunakan untuk alokasi memori : new

Operator yang digunakan untuk dealokasi memori : *delete*

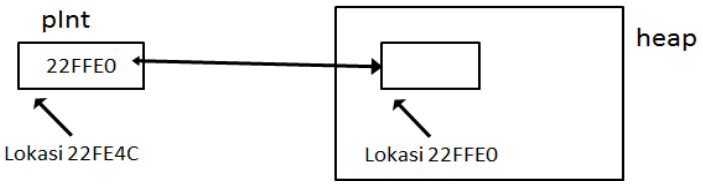
Contoh :

```
int *pInt;
pInt= new int;
```

Setelah int *pInt dieksekusi, program akan mempunyai satu variabel dengan pointer ke int. Tapi pointer ini belum menunjuk ke mana pun.



Setelah perintah pInt = new int dieksekusi, maka operating sistem mengalokasikan satu tempat di heap sebanyak tipe data int dan memberikan alamatnya ke variabel pInt.



Dengan mengetahui alamat di heap, maka program dapat memanipulasi nilai di heap dengan menggunakan alamat tersebut. Jangan lupa memakai operator * untuk mengakses nilai. Misal:

```
*pInt = 25;
```

Setelah program tidak lagi memakai heap, maka dapat dihapus (dikembalikan ke operating sistem) dengan perintah delete.

Peringatan:

Jangan lupa menghapus memori yang sudah dialokasikan ketika anda sudah selesai memakai memori tersebut. Kalau tidak dihapus, ini akan menyebabkan *memory leakage* (kebocoran memori) yaitu memori yang dialokasikan secara dinamis kemudian lupa dihapus. Terutama ketika *memory leakage* ini berada dalam satu fungsi yang berulang kali dipanggil. Maka *memory leakage* akan membuat program anda mengkonsumsi memori yang makin lama makin besar yang sebenarnya tidak dipakai.

```
delete pInt;
```

Perintah delete akan mengembalikan memori ke heap sehingga dapat dipakai oleh program lain.

Contoh Program 1

```
1  #include<iostream>
2  using namespace std;
3  int main(){
4          int *pArr;
5          pArr = new int[4];
```

```
6           pArr[0]=11;
7           pArr[1]=12;
8           pArr[2]=13;
9           pArr[3]=14;
10          for(int i=0;i<4;i++)
11                  printf("%3d ",pArr[i]);
12          delete []pArr;
13  }
```

Program di atas membuat alokasi memori dinamis sebesar 4 x int dengan perintah pArr = new int [4] di baris 5. Ini akan membuat larik 1 dimensi dengan tipe data int. Kemudian mengisi dengan nilai yang diinginkan. Setelah itu ditampilkan di baris 10-11. Selesai memakai, program akan menghapus memori dengan perintah delete []pArr di akhir program.

Keluaran dari program di atas adalah sebagai berikut.

```
11  12  13  14
```

Contoh Program 2

```
1  #include <iostream>
2  using namespace std;
3  int main () {
4    int i, n;
5    int *p;
6    int jumlah=0;
7    cout<<"Berapa angka yang ingin anda
8  masukkan? ";
9    cin >> i;
10   p = new int[i];
11   if (p == nullptr){
12     cout<<"Error mengalokasi memori";
13   }
14   for (n=0; n<i; n++) {
15      cout << "Masukan angka "<< n+1 << ": ";
16      cin >> p[n];
17      jumlah += p[n];
18   }
19   cout << "Anda memasukkan angka: ";
20   for (n=0; n<i; n++)
```

```
21      cout << p[n] << " ";
22    cout << endl;
23    cout << "Total: " << jumlah << endl;
24    delete[] p;
      }
```

Di baris 9 dialokasikan memori secara dinamis dengan perintah p = new int[i]. Kemudian diperiksa apakah sistem dapat memberikan memori dari heap. Jika (p==nullptr) berarti sistem tidak dapat mengalokasikan memori untuk program kita. Maka program tidak dilanjutkan. Sebaliknya, program akan melanjutkan dengan meminta masukan angka satu per satu dari pengguna.

Contoh keluaran dari program di atas adalah:

Berapa angka yang ingin anda masukkan? 5

Masukan angka 1: 27

Masukan angka 2: 64

Masukan angka 3: 90

Masukan angka 4: 154

Masukan angka 5: 47

Anda memasukkan angka: 27 64 90 154 47

Total: 382

Pengalokasian memori dilakukan di saat program berjalan, yaitu setelah pengguna memasukkan jumlah angka yang ingin dia masukkan. Jumlah angka ini tidak dapat diketahui saat kita membuat program. Maka perlu alokasi memori dinamis untuk ini.

C. LATIHAN SOAL

Soal 1

Buatlah program membuat larik 2 dimensi tipe data char untuk menyimpan nama dengan pengalokasian memori secara dinamis. Masing-masing nama memiliki panjang maksimum 30 huruf. Masukan dari pengguna adalah jumlah orang, dan program akan mengalokasi sejumlah nama untuk masing-masing orang. Kemudian minta pengguna memasukkan nama satu per satu. Akhir program akan menampilkan semua nama yang dimasukkan pengguna.

Soal 2

Untuk ujian nasional ada 3 mata pelajaran yang diujikan yaitu Bahasa Indonesia, Bahasa Inggris dan Matematika. Jumlah siswa tidak diketahui. Buatlah larik 2 dimensi untuk menampung ketiga nilai dari sejumlah siswa. Jumlah siswa akan ditentukan saat program dijalankan. Tampilkan masukan nilai dari pengguna, beserta rata-rata dari tiap siswa.

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan Java*. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

BAB IV

PROSEDUR DAN FUNGSI

A. CAPAIAN PEMBELAJARAN

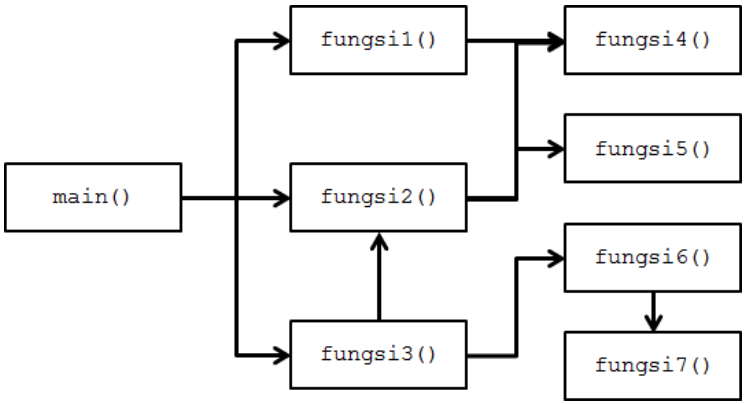
Mahasiswa dapat mengerti dan dapat menggunakan fungsi dan prosedur dalam pemrograman.

B. MATERI

1. Pengertian Prosedur dan Fungsi

Prosedur (*procedure*) dan fungsi (*function*) adalah suatu sub program yang mengerjakan suatu tugas tertentu. Perbedaan antara keduanya adalah fungsi akan mengembalikan suatu nilai tertentu ke buku ajar yang memanggilnya, sedangkan prosedur tidak mengembalikan suatu nilai. Tujuan dari prosedur dan fungsi adalah agar satu aplikasi dapat dipecah menjadi bagian-bagian kecil yang dapat dikelola dengan lebih mudah.

Untuk memudahkan penulisan, prosedur dan fungsi keduanya akan ditulis sebagai fungsi. Karena prosedur adalah fungsi secara khusus, yaitu fungsi yang tidak mengembalikan nilai.



Di gambar di atas, program dipecah menjadi 7 fungsi yang masing-masing mempunyai tugas khusus untuk diselesaikan. Tiap fungsi dapat saling memanggil.

Catatan:

Didalam C++, semua kode berada di dalam fungsi. Fungsi adalah fitur fundamental untuk semua bahasa pemrograman tingkat tinggi. Fungsi memungkinkan kita membuat program menangani tugas yang besar dan rumit dengan cara memecahnya menjadi banyak fungsi kecil-kecil yang mudah kita tangani.

Dalam tingkat lebih rendah, fungsi adalah alamat memori di mana kode badan fungsi disimpan. Di dalam kode sumber, alamat memori diberikan nama yang cukup menjelaskan kegunaan dari fungsi ini. Alamat memori ini dapat dipanggil, dan kontrol program akan beralih ke kode di alamat tersebut yang merupakan awal dari kode fungsi. Setelah selesai mengerjakan kode di badan fungsi, kontrol akan kembali ke instruksi setelahnya pemanggilan fungsi.

2. Pendefinisian Fungsi

Suatu fungsi dalam C++ harus diberi nama yang unik, yang membedakan satu sama lain. Sangat dianjurkan memberi nama fungsi sesuai dengan tugas yang dijalankan oleh fungsi tersebut.

Format umum definisi satu fungsi adalah seperti di bawah ini.

type name(parameter1, parameter2, ...) { statements }

Di mana:

- type adalah tipe data untuk nilai yang akan dikembalikan dari fungsi. Untuk prosedur, type ditulis sebagai void yang berarti tidak mengembalikan nilai apapun.
- name adalah nama fungsi.
- parameter. Setiap parameter terdiri dari tipe data dan nama. Parameter dapat didefinisikan sebanyak yang diperlukan dengan tiap parameter diantarai koma. Parameter merupakan pertukaran nilai antara pemanggil dengan fungsi yang dipanggil. Di dalam fungsi, parameter berlaku seperti layaknya variabel.
- statements adalah penjabaran aksi yang dilakukan oleh fungsi. Kadang disebut sebagai badan fungsi.

3. Lebih Jauh Mengenai Parameter

Parameter adalah nama-nama variabel yang dideklarasikan pada bagian header beserta tipe datanya.

Parameter ada dua:

- aktual (argumen) : parameter yang disertakan pada saat pemanggilan fungsi. Ini ada di buku ajar pemanggil.
- formal : parameter yang dideklarasikan pada bagian header fungsi. Ini ada di bagian fungsi.

4. Pemanggilan Fungsi

Fungsi bukan program yang berdiri sendiri, sehingga tidak dapat dijalankan secara langsung. Fungsi dipanggil namanya dari buku ajar pemanggil.

Ketika sebuah fungsi dipanggil, kendali program akan pindah ke fungsi tersebut. Instruksi-instruksi dalam badan fungsi akan dijalankan. Setelah semua instruksi selesai dilaksanakan, kendali program berpindah kembali ke buku ajar pemanggil, menjalankan instruksi yang ada sesudah pemanggilan fungsi.

Contoh Program 1

```
1  #include <iostream>
2  using namespace std;
3  int perkalian(int a, int b) {
4          int r=a*b;
5          return r;
6  }
7  int main () {
8          int z;
9          z = perkalian(3,4);
10         cout << "Hasil perkalian: " << z;
11 }
```

Program di atas dibagi atas dua fungsi, perkalian dan main. Ya, main juga merupakan fungsi. Hanya saja main adalah fungsi istimewa karena dia yang pertama kali dipanggil oleh sistem ketika program dijalankan. Fungsi lain akan dijalankan hanya jika dipanggil oleh main, langsung maupun tidak langsung.

Dalam contoh di atas, sub program main mulai dengan mendefinisikan variabel z dengan tipe int. Kemudian program memberi nilai z dengan cara memanggil fungsi perkalian dengan 2 parameter yaitu 3 dan 4. Ini disebut **parameter aktual**. Jika kita lihat deklarasi fungsi perkalian, fungsi ini membutuhkan dua parameter a dan b, keduanya dengan tipe int. Kedua parameter ini disebut **parameter formal**.

Maka ketika fungsi perkalian dipanggil, di dalam fungsi itu variabel a akan berisi nilai 3 dan variabel b akan berisi nilai 4. Kedua nilai ini didapat dari parameter aktual di fungsi main. Badan fungsi perkalian akan mengalikan a dengan b dan mengembalikan hasilnya ke buku ajar pemanggil yaitu main. Fungsi main akan menangkap nilai yang dikembalikan oleh fungsi perkalian dan menyimpan hasilnya di variabel z. Saat ditampilkan, z akan berisi nilai 12.

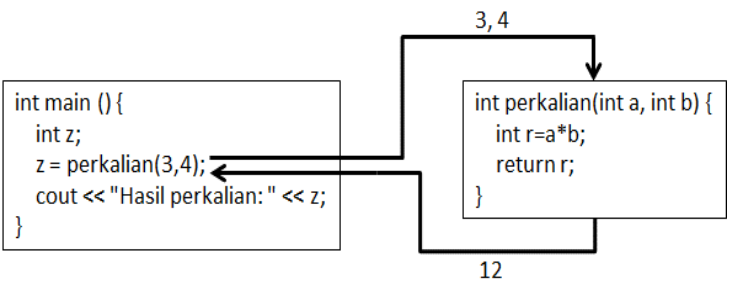
Keluaran dari program di atas adalah:

Hasil perkalian: 12

Perhatikan bahwa eksekusi program meninggalkan fungsi main di baris ke 9, yaitu ketika fungsi main

memanggil fungsi perkalian. Setelah selesai dari fungsi perkalian, kontrol program akan kembali ke fungsi main dan meneruskan eksekusi baris setelahnya, yaitu akan mengeksekusi baris ke 10.

Sub program mengembalikan nilai dengan perintah return value. Di contoh di atas, fungsi perkalian mengembalikan nilai ke fungsi main dengan perintah return r. Variabel r sebelumnya dihitung dengan mengalikan parameter pertama dan parameter kedua. Hasilnya adalah 12, dan ini dikembalikan ke fungsi main.



5. Prototipe Fungsi

Suatu fungsi harus didefinisikan di atas pemanggilnya. Sehingga ketika dipanggil, compiler sudah tahu bahwa ada sub program dengan nama tersebut. Contoh, fungsi perkalian ditulis di atas fungsi main. Cobalah mengubah program di atas dengan menuliskan fungsi perkalian di bawah fungsi main. Apakah program anda berhasil dikompilasi?

Kita dapat mendefinisikan suatu fungsi di bawah fungsi lain dan tetap dapat dipanggil dengan syarat prototipe fungsi dituliskan terlebih dahulu. Contoh, program di atas dapat dituliskan sebagai berikut.

Contoh Program 2

```
1  #include <iostream>
2  using namespace std;
3  int perkalian(int a, int b);
4  int main () {
5          int z;
6          z = perkalian(3,4);
7          cout << "Hasil perkalian: " << z;
8  }
9  int perkalian(int a, int b) {
10         int r=a*b;
11         return r;
12 }
```

Prototipe mempunyai format sebagai di bawah.

type name (parameter1, parameter2, ...);

Sedang badan fungsi dituliskan di bawah. Di contoh program di atas, prototipe fungsi perkalian dituliskan di baris ke 3.

```
int perkalian(int a, int b);
```

Sedang badan fungsi perkalian dituliskan di bawah di baris 9-12. Perhatikan bahwa header di badan fungsi harus dituliskan lagi secara lengkap.

Contoh Program 3

```
1  #include <iostream>
2  #include <math.h>
3  using namespace std;
4  int luas(int a, int b);
5  int keliling(int a, int b);
6  double diagonal(int a, int b);
7  int main() {
8          int panjang, lebar;
9          int l, k;
10         double d;
11         panjang=8;
12         lebar=5;
13         l = luas(panjang, lebar);
14         k = keliling(panjang, lebar);
```

```
15          d = diagonal(panjang, lebar);
16          cout << "Segi Empat" << endl;
17          cout << "Panjang : " << panjang << endl;
18          cout << "Lebar   : " << lebar << endl;
19          cout << "Luas    : " << l << endl;
20          cout << "Keliling: " << k << endl;
21          cout << "Diagonal: " << d << endl;
22  }
23  int luas(int a, int b){
24          return a*b;
25  }
26  int keliling (int a, int b){
27          return 2 * (a+b);
28  }
29  double diagonal(int a, int b){
30          return (double)sqrt(a*a+b*b);
31  }
```

Program mendefinisikan prototipe dari fungsi luas, keliling, dan diagonal di baris 4, 5, dan 6. Sedang badan fungsi dituliskan di baris 23 – 31. Dengan menuliskan prototipe di atas, maka fungsi main dapat memanggil

ketiga fungsi tersebut. Lakukan percobaan menghapus baris 4, 5, dan 6 di atas. Apakah C++ dapat mengkompilasi program? Mengapa?

Di dalam program di atas, di fungsi main pemanggilan fungsi luas dilakukan dengan memakai parameter aktual panjang dan lebar. Sedang di dalam fungsi luas, parameter formal bernama a dan b. Perbedaan nama antara parameter aktual dan parameter formal ini dibolehkan, asalkan tipe datanya sama. Dalam hal ini tipe data kedua parameter aktual haruslah int, sama dengan tipe data kedua parameter formal di fungsi luas.

Di dalam fungsi diagonal, dipanggil fungsi sqrt (akar) karena penghitungan diagonal adalah

$$\text{diagonal} = \sqrt{a^2 + b^2}$$

Untuk ini perlu dimasukkan definisi matematika dengan cara #include <math.h>. Juga perlu diberi tipe data double karena mengandung angka desimal. Dilakukan dengan cara casting ke double (baris 30).

Keluaran dari program di atas adalah sebagai berikut.

```
Segi Empat
Panjang : 8
Lebar   : 5
Luas    : 40
```

Keliling: 26

Diagonal: 9.43398

Contoh Program 4

```
1  #include <iostream>
2  using namespace std;
3  void judul();
4  int fungsiX(int x);
5  int main() {
6     int angka=-1;
7     judul();
8     while (angka!=0){
9        cout<<"Masukan angka (0 untuk berhenti): ";
10       cin>>angka;
11       if (angka==0) break;
12       cout<<"Hasil fungsi: "
13           <<fungsiX(angka)<<endl<<endl;
14       judul();
15    }
16 }
17 void judul() {
18    cout << "************************" <<endl;
```

```
19    cout << "Program menghitung hasil"<<endl;
20    cout << "   fungsi 3x^2+5*x+4"<<endl;
21    cout << "************************" <<endl;
22  }
23  int fungsiX(int x){
24    return 3*x*x + 5*x + 4;
25  }
```

Keluaran dari program di atas:

```
************************
Program menghitung hasil
   fungsi 3x^2+5*x+4
************************
Masukan angka (0 untuk berhenti): 4
Hasil fungsi: 72


************************
Program menghitung hasil
   fungsi 3x^2+5*x+4
************************
Masukan angka (0 untuk berhenti): 7
Hasil fungsi: 186
```

```
**********
Program menghitung hasil
  fungsi 3x^2+5*x+4
**********
Masukan angka (0 untuk berhenti): 0
```

Program di atas membuat 2 fungsi, judul dan fungsiX. Ini membuat fungsi main lebih rapi dan mudah dibaca. Terlihat bahwa main memakai pengulangan while untuk (1) meminta masukan dan (2) menghitung hasil fungsi dan (3) menampilkan judul.

Program ini juga menunjukkan satu keunggulan fungsi, yaitu dapat dipanggil berkali-kali. Di baris 14, pemanggilan ke fungsi judul dilakukan berkali-kali hanya dengan satu baris perintah. Ini menghemat penulisan.

6. Tugas

Soal 1

Buatlah satu program C++ dengan tiga fungsi sebagai berikut:

- Fungsi abs – mengembalikan nilai positif dari suatu angka.
- Fungsi minus – jika angka positif, kembalikan nilai negatif. Jika angka negatif, kembalikan nilai positif.

- Fungsi pangkat dua – kembalikan nilai angka dipangkatkan dua.

Ketiga fungsi memerlukan satu parameter masukan dan memberikan satu nilai kembalian. Mintalah masukan satu angka dari pengguna, dan hitunglah dengan ketiga fungsi di atas yaitu, nilai absolut,nilai minus, dan nilai pangkat dua.

Soal 2

Buatlah program C++ dengan 4 fungsi seperti di bawah ini.

- Fungsi max: mengembalikan maksimum dari 2 bilangan.
- Fungsi min: mengembalikan minimum dari 2 bilangan.
- Fungsi odd: mengembalikan bilangan yang ganjil. Bila keduanya ganjil, boleh dipilih salah satu.
- Fungsi even: mengembalikan bilangan yang genap. Bila keduanya genap, boleh dipilih salah satu.

Keempat fungsi memerlukan dua parameter masukan dan memberikan satu nilai kembalian. Mintalah masukan dua angka dari pengguna, dan tampilkan hasil dari keempat fungsi di atas.

Soal 3

Buatlah satu program C++ untuk lingkaran, memakai 3 fungsi dengan satu parameter jari-jari. Kembalian bertipe data double. Tiga fungsi adalah:

- Fungsi luas

- Fungsi keliling
- Fungsi diameter

Mintalah masukan dari pengguna satu angka yang merupakan jari-jari lingkaran. Kemudian tampilkan hasil dari ketiga fungsi di atas.

7. Fungsi yang Disediakan C++

Ada fungsi-fungsi yang sudah disediakan oleh C++ (*built-in functions*) dan dapat dipanggil oleh program kita. Misal perintah printf untuk mencetak ke layar. Fungsi-fungsi ini biasanya mengharuskan program memasukkan file header sebelum dapat dipakai. Misal fungsi sqrt untuk menghitung akar suatu angka. Fungsi ini mengharuskan kita memasukkan header math.h ke dalam program kita dengan perintah #include <math.h>. File header ini berisi definisi dan prototipe dari fungsi-fungsi matematika yang tersedia.

Contoh Program 5

```
1  #include <iostream>
2  #include <math.h>      // prototipe fungsi math
3  using namespace std;
4  #define PI 3.14159265  // definisi PI
5  void judul();
6  int main() {
```

```
7    double rad;   // menyimpan nilai radian
8    int sudut=10; // inisialisasi sembarang nilai > 0
9    while (sudut > 0) {     // berulang jika sudut > 0
10     judul();   // tampilkan judul
11     cout << "Masukkan sudut (0 untuk berhenti):
12 ";
13     cin >> sudut;          // masukan sudut
14     if (sudut <= 0) break;         // berhenti
15           // parameter harus dengan satuan radian.
16     // sudut derajat dikonversi ke radian
17     rad = (double)sudut * PI / 180;
18     // menampilkan sin, cos, tan
19     // dengan built-in function di math.h
20     cout << "Sinus  : " << sin(rad) << endl;
21     cout << "Cosinus: " << cos(rad) << endl;
22     cout << "Tangen : " << tan(rad) << endl;
23     cout << endl;
24   }
25   cout << "Keluar program"<< endl;
26 }
27 void judul() {
```

```
28    char bintang[]="*****************************";
29    cout << bintang << endl;
30    cout<<"Hitung sinus, cosinus, tangen"<<endl;
31    cout<<"dengan built-in function C++"<<endl;
32    cout << bintang << endl << endl;
      }
```

Contoh program di atas memakai fungsi-fungsi matematika yang sudah disediakan C++. Harus dimasukkan prototipe dari fungsi matematika dengan perintah #include <math.h> (baris 2). Selanjutnya fungsi sin, cos, tan dapat dipanggil dari program. Keluaran dari program di atas:

```
*****************************
Hitung sinus, cosinus, tangen
dengan built-in function C++
*****************************

Masukkan sudut (0 untuk berhenti): 49
Sinus  : 0.75471
Cosinus: 0.656059
Tangen : 1.15037

*****************************
Hitung sinus, cosinus, tangen
```

```
dengan built-in function C++
****************************

Masukkan sudut (0 untuk berhenti): 94
Sinus  : 0.997564
Cosinus: -0.0697565
Tangen : -14.3007

****************************
Hitung sinus, cosinus, tangen
dengan built-in function C++
****************************

Masukkan sudut (0 untuk berhenti): 0
Keluar program
```

Program akan menanyakan sudut yang akan dihitung. Sudut yang dimasukkan akan dikonversi ke radian dengan rumus:

radian = sudut * PI / 180

di mana PI = 3.14159265

Kemudian program memanggil dan menampilkan sinus, cosinus, dan tangen dari sudut yang diminta. Berhenti ketika masukan sudut 0 atau kurang dari 0.

Contoh Program 6

```
1  #include <iostream>
2  #include <time.h>      // prototipe fungsi time
3  using namespace std;
4  #define JML_DATA 50
5  void isi_random(int a[]);
6  void cetak(int a[]);
7  void cari(int a[], int x);
8  int main() {
9    int angka[JML_DATA];
10   isi_random(angka);
11   cetak(angka);
12   int n=10;
13   while (n>=0) {
14      cout<<endl
15          <<"Masukkan angka (-1 untuk berhenti): ";
16      cin>>n;
17      if (n<0) break;
18      cari(angka, n);
19   }
20 }
```

```
21  void isi_random(int a[]) {
22    srand(time(NULL));
23    for (int i=0; i<JML_DATA; i++)
24       a[i]=rand()%100;
25  }
26  void cetak(int a[]) {
27    char garis[]="--------------------------------";
28    cout << garis;
29    for (int i=0; i<JML_DATA; i++) {
30       if (i%10==0) cout << endl;
31       printf("%02d  ", a[i]);
32    }
33    cout<<endl<<garis<<endl;
34  }
35  void cari(int a[], int x) {
36    int i;
37    for (i=0; i<JML_DATA; i++) {
38       if (a[i]==x) {
39          cout<<"Ditemukan di indeks "<<i+1<<endl;
40          break;
41       }
```

```
42    }
43    if (i==JML_DATA)
44       cout << "Angka tidak ditemukan"<<endl;
45  }
```

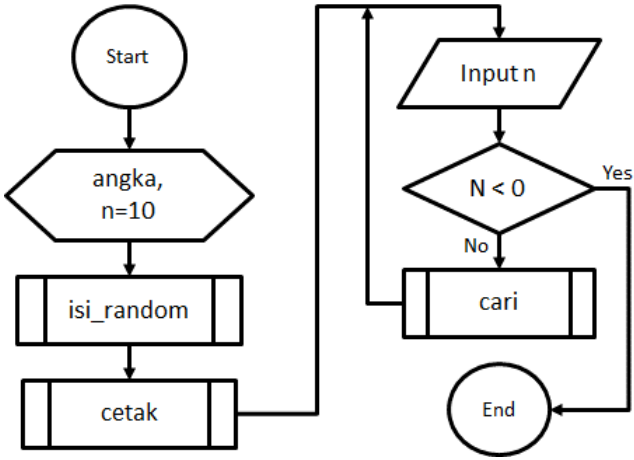
Untuk program di atas, kita dapat menerangkan per fungsi sebagai berikut.

Fungsi isi_random mengisi nilai random ke dalam larik 1-dimensi yang sudah dibuat.

Fungsi cetak mencetak seluruh nilai random ke layar.

Fungsi cari melihat seluruh nilai di larik dan membandingkan dengan angka yang dimasukkan pengguna. Jika ditemukan, akan ditampilkan indeksnya. Jika tidak ditemukan akan dituliskan di layar “Angka tidak ditemukan”.

Ini lebih mudah dimengerti dan lebih mudah dituliskan di dokumentasi nantinya. Sedang fungsi main sendiri dapat digambarkan dengan flowchart sebagai berikut.



C. LATIHAN

Buatlah satu program C++ dengan memakai fungsi yang sudah disediakan oleh C++:

- strcpy – menyalin isi string
- strcmp – membandingkan string
- strlen – menghitung panjang satu string
- strcat – menambahkan string

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan*

Java. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

BAB V

PROSEDUR DAN FUNGSI (LANJUTAN)

A. CAPAIAN PEMBELAJARAN

Mahasiswa dapat mengerti dan dapat menggunakan fungsi dan prosedur dalam pemrograman.

B. MATERI

1. Lingkup Variabel

Lingkup variabel menjelaskan jangkauan (*scope*) suatu variabel di dalam program. Yaitu daerah di mana variabel dapat diakses. Secara umum ada tiga jangkauan.

a. Di dalam satu blok pernyataan. Ini disebut variabel blok. Blok adalah kumpulan pernyataan yang dimulai dengan kurung kurawal buka “{“ ditutup dengan kurung kurawal tutup “}”.
b. Di dalam satu fungsi, disebut variable lokal. Ada juga yang menyamakan variabel blok dengan variabel lokal. Karena memang fungsi selalu dibuat dengan blok kurung kurawal buka - tutup.

c. Berlaku di semua fungsi, disebut variabel global. Variabel ini didefinisikan di luar semua fungsi dan dapat diakses di semua fungsi.

Di bawah akan dijelaskan satu per satu.

a. Variabel Blok

Contoh Program 1

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4          for (int i=0; i<10; i++){
5                  cout << i << " ";
6          }
7          cout << i;  // SALAH !
8  }
```

Dalam program di atas, variabel i dideklarasikan di dalam blok for. Karenanya lingkup variabel i hanya di dalam blok for. Setelah keluar dari blok itu, variabel i dihapus. Program di atas akan menerbitkan kesalahan saat kompilasi dengan pesan kesalahan:

[Error] 'i' was not declared in this scope

Ini karena variabel i sudah dihapus setelah keluar dari blok for. Bedakan dengan program di bawah ini.

Contoh Program 2

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4          int i;  // dideklarasikan di luar blok
5          for (i=0; i<10; i++){
6                  cout << i << " ";
7          }
8          cout << i;  // BENAR !
9  }
```

Variabel i dideklarasikan di luar blok for. Karenanya setelah keluar dari blok for, variabel i masih tetap ada.

b. Variabel Lokal

Variabel ini dideklarasikan di dalam satu fungsi dan hanya berlaku di dalam fungsi itu.

Contoh Program 3

```
1  #include <iostream>
2  using namespace std;
3  void judul(){
4          char bintang[]="**********";
5          cout << bintang <<endl;
6          cout << "Selamat datang" << endl;
7          cout << bintang <<endl;
8  }
9  int main(){
10         judul();
11         cout << bintang << endl;   // SALAH !
12 }
```

Fungsi main memanggil fungsi judul di baris 10. Fungsi judul membuat variabel bintang dan ditampilkan. Kembali ke fungsi main, baris 11 akan menampilkan variabel bintang. Tapi variabel bintang sudah dihapus ketika keluar dari fungsi judul. Maka ini akan menerbitkan pesan kesalahan:

[Error] 'bintang' was not declared in this scope

Di fungsi main, variabel bintang tidak dapat diakses karena variabel itu lokal, hanya berlaku di fungsi judul.

Contoh Program 4

```
1  #include <iostream>
2  using namespace std;
3  void f1() {
4     int angka = 44;  //variabel lokal di fungsi f1
5     cout << "Dalam f1, angka = " << angka << endl;
6  }
7  int main() {
8     int angka = 35; //variabel lokal di fungsi main
9     cout << "Dalam main, angka = " << angka <<
10 endl;
11    f1();
12    cout << "Dalam main, angka = " << angka <<
   endl;
   }
```

Keluaran dari program di atas adalah:

Dalam main, angka = 35

Dalam f1, angka = 44

Dalam main, angka = 35

Terlihat bahwa walau pun nama variabel sama, nilai variabel angka di fungsi main dan di fungsi f1 berbeda. Karena variabel angka di fungsi f1 hanya berlaku lokal. Nilainya terhapus ketika kembali ke buku ajar pemanggil.

c. Variabel Global

Variabel global dideklarasikan di luar semua blok. Dapat diakses di semua fungsi di dalam program.

Contoh Program 5

```
1  #include <iostream>
2  using namespace std;
3  int angka=50; // variabel global
4  void f1(){
5     cout << "Dalam f1 angka = " << angka << endl;
6  }
7  int main() {
8     cout << "Dalam main angka = " << angka <<
9  endl;
10    f1();
   }
```

Keluaran program di atas adalah:

Dalam main angka = 50

Dalam f1 angka = 50

Terlihat bahwa variabel angka dapat diakses di fungsi main maupun di fungsi f1. Walau pun di kedua fungsi itu variabel angka tidak dideklarasikan.

Contoh Program 6

```
1  #include <iostream>
2  using namespace std;
3  int angka=50; ; // variabel global
4  void f1() {
5    angka = 44;  // nilai variabel global diubah
6    cout << "Dalam f1, angka = " << angka << endl;
7  }
8  int main() {
9    cout << "Dalam main, angka = " << angka <<
10 endl;
11   f1();
12   cout << "Dalam main, angka = " << angka <<
   endl;
   }
```

Keluaran program di atas adalah:

Dalam main, angka = 50

Dalam f1, angka = 44

Dalam main, angka = 44

Variabel angka dapat diakses di fungsi main maupun di fungsi f1. Ketika nilai variabel ini diubah di fungsi f1, dan kontrol program kembali ke fungsi main, nilai variabel angka berubah menjadi 44. Bedakan dengan Contoh Program 4 di atas.

2. Latihan

Soal 1

Buatlah larik Akun dan Saldo sebanyak masing-masing 5 anggota. Buatlah fungsi HitungBunga untuk menghitung bunga setahun dengan bunga didefinisikan sebagai variabel global bernilai 6%. Tampilkan Akun, Saldo, dan Saldo Baru setelah HitungBunga. Buatlah semua variabel yang lain sebagai variabel lokal.

Soal 2

Global variabel juga bisa berupa constant. Buatlah satu variabel global const double PI=3.14159265. Kemudian buatlah program yang menghitung luas dan keliling lingkaran dengan jari-jari sebagai variabel lokal yang merupakan masukan dari pengguna.

3. Variabel Statis

Variabel statis adalah variabel lokal yang nilainya tidak berubah walau pun kontrol program sudah meninggalkan fungsi.

Sifat :

- Variabel lokal.
- Variabel tidak dihapus saat eksekusi fungsi berakhir
- Inisialisasi dalam deklarasi hanya dijalankan sekali selama aplikasi berjalan, yaitu saat fungsi pertama kali dipanggil.

Contoh Program 1

```
1  #include <iostream>
2  using namespace std;
3  void penghitung(); // prototipe fungsi
4  int main(){
5    penghitung();          // panggilan 1
6    penghitung();          // panggilan 2
7    penghitung();          // panggilan 3
8  }
9  void penghitung(){
10   static int cnt=0; //inisialisasi variabel statis
```

```
11    cnt++;   // increment
12    cout<<"Dipanggil "<<cnt<<" kali"<<endl;
13  }
```

Keluaran dari program di atas adalah;

Dipanggil 1 kali

Dipanggil 2 kali

Dipanggil 3 kali

Ketika fungsi penghitung dipanggil pertama kali di baris 5, variabel statis cnt diinisialisasi menjadi 0. Saat meninggalkan fungsi penghitung, variabel cnt tidak dihapus oleh sistem. Nilainya juga tetap dipegang. Saat meninggalkan fungsi penghitung, nilai cnt adalah 1, hasil dari increment di baris 11. Ketika dipanggil lagi oleh main di baris 6, inisialisasi int cnt=0 di baris 10 TIDAK DIJALANKAN. Maka nilai cnt tetap 1. Ketika di-increment, nilai cnt menjadi 2. Demikian juga ketika dipanggil ketiga kali.

Cobalah kata const dihapus di program di atas. Apa keluaran dari program anda?

4. Parameter

Program biasanya memerlukan pertukaran informasi antara fungsi dan pemanggilnya. Mekansme ini

dapat dicapai dengan penggunaan parameter. Fungsi dengan parameter dipanggil dengan cara menyebut nama fungsi tersebut beserta parameternya. Parameter di fungsi yang dipanggil disebut parameter formal, parameter yang diberikan oleh fungsi pemanggil disebut parameter aktual.

Aturan yang harus diperhatikan antara parameter formal dan aktual adalah sebagai berikut:

- Jumlah parameter aktual harus sama dengan jumlah parameter formal pada deklarasi prosedurnya. Bukan hanya sama, urutannya juga harus sesuai.
- Parameter aktual harus mempunyai tipe data yang sama dengan tipe data parameter formal yang bersesuaian
- Parameter aktual dituliskan sesuai dengan jenis parameter formal, yaitu apakah parameter masukan atau keluaran.

Berdasarkan maksud penggunaannya, terdapat dua jenis parameter formal:

a. parameter masukan (*input parameter*)
b. parameter keluaran (*output parameter*)

a. Parameter Masukan

Parameter yang berlaku sebagai masukan untuk fungsi yang dipanggil. Parameter ini tidak berubah nilainya di sisi fungsi pemanggil. Disebut juga sebagai parameter nilai (*parameter by value*).

Contoh Program 1

```
1  #include <iostream>
2  using namespace std;
3  void keliling(int, int); // prototipe fungsi
4  int main(){
5     int panjang=7, lebar=4;
6     keliling(panjang, lebar);
7  }
8  void keliling(int a, int b){
9     int k;
10    k = 2 * (a + b);
11    cout << "Keliling: " << k << endl;
12 }
```

Dalam program di atas, panjang dan lebar adalah parameter aktual yang diberikan oleh fungsi main. Keduanya sebagai parameter masukan bagi fungsi keliling, nilainya tidak diubah oleh fungsi keliling.

Catatan:

Untuk prototipe, diperbolehkan tidak menyebut nama variabel hanya disebutkan tipe datanya. Contoh adalah di baris 3:

```
void keliling(int, int);
```

Karena memang nama variabel tidak diperlukan di sini. Contoh prototipe di atas dapat dituliskan dengan nama variabel, misal:

```
void keliling(int x, int y);
```

Tapi karena baik x maupun y tidak diproses, maka penulisan nama ini tidak diharuskan (*optional*). Sebaliknya, penulisan nama variabel di dalam definisi harus dilakukan. Misal di baris 8:

b. Parameter Keluaran

Parameter keluaran adalah hasil dari fungsi yang dipanggil. Nilai dari parameter ini diharapkan akan diisi oleh fungsi yang dipanggil. Disebut juga parameter referensi (*parameter by reference*).

Contoh Program 2

```
1  #include <iostream>
2  using namespace std;
3  void keliling(int, int, int *); // prototipe fungsi
4  int main(){
5     int panjang=7, lebar=4, kel;
```

```
6     keliling(panjang, lebar, &kel);
7     cout << "Keliling: " << kel << endl;
8  }
9  void keliling(int a, int b, int *k){
10    *k = 2 * (a + b);
11 }
```

Fungsi keliling mengharapkan parameter yang diberikan adalah referensi. Fungsi pemanggil akan memberikan alamat dari variabel tempat menampung hasil. Fungsi pemanggil melakukan ini dengan memakai operator reference (&). Fungsi yang dipanggil mendeklarasikan parameter keluaran ini dengan dituliskan sebagai pointer (*). Contoh adalah:

```
void keliling(int, int, int *);
```

Baik prototipe maupun definisi fungsi keliling harus menuliskan parameter ketiga ini sebagai pointer. Dan mengharapkan fungsi pemanggil memberikan alamat dari variabel penerima sebagai parameter aktual.

5. Tugas

Soal 1

Buatlah satu fungsi untuk menukar nilai dari dua variabel. Dua parameter adalah parameter keluaran.

Soal 2

Buatlah fungsi untuk perjumlahan matriks. Dua matriks A dan B dijumlahkan menghasilkan matriks C. Hasilnya dikembalikan ke fungsi pemanggil.

Soal 3

Buatlah fungsi untuk mentranspose matriks. Matrik A ditranspose menjadi matrik B.

Contoh: Matrik A berukuran 4x3.

34	78	12
72	19	44
14	92	77
64	90	34

Menjadi matriks B berukuran 3x4.

34	72	14	64
78	19	92	90
12	44	77	34

Soal 4

Buatlah fungsi untuk perkalian matriks. Dua matriks A dan B dikalikan menghasilkan matriks C. Hasilnya dikembalikan ke fungsi pemanggil.

Contoh: Matrik A berukuran 2x3.

4	8	2
2	9	4

Dikalikan matriks B berukuran 3x4.

3	2	4	5
8	7	7	1
2	5	9	3

Hasilnya matriks C berukuran 2x4.

80	74	90	34
86	87	116	31

6. Fungsi Rekursif

Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri. Oleh karena itu, algoritma rekursif harus dinyatakan dalam prosedur atau fungsi karena hanya prosedur atau fungsi yang dapat dipanggil dalam sebuah program. Fungsi rekursif adalah fungsi yang memanggil

dirinya sendiri. Proses pemanggilan fungsi itu sendiri disebut rekursi.

Algoritma rekursif adalah algoritma yang memanggil dirinya sendiri dengan input yang telah kecil atau lebih sederhana. Karenanya, algoritma rekursif harus dinyatakan dalam fungsi karena hanya fungsi yang dapat dipanggil. Fungsi rekursif adalah fungsi yang memanggil diri sendiri. Proses pemanggilan fungsi rekursif disebut rekursi.

Badan fungsi rekursif disusun oleh dua bagian :

a. Basis : bagian ini berisi proses yang memberikan sebuah nilai definitif pada fungsi rekursif. Bagian ini menghentikan fungsi rekursif.
b. Rekurens : bagian yang mendefinisikan nilai dalam terminologi dirinya sendiri. Bagian ini akan memanggil diri sendiri dengan nilai input yang lebih sedikit atau lebih sederhana.

Contoh : Masalah Bilangan Genap

Mencari angka genap urutan ke-n dari bilangan natural. Bilangan natural adalah 0, 1, 2, ... Dapat digambarkan sebagai berikut.

Definisi angka genap	Dapat ditulis:
Genap ke-1 = 0	Genap(1) = 0

Genap ke-2 = 2 = 2 Genap ke-3 = 4 = 2+2 Genap ke-4 = 6 = 2+2+2 Genap ke-5 = 8 = 2+2+2+2	Genap(2) = 2+Genap(1) Genap(3) = 2+Genap(2) Genap(4) = 2+Genap(3) Genap(5) = 2+Genap(4) . . Genap(n)= 2+Genap(n-1)

Dari tabel di atas, dapat dibuat:

- Basis: jika n=1, genap = 0.
- Rekurens: jika n>1, genap = 2 + genap (n-1)

Di bawah ini adalah program C++ untuk mencari bilangan genap ke-n dengan metode rekursif.

```
1  #include <iostream>
2  using namespace std;
3  int genap(int);   // prototipe
4  int main() {
5     int k=8;
6     cout<<"Genap ke "<<k<<" adalah "<<genap(k);
7  }
```

```
8  int genap(int n) {
9     if (n==1) return 0;    // basis
10    return 2 + genap(n-1); // rekurens
11 }
```

Keluaran dari program di atas adalah:

Genap ke 8 adalah 14

Baris 6 terlihat fungsi main memanggil fungsi genap. Baris ke 8-11 adalah fungsi genap. Jika n==1 maka hasilnya adalah 0. Ini disebut basis. Sedang di baris 10 adalah rekurens. Terlihat bahwa fungsi genap memanggil dirinya sendiri dengan nilai masukan n-1, yaitu nilai yang lebih kecil. Nilai n akan terus berkurang sampai mencapai n=1, di mana fungsi akan mengembalikan nilai basis. Dan siklus akan berbalik ke nilai yang lebih besar, satu per satu sampai mencapai nilai n yang diminta.

Catatan:

Nilai genap ke-n ini sebenarnya dapat dengan mudah dicari memakai algoritma sederhana genap=2*(n-1). Tapi tujuan dari algoritma di atas adalah menunjukkan fungsi rekursif.

Contoh : Masalah Faktorial

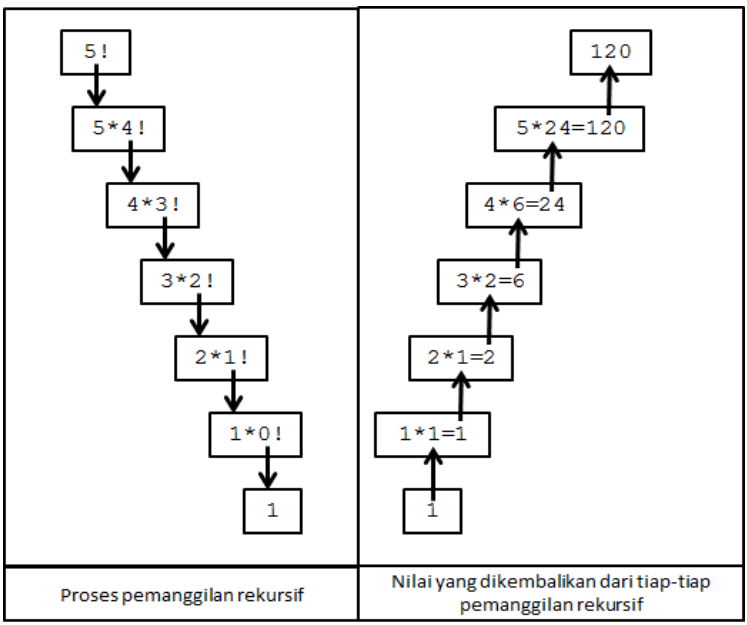
Definisi faktorial (!)	Dapat ditulis:
0! = 1	0! = 1
1! = 1	1! = 1*0!
2! = 2*1	2! = 2*1!
3! = 3*2*1	3! = 3*2!
4! = 4*3*2*1	4! = 4*3!
5! = 5*4*3*2*1	5! = 5*4!
.	.
.	.
n! = n*(n-1)*(n-2)*...*3*2*1	n! = n*(n-1)!

Basis: jika n=1 maka n! = 1

Rekurens: jika n>0 maka n! = n * (n-1)!

Pseudo code untuk program faktorial dapat kita tuliskan sebagai berikut.

function Fakt (input n : integer) -> integer



DEKLARASI

DESKRIPSI

if n = 0 then

    return 1 (basis)

else

    n*Fakt(n-1) (rekurens)

end if

Misalkan n=5, alur fungsi rekursif dapat digambarkan sebagai di bawah.

Program untuk faktorial:

```
1  #include <iostream>
2  using namespace std;
3  int Fakt(int);   // prototipe
4  int main() {
5     int k=8;
6     cout<<"Faktorial ke "<<k<<" adalah "<<Fakt(k);
7  }
8  int Fakt(int n) {
9     if (n==0) return 1;   // basis
10    return n * Fakt(n-1); // rekurens
11 }
```

Keluaran dari program di atas adalah:

Faktorial ke 8 adalah 40320

Di fungsi Fakt baris 8-11, kita melihat ada dua bagian, basis dan rekurens. Di dalam basis, jika n=0 hasilnya adalah 1. Di dalam bagian rekurens, fungsi memanggil diri sendiri dengan nilai masukan lebih kecil, yaitu n-1.

C. LATIHAN

Soal 1

Buatlah fungsi rekursif pangkat dengan 2 parameter masukan yaitu x dan y. Hasil dari fungsi pangkat adalah x pangkat y. Contoh keluaran program:

5 pangkat 4 adalah 625

Soal 2

Buatlah fungsi rekursif deret bilangan genap yang menampilkan deret bilangan genap dari besar ke kecil sebanyak cacah yang diminta. Contoh keluaran program:

7 bilangan genap: 14 12 10 8 6 4 2

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan Java*. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

BAB VI

PENCARIAN

A. CAPAIAN PEMBELAJARAN

Mahasiswa dapat mengerti dan dapat menggunakan algoritma pencarian beruntun (*sequential search*) dalam pemrograman.

B. MATERI

1. Konsep Pencarian

Pencarian (*Searching*) adalah proses menemukan nilai (data) pada larik atau array, baik data yang terdapat pada *primary memory* atau dalam *secondary memory*.

Ada 2 tipe pencarian berdasarkan lokasi memori:

a. Pencarian Internal

Pencarian internal adalah pencarian pada data yang tersimpan di dalam larik atau array.

b. Pencarian eksternal

Pencarian eksternal adalah pencarian pada data yang tersimpan dalam sebuah arsip atau File, misal file dengan ekstensi .txt

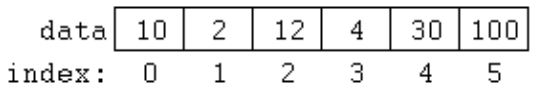
Contoh:

Diketahui int data[]={10,2,12,4,30,100};

int d=4 ;

Tentukan d berada pada indeks berapa?

Jawab:



Data d berada pada indeks ke-3

2. Pencarian Beruntun (sequential search)

Pencarian beruntun (*sequential search*) adalah proses pencarian dengan membandingkan data secara urut dari satu elemen ke elemen lainnya sampai data ditemukan atau sudah habis dibandingkan.

Algoritma sequential search:

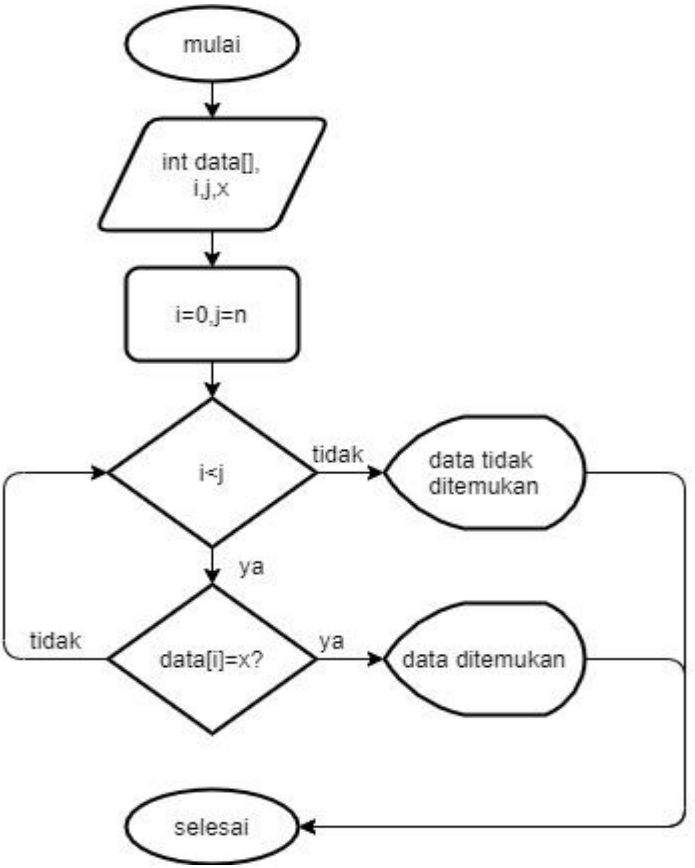
a. Input x (data yang dicari)

b. Bandingkan x dengan data ke-i sampai n

c. Jika ada data yang sama dengan x maka cetak pesan "ada"

d. Jika tidak ada data yang sama dengan x cetak pesan "tidak ada"

Flowchart



Contoh 1 :

Diketahui int data1[]={10,2,12,4,30,100};

int d=4 ;

Tentukan d berada pada indeks berapa menggunakan metode pencarian beruntun (*sequential search*)?

Jawab:

data1[]	10	2	12	4	30	100
index	0	1	2	3	4	5

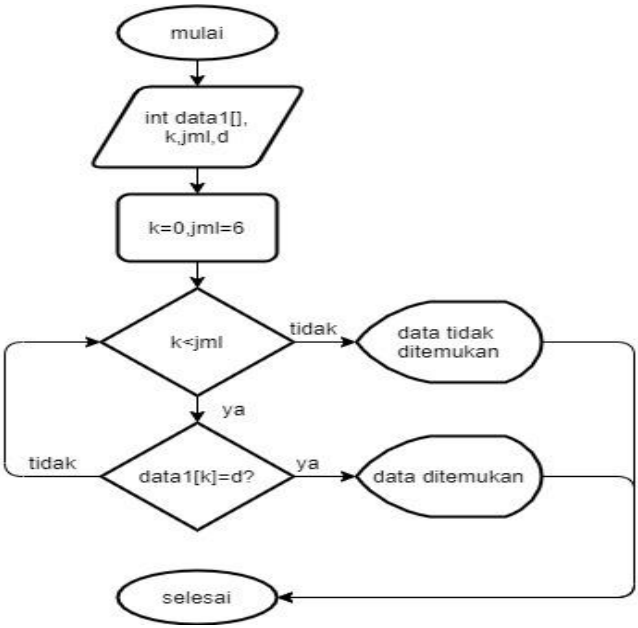
a. data1[0]=10, apakah data1[0]==d (Tidak maka geser ke elemen berikutnya)

b. data1[1]=2, apakah data1[1]==d (Tidak maka geser ke elemen berikutnya)

c. data1[2]=12, apakah data1[2]==d (Tidak maka geser ke elemen berikutnya)

d. data1[3]=4, apakah data1[3]==d (Ya, pencarian selesai dan data ditemukan pada indeks ke-3).

Flow chart



Program:

```
1   #include<iostream>
2   #include<cstdlib>
3   using namespace std;
4   void cari1(int data1[],int jml,int d, int *dx);
5   int main()
6   {
7           system("cls");
8           int data1[]={10,2,12,4,30,100};
9           int d,k,jml=6;int dx;
10          cout<<"Elemen Array :  ";
```

```
11          for(k=0;k<jml;k++)cout<<data1[k]<<"
";cout<<endl;
12          cout<<"Masukan data yang akan dicari ?
";cin>>d;
13          cari1(data1,jml,d,&dx);
14          if(dx!=-1)
15                  cout<<"Data yang dicari
ditemukan pada indeks ke-"<<dx<<endl;
            else
16                  cout<<"Data yang dicari tidak ada
17  dalam array"<<endl;
    }
18  void cari1(int data1[],int jml,int d, int *dx)
19  {
20          int k=0;
21          while(k<jml-1 && data1[k]!=d)k++;
22          if(data1[k]==d)*dx=k;
23          else *dx=-1;
24  }
25
```

Contoh 2 :

Diketahui int data2[]={20,12,120,9,8,40};

int d=40 ;

Tentukan d berada pada indeks berapa menggunakan metode pencarian beruntun (*sequential search*)?

Jawab:

data2[]	20	12	120	9	8	40
index	0	1	2	3	4	5

a. data2[0]=20, apakah data2[0]==d (Tidak maka geser ke elemen berikutnya)

b. data2[1]=12, apakah data2[1]==d (Tidak maka geser ke elemen berikutnya)

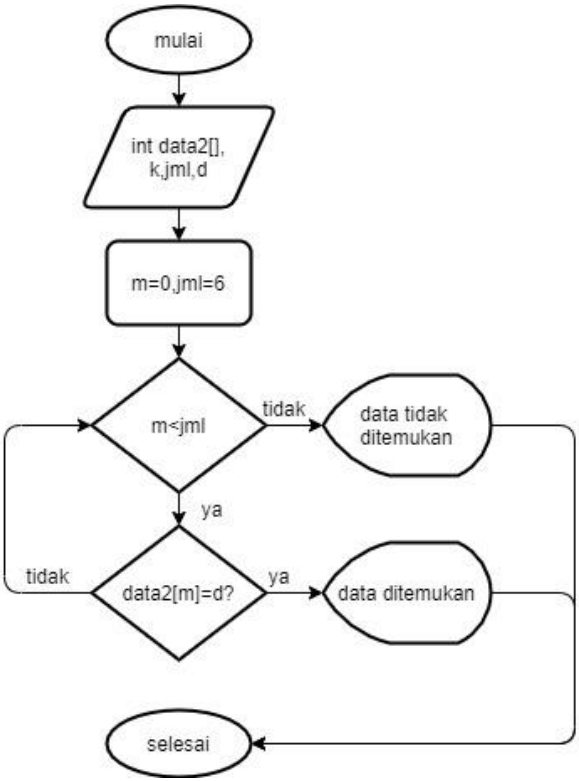
c. data2[2]=120, apakah data2[2]==d (Tidak maka geser ke elemen berikutnya)

d. data2[3]=9, apakah data2[3]==d (Tidak maka geser ke elemen berikutnya).

e. data2[4]=8, apakah data2[4]==d (Tidak maka geser ke elemen berikutnya).

f. data2[5]=40, apakah data2[5]==d (Ya, pencarian selesai dan data ditemukan pada indeks ke-5).

Flowchart



Program:

```
1  #include<iostream>
2  #include <cstdlib>
3  using namespace std;
4  void cari2(int data2[],int jml,int d, int *dx);
5  int main()
6  {
7          system("cls");
```

```
8       int data2[]={20,12,120,9,8,40};
9       int d,m,jml=6;int dx;
10      cout<<"Elemen Array :  ";
11      for(m=0;m<jml;m++)cout<<data2[m]<<"
";cout<<endl;
12      cout<<"Masukan data yang akan dicari ?
";cin>>d;
        cari2(data2,jml,d,&dx);
13
        if(dx!=-1)
14
        cout<<"Data yang dicari ditemukan pada
15  indeks ke-"<<dx<<endl;
        else
16
        cout<<"Data yang dicari tidak ada dalam
17  array"<<endl;
    }
18  void cari2(int data2[],int jml,int d, int *dx)
19  {
20      int m=0;
21      while(m<jml-1 && data2[m]!=d)m++;
22      if(data2[m]==d)*dx=m;
23      else *dx=-1;
24  }
```

25	

Contoh 3 :

Diketahui int data3[]={20,12,120,9,8,40};

int d=90 ;

Tentukan nilai dari variabel d berada pada indeks berapa menggunakan metode pencarian beruntun (*sequential search*)?

Jawab:

data3[]	20	12	120	9	8	40
index	0	1	2	3	4	5

a. data3[0]=20, apakah data3[0]==d (Tidak maka geser ke elemen berikutnya)

b. data3[1]=12, apakah data3[1]==d (Tidak maka geser ke elemen berikutnya)

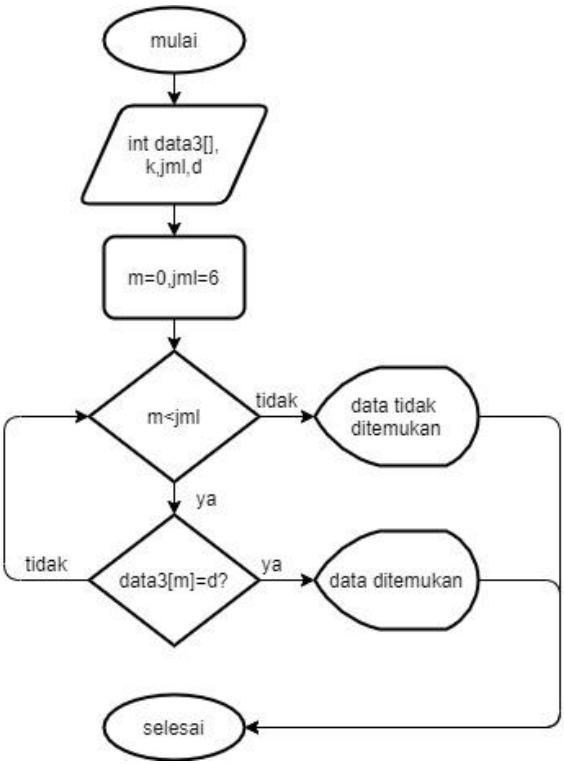
c. data3[2]=120, apakah data3[2]==d (Tidak maka geser ke elemen berikutnya)

d. data3[3]=9, apakah data3[3]==d (Tidak maka geser ke elemen berikutnya).

e. data3[4]=8, apakah data3[4]==d (Tidak maka geser ke elemen berikutnya).

f. data3[5]=40, apakah data3[5]==d (Tidak, index array sudah habis data tidak ditemukan).

Flowchart



Program:

```
1  #include<iostream>
2  #include <cstdlib>
3  using namespace std;
4  void cari3(int data3[],int jml,int d, int *dx);
5  int main()
```

```
{
        system("cls");
        int data3[]={20,12,120,9,8,40};
        int d,m,jml=6;int dx;
        cout<<"Elemen Array :  ";
        for(m=0;m<jml;m++)cout<<data3[m]<<" ";cout<<endl;
        cout<<"Masukan data yang akan dicari ? ";cin>>d;
        cari3(data3,jml,d,&dx);
        if(dx!=-1)
        cout<<"Data yang dicari ditemukan pada indeks ke-"<<dx<<endl;
        else
        cout<<"Data yang dicari tidak ada dalam array"<<endl;
}
void cari3(int data3[],int jml,int d, int *dx)
{
        int m=0;
        while(m<jml-1 && data3[m]!=d)m++;
        if(data3[m]==d)*dx=m;
```

```
23          else *dx=-1;
24  }
25
```

Contoh 4 :

Diketahui int data4[]={20,12,120,9,8,40};

int cari=12 ;

Tentukan nilai dari variabel cari berada pada indeks berapa menggunakan metode pencarian beruntun (*sequential search*)?

Jawab:

data4[]	20	12	120	9	8	40
index	0	1	2	3	4	5

a. data4[5]=40, apakah data4[5]==cari (Tidak maka geser ke elemen sebelumnya)

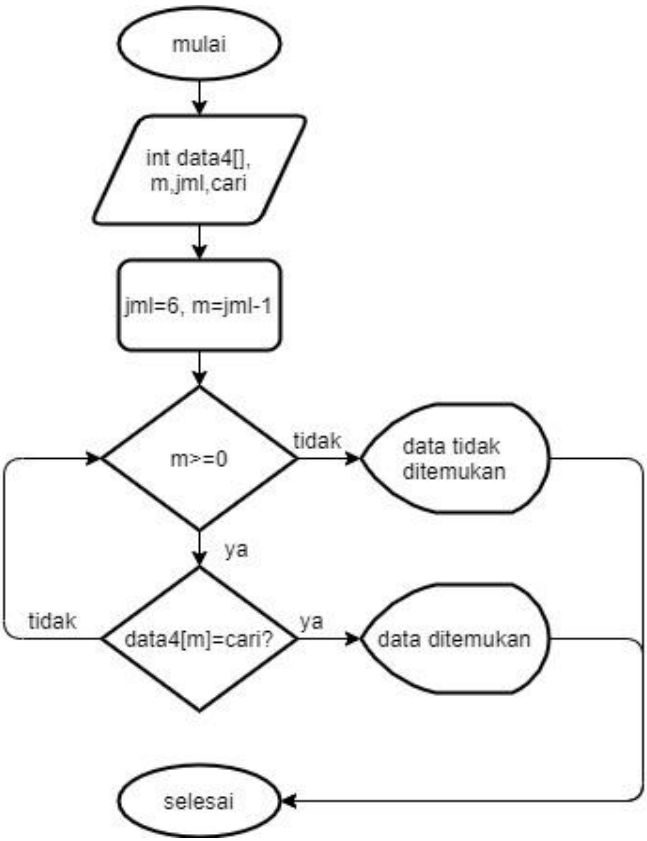
b. data4[4]=8, apakah data4[4]==cari (Tidak maka geser ke elemen sebelumnya)

c. data4[3]=9, apakah data4[3]==cari (Tidak maka geser ke elemen sebelumnya)

d. data4[2]=120, apakah data4[2]==cari (Tidak maka geser ke elemen sebelumnya).

e. data4[1]=12, apakah data4[1]==cari (Ya maka selesai data ditemukan).

Flowchart



Program:

```
1  #include<iostream>
2  #include<cstdlib>
3  using namespace std;
```

```
4   void cari4(int data4[],int jml,int cari, int *dx);
5   int main()
6   {
7           system("cls");
8           int data4[]={20,12,120,9,8,40};
9           int cari=12,m,jml=6;int dx;
            cout<<"Elemen Array :  ";
10          for(m=0;m<jml;m++)cout<<data4[m]<<"
11  ";cout<<endl;
12          cout<<"Data yang dicari adalah "<<cari;
13          cari4(data4,jml,cari,&dx);
            if(dx!=-1)
14          cout<<"\nData yang dicari ditemukan
    pada indeks ke-"<<dx<<endl;
15          else
16          cout<<"\nData yang dicari tidak
    ditemukan"<<endl;
17  }
18  void cari4(int data4[],int jml,int cari, int *dx)
19  {
20          int m=jml-1;
21          while(m>=0 && data4[m]!=cari)m--;
```

22	if(data4[m]==cari)*dx=m;
23	else *dx=-1;
	}

Contoh 5 :

Diketahui int data5[]={20,12,120,9,8,40};

int cari=12 ;

Tentukan nilai dari variabel cari berada pada indeks berapa menggunakan metode pencarian beruntun (*sequential search*)?

Jawab:

data5[]	20	12	120	9	8	40
index	0	1	2	3	4	5

a. data5[5]=40, apakah data5[5]==cari (Tidak maka geser ke elemen berikutnya)

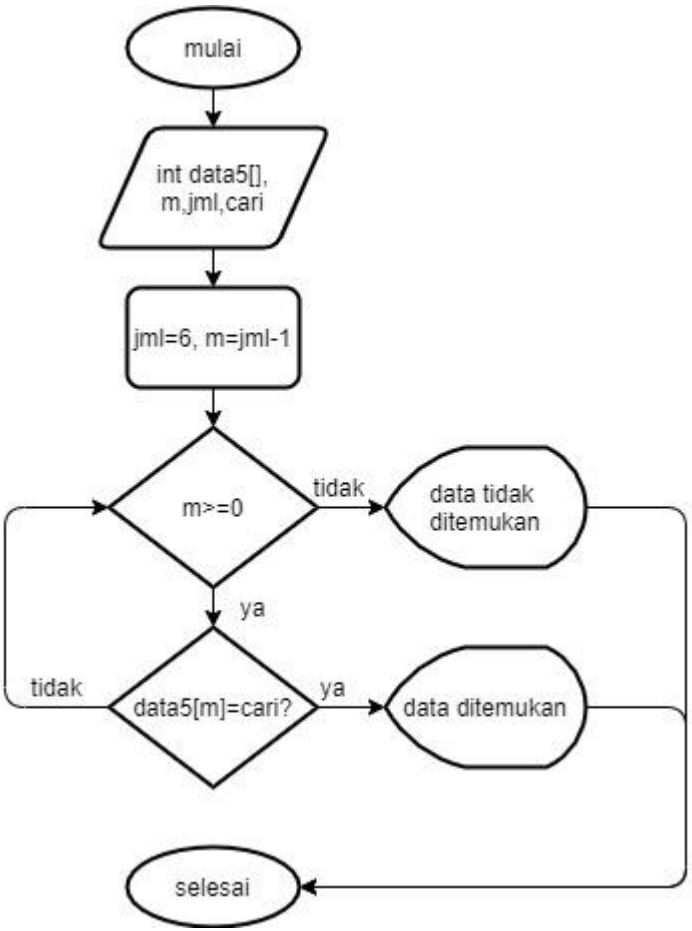
b. data5[4]=8, apakah data5[4]==cari (Tidak maka geser ke elemen berikutnya)

c. data5[3]=9, apakah data5[3]==cari (Tidak maka geser ke elemen berikutnya)

d. data5[2]=120, apakah data5[2]==cari (Tidak maka geser ke elemen berikutnya).

e. data5[1]=12, apakah data5[1]==cari (Ya maka selesai data ditemukan).

Flowchart



Program:

1	#include<iostream>
2	#include<cstdlib>

```
3   using namespace std;
4   void cari5(int data5[],int jml,int cari, int *dx);
5   int main()
6   {
7           system("cls");
8           int data5[]={20,12,120,9,8,40};
9           int cari=12,m,jml=6;int dx;
10          cout<<"Elemen Array :  ";
11          for(m=0;m<jml;m++)cout<<data5[m]<<"
    ";cout<<endl;
12          cout<<"Data yang dicari adalah "<<cari;
13          cari5(data5,jml,cari,&dx);
14          if(dx!=-1)
15          cout<<"\nData yang dicari ditemukan
    pada indeks ke-"<<dx<<endl;
16          else
17          cout<<"\nData yang dicari tidak
    ditemukan"<<endl;
    }
18  void cari5(int data5[],int jml,int cari, int *dx)
19  {
20          int m;
```

```
21          for(int m=jml-1;m>=0;m++){
22                  if(data5[m]==cari)break;
23          }
24          if(data5[m]==cari)*dx=m;
25          else *dx=-m;
26  }
27
```

C. LATIHAN

Soal 1

Buatlah program pencarian beruntun dengan membandingkan data dari indeks terbesar ke indeks terkecil.

Soal 2

Diketahui data[]={22,45,32,11,20,55,70} dan data yang dicari adalah 55. Tuliskan proses paencarian data tersebut.

Soal 3

Buatlah sebuah program pencarian data pada arsip dengan extensi .txt

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan Java*. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

BAB VII

PENCARIAN (Lanjutan)

A. CAPAIAN PEMBELAJARAN

Mahasiswa dapat mengerti dan menggunakan algoritma pencarian bagidua (*binary search*) dalam pemrograman.

B. MATERI

1. Pencarian Bagi dua (*Binary Search*)

Pencarian bagidua (binary search) adalah proses pencarian dengan membagi dua bagian data kemudian membandingkan isi data bagian pertama atau bagian kedua dengan data yang akan dicari. Pada metode ini membutuhkan nilai indeks terkecil dan terbesar untuk dijumlahkan kemudian dibagi dua. Pencarian bagi dua (binary search) hanya bisa dilakukan pada data yang sudah urut. Metode ini lebih efisien dibandingkan dengan metode pencarian *sequential search* atau pencarian beruntun.

2. Algoritma Pecarian Bagidua data *ascending*

Step 1:

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal+akhir) div 2

Step 2:

Bandingkan data, apakah data[c] = d ?

d adalah data yang sedang dicari.

apabila data[c] = d, data ditemukan, selesai.

apabila data[c] > d, pencarian berpindah ke bagian kiri dengan cara

mengubah nilai c menjadi: akhir = c-1

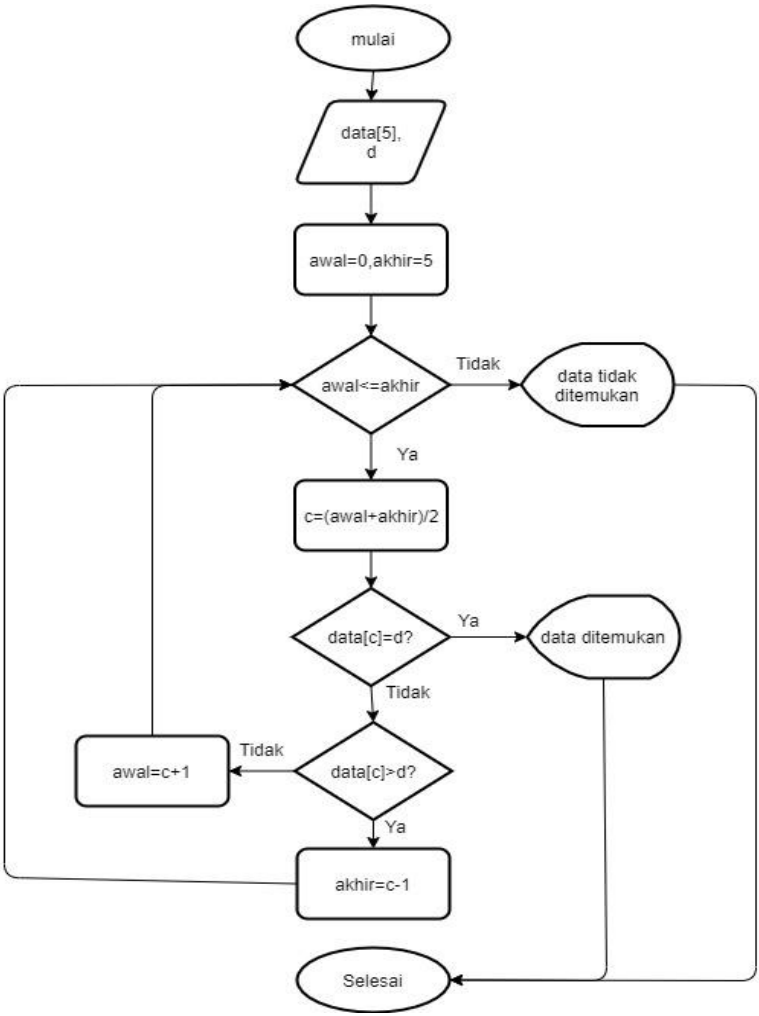
apabila data[c] < d, pencarian berpindah ke bagian kanan dengan cara

mengubah nilai c menjadi awal = c + 1

step 3 :

Kembali ke **step 1** hingga d (data yang dicari) ditemukan atau hingga nilai awal > akhir .

3. Flowchart Algoritma Pecarian Bagidua data *ascending*



4. Algoritma Binary *Search* data *Descending*

Step 1:

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal+akhir) div 2

Step 2:

Bandingkan data, apakah data[c] = d ?

d adalah data yang sedang dicari.

apabila data[c] = d, data ditemukan, selesai.

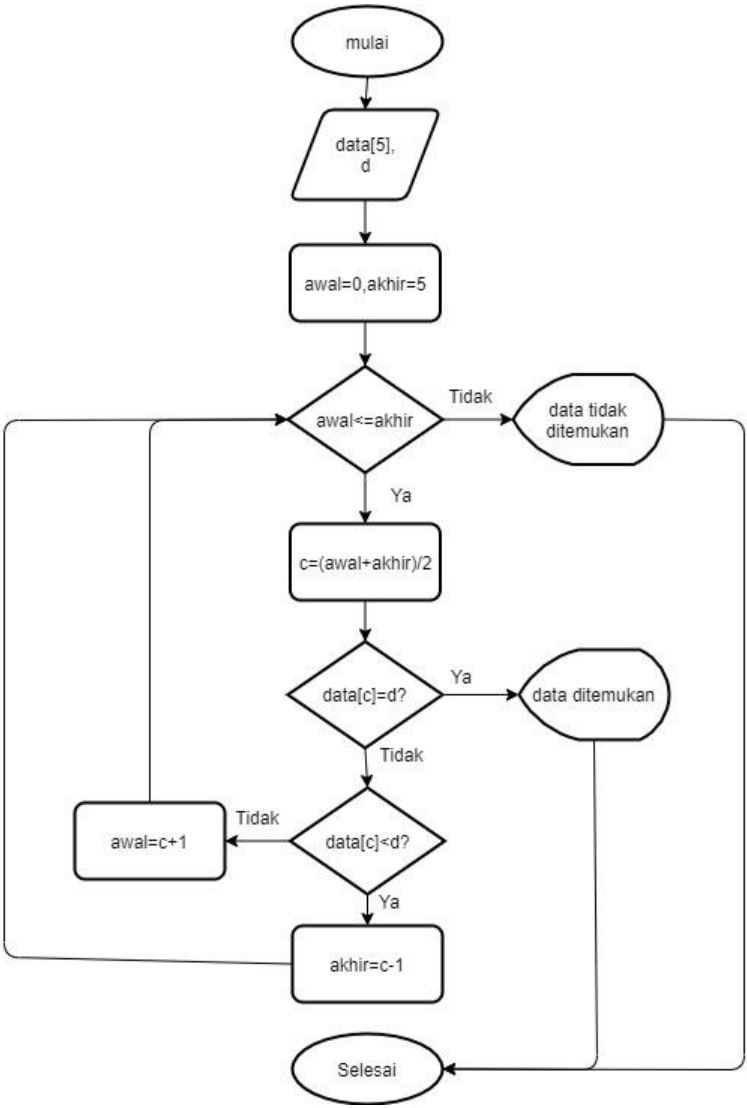
apabila data[c] < d, pencarian berpindah ke bagian kiri dengan cara mengubah nilai c menjadi: akhir = c-1

apabila data[c] > d, pencarian berpindah ke bagian kanan dengan cara mengubah nilai c menjadi awal = c + 1

step 3 :

Kembali ke **step 1** hingga d (data yang dicari) ditemukan atau hingga nilai awal > akhir .

5. Flowchart Algoritma Binary Search data Descending



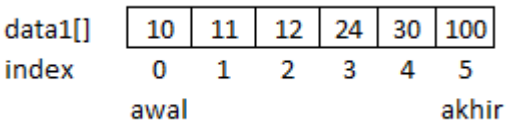
Contoh 1 :

Diketahui int data1[]={10,11,12,24,30,100};

int d=30 ;

Tentukan d berada pada indeks berapa menggunakan metode Pencarian bagidua (binary search)?

Jawab:



Step 1

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal+akhir) div 2

c = (0+5) div 2

c = 2

Step 2

Membandingkan data1[c] dengan data d,

data1[2]<d

11<30 pencarian berpindah ke bagian kanan dengan cara

mengubah nilai c menjadi awal = c + 1

awal = c + 1

awal = 3

Step 1’

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal + akhir) div 2

c = (3 + 5) div 2

c = 4

Step 2’

Membandingkan data1[c] dengan data d,

Program:

```
1  #include<iostream>
2  #include<cstdlib>
3  using namespace std;
4  main()
5  {
6          system("cls");
7          int data1[]={10,11,12,24,30,100};
8          int awal,akhir,c=-1,d=30;//data yang
9  dicari
```

```
10          awal=0,akhir=5;
11          while(awal<=akhir){
12                  c=(awal+akhir)/2;
13                  if(data1[c]==d){
14                  cout<<"data ditemukan pada index ke:"<<c<<endl;
15                  break;
16                  }else if(data1[c]>d)akhir=c-1;
17                  else awal=c+1;
18          }
19          if(data1[c]!=d)cout<<"data tidak ditemukan"<<endl;
}
```

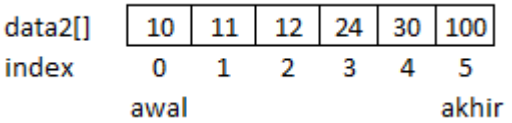
Contoh 2 :

Diketahui int data2[]={10,11,12,24,30,100};

int d=40 ;

Tentukan d berada pada indeks berapa menggunakan metode Pencarian bagidua (binary search)?

Jawab:



Step 1

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal + akhir) div 2

c = (0+5) div 2

c = 2

Step 2

Membandingkan data2[c] dengan data d,

data2[2]<d

11<30 pencarian berpindah ke bagian kanan dengan cara

mengubah nilai c menjadi awal = c + 1

awal = c + 1

awal = 3

Step 1’

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal + akhir) div 2

c = (3 + 5) div 2

c = 4

Step 2’

Membandingkan data2[c] dengan d

data2[4]<d

30<40 pencarian berpindah ke bagian kanan dengan cara

mengubah nilai c menjadi awal= c + 1

awal = 4 + 1

awal = 5

Step 1’’

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal + akhir) div 2

c = (5 + 5) div 2

c = 5

Step 2''

Membandingkan data2[c] dengan d

Data2[5]>d

100>40 pencarian berpindah ke bagian kiri dengan cara

mengubah nilai c menjadi akhir= c - 1

akhir = c - 1

akhir = 4

Pencarian Selesai karena awal=5 dan akhir=4 maka awal<akhir bernilai False looping berakhir, **data tidak ditemukan**.

Program:

```
1  #include<iostream>
2  #include<cstdlib>
3  using namespace std;
4  main()
5  {
```

```
6           system("cls");
7           int data2[]={10,11,12,24,30,100};
8           int awal,akhir,c=-1,d=30;//data yang
9   dicari
10          awal=0,akhir=5;
11          while(awal<=akhir){
12                  c=(awal+akhir)/2;
13                  if(data2[c]==d){
                    cout<<"data ditemukan pada
    index ke:"<<c<<endl;
14                  break;
15                  }else if(data2[c]>d)akhir=c-1;
16                  else awal=c+1;
17          }
18          if(data2[c]!=d)cout<<"data tidak
    ditemukan"<<endl;
19  }
```

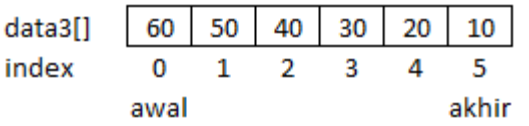
Contoh 3 :

Diketahui int data3[]={60,50,40,30,20,10};

int cari=20 ;

Tentukan cari berada pada indeks berapa menggunakan metode Pencarian bagidua (binary search)?

Jawab:



Step 1:

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal+akhir) div 2

c = 0 + 5 div 2

c = 2

Step 2:

Membandingkan data3[c] dengan data cari,

Data3[2]>cari

40>20 pencarian berpindah ke bagian kanan dengan cara mengubah nilai c menjadi awal = c + 1

awal = c + 1

awal = 3

step 1' :

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal+akhir) div 2

c = 3 + 5 div 2

c = 4

Step 2'

Membandingkan data3[c] dengan data cari,

data3[4]=cari

20=20 pencarian berpindah ke bagian kanan dengan cara

mengubah nilai c menjadi awal = c + 1

awal = c + 1

awal = 3

data[4]=cari

20=20 pencarian selesai data ditemukan pada indeks **c=4**

Program:

```
1  #include<iostream>
2  #include<cstdlib>
3  #define jmldt 6
4  using namespace std;
5  void cari3(int data3[], int jml, int cari, int *dx);
   int main()
6  {
7   int data3[jmldt]={60,50,40,30,20,10},cari,dx,awal;
8   cout<<"Elemen Array :  ";
9  for(awal=0;awal<jmldt;awal++)cout<<data3[awal]<<"
10 ";cout<<endl;
11  cout<<"Masukan data yang akan dicari
    ?:";cin>>cari;
12  cari3(data3,jmldt,cari,&dx);
    if(dx!=-1)cout<<"Data yang dicari berada pada
    indeks : "<<dx<<endl;
13  else cout<<"Data yang dicari tidak ada dalam
14 array"<<endl;
```

```
    }
15  void cari3(int data3[],int jml,int cari, int *dx)
    {
16   bool ketemu = false;
17   int akhir = jml-1,awal = 0,c;
18   while(awal<=akhir && !ketemu)
19   {
20            c=(awal+akhir)/2;
21            if(data3[c]==cari)ketemu=true;
22            else
23            if(data3[c]<cari)akhir=c-1;
24            else awal=c+1;
25    }
26   if(ketemu) *dx=c;
27   else *dx=-1;
28  }
29
30
31
```

Contoh 4 :

Diketahui int data4[]={60,50,40,30,20,10};

int cari=55 ;

Tentukan cari berada pada indeks berapa menggunakan metode Pencarian bagidua (binary search)?

Jawab:

data4[]	60	50	40	30	20	10
index	0	1	2	3	4	5
	awal					akhir

Step 1:

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal+akhir) div 2

c = 0 + 5 div 2

c = 2

Step 2:

Membandingkan data4[c] dengan data cari,

data4[2]<cari

40<55 pencarian berpindah ke bagian kiri dengan cara

mengubah nilai c menjadi akhir = c - 1

akhir = c - 1

akhir = 1

step 1’ :

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal+akhir) div 2

c = 0 + 1 div 2

c = 0

Step 2’

Membandingkan data4[c] dengan data cari,

data4[0]>cari

60>55 pencarian berpindah ke bagian kanan dengan cara

mengubah nilai c menjadi awal = c + 1

awal = 0 + 1

awal = 1

Pencarian Selesai karena awal=1 dan akhir=1 maka awal<akhir bernilai False

looping berakhir, **data tidak ditemukan**.

Program:

```
1  #include<iostream>
2  #include<cstdlib>
3  #define jmldt 6
4  using namespace std;
5  void cari4(int data4[], int jml, int cari, int *dx);
   int main()
6  {
7   int data4[jmldt]={60,50,40,30,20,10},cari,dx,awal;
8   cout<<"Elemen Array :  ";
9  for(awal=0;awal<jmldt;awal++)cout<<data4[awal]<<"
10 ";cout<<endl;
11  cout<<"Masukan data yang akan dicari
    ?:";cin>>cari;
    cari4(data4,jmldt,cari,&dx);
12  if(dx!=-1)cout<<"Data yang dicari berada pada
    indeks : "<<dx<<endl;
13
```

```
14  else cout<<"Data yang dicari tidak ada dalam
    array"<<endl;
15  }
    void cari4(int data4[],int jml,int cari, int *dx)
16  {
17   bool ketemu = false;
18   int akhir = jml-1,awal = 0,c;
19   while(awal<=akhir && !ketemu)
20   {
21            c=(awal+akhir)/2;
22            if(data4[c]==cari)ketemu=true;
23            else
24            if(data4[c]<cari)akhir=c-1;
25            else awal=c+1;
26    }
27   if(ketemu) *dx=c;
28   else *dx=-1;
29  }
30
31
```

Contoh 5 :

Diketahui int data5[]={100,90,80,70,60,50,40,30,20,10};

int cari=10 ;

Tentukan cari berada pada indeks berapa menggunakan metode Pencarian bagidua (binary search)?

Jawab:

data5[]	100	90	80	70	60	50	40	30	20	10
index	0	1	2	3	4	5	6	7	8	9
	awal									akhir

Step 1:

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal+akhir) div 2

c = 0 + 9 div 2

c = 4

Step 2:

Membandingkan data4[c] dengan data cari,

data5[4]>cari

60>10

pencarian berpindah ke bagian kanan dengan cara

mengubah nilai c menjadi awal = c + 1

awal = c + 1

awal = 5

step 1' :

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal+akhir) div 2

c = 5 + 9 div 2

c = 7

Step 2'

Membandingkan data5[c] dengan data cari,

data5[7]>cari

30>10 pencarian berpindah ke bagian kanan dengan cara

mengubah nilai c menjadi awal = c + 1

awal = 7 + 1

awal = 8

step 1'' :

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal+akhir) div 2

c = 8 + 9 div 2

c = 8

Step 2''

Membandingkan data5[c] dengan data cari,

data5[8]>cari

20>10 pencarian berpindah ke bagian kanan dengan cara

mengubah nilai c menjadi awal = c + 1

awal = 8 + 1

awal = 9

step 1''' :

Tentukan Nilai tengah, nilai tengah didapatkan dengan cara sebagi berikut:

Misal index awal =awal dan index akhir =akhir maka,

c = (awal+akhir) div 2

c = 9 + 9 div 2

c = 9

Step 2'''

Membandingkan data5[c] dengan data cari,

data5[9]=cari

10=10 Pencarian selesai **data ditemukan** pada index ke 9

Program:

```
1  #include<iostream>
2  #include<cstdlib>
3  #define jmldt 10
4  using namespace std;
5  void cari5(int data5[], int jml, int cari, int *dx);
6  int main()
7  {
8  int
9  data5[jmldt]={100,90,80,70,60,50,40,30,20,10},cari,
   dx,awal;
10  cout<<"Elemen Array :  ";
11
```

```
12  for(awal=0;awal<jmldt;awal++)cout<<data5[awal]<<" ";cout<<endl;
13  cout<<"Masukan data yang akan dicari ?:";cin>>cari;
14  cari5(data5,jmldt,cari,&dx);
15  if(dx!=-1)cout<<"Data yang dicari berada pada indeks : "<<dx<<endl;
16  else cout<<"Data yang dicari tidak ada dalam array"<<endl;
17  }
18  void cari5(int data5[],int jml,int cari, int *dx)
19  {
20  bool ketemu = false;
21  int akhir = jml-1,awal = 0,c;
22  while(awal<=akhir && !ketemu)
23  {
24          c=(awal+akhir)/2;
25          if(data5[c]==cari)ketemu=true;
26          else
27          if(data5[c]<cari)akhir=c-1;
28          else awal=c+1;
29   }
```

```
30    if(ketemu) *dx=c;
31    else *dx=-1;
      }
```

C. LATIHAN

Soal 1

Buatlah program pencarian bagidua dengan data urut *descending*

Soal 2

Diketahui data[]={100,30,23,10,9,7} dan data yang dicari adalah 9 Tuliskan langkah paencarian data tersebut menggunakan metode pencarian bagidua(*binary search*).

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan Java*. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

BAB VIII

PENGURUTAN GELEMBUNG (BUBBLE SORT)

A. CAPAIAN PEMBELAJARAN

Pada bab ini akan dijelaskan mengenai definisi pengurutan, Ascending-Descending, Metode Pengurutan Gelembung, Algoritma pengurutan Gelembung, Contoh Program C++ Pengurutan, Manfaat pengurutan sisip.

B. MATERI

1. Pengertian Pengurutan (Sorting)

Pengurutan atau sorting adalah suatu proses untuk menyusun kembali himpunan obyek dengan aturan tertentu.

Ada dua jenis urutan yang dapat digunakan dalam kegiatan pengurutan yaitu :

a. Ascending.
   Ascending adalah pengurutan data yang dimulai dari nilai yang paling kecil ke nilai yang paling besar.
b. Descending.
   Descending adalah pengurutan data yang dimulai dari nilai yang paling nesar ke nilai yang paling kecil.

Contoh : data array 10, 5, 15 dan 20 jika diurutkan secara Ascending menjadi 5, 10, 15, 20, Sedangkan jika diurutkan secara descending menjadi 20, 15, 10, 5. Pada data yang bertipe char, nilai data dikatakan lebih kecil atau lebih besar dari yang lain didasarkan pada urutan relatif (collating sequence) seperti dinyatakan dalam tabel ASCII.

Keuntungan dari data yang sudah dalam keadaan terurutkan antara lain :

a. Memudahkan dalam pencarian data
b. Memudahkan dalam melakukan pengecekan
c. Mempercepat dalam proses pencarian data jika harus dilakukan berulang kali.

2. Metode Pengurutan Gelembung (Bubble Sort)

Bubble sort atau metode gelembung adalah algoritma pengurutan sederhana. Algoritma pengurutan ini adalah algoritma berbasis perbandingan di mana setiap pasangan elemen yang berdekatan dibandingkan dan elemen-elemen tersebut ditukar jika tidak berurutan. Algoritme ini tidak cocok untuk kumpulan data besar karena kompleksitas kasus rata-rata dan terburuknya adalah (n^2) di mana **n** adalah jumlah item. Metode ini bisa dikatakan adalah metode yang paling mudah dipahami namun juga merupakan metode pengurutan yang paling tidak efisien jika terdapat banyak data tang harus diurutkan.

Pengurutan gelembung membutuhkan waktu (n^2) jadi kami membuatnya singkat dan tepat. Bubble sort dimulai dengan dua elemen pertama, membandingkannya untuk memeriksa mana yang lebih besar. Dalam hal ini jika nilai pertama lebih kecil dari nilai kedua, maka tidak diperlukan adanya pertukaran data, namun jika nilai pertama lebih besar dari pada nilai kedua, maka harus dilakukan proses pertukaran data agar masing2 data tepat pada posisinya. Proses perbandingan dan pertukaran akan tetap dilakukan sampai pada akhir array.

Dalam hal ini proses perbandingan dan pertukaran akan terus dilakukan sampai dengan batas iterasi telah selesai dilaksanakan, dalam setiap iterasi terdapat beberapa kali proses perbanidingan dan pertukaran.

3. Algoritma Metode Pengurutan Gelembung (Bubble Sort)

Prose algoritma bubble sort dapat dituliskan sebagai berikut :

1 x ← 0

2 selama (x < N-1), maka kerjakan baris 2 sampai dengan 5

3 y← N - 1

4 Selama (y >= i), maka kerjakan baris 3 sampai dengan 5

5 Jika (Dta[y-1] > Dta[y]) maka tukar Dta[y-1] dengan Dta[y]

6 y ← y – 1

7 x ← x + 1

Untuk memudahkan menjelaskan langkah-langkah dalam proses algoritma gelembung, maka dapat dilihat pada contoh dibawah Proses pengurutan dapat dijelaskan sebagai berikut:

Array Awal

20	**35**	**10**	**25**	**15**	**5**

a. Pada saat x=1 :

Iterasi		Array [0]	Array [1]	Array [2]	Array [3]	Array [4]	Array [5]
Awal		20	35	10	25	15	5
x=1	y=5	20	35	10	25	**5**	**15**
	y=4	20	35	10	**5**	**25**	15
	y=3	20	35	**5**	**10**	25	15
	y=2	20	**5**	**35**	10	25	15
	y=1	**5**	**20**	35	10	25	15

- Proses perbanidngan dan pertukaran nilai y akan diulang dari 5 sampai dengan 1. Dalam proses pengulangan pertama data pada Array[5] dibandingkan dengan data Array[4], karena 5 < 15 maka Array [5] dan Array [4] dilakukan proses penukaran.
- Dalam proses pengulangan yang kedua, Array[4] dibandingkan Array[3], karena data 5 < 25 maka Array[4] akan ditukar dengan Array[3].
- Proses ini dilakukan sedemikian rupa sampai dengan y=1.

b. Pada saat x=2 :

Iterasi		Array [0]	Array [1]	Array [2]	Array [3]	Array [4]	Array [5]
Hasil x=1		5	20	35	10	25	15
x=2	y=5	5	20	35	10	**15**	**25**
	y=4	5	20	35	10	15	25
	y=3	5	20	**10**	**35**	15	25
	y=2	5	**10**	**20**	35	15	25

- Proses perbanidngan dan pertukaran nilai y akan diulang dari 5 sampai dengan 2. Dalam proses pengulangan pertama data pada Array[5]

dibandingkan Array [4], karena 15 < 25 maka data pada Array[5] akan ditukar dengan data Array [4].

- Pada pengulangan kedua Data[4] dibandingkan Data[3], karena 15 > 10 maka dilanjutkan tanpa ada penukaran data.
- Pada pengulangan ketiga Data[3] dibandingkan Data[2], karena 10 < 35 maka Data[3] ditukar dengan Data[2]
- Pada pengulangan keempat Data[2] dibandingkan Data[1], karena 10 < 20 maka Data[2] ditukar dengan Data[1]
- Dilanjutkan iterasi ke-3

c. Pada saat x=3,

Iterasi		Array [0]	Array [1]	Array [2]	Array [3]	Array [4]	Array [5]
Hasil x=2		5	10	20	35	15	25
x=3	y=5	5	10	20	35	15	25
	y=4	5	10	20	**15**	**35**	25
	y=3	5	10	**15**	**20**	35	25

- nilai j diulang dari 5 sampai dengan 3. Pada pengulangan pertama Data[5] dibandingkan Data[4], karena 25 > 15 maka proses dilanjutkan.

- Pada pengulangan kedua Data[4] dibandingkan Data[3], karena 15 < 35 maka Data[4] ditukar dengan Data[3].
- Pada pengulangan ketiga Data[3] dibandingkan Data[2], karena 15 < 20 maka Data[3] ditukar dengan Data[2].
- Dilanjutkan iterasi ke-4

d. Pada iterasi ke-4 :

Iterasi		Array [0]	Array [1]	Array [2]	Array [3]	Array [4]	Array [5]
Hasil x=3		5	10	15	20	35	25
x=4	y=5	5	10	15	20	**25**	**35**
	y=4	5	10	15	20	25	35

- nilai j diulang dari 5 sampai dengan 4. Pada pengulangan pertama Data[5] dibandingkan Data[4], karena 25 < 35 maka Data[5] ditukar dengan Data[4].
- Pada pengulangan kedua Data[3] dibandingkan Data[2], karena 25 > 20 maka proses dilanjutkan tanpa penukaran data.
- Dilanjutkan iterasi ke-5

e. Pada iterasi ke-5 :

Iterasi		Array [0]	Array [1]	Array [2]	Array [3]	Array [4]	Array [5]
Hasil x=4		5	10	15	20	25	35
x=5	x=5	5	10	15	20	25	35

- nilai j diulang dari 5 sampai dengan 4. Pada pengulangan pertama Data[5] dibandingkan Data[4], karena 35 > 25 maka proses dilanjutkan tanpa penukaran data.

```
procedure Insertion(input/output L : LarikInt, input n : integer)
DEKLARASI
  I : integer {Pencacah tahap}
  j : integer {Pencacah untuk penelusuran larik}
  x : integer {Peubah bantu}
  tukar : boolean  {Untuk menyatakan posisi penukaran ditemukan}
DESKRIPSI
    void BubbleSort()
    {
       int i, j;
       for(i=1; i<Max-1; i++)
              for(j=Max-1; j>=i; j--)
                     if(Data[j-1] > Data[j])
                        Tukar(&Data[j-1], &Data[j]);
    }
```

Dari algoritma dan prosedur diatas, jumlah perbandingan (=C) metode gelembung selalu konstan yaitu :

C = (N2 – N) / 2

Jumlah penukaran data (=M) pada metode gelembung tergantung keadaan data. Jumlah penukaran minimum terjadi bila data sudah dalam keadaan urut, sebaliknya jumlah penukaran maksimum terjadi bila data dalam keadaan urut terbalik. Adapaun rumus dari jumlah penukaran pada metode gelembung adalah sebagai berikut :

$M_{min} = 0$

$M_{rata\text{-}rata} = 3(N^2 - N) / 4$

$M_{max} = 3(N^2 - N) / 2$

4. Program C++ untuk pengurutan Seleksi

a. Contoh Program Pengurutan Gelembung Secara Menaik

```
/*Mengurutkan data dengan metode bubble*/
#include<iostream.h>
#include<conio.h>
#include<iomanip.h
int main ( )
{
int Nilai [ 20 ];
int i, j, k ,N ;
```

```
int temp ;
bool tukar ;
cout<<"Masukan Banyak Bilangan :";
cin>>N;
for (i=0; i<N; i++)
{
    cout<<"Elemen ke-"<<i<<" : ";
    cin>>Nilai [ i ];
}
 //Proses Cetak Sebelum diurutkan

cout<<"\nData sebelumnya diurut :";
for (i=0; i<N ; i++)
 cout<<setw ( 3 )<<Nilai [ i ];
 //Proses Pengurutan
i=0;
 tukar = true;
 while ((i<=N-2) && (tukar))
 {
    tukar=false;
    for (j=N-1; j>=i+1; j--)
```

```
    {
        if( Nilai [ j ] < Nilai [ j-1 ] )
        {
            temp = Nilai [ j ];
          Nilai [ j ] = Nilai [ j1];
          Nilai [j-1 ]=temp;
          tukar= true;

          cout<<"\nUntuk j = "<<j<<" : ";
          for (k=0; k<N; k++)

            cout<<setw(3)<<Nilai [ k ] ;
           }
     }
     i++ ;
}

//Proses Cetak setelah diurutkan
cout<<"\nData setelah di urut : ";
for (i=0; i<N; i++ )
```

```
cout<<setw ( 3 )<<Nilai [ i ] ;
getch () ;
}
```

b. Contoh Program Pengurutan Gelembuk Secara Menurun

```
/*Mengurutkan data dengan metode gelembung*/

#include<iostream.h>
#include<conio.h>
#include<iomanip.h
int main ()
{

int Nilai [ 20 ];
int i, j, k ,N ;
int temp ;
bool tukar ;

cout<<"Masukan Banyak Bilangan : ";
cin>>N;
```

```
for (i=0; i<N; i++)
 {

        cout<<"Elemen ke-"<<i<<" : ";
        cin>>Nilai [ i ];

}

//Proses Cetak Sebelum diurutkan
cout<<"\nData sebelumnya diurut :";
for (i=0; i<N ; i++)

cout<<setw ( 3 )<<Nilai [ i ];

//Proses Pengurutan
i=0;
tukar = true;
while ((i<=N-2) && (tukar))
{
```

```
tukar=false;
for (j=N-1; j>=i+1; j--)
{
    if( Nilai [ j ] < Nilai [ j-1 ] )
    {

        temp = Nilai [ j ];
        Nilai [ j ] = Nilai [ j1];
        Nilai[j-1]=temp;
        tukar=true;

        cout<<endl;
        for(k=0; k<N; k++)
            cout<<setw (3)<<Nilai [ k ] ;
     }
}
i++ ;
}

//Proses Cetak setelah diurutkan
cout<<"\nData setelah di urut : ";
```

```
for (i=0; i<N; i++ )
cout<<setw ( 3 )<<Nilai [ i ] ;
getch () ;
}
```

C. LATIHAN

Buatlah program C++ dan proses iterasi untuk mengurutkan data berikut secara menaik dengan menggunakan metode Bubble Sort:

2	2	1	3	6	5	3	3	4	1	4
7	5	2	2	0	2	5	7	7	7	5

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan Java*. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung,

Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

BAB IX

PENGURUTAN SELEKSI

A. CAPAIAN PEMBELAJARAN

Pada bab ini akan dijelaskan mengenai pengurutan data yang berada pada suatu tempat penyimpanan, Metode Pengurutan Seleksi, Algoritma pengurutan seleksi, Program C++, Pengurutan, Manfaat pengurutan sisip.

B. MATERI

1. Metode Pengurutan Seleksi

Pengurutan data (sorting) didefinisikan sebagai suatu proses untuk menyusun kembali humpunan obyek menggunakan aturan tertentu. Mnurut Microsoft Book-shelf, definisi algoritma pengurutan adalah algoritma untuk meletakkan kumpulan elemen data ke dalam urutan tertentu berdasarkan satu atau beberapa kunci dalam tiap-tiap elemen.

Metode seleksi melakukan pengurutan dengan cara mencari data yang terkecil kemudian menukarkannya dengan data yang digunakan sebagai acuan atau sering dinamakan pivot.

Seleksi sort adalah algoritma pengurutan sederhana. Algoritma pengurutan ini adalah algoritma berbasis perbandingan di tempat di mana daftar dibagi menjadi dua bagian, bagian yang diurutkan di ujung kiri dan bagian yang tidak disortir di ujung kanan. Awalnya, bagian yang diurutkan kosong dan bagian yang tidak disortir adalah seluruh daftar.

Elemen terkecil dipilih dari array yang tidak disortir dan ditukar dengan elemen paling kiri, dan elemen tersebut menjadi bagian dari array yang diurutkan. Proses ini terus memindahkan batas array yang tidak disortir oleh satu elemen ke kanan.

Algoritma ini tidak cocok untuk kumpulan data besar karena kompleksitas rata-rata dan kasus terburuknya adalah (n2), di mana n adalah jumlah item.

2. Algoritma Selection Sort untuk pengurutan menaik

Proses pengurutan dengan metode seleksi dapat dijelaskan sebagai berikut :

Langkah pertama dicari data terkecil dari data pertama sampai data terakhir. Kemudian data terkecil ditukar dengan data pertama. Dengan demikian, data pertama sekarang mempunyai nilai paling kecil dibanding data yang lain. Langkah kedua, data terkecil kita cari mulai dari data kedua sampai terakhir. Data terkecil yang kita peroleh ditukar dengan data kedua dan demikian

seterusnya sampai semua elemen dalam keadaan terurutkan.

Algoritma seleksi dapat dituliskan sebagai berikut :

1. i ← 0
2. selama (i < N-1) kerjakan baris 3 sampai dengan 9
3. k ← i
4. j ← i + 1
5. Selama (j < N) kerjakan baris 6 dan 7
6. Jika (Data[k] > Data[j]) maka k ← j
7. j ← j + 1
8. Tukar Data[i] dengan Data[k]
9. i ← i + 1

Untuk lebih memperjelas langkah-langkah algoritma seleksi dapat dilihat pada tabel berikut. Proses pengurutan pada tabel dapat dijelaskan sebagai berikut:

- Pada saat i=0, data terkecil antara data ke-1 s/d ke-9 adalah data ke-4, yaitu 3, maka data ke-0 yaitu 12 ditukar dengan data ke-4 yaitu 3.

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=0	**12**	35	9	11	**3**	17	23	15	31	20

- Pada saat i=1, data terkecil antara data ke-2 s/d ke-9 adalah data ke-2, yaitu 9, maka data ke-1 yaitu 35 ditukar dengan data ke-2 yaitu 9.

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=0	**12**	35	9	11	**3**	17	23	15	31	20
i=1	3	**35**	**9**	11	12	17	23	15	31	20

- Pada saat i=2, data terkecil antara data ke-3 s/d ke-9 adalah data ke-3, yaitu 11, maka data ke-2 yaitu 35 ditukar dengan data ke-3 yaitu 11.

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=0	**12**	35	9	11	**3**	17	23	15	31	20
i=1	3	**35**	**9**	11	12	17	23	15	31	20
i=2	3	9	**35**	**11**	12	17	23	15	31	20

- Pada saat i=3, data terkecil antara data ke-4 s/d ke-9 adalah data ke-4, yaitu 12, maka data ke-3 yaitu 35 ditukar dengan data ke-4 yaitu 12.

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=0	**12**	35	9	11	**3**	17	23	15	31	20
i=1	3	**35**	**9**	11	12	17	23	15	31	20
i=2	3	9	**35**	**11**	12	17	23	15	31	20
i=3	3	9	11	**35**	**12**	17	23	15	31	20

- Pada saat i=4, data terkecil antara data ke-5 s/d ke-9 adalah data ke-7, yaitu 15, maka data ke-4 yaitu 35 ditukar dengan data ke-7 yaitu 15.

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=0	**12**	35	9	11	**3**	17	23	15	31	20
i=1	3	**35**	**9**	11	12	17	23	15	31	20
i=2	3	9	**35**	**11**	12	17	23	15	31	20
i=3	3	9	11	**35**	**12**	17	23	15	31	20
i=4	3	9	11	12	**35**	17	23	**15**	31	20

- Pada saat i=5, data terkecil antara data ke-6 s/d ke-9 adalah data ke-5, yaitu 17, maka data ke-5 tetap pada posisinya.

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=0	**12**	35	9	11	**3**	17	23	15	31	20
i=1	3	**35**	**9**	11	12	17	23	15	31	20
i=2	3	9	**35**	**11**	12	17	23	15	31	20
i=3	3	9	11	**35**	**12**	17	23	15	31	20
i=4	3	9	11	12	**35**	17	23	**15**	31	20
i=5	3	9	11	12	15	**17**	23	35	31	20

- Pada saat i=6, data terkecil antara data ke-7 s/d ke-9 adalah data ke-9, yaitu 20, maka data ke-6 yaitu 23 ditukar dengan data ke-9 yaitu 20.

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=0	**12**	35	9	11	**3**	17	23	15	31	20
i=1	3	**35**	**9**	11	12	17	23	15	31	20
i=2	3	9	**35**	**11**	12	17	23	15	31	20
i=3	3	9	11	**35**	**12**	17	23	15	31	20
i=4	3	9	11	12	**35**	17	23	**15**	31	20
i=5	3	9	11	12	15	**17**	23	35	31	20
i=6	3	9	11	12	15	17	**23**	35	31	**20**

- Pada saat i=7, data terkecil antara data ke-8 s/d ke-9 adalah data ke-9, yaitu 23, maka data ke-7 yaitu 35 ditukar dengan data ke-9 yaitu 23.

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=0	**12**	35	9	11	**3**	17	23	15	31	20
i=1	3	**35**	**9**	11	12	17	23	15	31	20
i=2	3	9	**35**	**11**	12	17	23	15	31	20
i=3	3	9	11	**35**	**12**	17	23	15	31	20
i=4	3	9	11	12	**35**	17	23	**15**	31	20
i=5	3	9	11	12	15	**17**	23	35	31	20
i=6	3	9	11	12	15	17	**23**	35	31	**20**
i=7	3	9	11	12	15	17	20	**35**	31	**23**

- Pada saat i=8, data terkecil antara data ke-8 s/d ke-9 adalah data ke-8, yaitu 31, maka data ke-8 yaitu 31 tetap pada posisinya.

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=0	**12**	35	9	11	**3**	17	23	15	31	20
i=1	3	**35**	**9**	11	12	17	23	15	31	20
i=2	3	9	**35**	**11**	12	17	23	15	31	20
i=3	3	9	11	**35**	**12**	17	23	15	31	20
i=4	3	9	11	12	**35**	17	23	**15**	31	20
i=5	3	9	11	12	15	**17**	23	35	31	20
i=6	3	9	11	12	15	17	**23**	35	31	**20**
i=7	3	9	11	12	15	17	20	**35**	31	**23**
i=8	3	9	11	12	15	17	20	23	**31**	35

- Adapun hasil akhir dari proses sortingnya adalah sebagai berikut :

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=0	**12**	35	9	11	**3**	17	23	15	31	20
i=1	3	**35**	**9**	11	12	17	23	15	31	20
i=2	3	9	**35**	**11**	12	17	23	15	31	20
i=3	3	9	11	**35**	**12**	17	23	15	31	20
i=4	3	9	11	12	**35**	17	23	**15**	31	20
i=5	3	9	11	12	15	**17**	23	35	31	20
i=6	3	9	11	12	15	17	**23**	35	31	**20**
i=7	3	9	11	12	15	17	20	**35**	31	**23**
i=8	3	9	11	12	15	17	20	23	**31**	35
Akhir	3	9	11	12	15	17	20	23	31	35

```
procedure Insertion(input/output L : LarikInt, input n : integer)
DEKLARASI
  i : integer {Pencacah tahap}
  j : integer {Pencacah untuk penelusuran larik}
  x : integer {Peubah bantu}
  ketemu : boolean {Untuk menyatakan posisi penyisipan ditemukan}
DESKRIPSI
    void SelectionSort()
    {
     int i, j, k;
     for(i=0; i<Max-1;i++){
      k = i;
    for (j=i+1; j<Max; j++)
      if(Data[k] > Data[j])
       k = j;
      Tukar(&Data[i], &Data[k]);
     }
    }
```

Dari algoritma dan prosedur diatas, jumlah perbandingan (=C) metode seleksi adalah

$C = N(N - 1) / 2$

Jumlah penukaran (=M) pada metode seleksi tergantung keadaan datanya. Penukaran minimum terjadi bila data sudah dalam keadaan urut, sebaliknya jumlah penukaran maksimum terjadi bila data dalam keadaan urut terbalik. Jumlah penukaran minimum dan maksimum dapat dirumuskan sebagai berikut :

$Mmin = 3(N - 1)$

$Mmax = [N^2 / 4] + 3(N - 1)$

3. Program C++ untuk pengurutan Seleksi

a. Contoh Program Seleksi Minimum Secara Menaik

```
/*Mengurutkan data dengan metode Seleksi*/
#include<iostream.h>
#include<conio.h>
#include<iomanip.h
int main ( )
{
  int Nilai [ 20 ];
  int i, j, N, l;
  int temp, Imin;
  cout<<"Masukan Banyak bilangan : ";
```

```
  cin>>N;
  for (i=0; i<N; i++ )
  {
     cout<<"Elemen ke-"<<i<<" : ";
     cin>>Nilai [ i ] ;
  }
//Proses Cetak Sebelum Diurutkan
cout<<"\nData sebelum diurut :";
for(i=0; i<N; i++)
cout<<setw (3)<<Nilai [ i ];
//Proses pengurutan
for (i=0;i<=N-2; i++)
{
Imin = i; for(j=i+1;
j<N; j++)
{
if(Nilai [j] < Nilai [Imin])
Imin = j;
}
temp = Nilai [i]; Nilai
[i] = Nilai [Imin];
```

```
Nilai [Imin] = temp;
cout<<endl;
for(l=0; l<N; l++)
cout<<setw(3)<<Nilai [l];
}
cout<<"\nData Setelah di urut ; ";
for(i=0; i<N; i++)
cout<<setw(3)<<Nilai [i];
getch ( );
}
```

b. Contoh Program Seleksi Maximum Secara Menaik

```
/*Mengurutkan data dengan metode Seleksi Maximum*/
#include<iostream.h>
#include<conio.h>
#include<iomanip.h
int main ()
{
int Nilai [ 20 ];
int i, j , N, l ;
```

```
int temp, U, Imaks;
cout<<"Masukan Banyaknya Bilangan:";
cin>>N;


for(i=0; i<N; i++)
{
    cout<<"Elemen ke-"<<i<<" : ";
    cin>>Nilai [ i ];
}
 //Proses Cetak sebelum diurutkan
cout<<"\nData sebelum diurut :";
for(i=0; i<N; i++)
cout<<setw ( 3 )<<Nilai [ i ];
//Peroses Pengurutan U=N-1;


for (i=0 ; i<=N-2; i++)
{
   Imaks =0; for(j=1;
   j<=U; j++)
   {
      if( Nilai [ j ] > Nilai [ Imaks] )
```

```
        Imaks = j;
    }
  temp =Nilai [ U ];
  Nilai [ U ] = Nilai [ Imaks];
  Nilai [ Imaks ]= temp; U-;
  cout<<endl;
  for(l=0; l<N; l++)
  cout<<setw ( 3 )<<Nilai [ 1 ];
 }

cout<<"\nData Setelah di urut : ";
for(i=0; i<N; i++)
cout<<setw (3 )<<Nilai [ i ] ;
getch ( ) ;
}
```

C. LATIHAN

1. Buatlah program C++ untuk mengurutkan data berikut secara menurun dengan menggunakan metode Seleksi Minimum:

A :	2 7	2 5	1 2	3 2	6 0	5 2	3 5	3 7	4 7	1 7	4 5	1 0	5 1 5

2. Buatlah program C++ untuk mengurutkan data berikut secara menaik dengan menggunakan metode Maksimum:

A :	12	15	7	10	25	2	17	25	5	20

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan Java*. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

BAB X

INSERT SORT

A. CAPAIAN PEMBELAJARAN

Mahasiswa dapat mengerti dan dapat menggunakan insert sort dalam pemrograman.

1. 10.1 Pengertian Insert Sort
2. 10.2 Perhitungan Insert Sort
3. 10.3 Program C++

B. MATERI

1. Pengertian Insert Sort

Insert sort merupakan pengurutan sisipan atau menyisipkan antara beberapa elemen array yang tepat sehingga elemen array tersebut terurut ascending ataupun descending.

Insert sort ialah algoritma sederhana untuk pengurutan sekumpulan data dengan membandingkan semua data mulai dari data ke 2 dengan data ke 1 dan mulai dari datanya terus bertambah sampai beberapa jumlah sekumpulan data dan dibandingkan datanya

sampai data ke i (sampai elemen array terakhir) mengurutkan secara ascending ataupu descending.

Buatlah pengurutan dari elemen array 0 sampai ke 7 dengan nilai array tersebut seperti dibawah ini:

0	1	2	3	4	5	6	7
36	2	5	31	40	4	7	3

2. Perhitungan Insert Sort

Cara Pengurutan insert sort dengan ascending:

step 1	36	2	5	31	40	4	7	3	X=a[0]=2
	36	2	5	31	40	4	7	3	Satu kali langkah pertukaran
	36	2	5	31	40	4	7	3	
	2	36	5	31	40	4	7	3	

Step 1 a[0]=2 X yang akan dicari pada elemen array a[0] dengan a[0]=2, move

(geser) masing-masing 1 langkah ke kanan. Dari a[0]=36 pindah ke elemen ke a[1] sedangkan a[1]=2 pindah ke elemen array a[0]. Karena 2 lebih kecil dari 36.

step 2	2	36	5	31	40	4	7	3	X=a[1]=3
	2	36	5	31	40	4	3	7	5 kali langkah pertukaran
	2	36	5	31	40	3	4	7	
	2	36	5	31	3	40	4	7	
	2	36	5	3	31	40	4	7	
	2	36	3	5	31	40	4	7	
	0	1	2	3	4	5	6	7	
	2	3	36	5	31	40	4	7	

Step 2 a[1]=3 X yang akan dicari pada elemen array a[1] dengan a[1]=3, move (geser) masing-masing 1 langkah ke kanan:

1. Dari a[7]=3 pindah ke elemen ke a[6] sedangkan a[6]=7 pindah ke elemen array a[7]. Karena 3 lebih kecil dari 7. Jadi a[6]=3.
2. Dari a[6]=3 pindah ke elemen ke a[5] sedangkan a[5]=4 pindah ke elemen array a[6]. Karena 3 lebih kecil dari 4. Jadi a[5]=3.
3. Dari a[5]=3 pindah ke elemen ke a[4] sedangkan a[4]=40 pindah ke elemen array a[5]. Karena 3 lebih kecil dari 40. Jadi a[4]=3.
4. Dari a[4]=3 pindah ke elemen ke a[3] sedangkan a[3]=31 pindah ke elemen array a[4]. Karena 3 lebih kecil dari 31. Jadi a[3]=3.
5. Dari a[3]=3 pindah ke elemen ke a[2] sedangkan a[3]=5 pindah ke elemen array a[3]. Karena 3 lebih kecil dari 5. Jadi a[2]=3.
6. Dari a[2]=3 pindah ke elemen ke a[1] sedangkan a[1]=36 pindah ke elemen

array a[2]. Karena 3 lebih kecil dari 36. Jadi a[1]=3.

step 3	2	3	36	5	31	40	4	7	X=a[2]=4
	2	3	36	5	31	4	40	7	3 kali langkah pertukaran
	2	3	36	5	4	31	40	7	
	2	3	36	4	5	31	40	7	
	0	1	2	3	4	5	6	7	
	2	3	4	36	5	31	40	7	

Step 3 a[2]=4 X yang akan dicari pada elemen array a[2] dengan a[2]=4, move (geser) masing-masing 1 langkah ke kanan:

1. Dari a[6]=4 pindah ke elemen ke a[5] sedangkan a[5]=40 pindah ke elemen array a[6]. Karena 4 lebih kecil dari 40. Jadi a[5]=4.
2. Dari a[5]=4 pindah ke elemen ke a[4] sedangkan a[4]=31 pindah ke elemen

array a[5]. Karena 4 lebih kecil dari 31. Jadi a[4]=4.

3. Dari a[4]=4 pindah ke elemen ke a[3] sedangkan a[3]=5 pindah ke elemen array a[4]. Karena 4 lebih kecil dari 5. Jadi a[3]=4.
4. Dari a[3]=4 pindah ke elemen ke a[2] sedangkan a[2]=36 pindah ke elemen array a[3]. Karena 4 lebih kecil dari 36. Jadi a[2]=4.

step 4	2	3	4	36	5	31	40	7	X=a[3]=5
	0	1	2	3	4	5	6	7	1 kali langkah pertukaran
	2	3	4	5	36	31	40	7	

Step 4 a[3]=5 X yang akan dicari pada elemen array a[3] dengan a[3]=5, move (geser) masing-masing 1 langkah ke kanan. Dari a[4]=5 pindah ke elemen ke a[3] sedangkan a[3]=36 pindah ke elemen array a[4]. Karena 5 lebih kecil dari 36. Jadi a[3]=5.

step 5	2	3	4	5	36	31	40	7	X=a[4]=7
	2	3	4	5	36	31	7	40	
									2 kali langkah pertukaran
	2	3	4	5	36	7	31	40	
	2	3	4	5	7	36	31	40	

Step 5 a[4]=7 X yang akan dicari pada elemen array a[4] dengan a[4]=7, move (geser) masing-masing 1 langkah ke kanan:

1. Dari a[7]=7 pindah ke elemen ke a[6] sedangkan a[6]=40 pindah ke elemen array a[7]. Karena 7 lebih kecil dari 40. Jadi a[6]=7.
2. Dari a[6]=7 pindah ke elemen ke a[5] sedangkan a[5]=31 pindah ke elemen array a[6]. Karena 7 lebih kecil dari 31. Jadi a[5]=7.
3. Dari a[5]=7 pindah ke elemen ke a[4] sedangkan a[4]=36 pindah ke elemen array a[5]. Karena 7 lebih kecil dari 36. Jadi a[4]=7.

step 6	2	3	4	5	7	36	31	40	X=a[5]=31
									1 kali langkah pertukaran
	2	3	4	5	7	31	36	40	

Step 6 a[5]=31 X yang akan dicari pada elemen array a[5] dengan a[5]=31, move (geser) masing-masing 1 langkah ke kanan. Dari a[6]=31 pindah ke elemen ke a[5] sedangkan a[5]=36 pindah ke elemen array a[6]. Karena 31 lebih kecil dari 36. Jadi a[5]=31.

step 7	2	3	4	5	7	31	36	40	X=a[6]=40
									Tidak melakukan penukaran karena 36 lebih kecil dari 40
	2	3	4	5	7	31	36	40	

Step 7 a[6]=40 X yang akan dicari pada elemen array a[6] dengan a[6]=40, jadi tidak ada perpindahan nilai array. Tetap di a[6] karena nilai a[6]=40.

Hasil Pengurutan dari Insert Sort :

2	3	4	5	7	31	36	40

3. Program C++

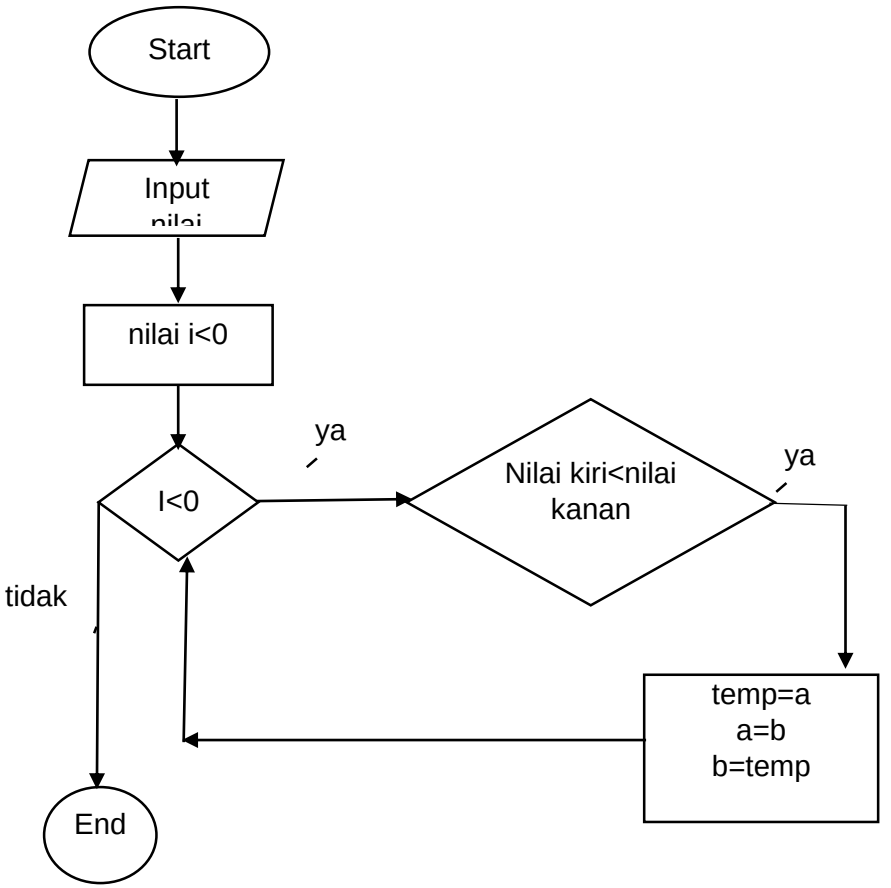
```
#include<stdio.h>
#define N 8 /*Jumlah data*/
void SisipanSort(int data[],int n);
main(void)
{
	int i, n=N-1;
	int data[]={36,2,5,31,40,4,7,3};
	printf("Sebelum diurutkan: ");
	for(i=0;i<=n;i++)printf(" %i",data[i]);
	SisipanSort(data,n);

	printf(" \n");
	printf("Setelah diurutkan: ");
```

```
	for(i=0;i<=n;i++)printf(" %i",data[i]);
}
void SisipanSort(int data[],int n)
{
	int i,j,x;
	bool ketemu;
	for(i=1;i<=n;i++)
	{
		x=data[i];
		j=i-1;
		ketemu=false;
		while((j>=0) && (!ketemu))
	{
		if(x<data[j])
	{
		  data[j+1]=data[j];
		  j=j-1;
	}
	else
	ketemu=true;
	}
```

```
            data[j+1]=x;
        }
}
```

4. Flowchart Insert Sort



C. LATIHAN

Buatlah pengurutan sisipan (*insert sort*) dalam mengurutkan jumlah dan ukuran sepatu yang di *dispay* (dengan jumlah dan ukuran sepatu silahkan tentukan sendiri).

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan Java*. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

BAB XI

PENGURUTAN SHELL

A. CAPAIAN PEMBELAJARAN

Mahasiswa dapat mengerti dan dapat menggunakan pengurutan shell dalam pemrograman.

B. MATERI

1. Definisi Pengurutan Shell

Shell sort ialah algoritma pertambahan menurun (dimising increment sort) jenis perbandingan antara jarak tertentu ditempat. Pengurutan shell sort itu membandingkan gap (jarak) selanjutnya dilakukan penukaran.

Pengurutan shell yaitu mengurutkan data dengan cara membandingkan pasangan data tertentu yang memiliki rentang jarak secara bertahap sehingga membentuk sebuah sub-list. Kemudian dilakukan penukaran apabila memenuhi syarat kondisi tertentu.

Pengurutan Shell adalah optimasi insert sort yang memungkinkan pertukaran yang berjalan. Shell sort berfungsi untuk mengatur daftar element sehingga dapat

dimulai dari mana saja dengan mengambil setiap data ke-I dan menghasilkan daftar yang diurutkan.

2. Metode Pengurutan Shell (Shell SORT)

Diberikan nama pengurutan shell karena penemu pengurutan ini bernama Donald Shell pada tahun 1959, Maka diberi nama pengurutan shell.

Algoritma shell sort ini efektif untuk data set kecil, untuk data yang hamper disorting. Algoritma shell shot tidak efesien saat element harus bergerak jauh dalam susunan.

Kekurangan dari metode shell sort ini yaitu membutuhkan metode tambahan dan sulit membagi masukan.

Kelebihan dari shell sort ini ialah:

a. Algoritma yang sangat rapat dan mudah di implementasikan
b. Operasi pertukaran dilakukan bebas kapan saja
c. Waktu pengurutan dapat lebih ditekan
d. Mudah menggunakannya Kembali

3. Pegoperasian Pengurutan Shell:

a. Gap (Jarak) yang akan diperlukan untuk pengurutan yaitu jarak antardata
b. Terlebih dahulu diketahui jumlah data (N)
c. Gap dimulai dari N/2

d. seiring berjalannya waktu gap (jarak) memperkecil

4. Dicari pengurutan Descending

Nilai awal dari array sebagai berikut:

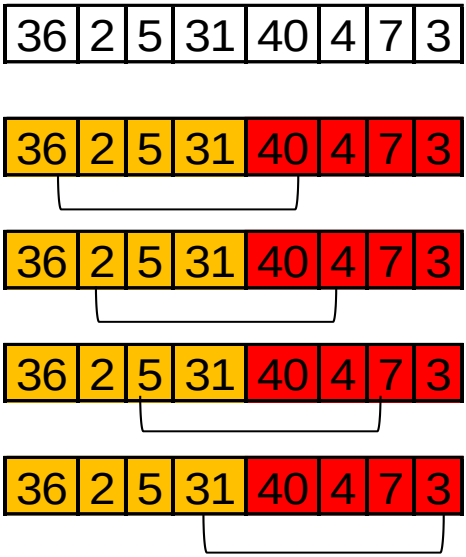
36	2	5	31	40	4	7	3

N : 8

Gap= 8/2=4

a. Pengurutan descending yaitu pengurutan terbesar ke terkecil (menurun).
b. Nilai N=8 dari jumlah dari elemen array diatas.
c. Nilai gap (jarak) didapat dari hasil N/2 yaitu 8/2=4, 4 langkah (gap) antardata.
d. 36 akan dibandingkan dengan 4 langkah nilai data yaitu 40, karena pengurutan descending yaitu pengurutan terbesar ke terkecil (menurun) yang lebih besar disebelah kiri dari yang kecil dan seterusnya sampai habis nilai gap nya.

Menerapkan 4 gap pada array awal sebelum di urutkan.



Proses pengurutan dengan 4 gap dengan pengurutan Descending

36	2	5	31	40	4	7	3
40	2	5	31	36	4	7	3
40	4	5	31	36	2	7	3
40	4	7	31	36	2	5	3
40	4	7	31	36	2	5	3
40	4	7	31	36	2	5	3

Hasil nilai yang diurutkan secara descending berdasarkan 4 gap.

40	4	7	31	36	2	5	3

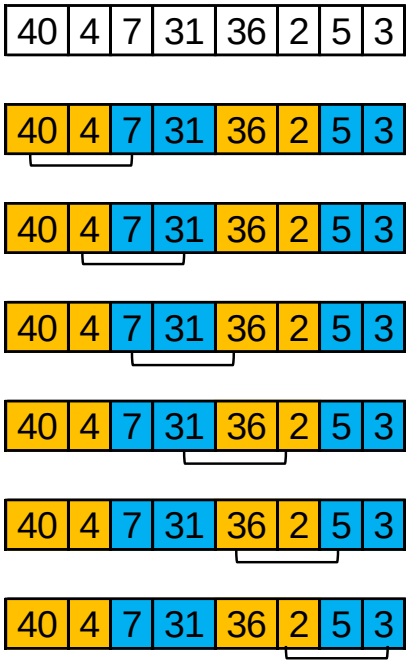
Nilai awal dari array sebagai berikut:

40	4	7	31	36	2	5	3

Gap= 4/2=2

a. Pengurutan descending yaitu pengurutan terbesar ke terkecil (menurun).
b. Nilai N didapat dari hasil gap sebelumnya adalah 4 maka nilai gap sekarang hasil N/2 yaitu 4/2=2, maka nilai N sekarang 2, 2 langkah (gap) antardata.
c. 40 akan dibandingkan dengan 2 langkah nilai data yaitu 7, karena pengurutan descending ialah pengurutan terbesar ke terkecil (menurun) yang lebih besar disebelah kiri dari yang kecil dan seterusnya sampai habis nilai gap nya.

   Menerapkan 2 gap pada array awal sebelum di urutkan.



Proses pengurutan dengan 2 gap dengan pengurutan Descending

40	4	7	31	36	2	5	3
40	4	7	31	36	2	5	3
40	31	36	4	7	2	5	3
40	31	36	4	7	2	5	3
40	31	36	4	7	2	5	3
40	31	36	4	7	2	5	3

40	31	36	4	7	3	5	2

Hasil nilai yang diurutkan secara descending berdasarkan 2 gap.

40	31	36	4	7	3	5	2

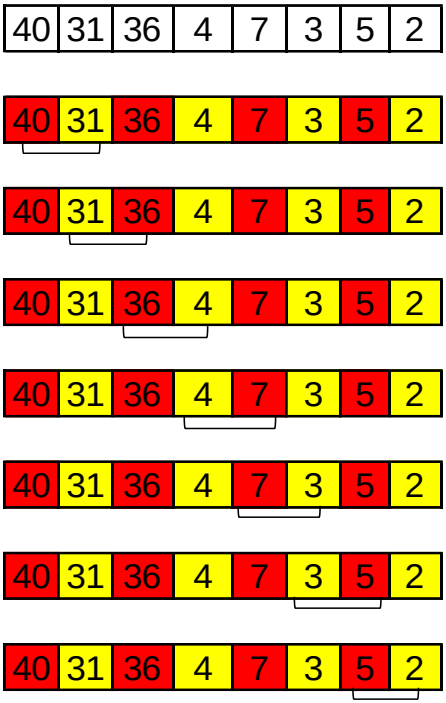
Nilai awal dari array sebagai berikut:

40	31	36	4	7	3	5	2

Gap= 2/2=1

a. Pengurutan descending yaitu pengurutan terbesar ke terkecil (menurun).
b. Nilai N didapat dari hasil gap sebelumnya adalah 2 maka nilai gap sekarang hasil N/2 yaitu 2/2=1, maka nilai N sekarang 2, 2 langkah (gap) antardata.
c. 40 akan dibandingkan dengan 1 langkah nilai data yaitu 31, karena pengurutan descending ialah pengurutan terbesar ke terkecil (menurun) yang lebih besar disebelah kiri dari yang kecil dan seterusnya sampai habis nilai gap nya.

Menerapkan 1 gap pada array awal sebelum di urutkan.



Proses pengurutan dengan 1 gap dengan pengurutan Descending.

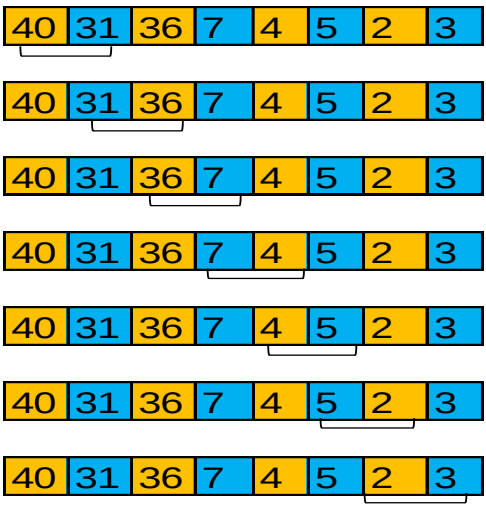
40	31	36	4	7	3	5	2
40	31	36	4	7	3	5	2
40	31	36	4	7	3	5	2
40	31	36	4	7	3	5	2
40	31	36	7	4	3	5	2

40	31	36	7	4	3	5	2
40	31	36	7	4	5	3	2
40	31	36	7	4	5	2	3

Hasil nilai yang diurutkan secara descending berdasarkan 1 gap yang belum selesai di urutkan.

40	31	36	7	4	5	2	3

Menerapkan 1 gap pada array hasil pengurutan sebelumnya, selanjutnya pengurutan lagi dengan 1 gap.



40	31	36	7	4	5	2	3
40	31	36	7	4	5	2	3
40	36	31	7	4	5	2	3
40	36	31	7	4	5	2	3
40	36	31	7	4	5	2	3
40	36	31	7	5	4	2	3
40	36	31	7	5	4	2	3
40	36	31	7	5	4	3	2

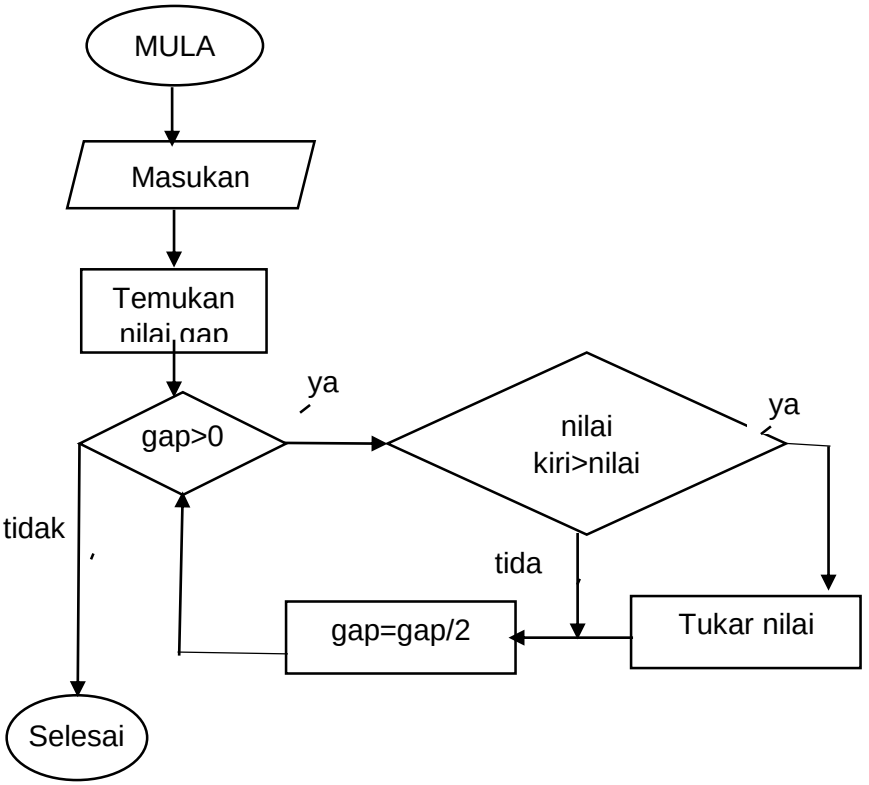
Hasil nilai yang diurutkan secara descending berdasarkan 1 gap yang sudah selesai diurutkan dengan nilai array.

40	36	31	7	5	4	3	2

Hasil akhir secara descending pengurutan shell.

40	36	31	7	5	4	3	2

5. Flowchat Shell Sort



6. Koding Shell Sort:

```
#include<stdio.h>
#define N 8 /*Banyaknya data*/
void PengurutanSisipan(int data[],int n,int mulai,int melangkah);
void pengurutanShell(int data[],int n);
 main(void)
{
     int i, n=N-1;
     int data[]={36,2,5,31,40,4,7,3};
     printf("Sebelum diurutkan: ");
     for(i=0;i<=n;i++)printf(" %i",data[i]);
     pengurutanShell(data,n);
     printf(" \n");
     printf("Setelah diurutkan: ");
     for(i=0;i<=n;i++)printf(" %i",data[i]);
}
void pengurutanShell(int data[],int n)
{
       int mulai,melangkah;
       for(melangkah=4;melangkah>=1;melangkah-=2)
```

```
    {
        for(mulai=0; mulai<=melangkah;mulai++)
      PengurutanSisipan(data,n,mulai,melangkah);
    }
}
void PengurutanSisipan(int data[],int n, int mulai, int melangkah)
{
    int i,j,x;
    bool ketemu;
    i=mulai+melangkah;
    while(i<=n)
    {
        x=data[i];
        j=i-melangkah;
        ketemu=false;
        while((j>=0) && (!ketemu))
        {
          if(x<data[j])
          {
                data[j+melangkah]=data[j];
```

```
                    j=j-melangkah;
                }
                else
                ketemu=true;
            }
            data[j+melangkah]=x;
            i=i+melangkah;
    }
}
```

C. LATIHAN

Buatlah pengurutan shell (shell sort) dalam memilih kendaraan bermotor dari harga termurah sampai harga termahal (dengan harga silahkan tentukan sendiri).

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan*

Java. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

BAB XII

MERGER SORT DAN QUICK SORT

A. CAPAIAN PEMBELAJARAN

Mahasiswa dapat mengerti dan dapat menggunakan merger sort dan quick sort dalam pemrograman.

B. MATERI

1. Merger Sort

Merger Sort adalah algoritma pengurutan yang dilakukan dengan cara memecah kemudian menggabungkan sekaligus mengurut.

Pengurutan Merger merupakan algoritma yang dirancang untuk memenuhi kebutuhan pengurutan atas suatu rangkaian data yang tidak memungkinkan untuk ditampungdalam memori komputer karena jumlahnya yang terlalu besar.

2. Proses pengurutan descending dengan merger sort:

1.	Array di pecah di bagi 2 menjadi array kanan dan array kiri.
2.	Setelah itu aray yang sebelah kiri di bagi lagi menjadi 2 tersisa dari masing-masing array 2 elemen array.
3.	Setelah itu aray yang sebelah kiri di bagi lagi menjadi 2 tersisa dari masing-masing array 1 elemen array.
4.	Begitu seterusnya sampai masing-masing array tersisa 1 elemen array.
5.	Kalo sudah sampai masing-masing elemen array sisa 1 elemen array selanjutnya digabung. Melakukan merger descending (mengurutkan dari nilai terbesar ke nilai terkecil). Digabung dari masing-masing array yang isinya 1 elemen array dan di urutkan di dalam 1 array.begitu seterusnya sampai selesai pengagungan dan pengurutan.

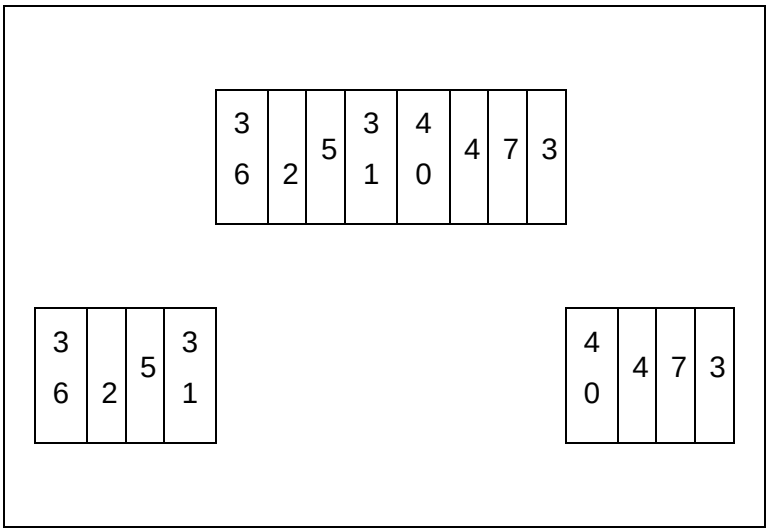
Tahap 1

Array dengan nilai awal.

36	2	5	31	40	4	7	3

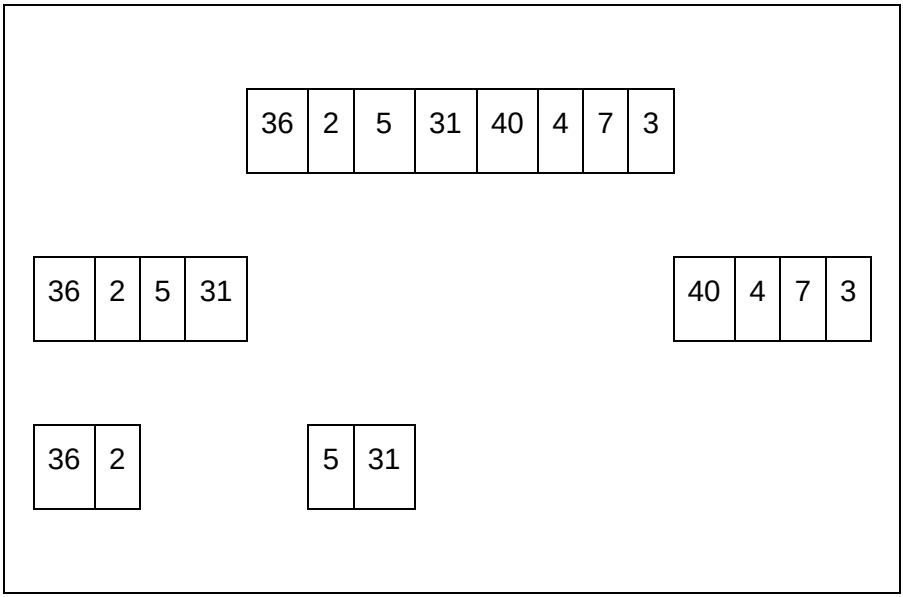
Tahap 2

Array di pecah di bagi 2.



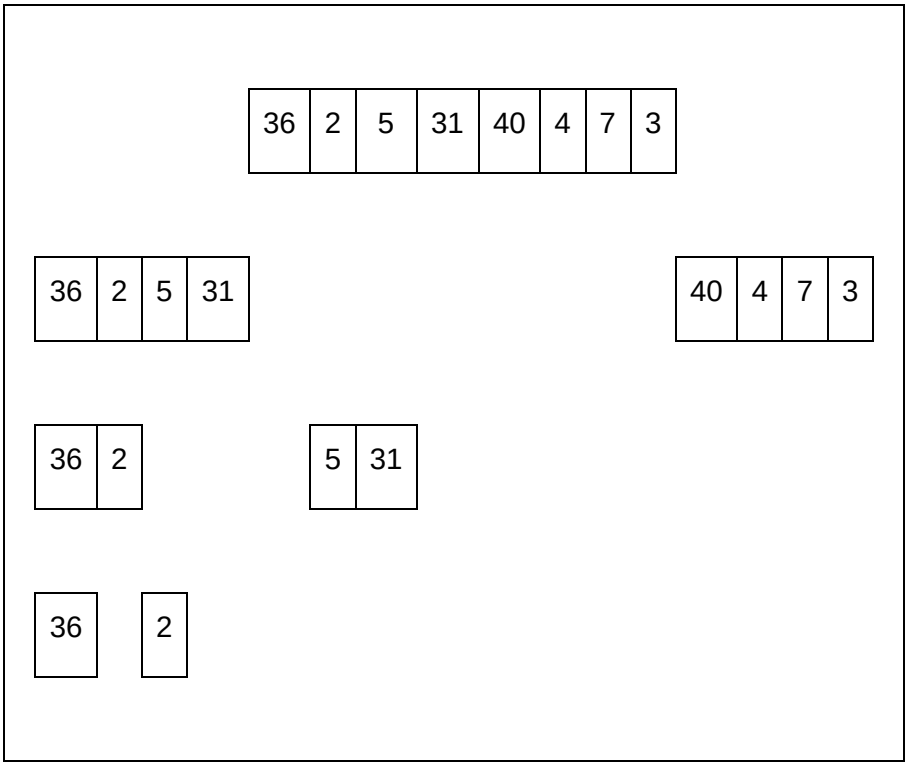
Tahap 3

Array di pecah di bagi 2.



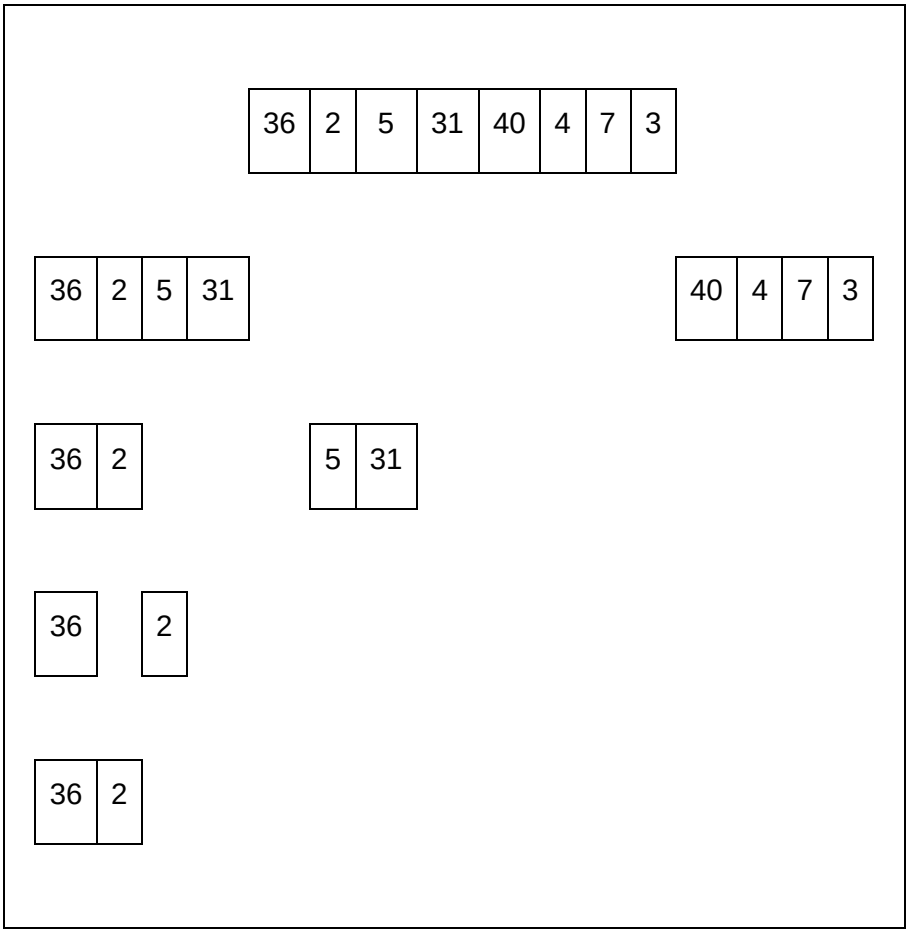
Tahap 4

Array di pecah di bagi 2.



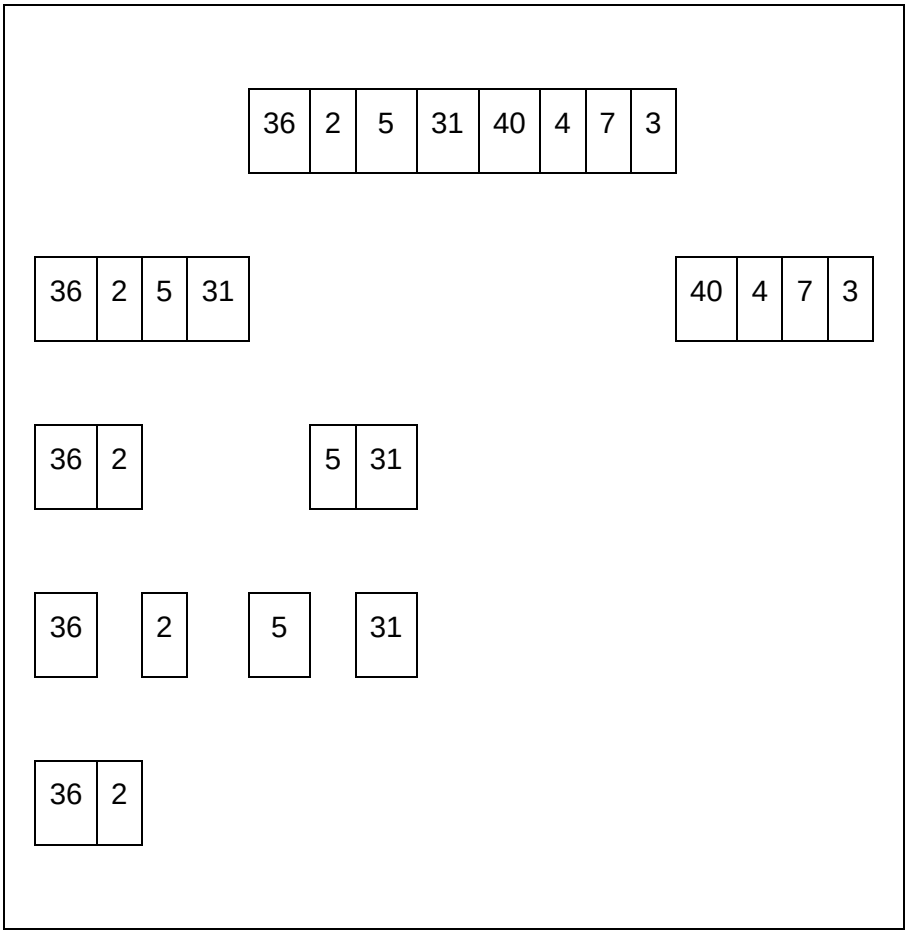
Tahap 5

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).


36 2 5 31 40 4 7 3
36 2 5 31
40 4 7 3
36 2
5 31
36
2
36 2

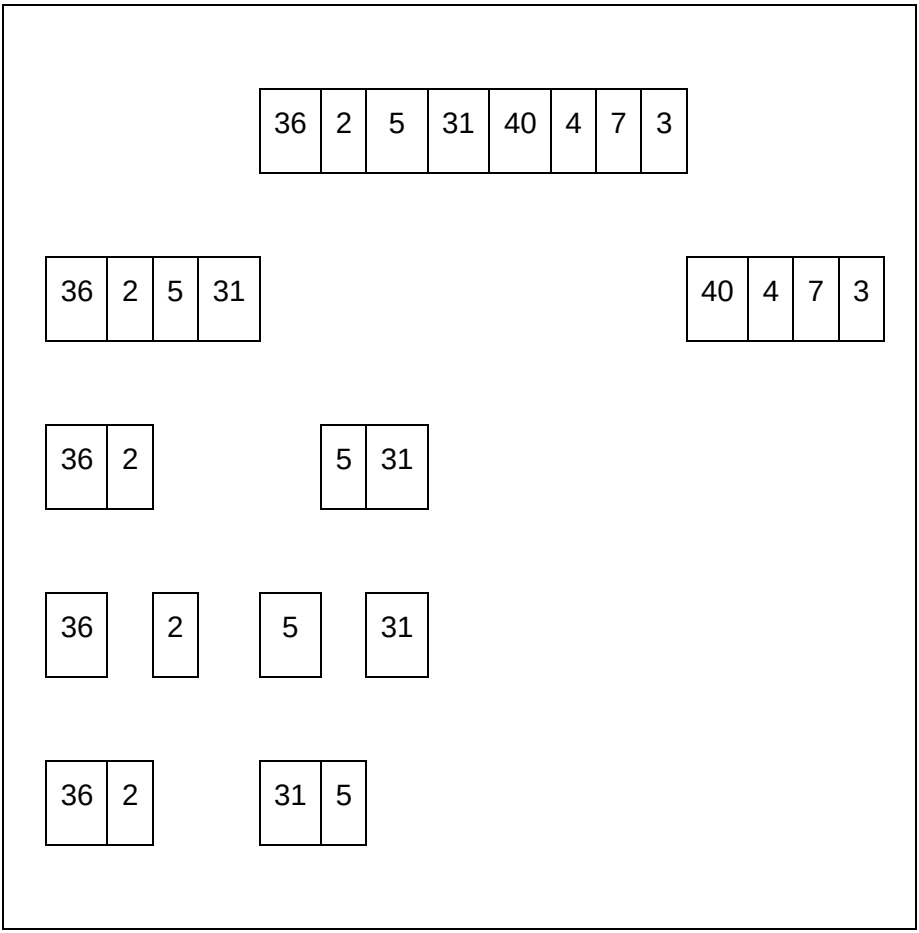
Tahap 6

Array di pecah di bagi 2.



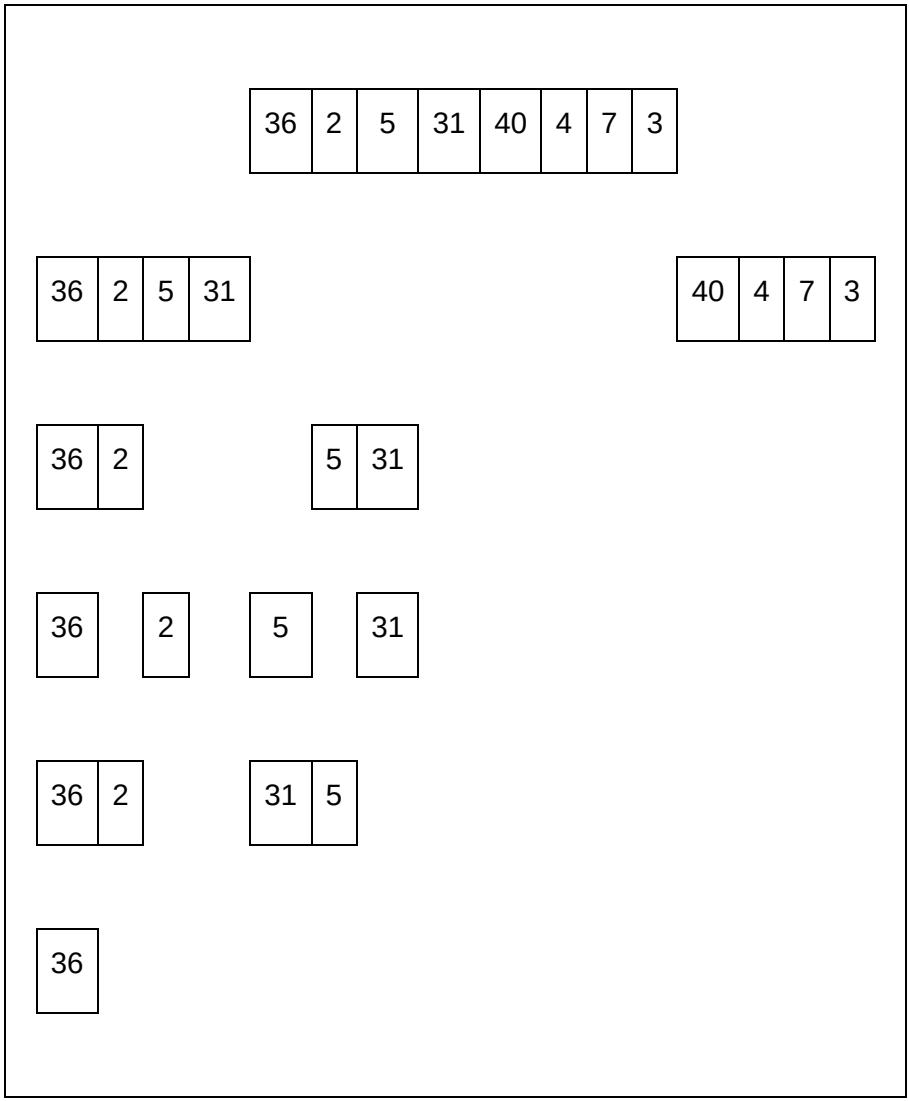
Tahap 7

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



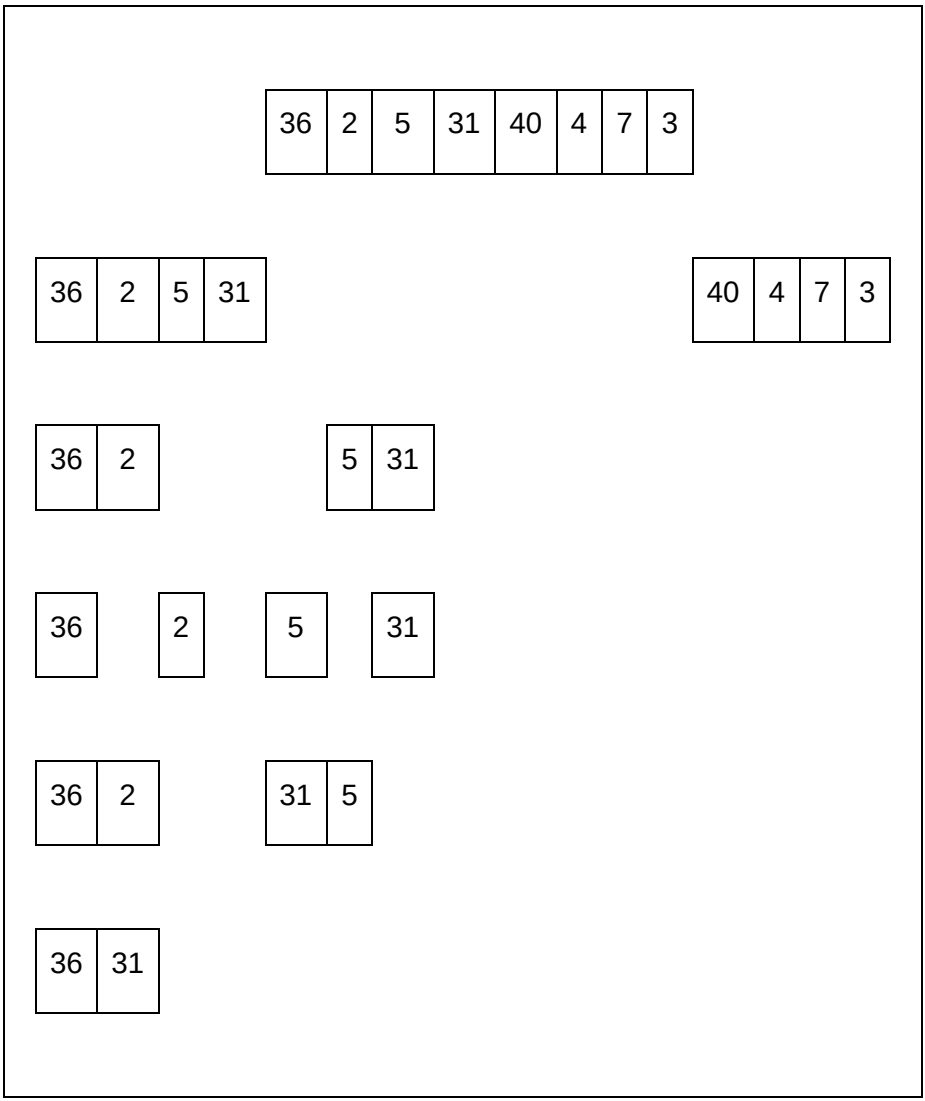
Tahap 8

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



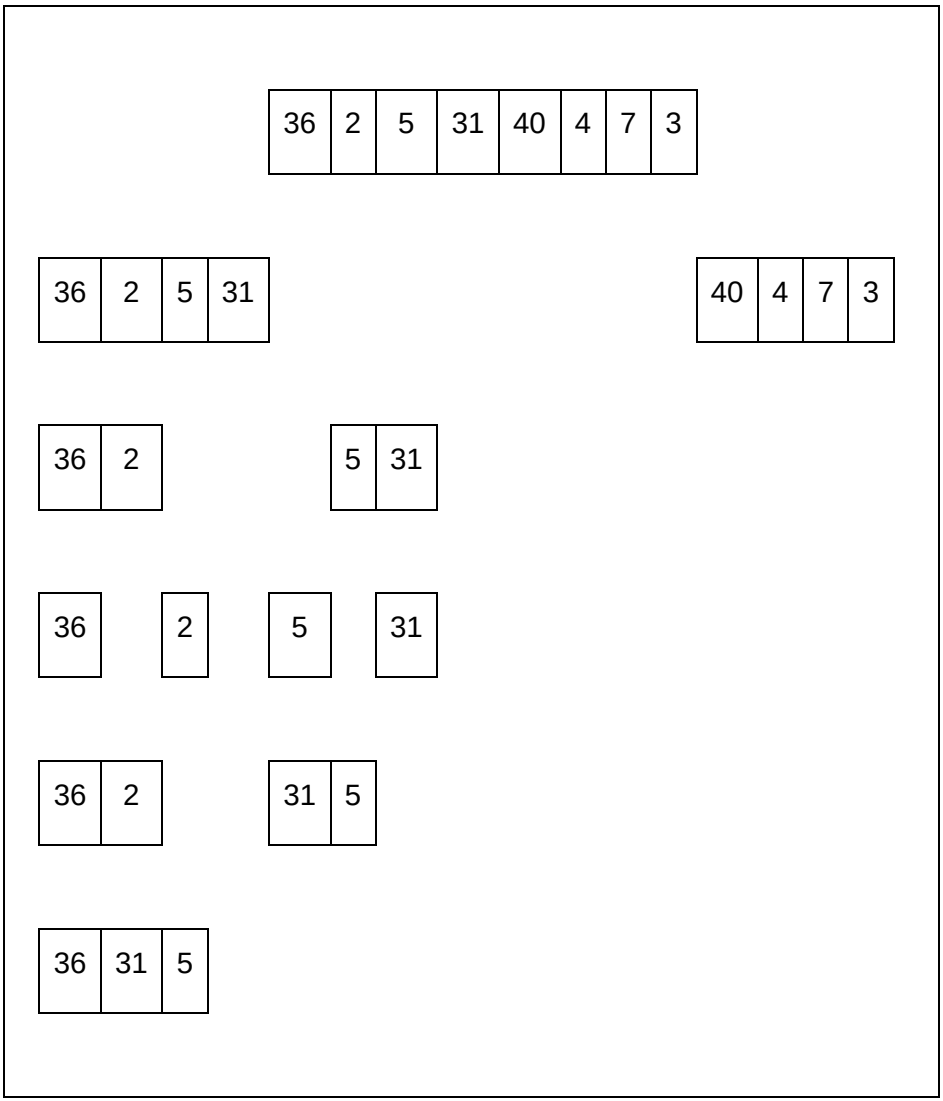
Tahap 9

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



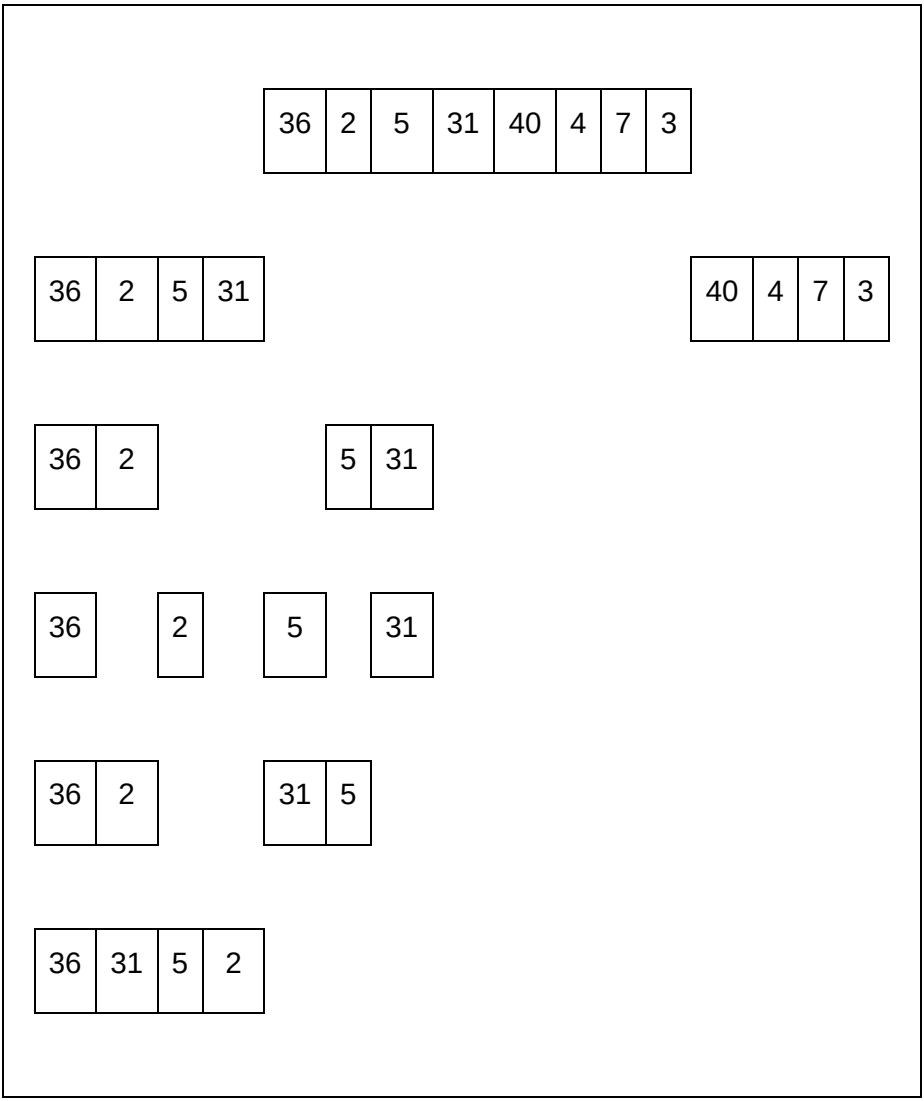
Tahap 10

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



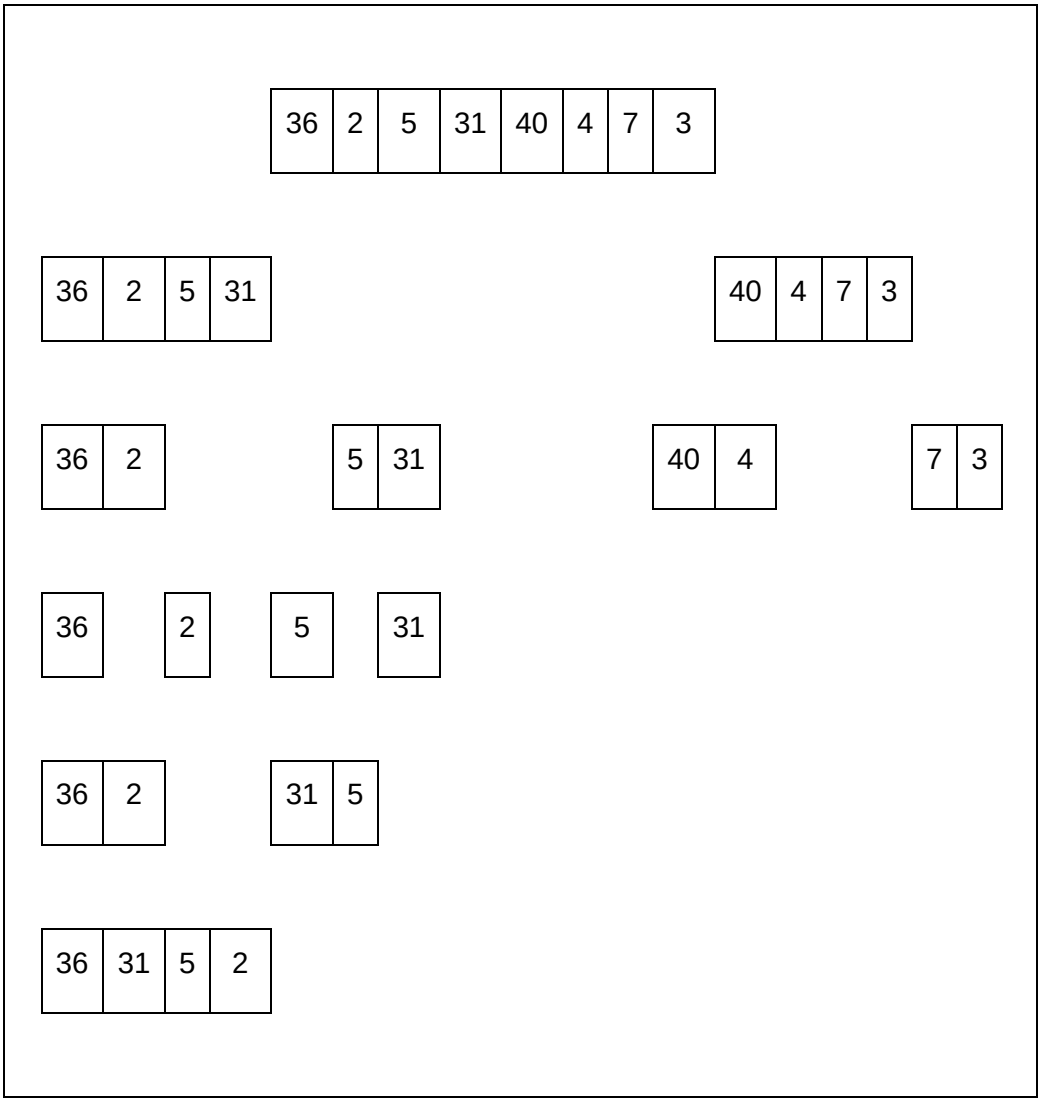
Tahap 11

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



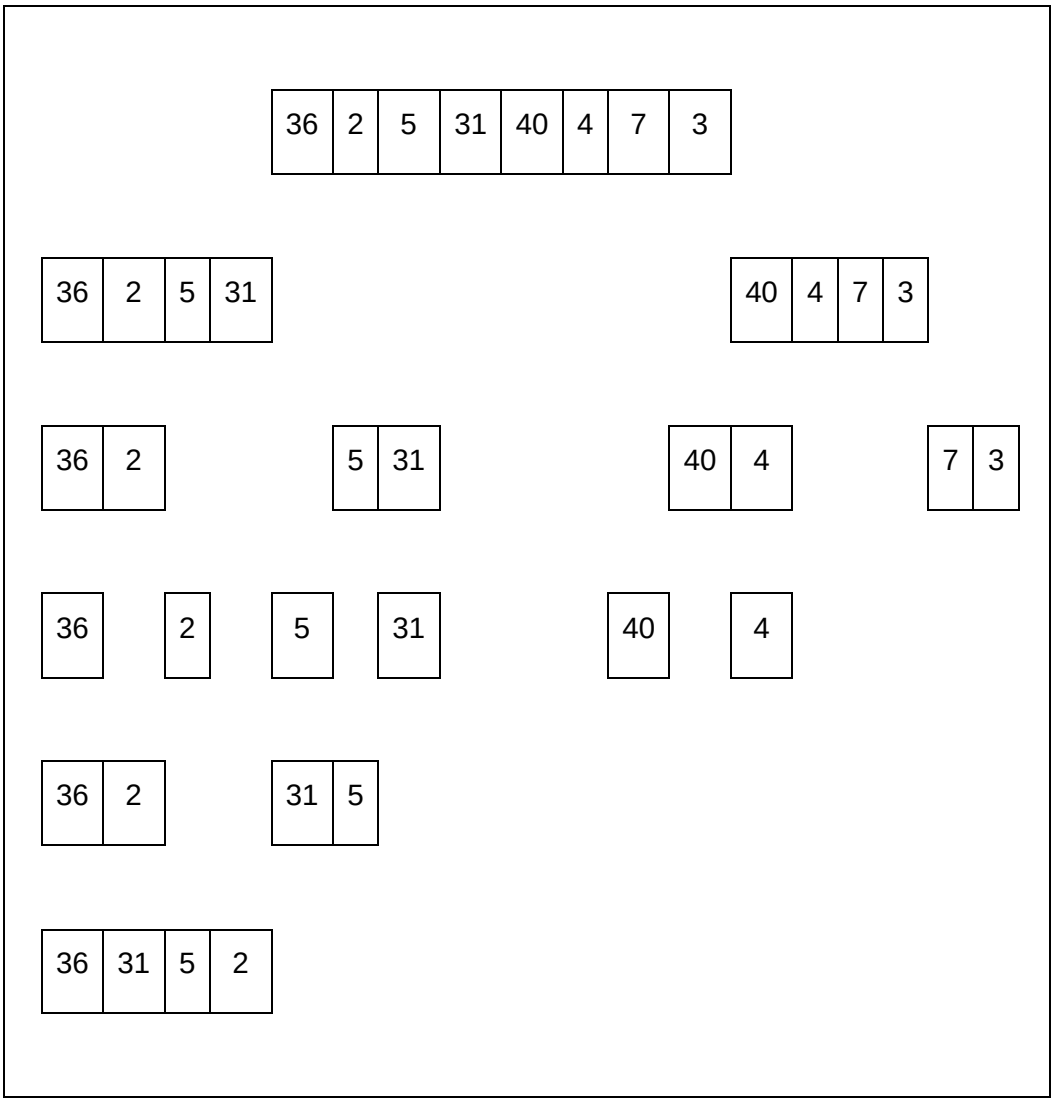
Tahap 12

Array di pecah di bagi 2.



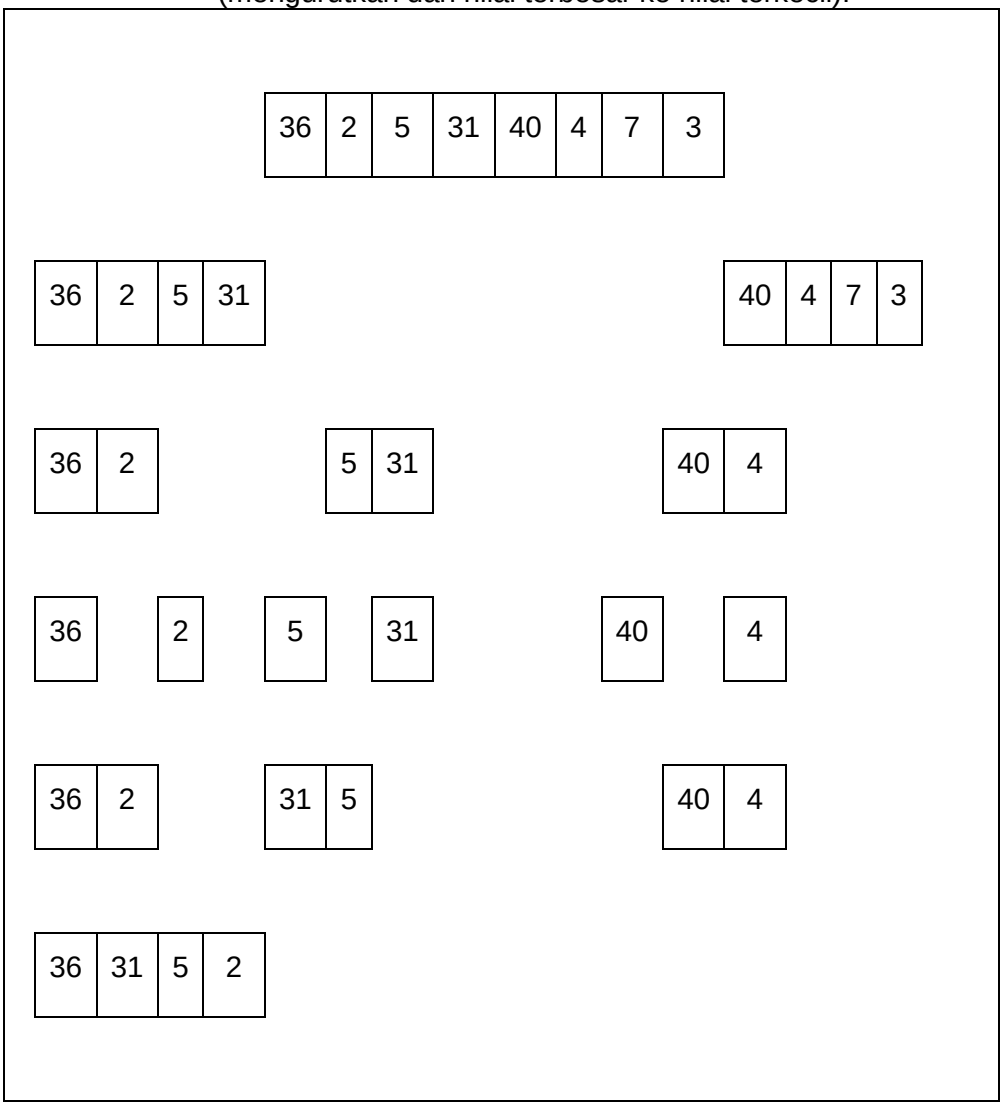
Tahap 13

Array di pecah di bagi 2.



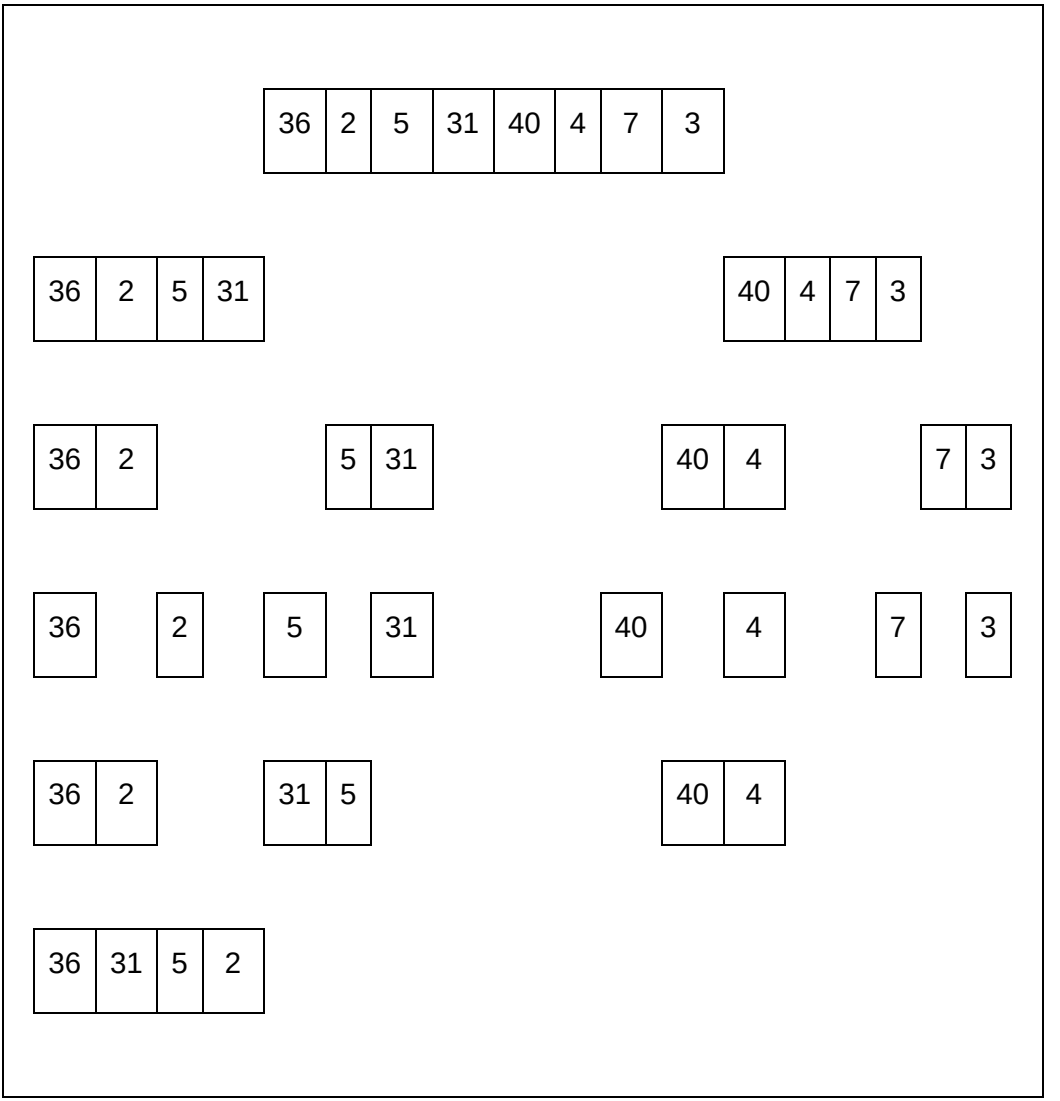
Tahap 14

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



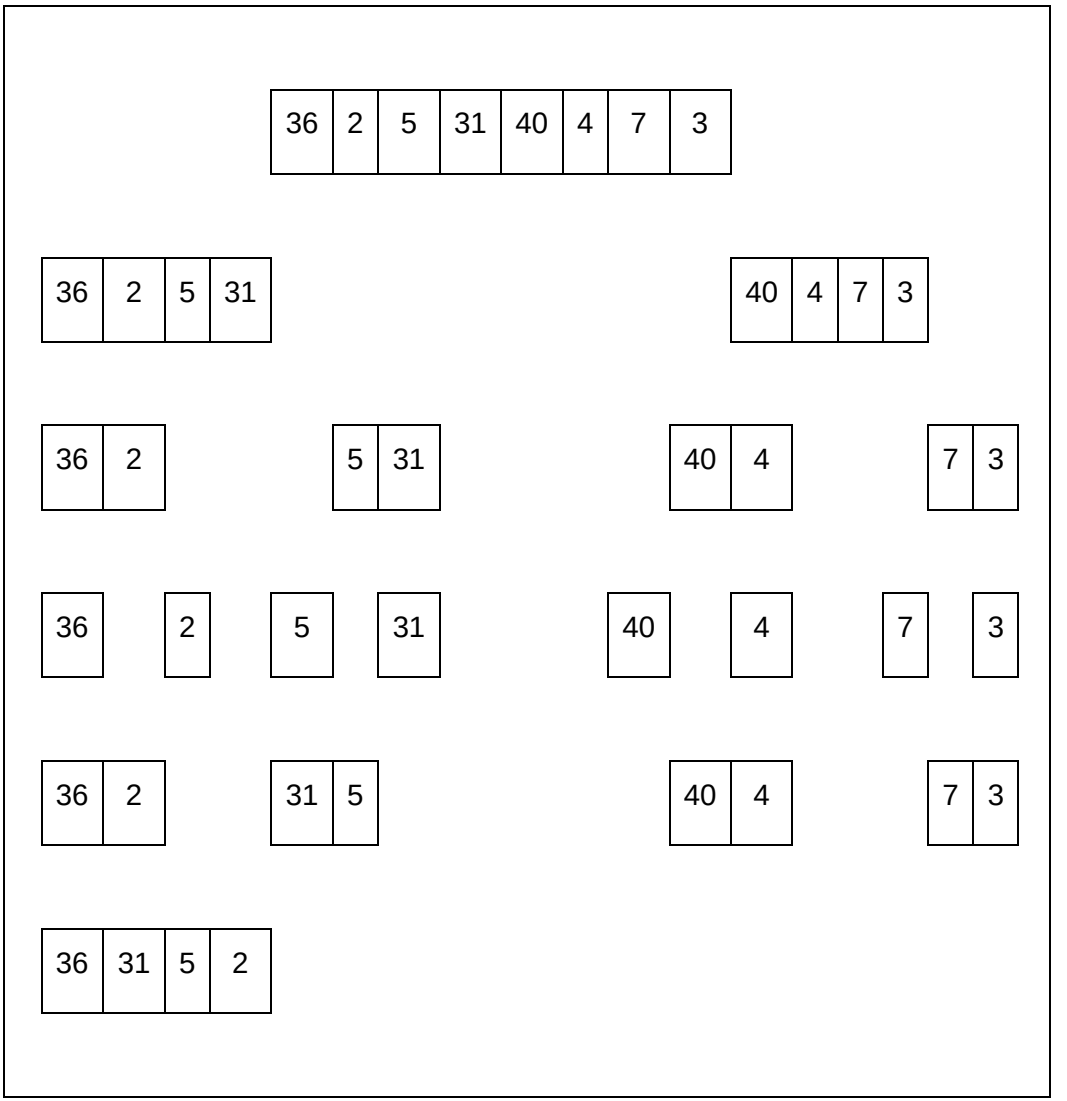
Tahap 15

Array di pecah di bagi 2.



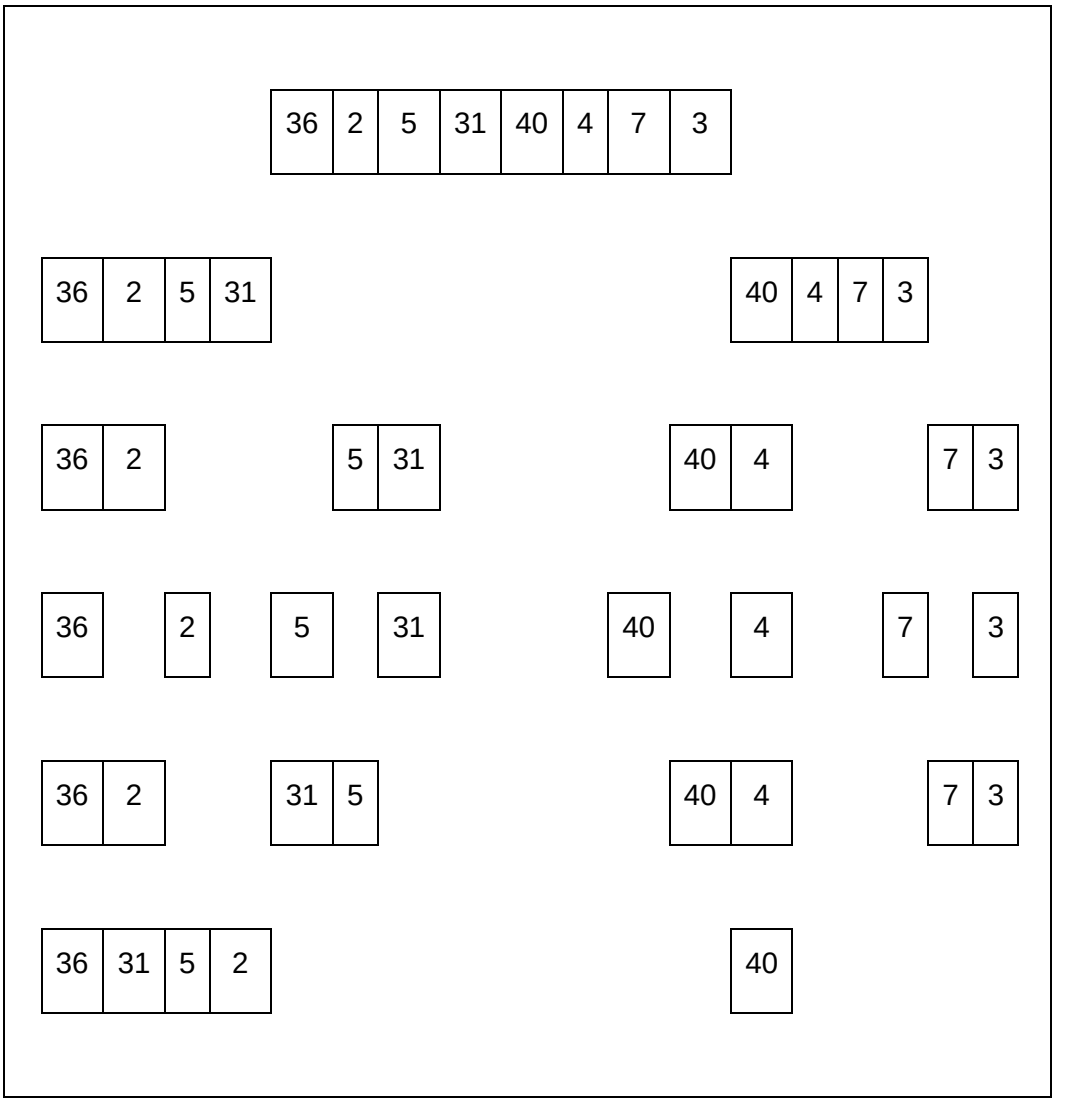
Tahap 16

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



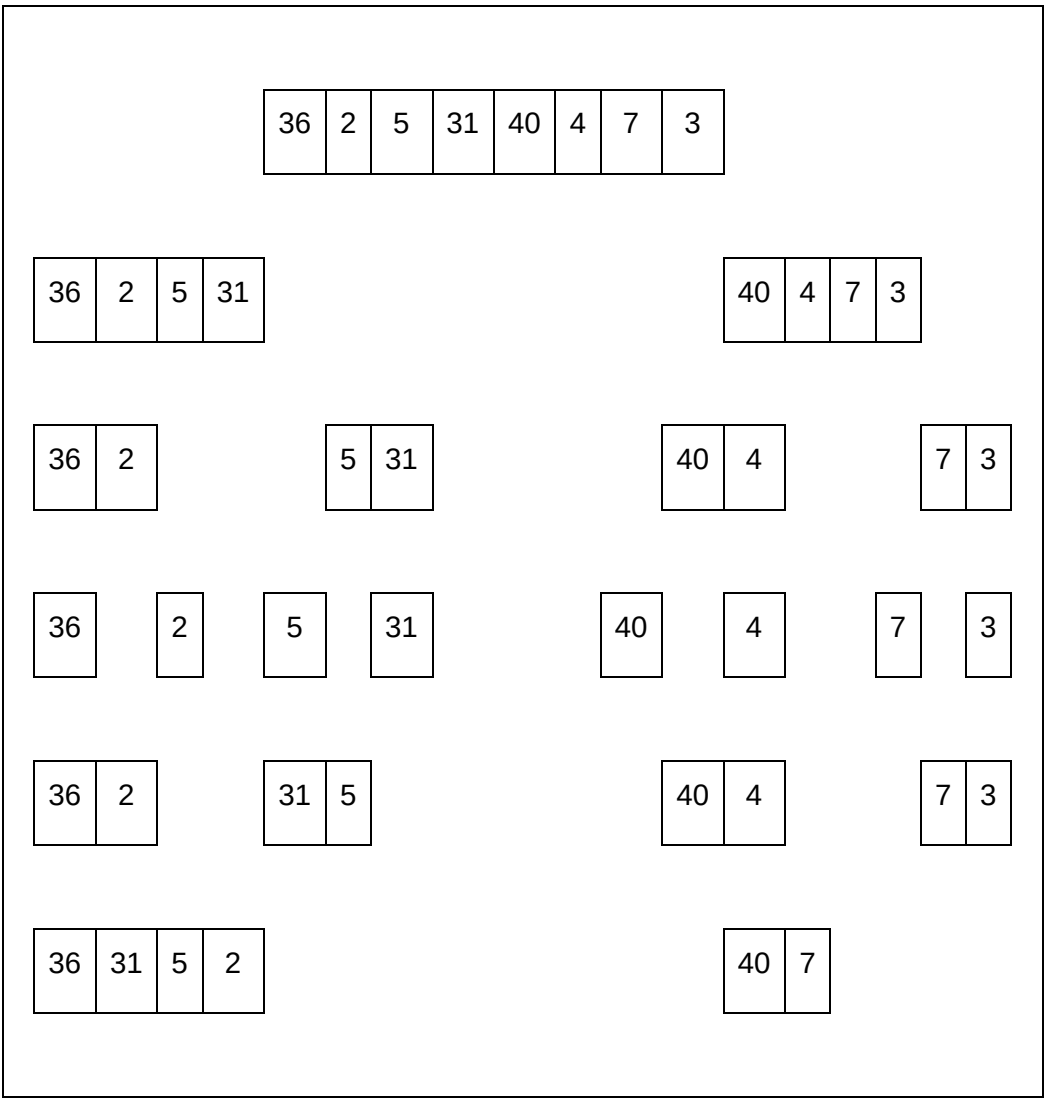
Tahap 17

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



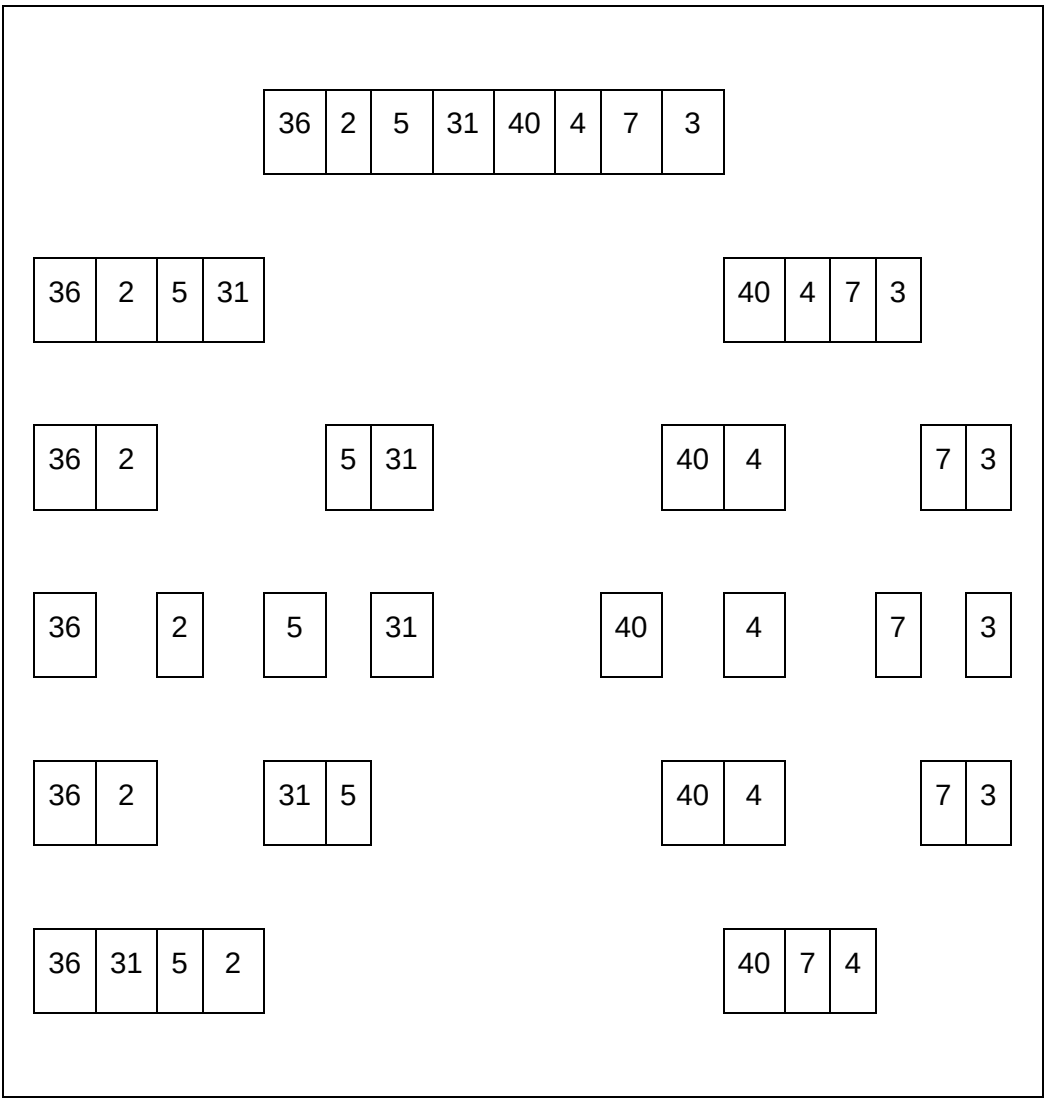
Tahap 18

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



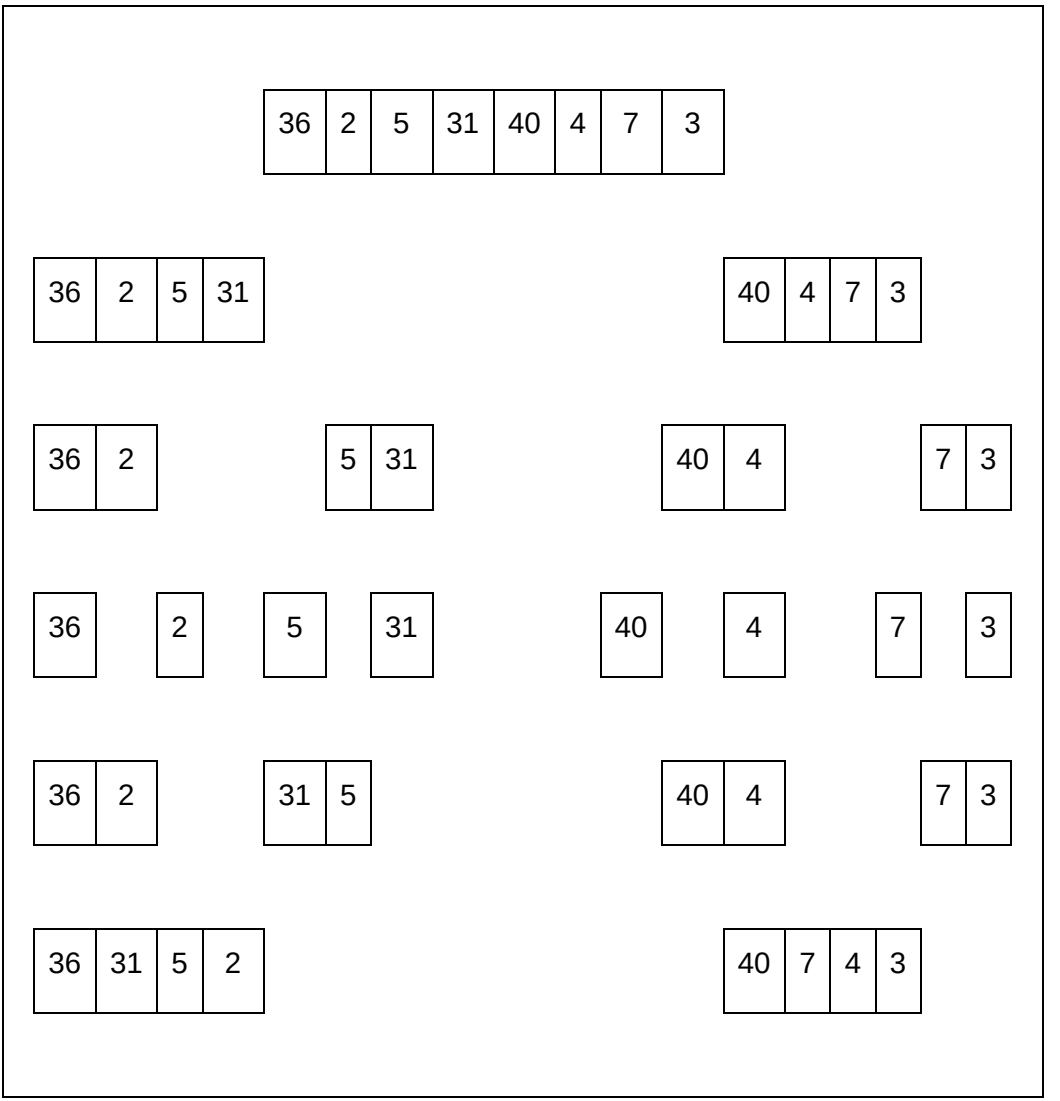
Tahap 19

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



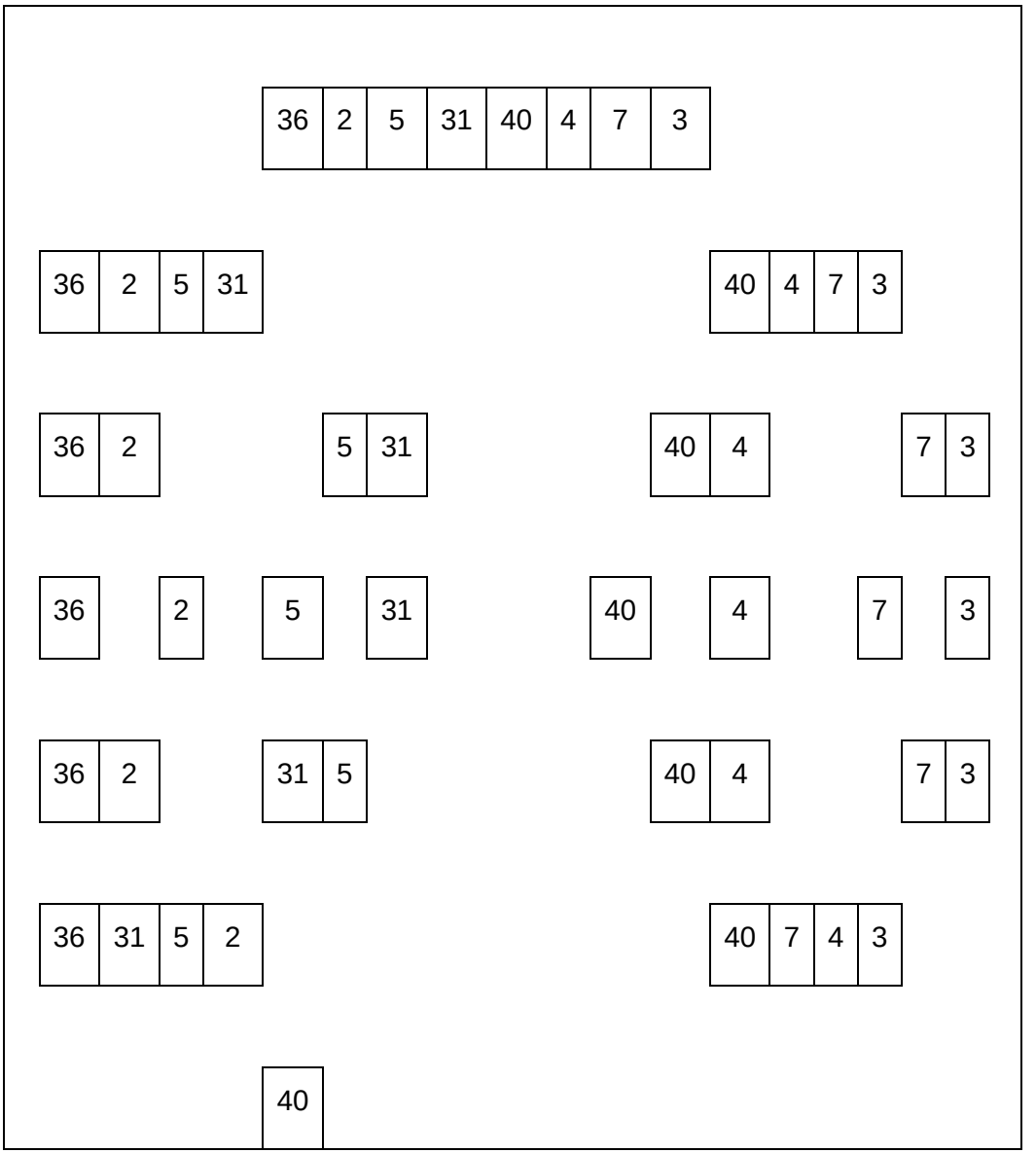
Tahap 20

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



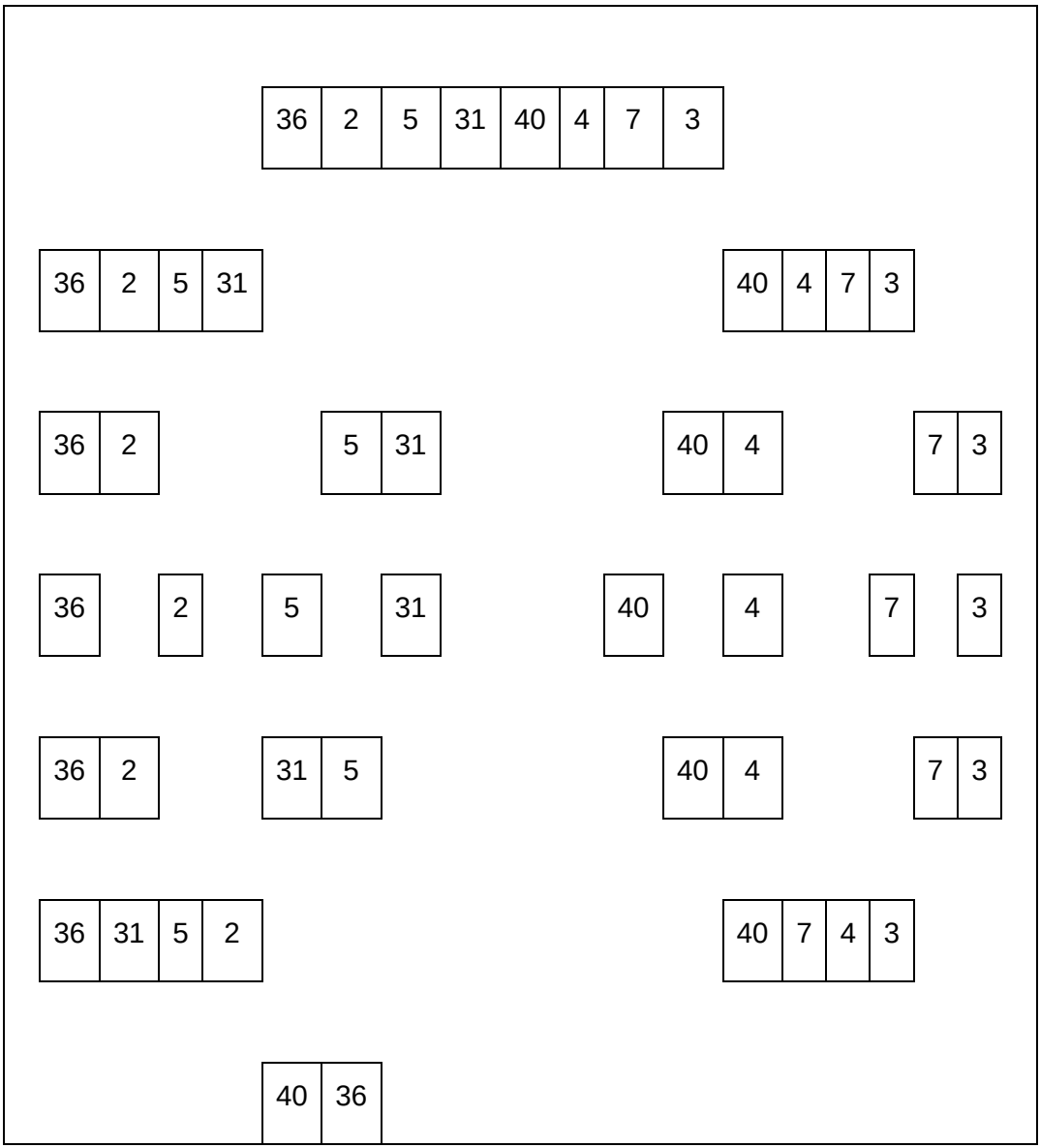
Tahap 21

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



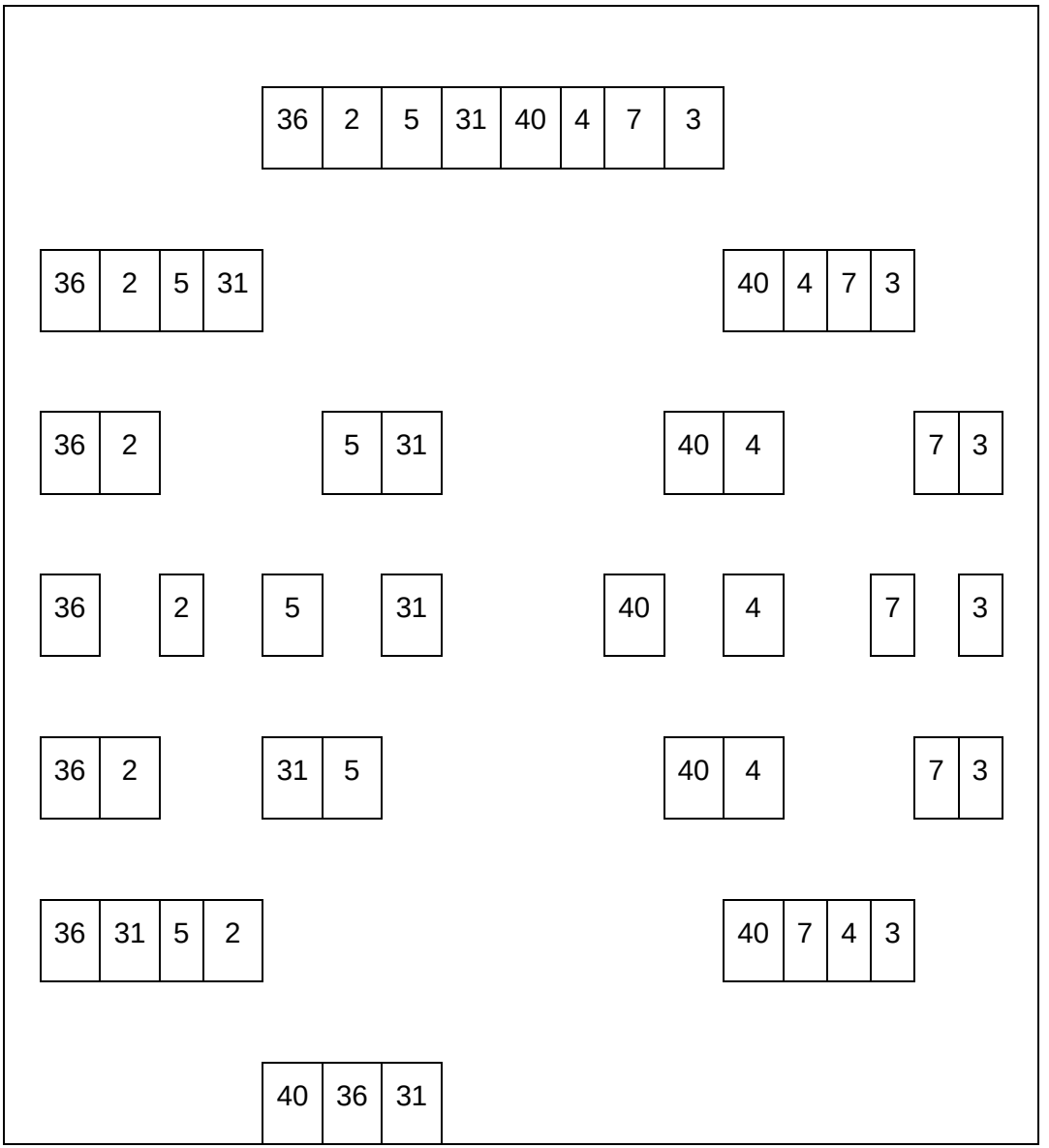
Tahap 22

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



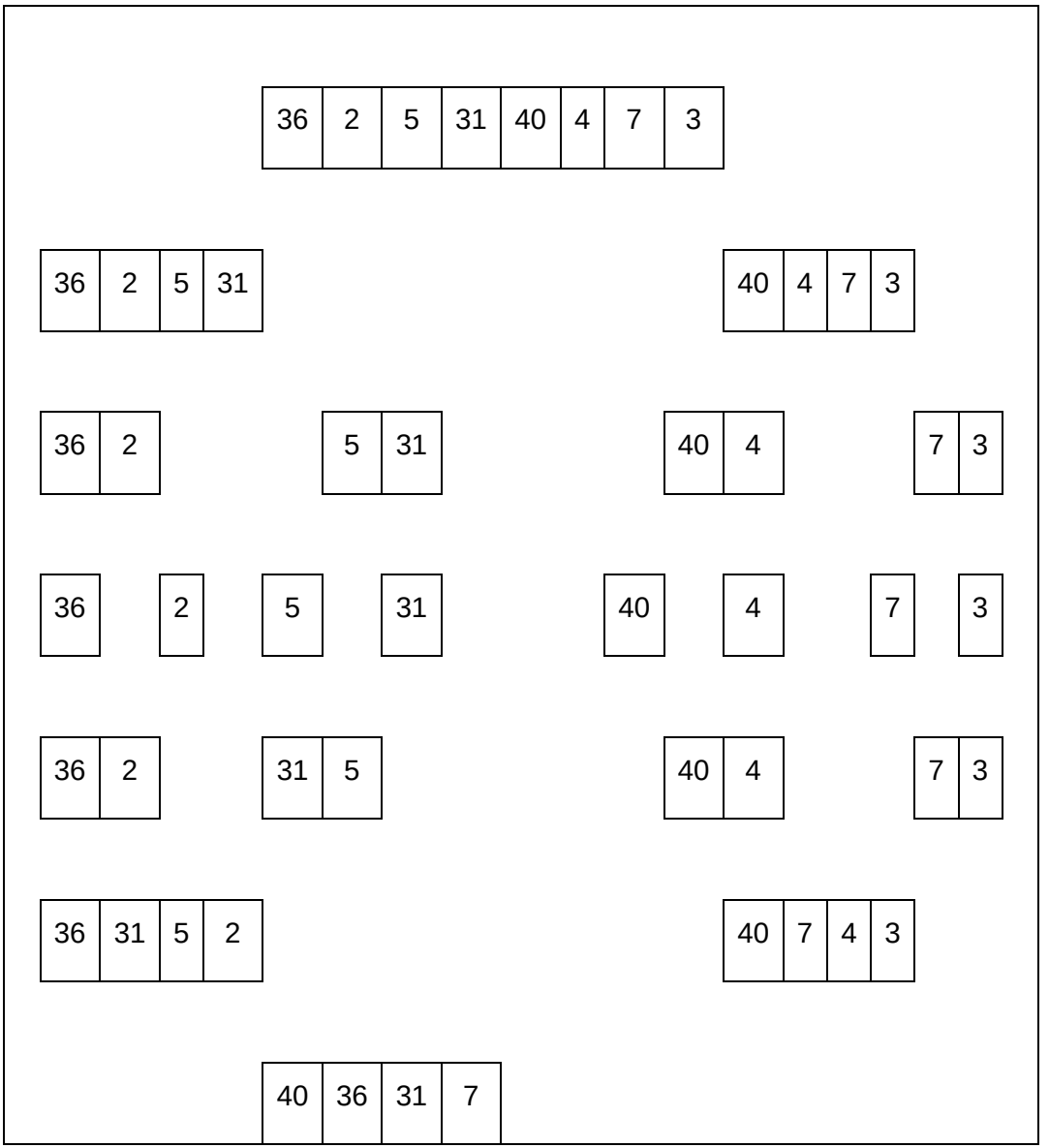
Tahap 23

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



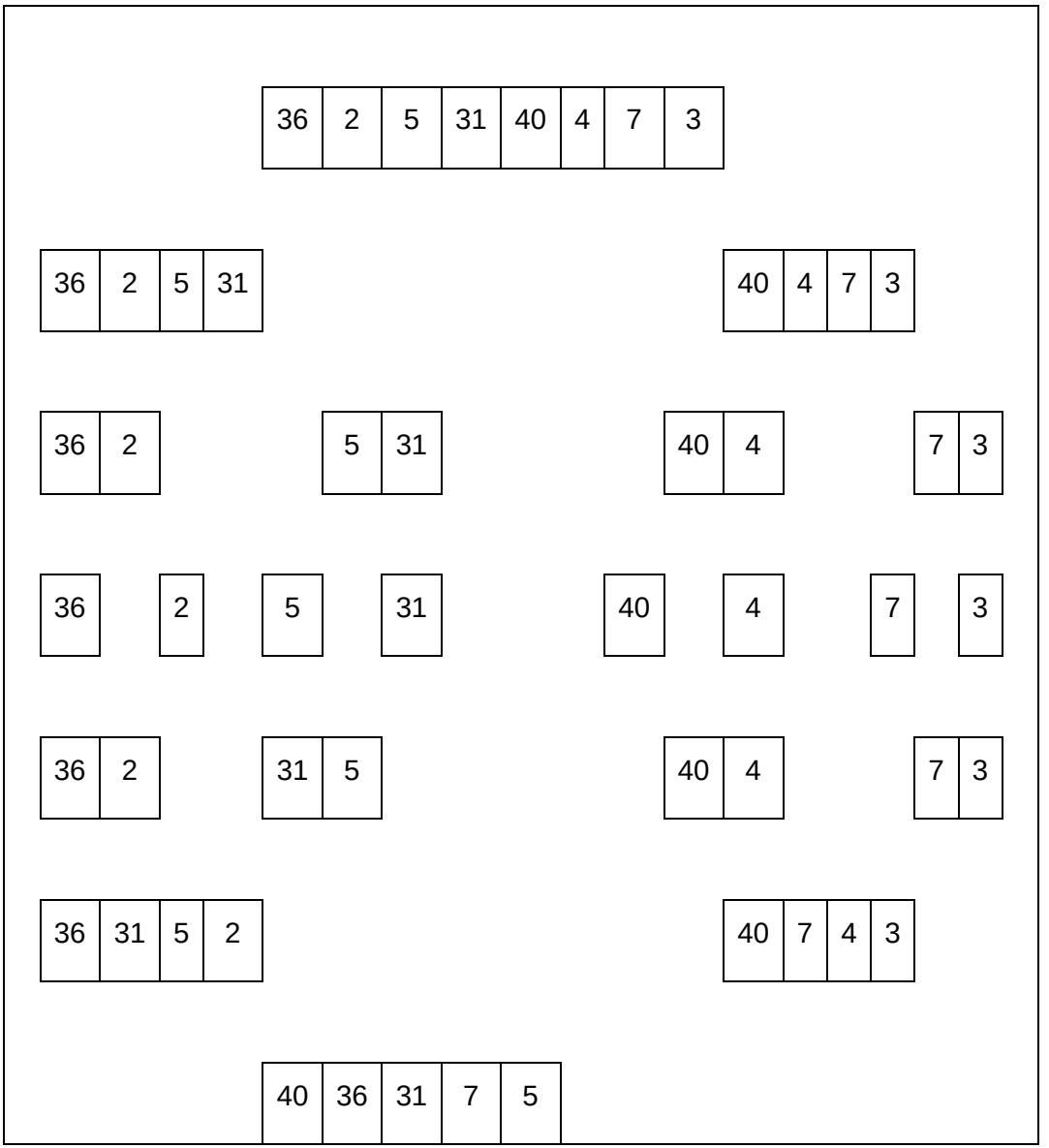
Tahap 24

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



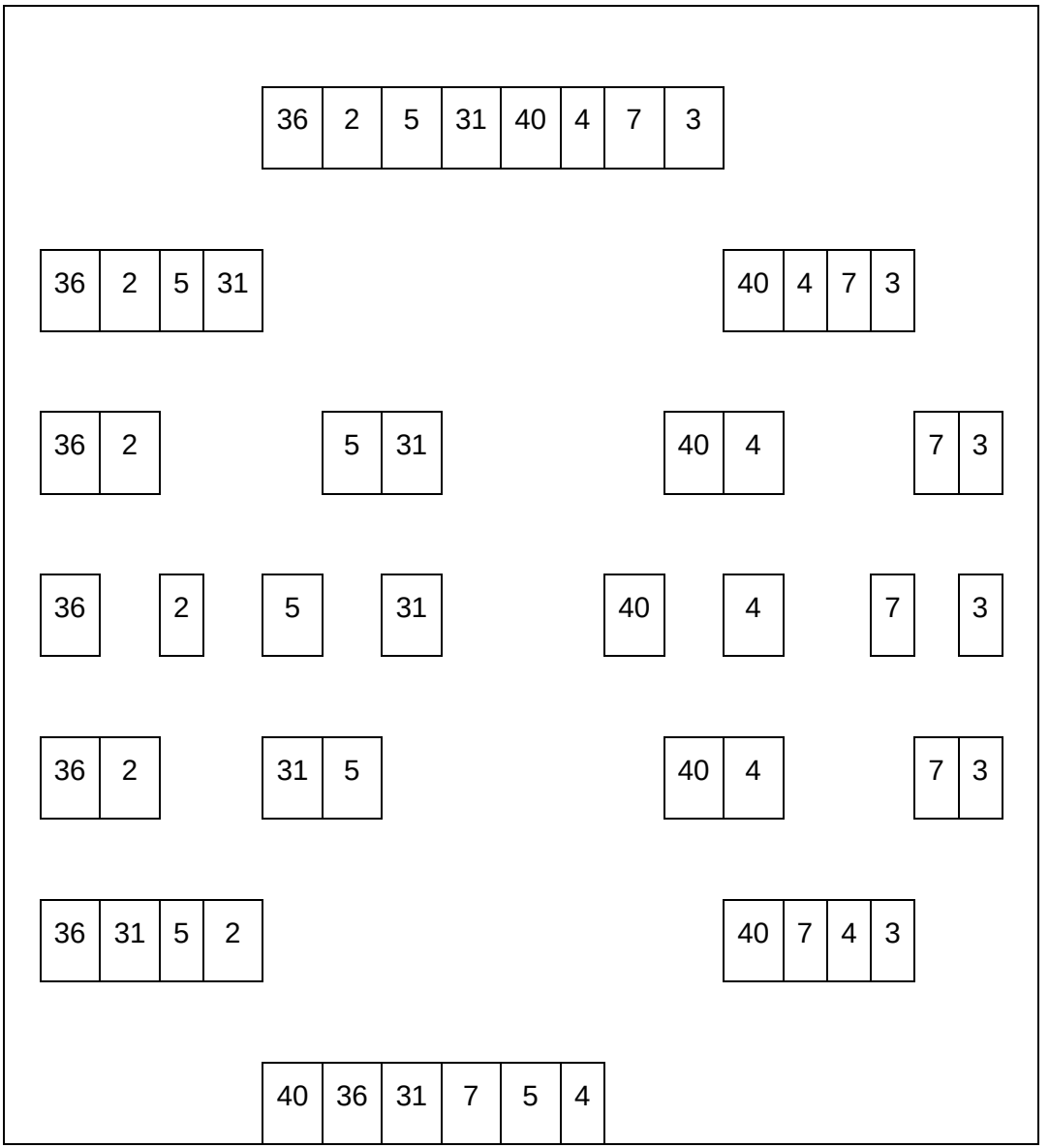
Tahap 25

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



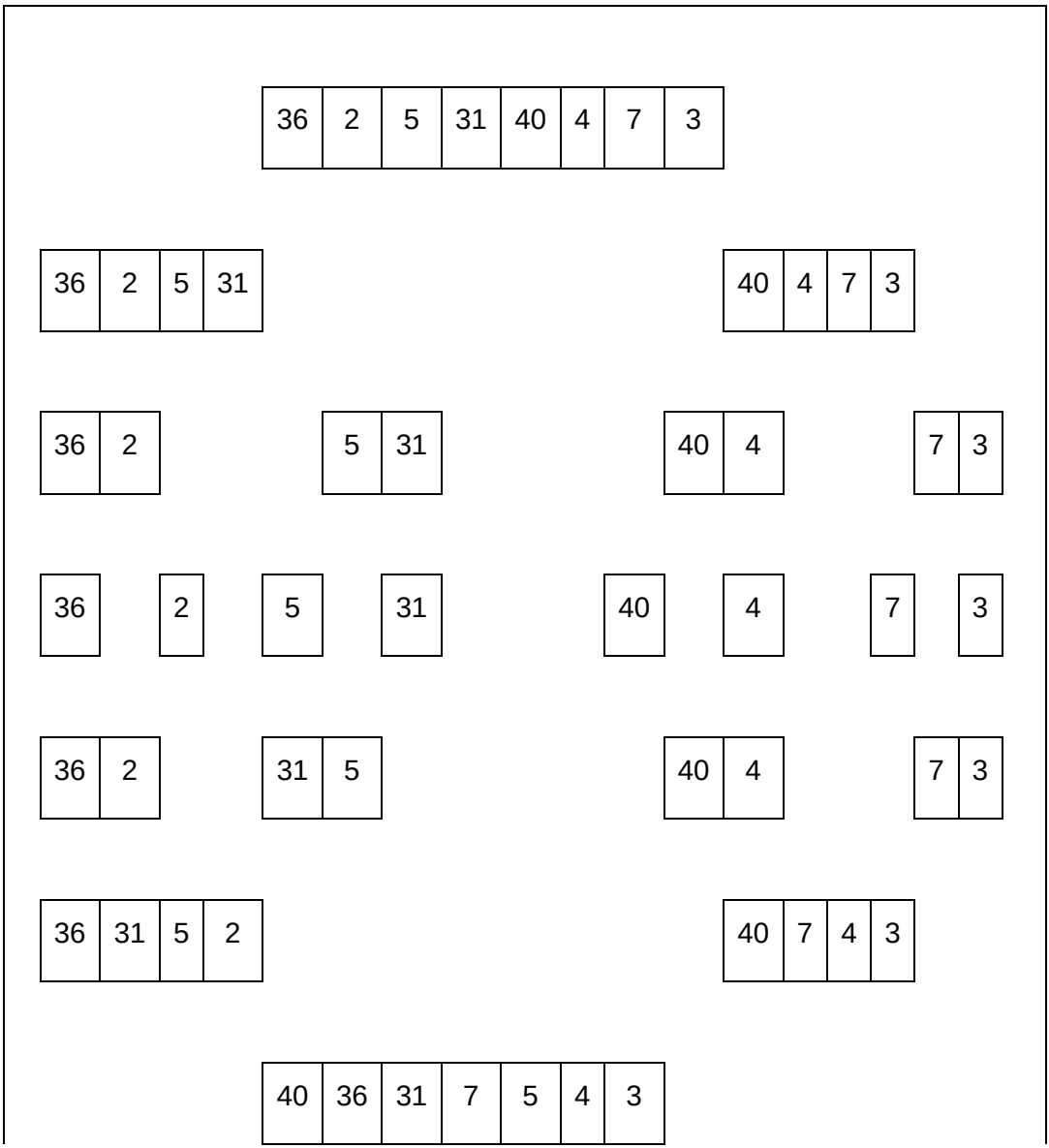
Tahap 26

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



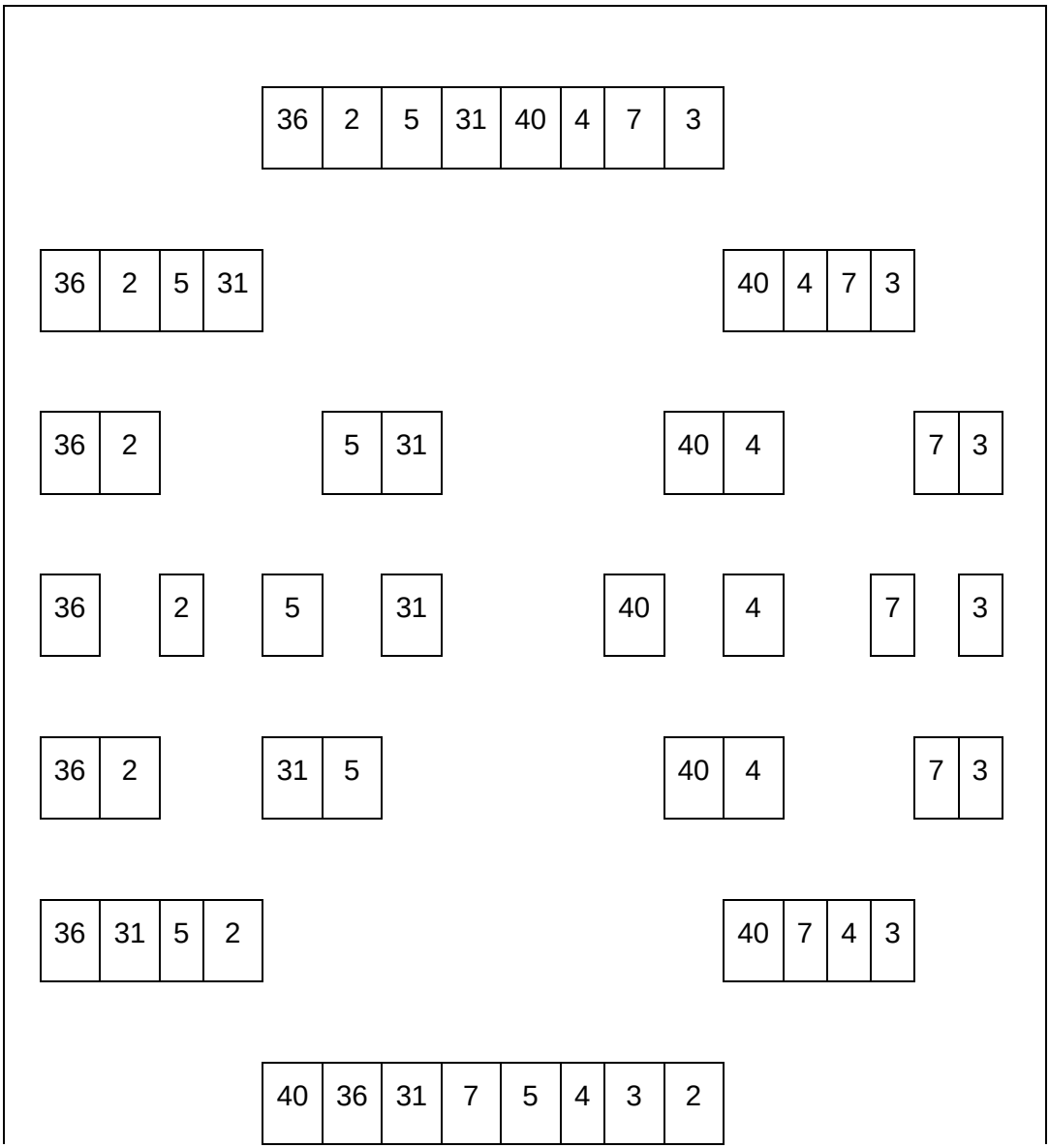
Tahap 27

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



Tahap 28

Melakukan merger sekaligus pengurutan descending (mengurutkan dari nilai terbesar ke nilai terkecil).



Hasil pengurutan descending dari merger sort.

40	36	31	7	5	4	3	2

3. Program C++ Merger Sort

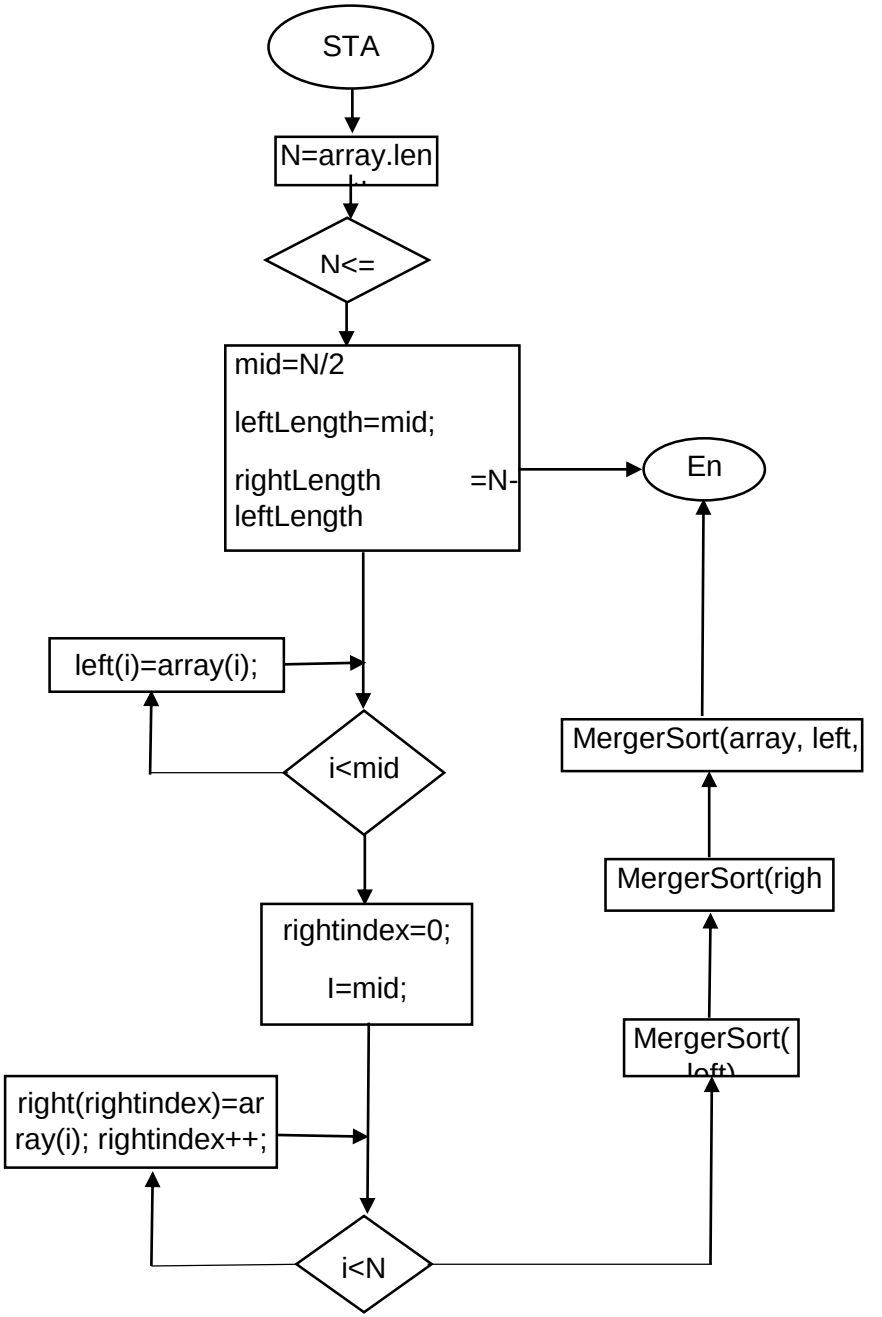
```
#include <iostream>
using namespace std;
void merge(int low, int mid, int up);
void mergeSort(int low, int up);
int a[50];
int main()
{
  int jumlahBil,i;
  cout<<"Masukkan Jumlah element Array"<< endl;
  cin>>jumlahBil;
  for(int i=0; i<jumlahBil;i++)
  {
    cout<<"Bilangan ke-"<< i+1 << endl;
    cin>>a[i];
  }
  mergeSort(1,jumlahBil);
```

```
  for(i=1;i<=jumlahBil;i++)
  cout<<a[i]<<"    ";
  cout<<endl;
  return 0;
}
void merge(int low, int mid, int up)
{
  int h, i,j,k;
  int b[50];
  h = low;
  i = low;
  j = mid+1;
  while((h<=mid)&&(j<=up))
  {
    if(a[h] < a[j])
      {
      b[i]=a[h];
      h++;
    }
    else
    {
      b[i]=a[j];
      j++;
    }
    i++;
```

```
  }
  if(h>mid)
  {
    for(k=j;k<=up;k++){
    b[i]=a[k];
    i++;
    }
  }
  else
  {
    for(k=h;k<=mid;k++)
    {
      b[i]=a[k];
      i++;
    }
  }
  for(k=low;k<=up;k++)
  a[k]=b[k];
}
void mergerSort(int low, int up)
{
  int mid;
  if(low<up)
  {
    mid=(low+up)/2;
    mergeSort(low,mid);
    mergeSort(mid+1,up);
```

```
      merge(low,mid,up);
    }
}
```

4. Flowchart Merger Sort



5. Quick Sort

Quick sort hampir sama seperti merger sort yaitu membagi 2 tetapi menggunakan pivot.

Quick sort ialah algoritma pengurutan data teknik pemecahan data menjadi partisi-partisi, sehingga metode ini disebut juga dengan nama partition exchange sort. Untuk memakai irterasi pengurutan, pertama-tama sebuah elemen data akan diurutkan diatur sedemikian rupa.

Algoritma ini mengambil salah satu elemen secara acak (biasanya dari tengah) yang disebut dengan pivot lalu menyimpan semua elemen yang lebih kecil di sebelah kiri pivot dan semua yang lebih kecil di sebelah kiri pivot dan semua elemen yang lebih besar disebelah kanan pivot. Hal ini dilakukan secara rekursif terhadap elemen disebelah kiri dan kanannya sampai semua elemen sudah terurut.

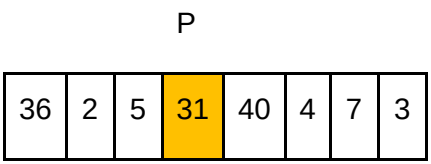
Tips memilih pivot:

a. Pilih pada nilai awal, tengah, atau akhir dari sebuah array.
b. Pilih nilai secara acak dari sebuah array.
c. Pilih nilai median dari sebuah array.

36	2	5	31	40	4	7	3

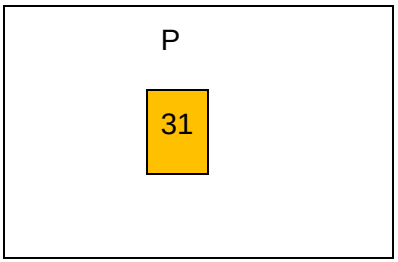
Pilih pivot pada nilai tengah yaitu di nilai array 31.

pivot = p



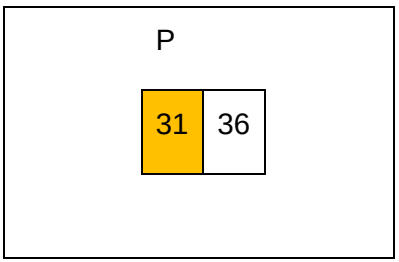
Tahap 1

Memilih pivot dengan nilai array 31.



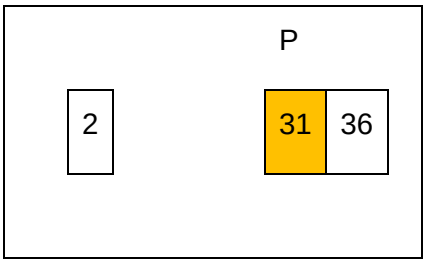
Tahap 2

Setelah memilih pivot kita pilih acak nilai array yang akan dibandingkan dengan pivot. 36 lebih besar dari 31 maka 36 diletakan dibelakang pivot.



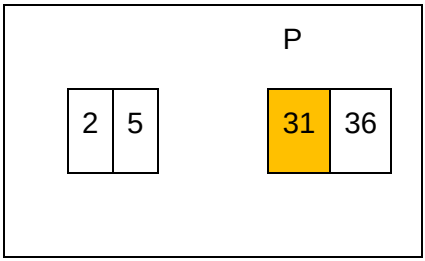
Tahap 3

Nilai array yang akan dibandingkan dengan pivot. 2 lebih kecil dari 31 maka diletakan di depan pivot.



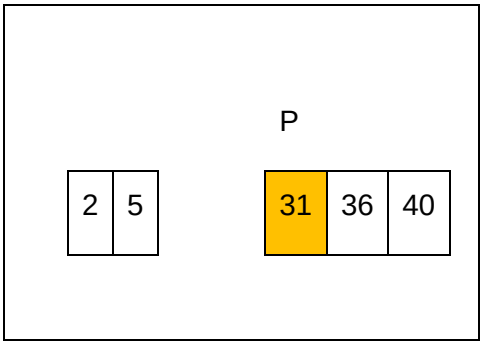
Tahap 4

Nilai array yang akan dibandingkan dengan pivot. 5 lebih kecil dari 31 maka diletakan didepan pivot.



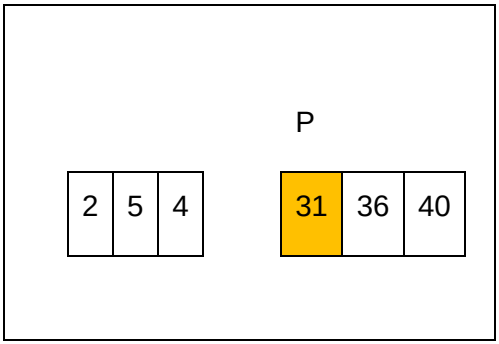
Tahap 5

Nilai array yang akan dibandingkan dengan pivot. 40 lebih besar dari 31 maka diletakan dibelakang pivot.



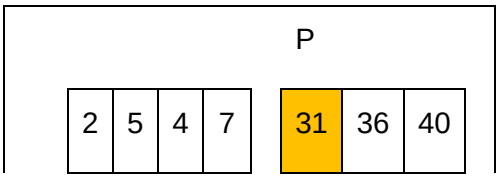
Tahap 6

Nilai array yang akan dibandingkan dengan pivot. 4 lebih kecil dari 31 maka diletakan didepan pivot.



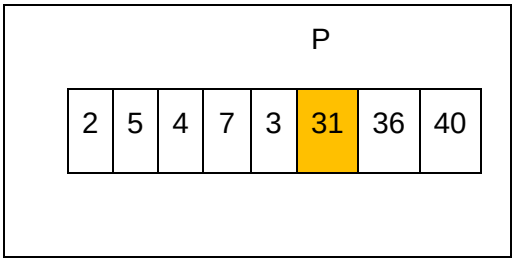
Tahap 7

Nilai array yang akan dibandingkan dengan pivot. 7 lebih kecil dari 31 maka diletakan didepan pivot.



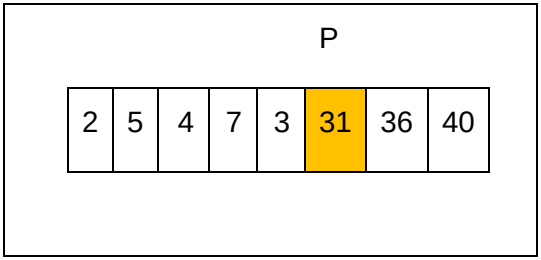
Tahap 8

Nilai array yang akan dibandingkan dengan pivot. 3 lebih kecil dari 31 maka diletakan didepan pivot.



Tahap 9

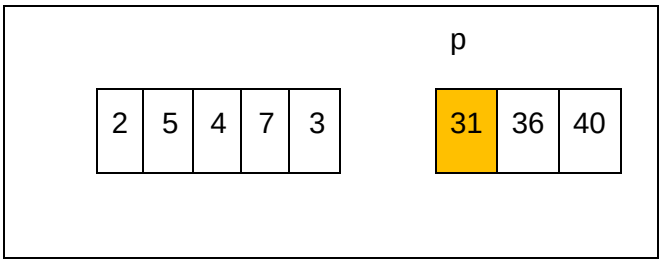
Setelah semua array di bandingkan dengan pivot maka terbentuklah array baru.



Tahap 11

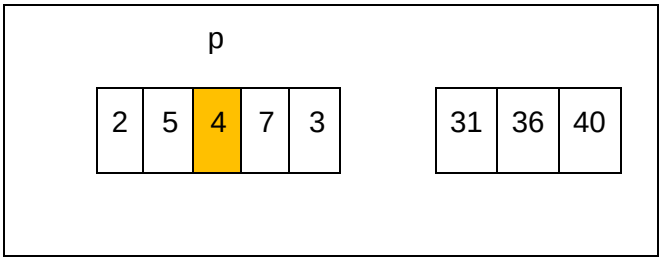
Setelah terbentuk array baru selanjutnya array tersebut dipecah menjadi 2 array. Yang sudah terurut baru 1 array yang terdiri dari nilai array 31,36,40.

Array yang satu lagi arraynya belum terurut.



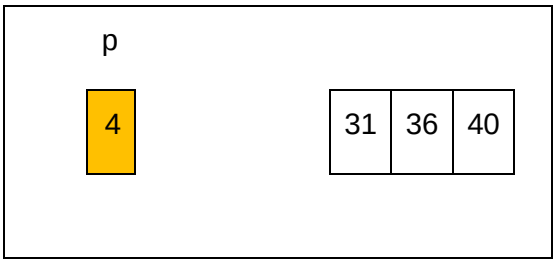
Tahap 12

Memilih pivot dengan nilai array 4.



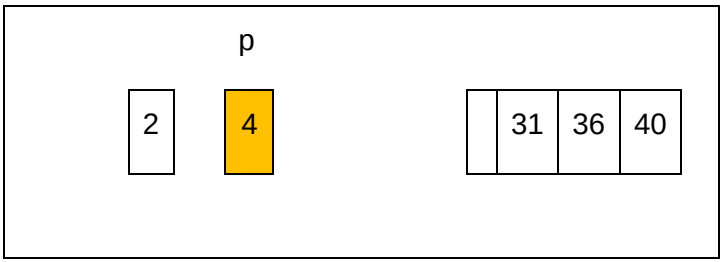
Tahap 13

Setelah memilih pivot, salah satu array yang sudah terurut nanti digabungkan dengan array baru setelah dibandingkan dengan pivot.



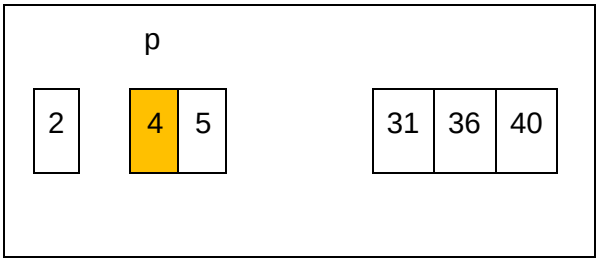
Tahap 14

Nilai array yang akan dibandingkan dengan pivot. 2 lebih kecil dari 4 maka diletakan didepan pivot.



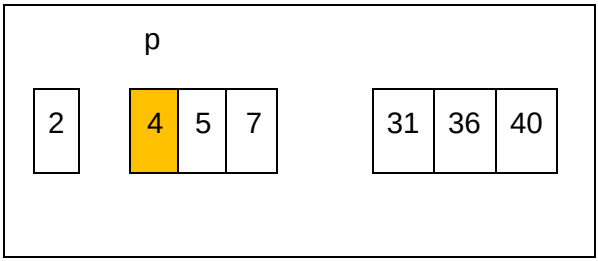
Tahap 15

Nilai array yang akan dibandingkan dengan pivot. 5 lebih besar dari 4 maka diletakan dibelakang pivot.



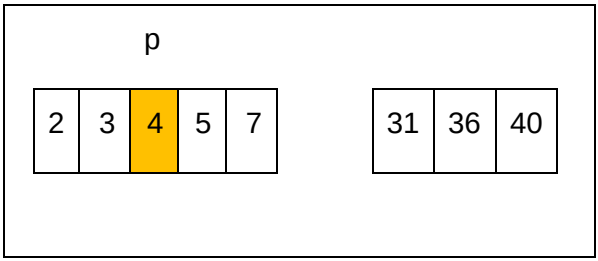
Tahap 16

Nilai array yang akan dibandingkan dengan pivot. 7 lebih besar dari 4 maka diletakan dibelakang pivot.



Tahap 17

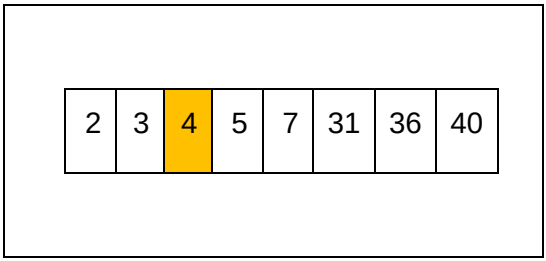
Nilai array yang akan dibandingkan dengan pivot. 3 lebih kecil dari 4 maka diletakan didepan pivot. Setelah semua array di bandingkan dengan pivot maka terbentuklah array baru.



Tahap 18

Setelah semua array di bandingkan dengan pivot maka terbentuklah array baru.

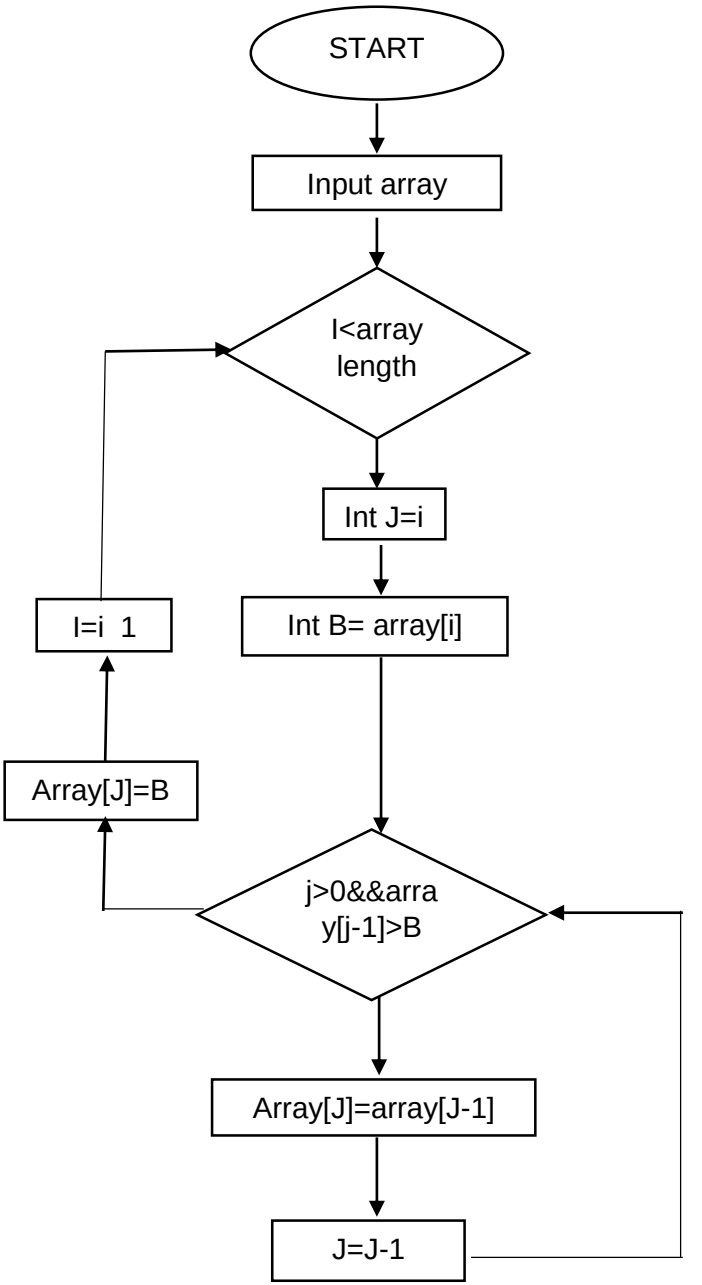
Selanjutnya digabung dengan array yang sudah terurut sebelumnya.



Hasil pengurutan descending dari quick sort.

2	3	4	5	7	31	36	40

6. Flowchart Quick Sort



7. Program C++ Quick Sort

```
#include <iostream>
#include <conio.h>
using namespace std;
void quick_sort(int arr[], int left, int right)
{
  int i = left, j = right;int tmp;
  int pivot = arr[(left+right)/2];/* partition */
  while (i<j){
    while (arr[i] < pivot)
    i++;
    while (arr[j] > pivot)
    j--;
    if (i<=j){
      tmp = arr[i];
      arr[i] = arr[j];
      arr[j] = tmp;
      i++;j--;
        };
  }; /* recursion */
  if (left < j)
    quick_sort(arr, left, j);
```

```
    if (i < right)
    quick_sort(arr, i, right);
}
int main()
{
  int i,n,data[50];
  cout<<"Masukan banyak data: ";cin>>n;
  for(i=0;i<n;i++)
  {
    cout<<"Masukan data ["<<i<<"] : ";cin>>data[i];
  }
  cout<<"\nData sebelum diurutkan: "<<endl;
  for(i=0;i<n;i++)
  {
    cout<<data[i]<<" ";
  }
  cout<<"\n";
  quick_sort(data,0,n-1);//hasil pengurutan
  cout<<"\nHasil pengurutan:\n";
  {
    int i;
    for (i=0;i<n;i++)
```

```
      cout<<data[i]<<" ";
      cout<<"\n";
   }
   getch();
}
```

C. LATIHAN

Buatlah pengurutan *merger* (*merger sort*) dalam mengurutkan jumlah dan ukuran sepatu yang di *dispay* (dengan jumlah dan ukuran sepatu silahkan tentukan sendiri).

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan Java*. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

BAB XIII

FILE

A. CAPAIAN PEMBELAJARAN

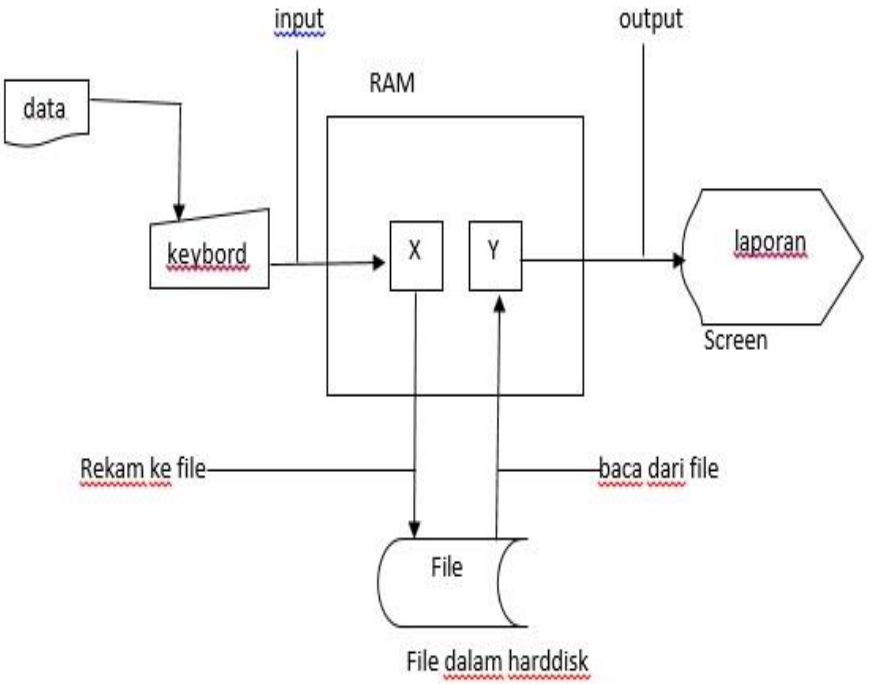
Mahasiswa mampu memahami konsep merekam ke file dan membaca dari file

B. MATERI

1. Pengertian File

File ialah tempat untuk merekam data yang terdapat pada suatu area di dalam *hardisk*.

a. Sesuatu yang di input dari keybord disebut dengan data.
b. Hasil akhir atau laporan yang dicetak dilayar.



Secara umum, komputer menangani tiga hal yang dinamakan masukan (*input*), proses, dan keluaran (*output*). Masukan adalah segala data yang diperlukan oleh komputer untuk diproses. Data bisa berupa bilang dan deretan karakter yang dimasukkan dari papan ketik, gambar dan video yang diambil dari kamera, suara yang ditangkap melalui mikrofon.

Proses adalah bagian yang mengolah masukan menjadi keluar. Proses ini ditangani oleh oleh program atau yang terkadang dinamakan perangkat lunak. Keluaran menyatakan hasil akhir pemrosesan oleh Komputer. Keluaran ini dapat diwujudkan dalam bentuk

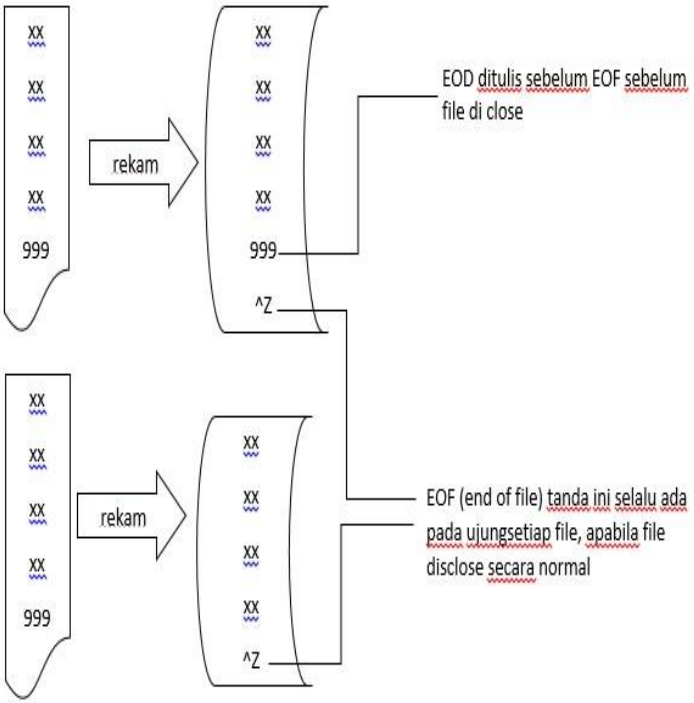
tampilan dilayar monitor, cetakan melalui printer suara pada pengeras suara, rekaman dalam media penyimpanan.

a. Data dan End of Data (EOD)

Proses perekaman selesai yaitu dengan end of data (EOD) yang merupakan bahwa file harus ada tanda dari setiap data yang direkam. Tanda tersebut bisa dibuat sendiri atau boleh dari komputer yang penting sudah dikenal oleh komputer,

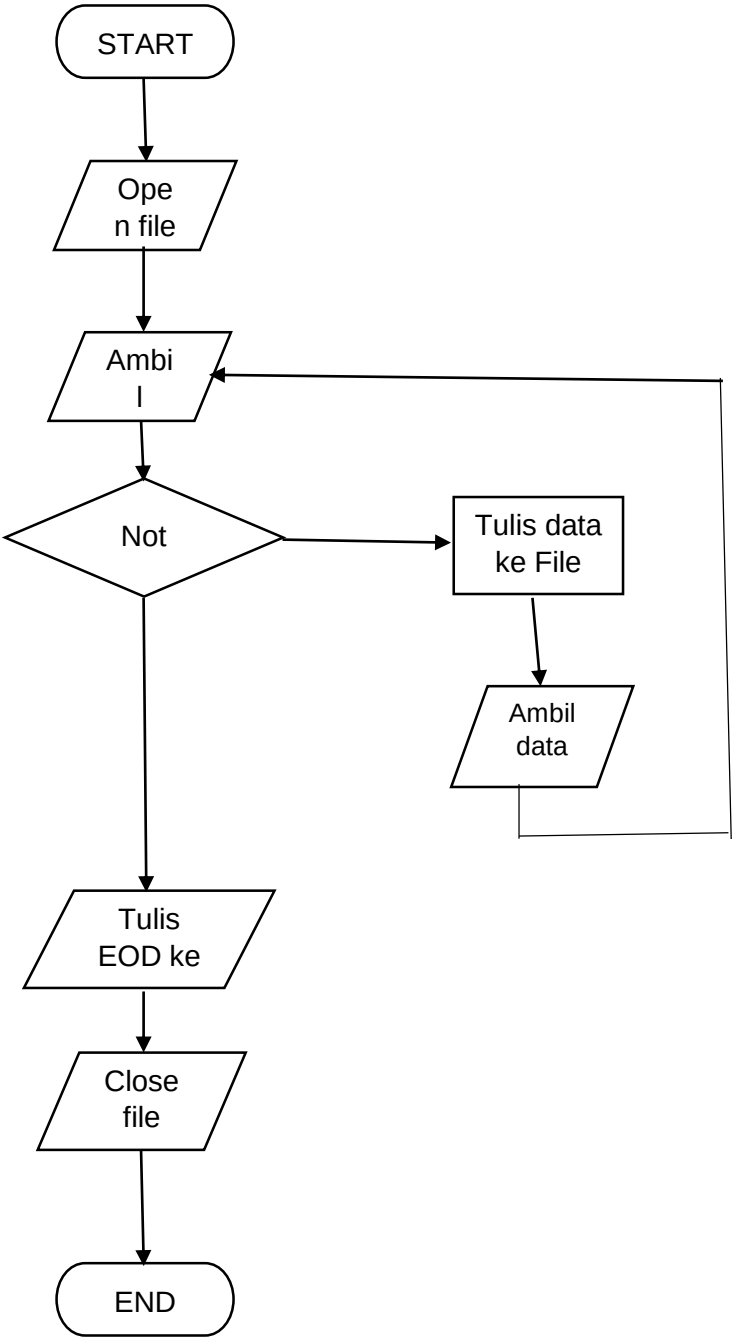
b. EOD dan EOF

Anda juga dapat mencatat EOD ke file yang anda inginkan. Namun jika file sudah memiliki batas end of file (EOF) ke file yang anda inginkan. EOD biasanya tidak disertakan pada file yang sudah memiliki end of file (EOF), sehingga EOD tidak ditulis pada file tersebut.

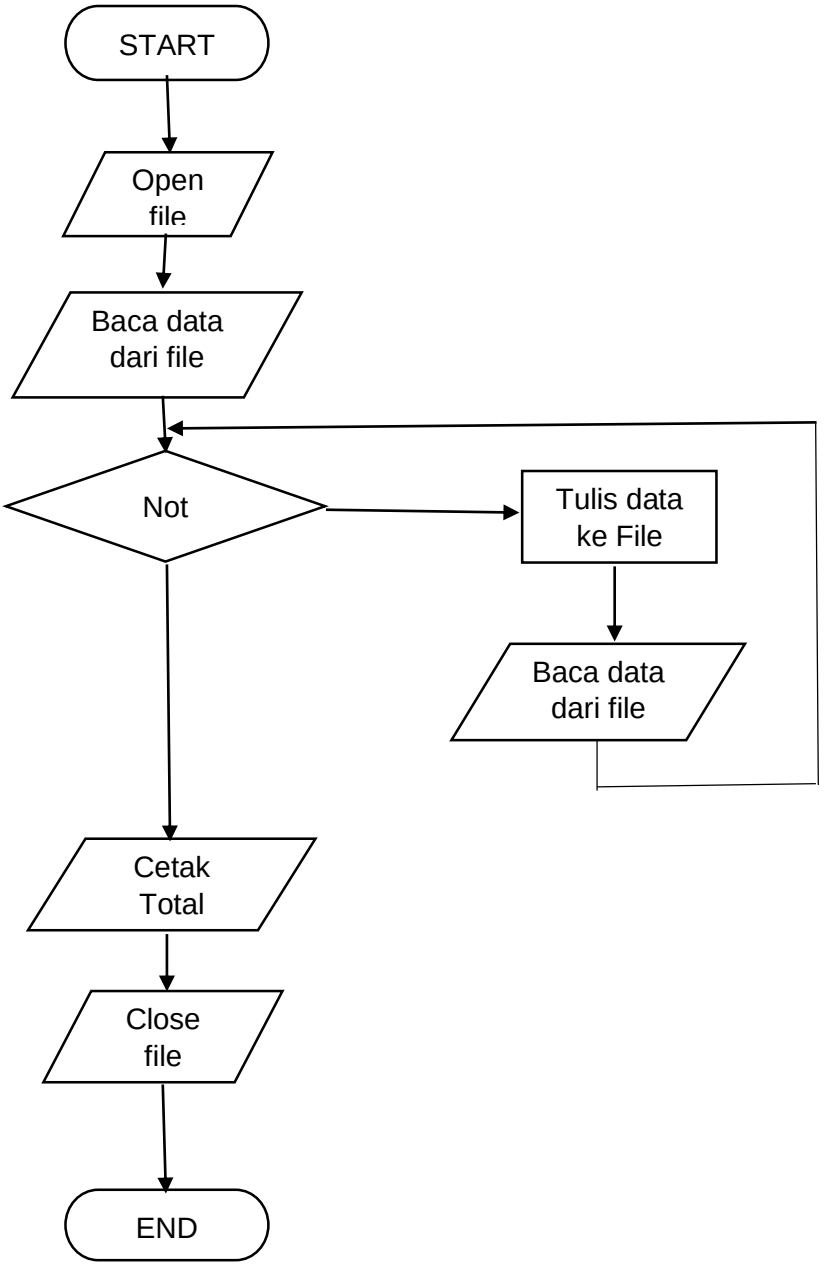


2. Merekam ke dan Membaca dari file pada C

a. Algoritma pokok untuk menulis data dari file.


START
Ope
n file
Ambi
l
Not
Tulis data
ke File
Ambil
data
Tulis
EOD ke
Close
file
END

b. Algoritma pokok untuk membaca data dari file.



3. Konsep Merekam ke dan membaca dari File pada C

Konsep Proses:

a. Buffer

   Untuk menghubungkan ke file, anda perlu mengatur buffer sebagai area perantara. Struktur buffer ini sudah digunakan dalam bahasa C dalam file <studio.h> selanjutnya kita cukup salin (copy) ke program yang sudah dibuat. File*PF. File adalah nama yang diberikan oleh bahasa c dan tidak boleh diganti dengan nama lain. Nama buffer yang dibuat seperti PF.

b. Buka (Open)

   Untuk melakukan apapun (menulis atau membaca) harus dibuka terlebih dahulu.

   Contoh untuk membuka file:

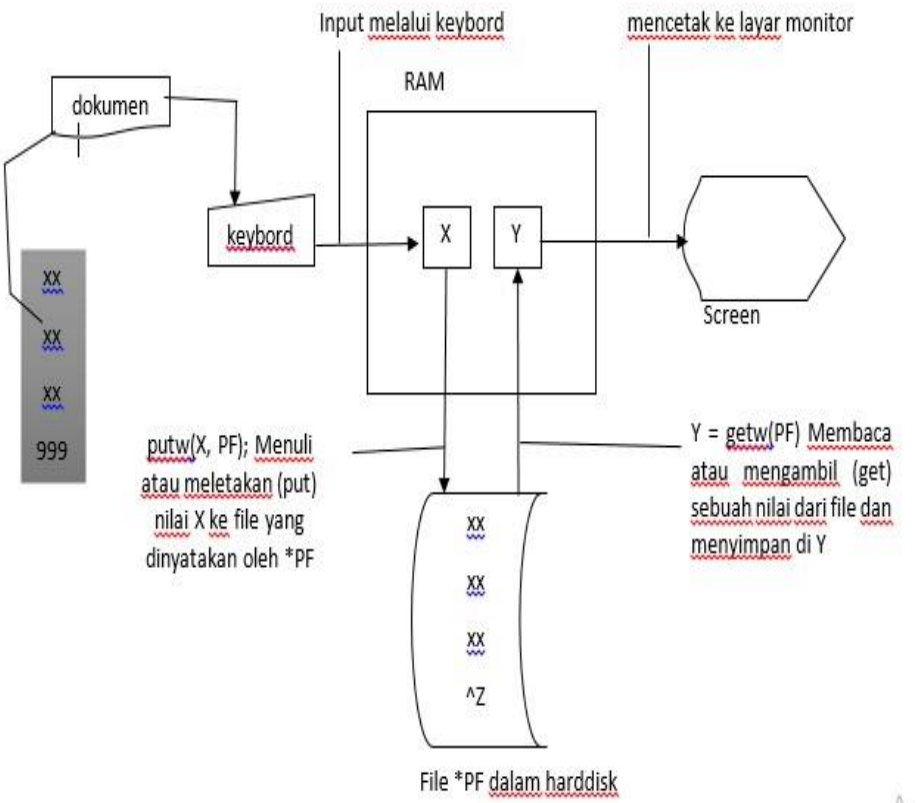
   PF = fopen(…);

c. Close

   Setelah selesai, file harus ditutup (close).

   Contoh untuk menutup file:

   fclose(PF);



4. Pengaksesan Arsip Beruntun

Perbandingan media penyimpanan data pada memori utama dan memori sekunder.

Arsip (file) : informasi yang tersimpan dalam memori sekunder.

Rekaman(record) : informasi yang direkam dalam arsip.

Pengertian record adalah data suatu tipe data terstruktur / bentukan yang terdiri dari beberapa elemen (field)

Misalkan data mahasiswa pada satu mata kuliah yang terdiri dari nim, nama, nilai

Nim	Nama	nilai
2021140120	amar	B
2021140121	Fitri	B
2021140122	Ari	B

Arsip beruntun adalah sekumpulan record dengan kolom-kolom data tertentu sesuai kebutuhan. Dalam arsip beluntun, file hanya berisi kumpulan catatan karena nama kolom tidak diikut sertakan dalam file.

Nim	Nama	nilai
2021140120	amar	B
2021140121	Fitri	B
2021140122	Ari	B

File tersebut memiliki sesuatu yang disebut end of file (EOF) pada akhir file. Secara umum, arsip harus

memiliki catatan untuk menunjukan kapan pembaca catatan dalam arsip beruntun telah mencapai catatan terakhir dari file tersebut. Misalkan xxxxx, xxx, x dari record yang berisi nim, nama, nilai.

5. Operasi Berkas

Arsip serial pada dasarnya adalah file yang dapat ditulis untuk menyimpan data, sehingga operasi yang digunakan dalam arsip beruntun pada dasarnya adalah operasi dalam file. Hanya saja memerlukan perlakuan khusus dalam menggunakan fungsi atau prosedur.

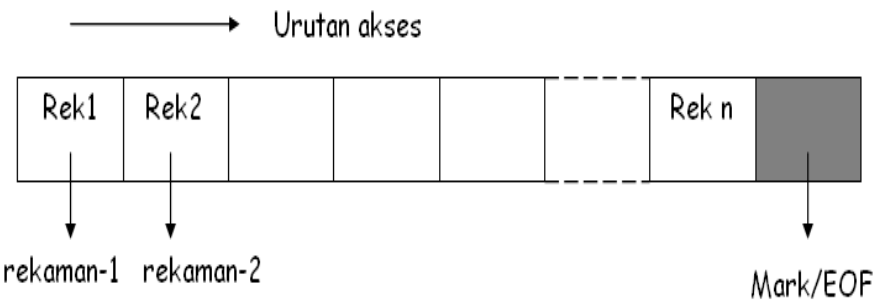
6. Operasi Pada Arsip Beruntun

a. Menyalin Arsip Beruntun
b. Mengabungkan Arsip Beruntun

Membuat arsip beruntun:

a. Tulis rekaman yang ingin anda tulis, ile arsip beruntun yang tidak kosong harus berisi setidaknya satu record dengan tipe nilai yang sama.
b. Menulis rekaman dummy diakhir pembacaan rekaman.

c. Jika file arsip beruntun dianggap sebagai file kosong, maka harus ada setidaknya satu dummy sebagai tanda akhir rekaman arsip beruntun. Tetapi jika arsip beruntun yang dibuat bukan arsip (file) kosong maka rekaman dummy akan berakhir, pada data rekaman.



FILE *arsip;

Sebelum kita memproses arsip, kita harus mendeklarasikannya terlebih dahulu bentuk umum dari deklarasi bahasa C.

Perintah standar yang dapat anda gunakan ialah:

fopen : menyiapkan arsip untuk menulis atau membacaan

fclose : menutup arsip

fscanf : membaca rekaman dari arsip

fprintf : menulis rekaman ke arsip

```
#include<iostream.h>
#include<stdio.h>
FILE *fp;
void main(void)
{
	int i=0,k;
	//Menyimpan data ke file
	fp=fopen("d:\\kampus\\program\\c++\\FileIO1\\data1.dat","w");
	if(fp==NULL)cout<<"Error membuka berkas"<<endl;
	else
	{
		for(i=0;i<=10;i++)fprintf(fp,"%d ",i);
		fclose(fp);
	}
	//Membaca data dari file
	fp=fopen("d:\\kampus\\program\\c++\\FileIO1\\data1.dat","r");
	if(fp==NULL)cout<<"Error membuka berkas"<<endl;
	else
	{
		while(!feof(fp))
		{
```

```
                    fscanf(fp,"%d ",&k);cout<<k<<" ";
                }
                cout<<endl;
                fclose(fp);
        }
    }
```

7. Pengaksesan File Secara Random

Rekaman dalam file diakses secara acak yang memiliki random tetap dan dapat diakses secara langsung tanpa melalui rekaman lain. Oleh karena itu, file yang diakses secara random cocok untuk digunakan dalam reservasi maskapai penerbangan. Sistem perbankan dan aplikasi lainnya yag memerlukan akses data cepat ke data tertentu. Karena panjang record tetap, posisi record relative terhadap awal file dapat dihitung sebagai fungsi dari kunci record.

```
FILE *arsip;
```

Sebelum kita memproses arsip, kita harus mendeklarasikannya terlebih dahulu bentuk umum dari deklarasi bahasa C.

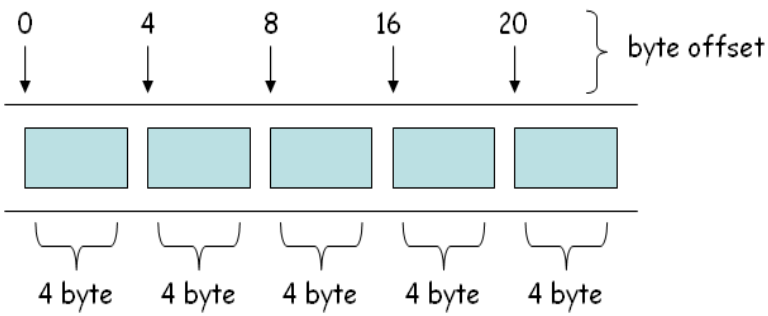
Perintah standar yang dapat anda gunakan ialah:

fopen : menyiapkan arsip untuk menulis atau membacaan

fclose : menutup arsip

fread : membaca rekaman dari arsip

fwrite : menulis rekaman ke arsip




```
//Penyimpanan & Pembacaan data secara random
#include<iostream.h>
#include<stdio.h>
FILE *fp;
void main(void)
{
        int i,j,jml_dat;
```

```
int k[100];
//Menulis data ke arsip
fp=fopen("d:\\kampus\\program\\c++\\FileIO3\\data1.dat","w");
if(fp==NULL)cout<<"Error membuka file"<<endl;
else
{
        for(i=0;i<=10;i++)fwrite(&i,sizeof(i),1,fp);
        fclose(fp);
}
//Membaca seluruh data data dari arsip
fp=fopen("d:\\kampus\\program\\c++\\FileIO3\\data1.dat","r");
if(fp==NULL)cout<<"Error membuka file"<<endl;
else
{
        i=0;
        while(!feof(fp))
        {
                fread(&k[i],sizeof(int),1,fp);
                i++;
        }
        fclose(fp);
```

```
}
//Mencetak selauruh data ke layar
jml_dat=i-1;
cout<<"Jumlah data = "<<jml_dat<<endl;
for(j=0;j<jml_dat;j++)cout<<k[j]<<" ";
cout<<endl;
fp=fopen("d:\\kampus\\program\\c++\\FileIO3\\data1.dat","r");
if(fp==NULL)cout<<"Error membuka file"<<endl;
else
{
        //Baca data ke-0
        fseek(fp,0*sizeof(j),SEEK_SET);
        fread(&j,sizeof(j),1,fp);
        cout<<"Data ke-0 = "<<j<<endl;
        //Baca data ke-1
        fseek(fp,1*sizeof(j),SEEK_SET);
        fread(&j,sizeof(j),1,fp);
        cout<<"Data ke-1 = "<<j<<endl;
        //Baca data ke-5
        fseek(fp,5*sizeof(j),SEEK_SET);
        fread(&j,sizeof(j),1,fp);
```

```
                cout<<"Data ke-5 = "<<j<<endl;
                //Baca data ke-0
                rewind(fp);
                fread(&j,sizeof(j),1,fp);
                cout<<"Data ke-0 = "<<j<<endl;
        }
}
```

C. LATIHAN

1. Susunlah program seperti dibawah ini:
   a. Menginput 3 buah nama kota dan merekam ke file “kota.txt”. misalkan:

      BANDUNG

      ACEH

      BANTEN

   b. Membaca file “kota.txt dan mencetak isinya sehingga tercetak:

      BANDUNG

      ACEH

      BANTEN

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan Java*. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

BAB XIV
STUDY KASUS

A. CAPAIAN PEMBELAJARAN

Mahasiswa dapat mengaplikasikan bahasan-bahasan yang telah dipelajari.

B. STUDY KASUS

1. Pointer

   Apa tampilan potongan program berikut? (Buatlah program lengkap untuk dapat menjalankan program-program berikut):

   Soal 1.

```
int *pa;
int a[10]={1,2,3,4,5,6,7,8,9,10};
*pa=a[0];
cout<<*pa++<<endl;
cout<<*pa<<endl;
```

Soal 2.

```
int *pa;
int a[10]={1,2,3,4,5,6,7,8,9,10};
*pa=a[0];
cout<<(*pa)++<<endl;
cout<<*pa<<endl;
```

Soal 3.

```
int *pa;
int a[10]={1,2,3,4,5,6,7,8,9,10};
*pa=a[0];
cout<<*++pa<<endl;
cout<<*pa<<endl;
```

Soal 4.

```
int *pa;
int a[10]={1,2,3,4,5,6,7,8,9,10};
*pa=a[0];
cout<<++*pa<<endl;
cout<<*pa<<endl;
```

2. Pointer dan Larik 2 Dimensi

   Soal 1

Tapilkan output program berikut:

```
1  #include<iostream>
2  #include<iomanip>
3  using namespace std;
4  int main()
5  {
6    // mendefinisikan string char
7  kalimat[]={"Selamat Datang Di Universitas
   Pamulang"};
8    char *pKarakter; // pointer ke char
9    int LowerCase=0; // penghitung huruf kecil
10   int UpperCase=0; // penghitung huruf besar
11   pKarakter=kalimat;  //pointer diisi alamat string
     while(*pKarakter) {   // looping
12     char kar=*pKarakter;        // huruf di posisi ini
13     if(kar>='a' && kar<='z')     // cek huruf kecil
14       LowerCase++; //increment penghitung
       if(kar>='A' && kar<='Z')     // cek huruf besar
15       UpperCase++; //increment penghitung
       pKarakter++; // maju satu posisi
16   }
```

```
17    // tampilkan hasil
18    cout<<"Jumlah Huruf Kecil =
"<<LowerCase<<endl;
19    cout<<"Jumlah Huruf Besar =
20  "<<UpperCase<<endl;
21  }
22
23
```

Soal 2

Ubahlah contoh program pada soal 1 di atas dengan menghitung

- jumlah angka dan
- jumlah karakter selain angka dan huruf.

3. Pointer dan Larik 3 Dimensi

   Soal

   Tuliskan kode program untuk menampilkan data berikut menggunakan pointer dan larik 3 Dimensi

NIM	N1	N2	N3	N4
1901501234	84	65	94	61
1901501235	65	49	83	72
1901501236	49	85	76	82

4. Prosedur

   Soal

   Buatlah program Menetukan Bilangan Ganjil atau Genap menggunakan Prosedur.

5. Fungsi

   Soal

   Buatlah program Menetukan Bilangan nilai terbesar atau nilai terkecil dari sekumpulan data dalam larik menggunakan Fungsi yang mengembalikan nilai.

6. Sequential search

   Soal

   Diketahui data[]={22,45,32,11,20,55,70} dan data yang dicari adalah 22. Tuliskan proses paencarian data tersebut menggunakan metode *Sequential search*.

7. *Binary Serach*

   Soal

   Diketahui data[]={100,30,23,10,9,7} dan data yang dicari adalah 11 Tuliskan langkah paencarian data tersebut menggunakan metode pencarian bagi dua(*binary search*).

8. Bubble Sort

   Soal

Diketahui data[]={10,35,28,15,40,5,8}

Tuliskan langkah-langkah proses sorting dengan data terurut secara menaik dengan metode bubble sort.

9. Selection Sort

   Soal

   Diketahui data[]={10,35,28,15,40,5,8}

   Tuliskan langkah-langkah proses sorting dengan data terurut secara menaik dengan metode seleksi minimum.

10. Insertion Sort

    Soal

    Diketahui data[]={20,25,38,5,42,15,80,65}

    Tuliskan langkah-langkah proses sorting dengan data terurut secara menaik dengan metode pengurutan sisip (Insertion Sort).

11. Shell Sort

    Soal

    Diketahui data[]={20,25,38,5,42,15,80,65}

    Tuliskan langkah-langkah proses sorting dengan data terurut secara menurun dengan metode pengurutan shell (Shell Sort).

12. Merger Sort

    Soal

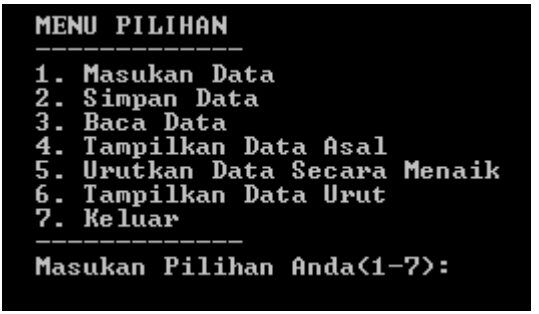
    Diketahui data[]={20,25,38,5,42,15,80,65}

Tuliskan langkah-langkah proses sorting dengan data terurut secara menurun dengan metode pengurutan merger (Merger Sort).

13. Pengurutan dan Akses berkas

Soal

Buatlah program C/C++ yang memberikan tampilan sbb :



Ket :

Pilihan 1 : Memasukan sejumlah data dari keyboard

Pilihan 2 : Menyimpan seluruh data kedalam sebuah arsip

Pilihan 3 : Membaca data dari arsip

Pilihan 4 : Menampilkan data

Pilihan 5 : Mengurutkan data

Pilihan 6 : Menampilkan data yang telah diurutkan

Pilihan 7 : Keluar Program

C. LATIHAN SAOL

1. Diketahui data[]={42,75,28,35,47,21,8,25}

   Tuliskan langkah-langkah proses sorting dengan data terurut secara menaik dengan metode pengurutan Quick (Quick Sort).

2. Apa tampilan potongan program berikut? (Buatlah program lengkap untuk dapat menjalankan program-program berikut):

```
int *pa;
int a[10]={11,21,34,14,65,61,27,48,91,10};
pa=a;
cout<<*pa<<endl;
pa++;
cout<<*pa<<endl;
```

D. REFERENSI

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan Java*. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.

DAFTAR PUSTAKA

Kirch-prinz U, Prinz P. *A Complete Guide to Programming in C++*. 1st ed. Sudbury, MA: Jones And Bartlett Publishers; 2002.

Arndt J. *Algorithms for Programmers*.; 2008.

Sjukani M. *Algoritma Dan Struktur Data 1 Dengan C, C++ Dan Java*. Jakarta, Indonesia: Mitra Wacana Media; 2014.

Edition E. *EIGHTH EDITION with an Introduction to C++ HOW TO PROGRAM*.

A.S. R. *Logika Algoritma Dan Pemrograman Dasar*. Bandung, Indonesia: Mandalajati; 2018.

Davis SR. *C++ for Dummies*. 7th Editio. New Jersey, USA: John Wiley & Sons, Inc; 2014.