Modul Praktikum

Algoritma dan Pemrograman

Program Studi Teknik Informatika
STMIK STIKOM Indonesia

DAFTAR ISI

MODUL 1
PENGENALAN PROGRAM C
(Pertemuan 1)

Tujuan :

1. Mengenal bahasa pemrograman dengan bahasa C
2. Memahami sistematika penulisan bahasa C
3. Mampu menulisakan bahasa pemrograman C dengan DevC++
4. Mampu membangun program sederhana di komputer.

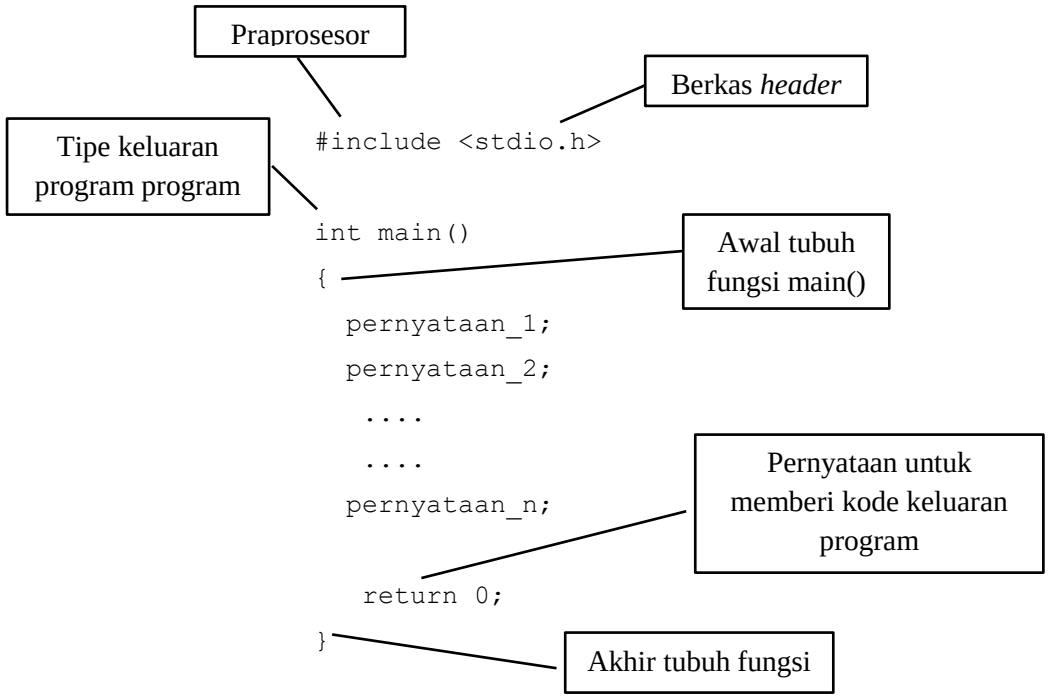
Tugas Pendahuluan :

Buatlah sebuah program sederhana yang dapat memasukkan dan menampilkan data pada C!

DASAR TEORI

Selamat datang di dunia pemrograman, modul ini akan memplajari bahasa C. Bahasa C merupakan bahasa komputer yang sangat singkat. Namanya tidak memiliki kepanjangan apapun, bahasa ini diciptakan oleh Dennis Ritchie sekitar tahun 1972. Hingga kini bahasa ini masih populer diberbagai platform; dari Windows sampai linux dan dari PC hingga mainframe.

Berikut struktur penulisan program dengan bahasa C



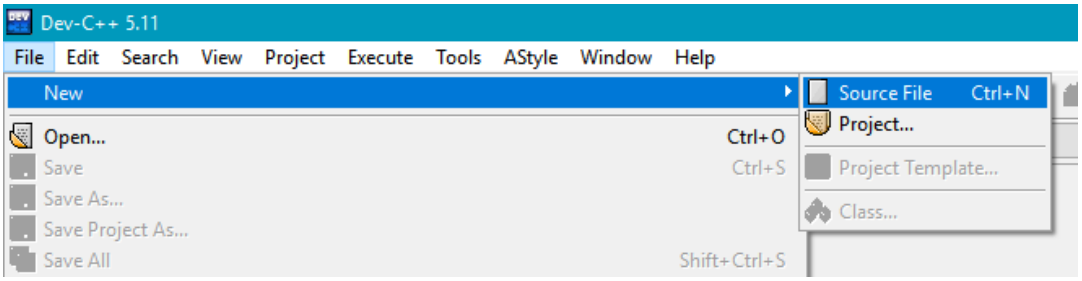
- Berkas *header* (berekstensi .h) adalah berkas yang berisi prototipe fungsi, definisi konstanta, dan defenisi variabel.

- Praprosesor **#include** adalah suatu perintah yang digunakan untuk mengatur kompiler agar membaca berkas *header* yang disertakan di belakang kata **include** saat pelaksanaan kompilasi.
- **main**() adalah fungsi yang akan dijalankan pertama kali ketika program dieksekusi. Kata **int** di depan **main**() menyatakan bahwa program memberikan nilai balik berupa integer (dibahas lebih lanjut dimodul 11)
- nilai balik program ditentukan oleh pernyataan **return**. Pada program di depan, **return 0** menyatakan bahwa nilai balik program adalah nol, nilai nol biasa digunakan untuk menyatakan bahwa program berhasil melaksanakan tugas yang dibebankan.
- Setiap pernyataan ditulis dengan akhiran tanda titik-koma (;)

KEGIATAN PRAKTIKUM

Langkah-langkah praktikum

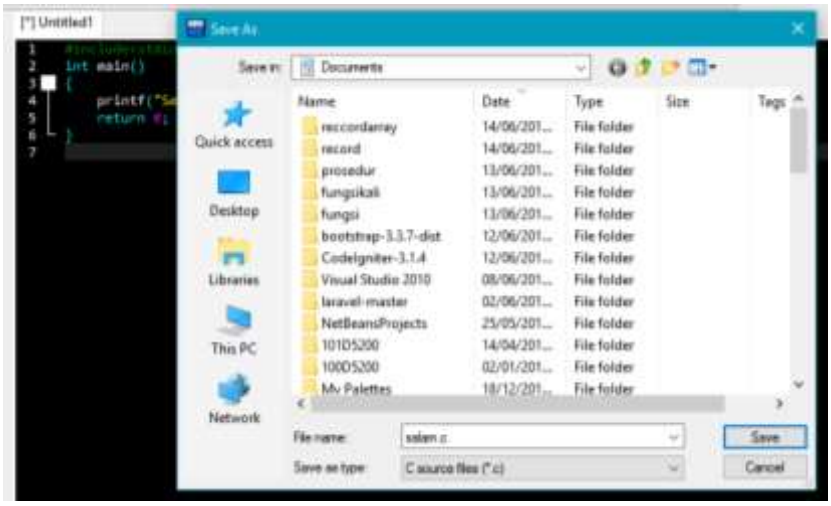
1. Buka editor bahasa C **DevC++ 5.11**
2. Buatlah file baru dengan membuka menu **file** > **new** > **source file** atau dengan *shortcut* **ctrl + N**



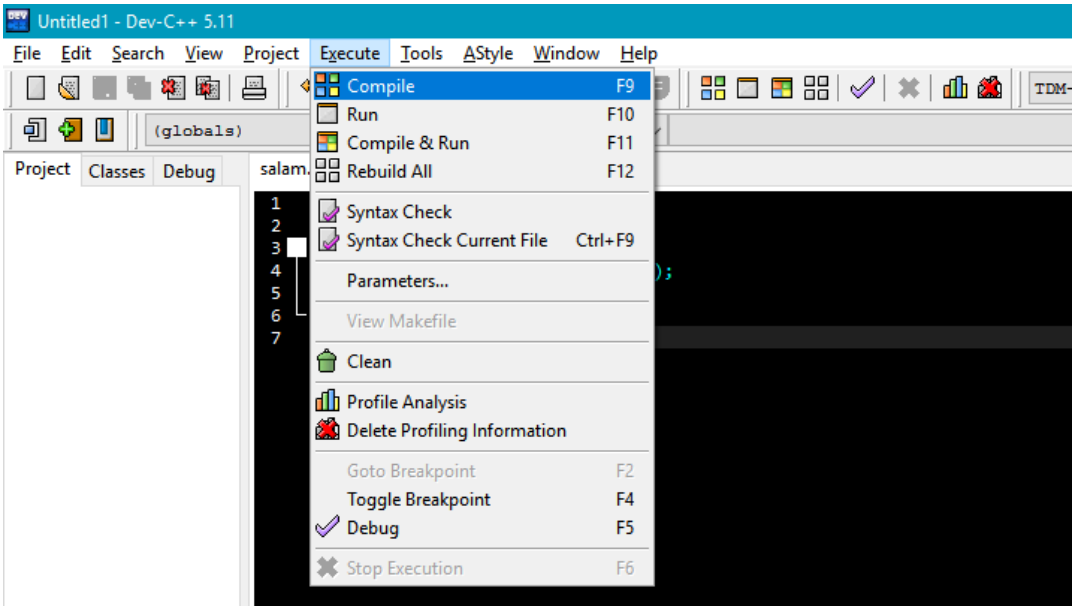
3. Tuliskan *coding* berikut

```
#include<stdio.h>
int main()
{
 printf("Selamat Belajar C\n");
 return 0;
}
```

4. Simpan *coding* yang telah dituliskan dengan membuka menu **file** > **save as**... pilih lokasi penyimpanannya dan beri nama file dengan "**salam.c**"

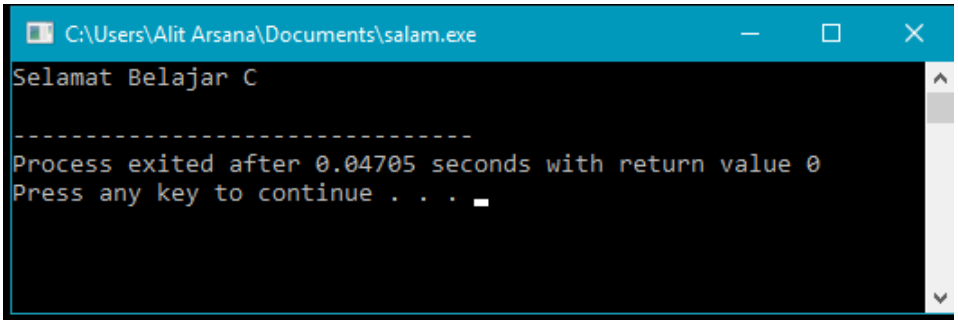


5. Lakukan kompilasi program melalui menu **Execute** > **Compile** atau dengan menekan *shortcut* **F9**.



6. Perhatikan hasil kompilasi program pada bagian "Compile Log". Jika kompilasi sukses, maka akan ditampilkan pesan kurang lebih sebagai berikut:

7. Jalankan program (*running*) melalui menu **Execute** > **Run** atau dengan menekan *shortcut* **F10**. Hasil program akan ditampilkan pada window seperti pada gambar berikut ini.



8. Proses kompilasi dan running dapat dilaksanakan sekaligus melalu menu **Execute** > **Compile & Run** atau dengan *shortcut* **F11**.
9. Selesai.

LATIHAN

1. Salinlah *coding* berikut **:**

```
#include<stdio.h>
int main()
{
   printf("%i",20+30);
   return 0;
}
```

Sekarang simpan program tersebut dengan memilih menu *File* lalu pilih *Save*, simpan dengan nama **total1.c**. Kompilasi program tersebut dengan menekan tombol *F9* dan jalankan program tersebut dengan menekan *f10*.

2. Salinlah coding berikut :

```
#include<stdio.h>
int main()
{
    int A,B;
    A=20;
    B=30;
   printf("%i",A+B);
   return 0;
}
```

Sekarang simpan program tersebut dengan memilih menu *File* lalu pilih *Save*, simpan dengan nama **total2.c**. Kompilasi program tersebut dengan menekan tombol *F9* dan jalankan program tersebut dengan menekan *f10*, kemudian amati hasilnya. Jelaskan perbedaan antara program total1 dengan total2!

TUGAS

1. Buatlah sebuah program yang dapat menampilkan nama NIM, Nama dan Jurusan Anda dilayar tanpan inputan melalui *keyboard*!
2. Buatlah sebuah program yang menghitung luas segitiga dengaan ukuran alas 3 dan tinggi 7!

MODUL 2
JENIS-JENIS DATA DAN VARIABEL
(Pertemuan 2)

Tujuan :

1. Mampu memahami tipe data dasar.
2. Mempraktekkan deklarasi data dalam bahasa pemrograman,
3. Mempraktekkan perintah *input* dan *output* dalam bahasa pemrograman.

Tugas Pendahuluan :

1. Apa yang anda ketahui tentang pemrograman dasar? Jelaskan!
2. Sebutkan tipe data apa saja yang terdapat pada pemrograman dengan C!

DASAR TEORI

Dalam bahasa C terdapat beberapa jenis tipe data yang bisa digunakan untuk sebuah variabel atau konstanta pada program. C menyediakan lima macam tipe data dasar, yaitu tipe data **integer** (nilai numerik bulat yang dideklarasikan dengan **int**), **floating-point** (nilai numerik pecahan ketetapan tunggal yang dideklarasikan dengan **float**), **double-precision** (nilai numerik pecahan ketetapan ganda yang dideklarasikan dengan **double**), **karakter** (dideklarasikan dengan **char**), **Boolean** (merupakan tipe data yang berisi nilai dengan kemungkinan hanya berupa *False* (nilai salah) dan *True* (nilai benar) yang dideklarasikan dengan **bool**). Berikut ini tabel tipe data yang ada dalam bahasa C beserta ukuran dan jangkauannya:

Tipe	Lebar	Jangkauan Nilai	
		Dari	Sampai Dengan
int	16 bit	-32768	32767
unsigned int	16 bit	0	65535
short int	16 bit	-32768	32767
long int	32 bit	-2147483648	2147483649
unsigned long int	32 bit	0	4294967295
float	32 bit	3.4E-38	3.4E+38
double	64 bit	1.7E-308	1.7E+308
long double	80 bit	3.4E-4932	3.4E+4932
char	8 bit	-128	128
unsigned char	8 bit	0	255

Variabel adalah suatu pengenal yang digunakan untuk mewakili suatu nilai tertentu didalam proses program. Variabel biasa digunakan di dalam program dengan tujuan untuk menampung data. Nilai yang terdapat pada variabel sewaktu-waktu dapat diubah. Jumlah variabel yang dibuat dapat tidak terbatas, namun masing-masing variabel tersebut harus bersifat unik dan tidak boleh ada nama variabel yang sama. Selain variabel terdapat konstanta yang juga dapat menampung data. Hanya saja dalam kostanta nilai yang ada tidak dapat diubah atau bernilai pasti. Dalam program, variabel dideklarasikan dengan **tipe_data** dan **nama_variabel**, sedangkan konstanta dideklarasikan dengan menggunakan preprocessor define **#define nama_konstanta** atau dengan dengan singkatan **const tipe_data** dan **nama_konstanta**.

Bentuk Umum :

```
konstanta
    #define nama_konstanta
atau
    const tipe_data nama_konstanta;
variabel
    tipe_data nama_variabel;
```

Adapun aturan-aturan penamaan variabel dalam bahasa C adalah sebagai berikut :

1. Tidak boleh mengandung spasi, simbol, atau tanda.
2. Tidak boleh diawali dengan angka.

Contoh :

```
Benar
    char nama[25];
    int usia;
    char kelas;

Salah :
    char @nama[25];
    int us!a;
```

Sebuah program dapat menampilkan kalimat ke layar. Hal ini biasanya dilakukan untuk menampilkan perintah untuk memasukkan masukan program kepada pemakai program (*user*). Berikut adalah cara untuk menampilkan kalimat ke layar :

```
printf("string");


Misal :
Printf("Hello World");
```

KEGIATAN PRAKTIKUM

Membuat Program Biodata

Salinlah *coding* program berikut ini ke dalam DevC++ :

```
#include<stdio.h>
#include<conio.h>

int main()
{
      char nama[25];
      int umur;
      char alamat[50];

      printf("Masukkan nama anda   : ");gets(nama);
      printf("Masukkan alamat anda : ");gets(alamat);
      printf("Masukkan umur anda   : ");scanf("%d",&umur);

      printf("\n Nama anda   : %s",nama);
      printf("\n Alamat anda : %s",alamat);
      printf("\n Umur anda   : %d",umur);


      return 0;
}
```

Sekarang simpan program tersebut dengan memilih menu *File* lalu pilih *Save*, simpan dengan nama **PRAK01.c**. Kompilasi program tersebut dengan menekan tombol *F9* dan jalankan program tersebut dengan menekan *f10*, kemudian amati hasilnya bila dimasukkan masukan tertentu.

Membuat Program Hitung Luas Lingkaran

Salinlah *coding* program berikut ini ke dalam DevC++ :

```
#include<stdio.h>
#include<conio.h>
#define phi 3.14

int main()
{
   float luas,jari2;

   printf("masukkan jari-jari lingkaran:");scanf("%f",&jari2);

   luas=phi*(jari2*jari2);
   printf("Luas lingkaran : %f",luas);

   return 0;
}
```

Simpan pekerjaan anda dengan nama **PRAK02.c**. Kompilasi dan jalankan program tersebut, kemudian amati hasilnya bila dimasukkan masukan tertentu. Amati kembali, apa yang

terjadi bila nilai jari-jari sama dengan nol? Apa pula yang terjadi jika nilai jari-jari diisi dengan huruf?

TUGAS

1. Buatlah program untuk menentukan hasil penjumlahan dan pengurangan dari 2 bilangan bulat!
2. Buatlah program yang menerima sebuah masukan dan menampilkan hasil kuadrat dari bilangan masukan!
3. Buatlah program yang menerima 2 buah masukan bilangan bulat untuk menghitung keliling persegi panjang!

MODUL 3
MENGENAL OPERATOR
(Pertemuan 3)

Tujuan :

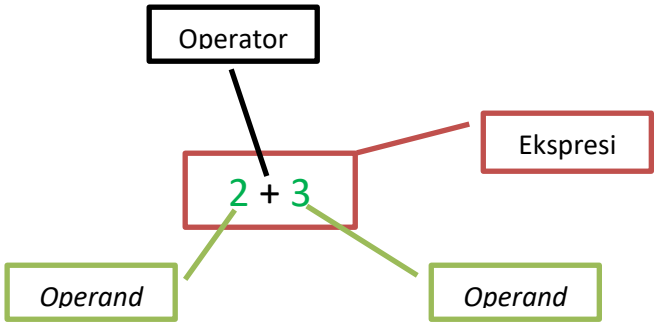
1. Mempraktekkan jenis-jenis Operator Logika dalam bahasa pemrograman C
2. Mempraktekkan jenis-jenis Ekspresi Logika dalam bahasa pemrograman C
3. Mempraktekkan Struktur Runtutan dalam bahasa pemrograman C

Tugas Pendahuluan :

1. Apa yang anda ketahui mengenai operator logika? Sebutkan!
2. Buatlah program dalam bahasa C dengan menggunakan ekspresi logika!
3. Buatlah program dalam bahasa C dengan struktur runtutan yang benar!

DASAR TEORI

Operator adalah simbol atau tanda yang jika diletakkan pada dua buah operand dapat menghasilkan sebuah hasil, operator berupa simbol yang digunakan untuk menyusun suatu ekspresi dengan melibatkan satu atau beberapa *operand*. Contohnya pada matematika dimana tanda tambah (‘+’) jika diletakkan di antara dua buah angka akan menghasilkan angka lain hasil pertambahan dari dua angka tersebut. Tanda tambah inilah yang disebut dengan operator.



Ditinjau dari jumlah *operand* yang dilibatkan dalam sebuah operator, terdapat 3 macam operator.

1. **Operator unary**, yaitu operator yang hanya melibatkan sebuah *operand.*

   Contoh :

   +1

   -1

2. **Operator binary**, yaitu operator yang melibatkan dua buah *operand.*

   Contoh :

   2*3

   5+2

3. **Operator tertiary**, yaitu operator yang melibatkan tiga buah *operand.*

   Contoh :

   a > b ? 1 : 0

   Ekspresi di atas berarti "jika a kebih besar dari pada b maka ekspresi menghasilkan nilai 1, sedangkan kalau tidak makas ekspresi menghasilkan 0".

Berdasarkan kelompok kegunaan, operator dapat dibagi menjadi operator aritmatika, operator pembanding, dan operator logika.

1. **Operator aritmatika**, operator aritmatika digunakan untuk melakukan perhitungan aritmatika. Daftar operator aritmatika dapat dilihat pada tabel berikut :

Operator	Prioritas	Keterangan	contoh
-	1	Unary minus	-1
+	1	Unary plus	+1
*	2	Perkalian	5*2
/	2	Pembagian	6/2
%	2	Sisa pembagian	8%2
+	3	Penjumlahan	3+2
-	3	Pengurangan	5-2

2. **Operator pembanding**, operator ini juga disebut operator relasional yang digunakan untuk melakukan perbandingan terhadap dua buah nilai. Hasil perbandingan bernilai 0 dan 1. Dalam hal ini :

   - Nilai nol berarti bahwa perbandingan memberikan hasil bernilai salah.
   - Nilai satu berarti bahwa perbandingan memberikan hasil bernilai benar.

   Daftar operator pembanding dapat dilihat pada tabel berikut :

Operator	Keterangan
>	Lebih dari
>=	Lebih dari atau sama dengan
<	Kurang dari
<=	Kurang dari atau sama dengan
!=	Tidak sama dengan
==	Sama dengan

3. **Operator logika**, operator logika digunakan membentuk suatu ekspresi pembandingan dari satu atau dua buah ekspresi pembandingan. Operator logika yang tersedia pada C dapat dilihat pada tabel berikut :

Operator	Keterangan
&&	Operator "dan"
||	Operator "atau"
!	Operator "bukan"

Operator && dan || melibatkan dua buah *operand*, sedangkan operator ! melibatkan sebuah *operand.*

KEGIATAN PRAKTIKUM

Menggunakan Operator Pembanding, Logika (OR) dan Operator Tertiary

Salinlah *coding* program berikut ini ke dalam DevC++ :

```
#include <stdio.h>
#include <conio.h>
#include <string.h>

int main()
{

	char nikah;
	char nama[10];
	int golongan;

	printf("\nMasukkan Nama Pegawai  : ");gets(nama);
	printf("Golongan Pegawai (1/2/3) : ");scanf("%d",&golongan);

	printf("-----------------------------------------\n");
	printf("Nama      : %s",nama);
	printf("\nGolongan  : %d",golongan);


	int tGolongan = (golongan == 1 ? 2000000 :(golongan == 2 ?
	2750000 : 3500000));
	printf("\nGaji Pokok     : Rp %d",tGolongan);

	int bonus = (golongan == 1 ? 150000 :(golongan == 2 ? 175000 :
	200000));
	printf("\nBonus          : Rp %d",bonus);

	printf("\nTotal Gaji     : Rp %d",tGolongan+bonus);

	return 0;

}
```

Simpan pekerjaan anda dengan nama **PRAK03.c**. Kompilasi dan jalankan program tersebut, kemudian amati hasilnya.

Membuat Program dengan Ekspresi Aritmatika

Salinlah *coding* program berikut ini ke dalam DevC++ :

```
#include<stdio.h>

int main()
{
	float x;

	x=1+2*3-4/2;

	printf("%.2f",x);
	return 0;

}
```

Simpan pekerjaan anda dengan nama **PRAK05.c** Kompilasi dan jalankan program tersebut, kemudian amati hasilnya. Berapakah hasilnya? Mengapa angka dibelakang koma dapat berjumlah dua buah? Cobalah mengganti angka yang akan dijumlahkan. Bagaimana hasilnya?

TUGAS

1. Buatlah algoritma dan program yang menerima masukan alas dan tinggi sebuah segitiga, dan akan mengeluarkan nilai luas segitiga tersebut!
2. Buatlah program yang menerima masukan Nama, Alamat, Tahun Lahir, dan Tahun Sekarang yangakan menampilkan Jumlah Usia dari tahun yang dimasukkan!

MODUL 4
TEKNIK PERCABANGAN (BAGIAN 1)
(Pertemuan 4)

Tujuan :

1. Memahami tentang pembacaan data secara percabangan dalam bahasa pemrograman
2. Mempraktekkan perintah-perintah percabangan dalam bahasa pemrograman

Tugas Pendahuluan :

1. Apa yang anda ketahui tentang percabangan dalam bahasa C? Jelaskan!
2. Mengapa percabangan begitu dibutuhkan pada bahasa pemrograman?
3. Buatlah program pernyataan *if* yangmenampilkan grade A untuk nilai di atas 7!

DASAR TEORI

Percabangan *if* merupakan sebuah blok program yang menyatakan bahwa sebuah aksi akan dijalankan jika kondisi percabangan dipenuhi jika tidak dipenuhi maka aksi tidak akan dijalankan. Percabangan *if* biasa digunakan untuk mengerjakan aksi yang memiliki syarat tertentu untuk menjalankannya. Pernyataan *if* diklasifikasikan lagi ke dalam tiga bagian, yaitu :

1. Pernyataan *if* dengan satu kondisi (*if* tunggal)
2. Pernyataan *if* dengan dua kondisi (*if - else*)
3. Pernyataan *if* bersarang (*if* di dalam *if*)

Pernyataan *If* tunggal hanya melibatkan satu kondisi yang akan diperiksa. Apabila kondisi yang diperiksa bernilai benar, maka program akan mengeksekusi bagian yang berada dalam blok. Jika sebaliknya, maka program akan mengabaikan pernyataan di dalam blok dan langsung melanjutkan eksekusi berikutnya.

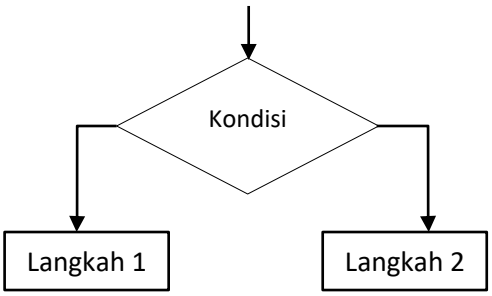
Bentuk Umum :

```
{terdiri dari satu statemen}
if (kondisi)
  pernyataan;

{terdiri atas beberapa statemen}
if (kondisi)
  {
    pernyataan1;
    pernyataan2;
    ...
  }
```

Pernyataan *if* dengan dua kondisi (*if - else*) dipergunakan untuk menyatakan pernyataan percabangan dua kondisi dimana ada dua blok aksi yang dipilih untuk dikerjakan jika syarat kondisi aksi terpenuhi. Saat pembacaan program sampai pada blok *if-else* maka akan dilakukan pemeriksaan terhadap syarat kondisi percabangan yang ada pada deklarasi *if*, jika syarat dipenuhi maka yang akan dijalankan adalah aksi yang ada di dalam blok *if*, tapi jika syarat tidak dipenuhi maka aksi yang dikerjakan adalah yang ada di dalam blok *else*.

Bentuk Umum :

```
if (kondisi)
  pernyataan1;
else
  pernyataan2;

atau

if (kondisi)
  {
    pernyataan1a;
    pernyataan1b;
    ...
  }
  else
  {
    pernyataan2a;
    pernyataan2b;
    ...
  }
```



KEGIATAN PRAKTIKUM

Penentuan Nilai Menggunakan Percabangan Sederhana

Salinlah *coding* program berikut ini ke dalam DevC++ :

```
#include<stdio.h>
#include<conio.h>

int main()
{
     int skor;
     char nilai;

     printf("masukkan skor : ");scanf("%d",&skor);

     if(skor>7)
      nilai='A';

     printf("Nilai : %c",nilai);

     return 0;
}
```

Simpan dengan nama **PRAK06.c**. Kompilasi program tersebut dengan menekan *F9* dan jalankan program tersebut dengan menekan *F10*, kemudian amati hasilnya bila dimasukkan masukan skor yang berbeda.

Penentuan Nilai dengan Banyak Pernyataan

Salinlah *coding* program berikut ini ke dalam DevC++ :

```
#include<stdio.h>
#include<conio.h>

int main()
{
     int skor;
     char nilai;
     char lulus;
     int bonus;

     printf("masukkan skor : ");scanf("%d",&skor);

     if(skor>7)
     {
      nilai='A';
      lulus='L';
      bonus=50000;
     }

      printf("Nilai : %c\n",nilai);
      printf("Lulus : %c\n",lulus);
      printf("Bonus : %d\n",bonus);

     return 0;
}
```

Simpan pekerjaan anda dengan nama **PRAK07.c.** Kompilasi dan jalankan program tersebut, kemudian amati hasilnya dan pahami logikanya.

Menentukan Kubus dan Bukan Kubus dengan *If*

Salinlah *coding* program berikut ini ke dalam DevC++ :

```
#include<stdio.h>
#include<conio.h>

int main()
{
     int s1,s2,s3;

     printf("Masukkan sisi pertama :");scanf("%d",&s1);
     printf("Masukkan sisi kedua   :");scanf("%d",&s2);
     printf("Masukkan sisi ketiga  :");scanf("%d",&s3);

     if(s1==s2 && s2==s3)
      {
            printf("\nTermasuk Kubus");
      }
      else
      {
            printf("\nBukan Kubus");
      }

     return 0;
```

Sekarang simpan program tersebut dengan nama **PRAK08.c**. Kompilasi program tersebut dan jalankan, kemudian amati hasilnya bila dimasukkan tiga buah bilangan sisi. Program tersebut menggunakan percabangan *if* dengan dua kondisi.

TUGAS

1. Buatlah program yang menerima dua masukan bilangan yang memiliki syarat bilangan pertama tidak boleh lebih kecil dari 3 dan bilangan kedua tidak boleh lebih kecil dari 4. Jika syarat dipenuhi, maka akan muncul kalimat “Syarat terpenuhi”. Gunakan percabangan *if* satu kondisi yang disertai dengan logika *and*!
2. Buatlah program penentuan bilangan ganjil yang menerima masukan sebuah bilangan kemudian menampilkan apakah bilang tersebut adalah bilangan ganjil dengan menggunakan percabangan satu *if*!
3. Buatlah program bilangan terbesar di antara 3 buah bilangan yang dimasukkan dengan menggunakan *if* bersarang!

MODUL 5
TEKNIK PERCABANGAN (BAGIAN 2)
(Pertemuan 5)

Tujuan :

1. Mempraktekkan tentang pembacaan data secara percabangan dalam bahasa pemrograman
2. Mempraktekkan perintah-perintah percabangan dalam bahasa pemrograman

Tugas Pendahuluan :

1. Jelaskan menurut pengetahuan anda apa yang dimaksud dengan pernyataan *if multiple condition*!
2. Buatlah program pemilihan menu dengan menggunakan seleksi *case*!
3. Buatlah program kalkulator dengan menggunakan pernyataan *if* bersarang!

DASAR TEORI

Selain pernyataan *if* dengan dua kondisi, suatu pernyataan *if* dapat mengandung pernyataan *if* yang lain. Bentuk seperti ini biasa disebut *if* bersarang (*nested if*). Sebuah program mengijinkan blok percabangan *if* di dalam blok percabangan lainnya, dan tidak membatasi jenis percabangan apa yang boleh berada di dalam percabangan lainnya.

Bentuk Umum :

```
if(kondisi1)
    if(kondisi2)
        .
        .
      if(kondisi n)
         pernyataan;
      else
         pernyataan;
        .
        .
   else
     pernyataan;
else
 pernyataan;
```

Pernyataan *switch case* merupakan alternatif dari pernyataan *if* untuk masalah dengan pilihan berganda. Pada masalah tertentu, *switch case* lebih memberikan kejelasan dari pada *if*. Namun perlu diketahui bahwa semua persoalan yang dapat ditangani *switch case* bisa ditangani oleh *if*, tetapi tidak sebaliknya. S*witch case* biasanya digunakan untuk memilih di antara lebih dari 2 pilihan. *Switch case* digunakan untuk menggantikan struktur *if-else-if* dimana kondisinya mengacu pada variabel yang sama.

Bentuk Umum :

```
switch(kondisi){
  case konstanta1:
      pernyataan1a;
      pernyataan1b;
    break;
  case konstanta2:
      pernyataan2a;
      pernyataan2b;
   break;
      .
      .
   default:
    pernyataan;
}
```

Pernyataan *switch* akan menyelesaikan kondisi yang diberikan dan kemudian membandingkan hasilnya dengan konstanta-konstanta yang berada di *case*. Jika semua konstanta-konstanta yang dibandingkan tidak ada yang sama, maka pernyataan yang berada pada default yang akan diproses.

KEGIATAN PRAKTIKUM

Menentukan Harga Berdasarkan Status dan Jabatan

Salinlah *coding* program berikut ini ke dalam DevC++ :

```
#include <stdio.h>
#include <string.h>
#include <conio.h>

int main() {
    char status[2],jabatan[2];
    int harga;

    printf("masukkan status (1. mahasiswa/2. non mahasiswa) : ");
scanf("%s",&status);

if(strcmp(status,"1")==0)
     {
      printf("masukkan jabatan (1. panitia/2. non panitia)    :
");scanf("%s",&jabatan);
       if(strcmp(jabatan,"1")==0)
       {
              harga=80000;
              printf("\nHarga: %d", harga);
       }
       else if (strcmp(jabatan,"2")==0)
       {
              harga=120000;
              printf("\nHarga: %d", harga);
       }
       else
       {
              printf("\nInput jabatan salah!!");
       }
     }
```

```
else if(strcmp(status,"2")==0)
     {
      printf("masukkan jabatan (1. panitia/2. non panitia)    :
");scanf("%s",&jabatan);
       if(strcmp(jabatan,"1")==0)
       {
              harga=100000;
              printf("\nHarga: %d", harga);
       }
       else if (strcmp(jabatan,"2")==0)
       {
              harga=150000;
              printf("\nHarga: %d", harga);
       }
       else
       {
              printf("\nInput jabatan salah!!");
       }
     }
     else
     {
      printf("input status salah!!");
     }

    return 0;
}
```

Simpan pekerjaan anda dengan nama **PRAK09.c** Kompilasi dan jalankan program tersebut, kemudian amati hasilnya dan pahami logikanya. Bagaimana jika inputan yang dimasukkan adalah angka atau huruf yang berbeda? Menngapa hal tersebut bisa terjadi? Pahamilah proses yang terjadi pada program tersebut.

Keterangan Grade Menggunakan *Case Of*

Salinlah *coding* program berikut ini ke dalam CevC++ :

```
#include <stdio.h>
#include <conio.h>

int main()
{
      int nilai;

      printf("Masukkan nilai 1-3 : ");scanf("%d",&nilai);

      switch(nilai)
      {
            case 1:
                  printf("Satu\n");
            break;
            case 2:
                  printf("Dua\n");
            break;
            case 3:
                  printf("Tiga\n");
            break;
            default:
                  printf("Bukan nilai 1, 2 atau 3\n");
            break;
      }
}
```

Simpan pekerjaan anda dengan nama **PRAK10.c** Kompilasi dan jalankan program tersebut, kemudian amati hasilnya dan pahami logikanya. Pahamilah proses yang terjadi pada program tersebut.

TUGAS

1. Buatlah program kalkulator yang menerima masukan dua buah bilangan, kemudian menerima masukan pilihan menu berupa penjumlahan, pengurangan, dan perkalian. Selanjutnya kedua buah bilangan yang telah dimasukkan tersebut akan diproses sesuai dengan menu yang telah dipilih!
2. Buatlah program penentuan bonus bagi pembeli berdasarkan total pembelian yang dimasukkan, dimana kriterianya adalah jika total pembelian lebih dari 100.000 maka pembeli mendapatkan diskon sebesar 10%, jika total pembelian kurang dari 100.000 dan lebih dari 50.000 maka pembeli mendapatkan sebuah piring cantik, jika total pembelian kurang dari 50.000 dan lebih dari 10.000 maka pembeli mendapatkan sebuah gelas cantik, selanjutnya jika total pembelian kurang dari 10.000 maka pembeli tidak akan mendapatkan bonus!
3. Dapatkah soal nomor 2 dipecahkan menggunakan *case* (tanpa *if*)? Berikan penjelasan anda!

MODUL 6
TEKNIK PENGULANGAN (BAGIAN 1)
(Pertemuan 6)

Tujuan :

1. Mempraktekkan tentang pembacaan data secara berulang dalam bahasa pemrograman
2. Mempraktekkan perintah-perintah perulangan serta analisa kondisi dan aksi dengan perulangan dalam bahasa pemrograman.

Tugas Pendahuluan :

1. Apakah yang dimaksud dengan pengulangan pada pemrograman dasar? Jelaskan!
2. Ada berapa macam pengulangan yang terdapat pada bahasa pemrograman C? Sebutkan!
3. Buatlah program pengulangan sederhana dengan menggunakan *for* dan *while do*!

DASAR TEORI

Pengulangan (*looping*) adalah suatu bagian yang bertugas melakukan kegiatan mengulang suatu proses sesuai dengan yang diinginkan. Banyak dari aplikasi perangkat lunak yang melakukan pekerjaan berulang-ulang sampai sebuah kondisi yang diinginkan, oleh karena itu pengulangan merupakan bagian yang penting dalam pemrograman karena dengan adanya pengulangan pembuat program tidak perlu menulis kode program sebanyak pengulangan yang diinginkan.

Dalam bahasa C menyediakan beberapa konstruksi perintah untuk melakukan proses-proses pengulangan, yaitu :

1. FOR
2. While
3. Do While

Struktur *for* digunakan untuk melakukan perulangan yang tidak berkondisi.Artinya jumlah perulangannya telah diketahui dengan pasti.

Bentuk Umum :

```
for(variable = nilai awal; kondisi_akhir; counter)
{
     Pernyataan;
}
```

`Nilai awal` suatu variabel untuk perulangan (misalnya i=0), **`kondisi_akhir`** adalah suatu ungkapan yang menunjukkan suatu kondisi yang harus dipenuhi agar perulangan dapat terus dilakukan (misalnya i<=20), **`counter`** adalah suatu ungkapan yang merubah nilai-nilai variabel pengontrol perulangan setiap saat perlungan dilakukan (misalnya i++).

Pengulangan *while* biasa digunakan jika jumlah pengulangan tidak diketahui atau memiliki kemungkinan dapat dilakukan kurang dari batas pengulangan yang telah ditentukan. Pengulangan *while* hanya akan melakukan pengulangan selama kondisi pengulangan terpenuhi. Perintah-perintah akan dilaksanakan apabila ekspresi boolean dalam keadaan *true*. Di dalam *loop* ada nilai yang mengontrol *loop* dan nilainya harus berubah, sehingga pada akhir program akan keluar dari *loop.*

Bentuk Umum :

```
variable = nilai awal;

while(kondisi)
{
      Pernyataan_1;
      Pernyataan_2;
      ......
      Pernyataan_n;

      counter;

}
```

KEGIATAN PRAKTIKUM

Membuat Perulangan Positif

Salinlah *coding* program berikut ke dalam DevC++ :

```
#include<stdio.h>

int main()
{
       int i;
       for(i=0;i<5;i++)
       {
              printf("\nIsi dari i ke-%d adalah : %d",i,i);

       }

       return 0;
}
```

Simpan pekerjaan anda dengan nama **PRAK11.c** kompilasi dan jalankan program tersebut.Apa yang terjadi? Amati dan pahamilah logikanya.

Membuat Perulangan Negatif

Salinlah *coding* program berikut ke dalam DevC++ :

```
#include<stdio.h>

int main()
{
  int i;
  for(i=0;i>-5;i--)
   {
     printf("\nIsi dari i ke 8%d adalah : %d",i,i);
   }

  return 0;
}
```

Kompilasi dan jalankan program tersebut. Apa yang terjadi? Amati dan pahamilah logikanya. Apa perbedaan antara perulangan positif dan perulangan negatif menurut anda setelah menjalankan program di atas?

Menghitung Rata-rata dari Sejumlah Nilai

Salinlah *coding* program berikut ke dalam DevC++ :

```
#include<stdio.h>

int main()
{
	float nilai,jumlah;
	int i;

	jumlah=0;

	for(i=0;i<5;i++)
	{
		printf("Nilai ke-%d : ",i);scanf("%f",&nilai);
		jumlah=jumlah+nilai;
	}

	printf("\nJumlah nilai : %f",jumlah);
	printf("\nNilai rata-rata : %f",jumlah/5);

	return 0;
}
```

Simpan pekerjaan anda dengan nama **PRAK12.c** Kompilasi dan jalankan program tersebut. Apa yang terjadi? Amati dan pahamilah logikanya.

Program *For* Bersarang

Salinlah *coding* program berikut ke dalam DevC++ :

```
#include<stdio.h>

int main()
{
	int i,j;

	for(i=0;i<8;i++)
	{
		for(j=1;j<=i;j++)
		{
			printf("*");

		}
		printf("\n");
	}
	return 0;
}
```

Simpan pekerjaan anda dengan nama **PRAK13.c**. Kompilasi dan jalankan program tersebut. Seperti apa tampilan yang muncul? Amati dan pahamilah logikanya.

Mengurut Bilangan dengan *While Do*

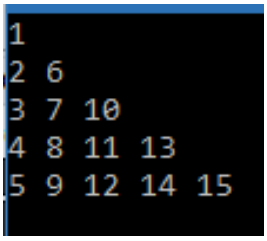
Salinlah *coding* program berikut ke dalam DevC++ :

```
#include<stdio.h>

int main()
{
      int i=0;

      while(i<10)
      {
         printf(" %d ",i);
         i++;
      }

      return 0;
}
```

Simpan pekerjaan anda dengan nama **PRAK14.c**. Kompilasi dan jalankan program tersebut. Amati dan pahamilah logikanya.

TUGAS

1. Buatlah program yang menerima masukan batas awal dan batas akhir dan menampilkan perkalian deret dari bilangan yang telah dibatasi!
2. Buatlah program yang akan menampilkan deretan bilangan ganjil seperti ini : (1 3 5 7 9 11 13 15) menggunakan pengulangan *for*!
3. Buatlah program pengulangan *for* bersarang yang menampilkan tampilan seperti berikut!



MODUL 7
TEKNIK PENGULANGAN (BAGIAN 2)
(Pertemuan 7)

Tujuan :

1. Mempraktekkan tentang pembacaan data secara berulang dalam bahasa pemrograman
2. Mempraktekkan perintah-perintah perulangan serta analisa kondisi dan aksi dengan perulangan dalam bahasa pemrograman.

Tugas Pendahuluan :

1. Jelaskan perbedaan dari *for, while ... do, do ... while*!
2. Apakah fungsi dari *do ... while* dan pada saat kondisi apa harus menggunakan *do ....while*?

DASAR TEORI

Pengulangan *do ... while* biasa digunakan jika jumlah pengulangan tidak diketahui atau belum pasti, namun berbeda dengan *while* karena kondisi pengulangan *do ... while* ada di bagian bawah blok pengulangan. Pengulangan *do ... while* minimal selalu dilakukan sekali karena kondisi pengulangan ada di bagian bawah, berbeda dengan pengulangan *while* yang saat pertama kali masuk blok pengulangan dilakukan pemeriksaan kondisi pengulangan.

Bentuk Umum :

```
Do
{
    pernyataan_1;
    pernyataan_2;
    ...
    pernyataan_n;
}
while (kondisi);
```

Di dalam teknik pengulangan terdapat teknik counter yang berfungsi untuk mengontrol pengulangan proses. Pengontrolan ini dilakukan dengan memeriksa isi variabel yang digunakan sebagai counter, sehingga jumlah pengulangan dapat diketahui.

KEGIATAN PRAKTIKUM

Program *do ... while*

Salinlah *coding* program berikut ke dalam DevC++ :

```
#include<stdio.h>

int main()
{
    int i;
    i=0;

    do{
       printf("%d",i);
       i=i+1;

    }while(i<5);
    return 0;

}
```

Simpan pekerjaan anda dengan nama **PRAK15.c.** Kompilasi dan jalankan program tersebut. Amati dan pahamilah logikanya.

Pilihan Pengulangan Tampilan

Salinlah *coding* program berikut ke dalam DevC++ :

```
#include<stdio.h>
#include<string.h>

int main()
{
    char jawaban[2];

    do
    {
       printf("\nProgram Dasar");

       printf("\nApakah anda ingin menampilkan lagi ?
(y/t)");scanf("%s",&jawaban);

    }while(strcmp(jawaban,"t")==1);
    return 0;
}
```

Simpan pekerjaan anda dengan nama **PRAK16.c**. Kompilasi dan jalankan program tersebut. Amati dan pahamilah logikanya.

TUGAS

1. Buatlah program untuk menampilkan bilangan genap dari 1 sampai dengan 100 menggunakan *do.....while*!
2. Buatlah program yang menerima masukan jumlah bintang dan menampilkan pola bintang sesuai dengan jumlah yang dimasukkan, seperti contoh di bawah ini!

```
Masukkan jumlah bintang : 5
*****
 *****
  *****
   *****
    *****
```

3. Selanjutnya buatlah program seperti nomor 2, namun dengan pola bintang seperti di bawah ini!

```
Masukkan jumlah bintang : 5
*****
 *****
  *****
 *****
*****
```

MODUL 8
ARRAY 1 DIMENSI
(Pertemuan 9)

Tujuan:

- Mahasiswa mengetahui tentang *array* dan jenis-jenis *array*, serta penggunaan *array* pada pemrograman.
- Mahasiswa dapat mempraktekkan penggunaan *array* satu dimensi dalam pemrograman.

Tugas Pendahuluan:

1. Apa yang Anda ketahui mengenai *array*?
2. Apakah kegunaan dari *array*?

DASAR TEORI

Array merupakan suatu tipe data yang terstruktur dan dapat digunakan untuk menyimpan data yang memiliki tipe data yang sama. Dengan kata lain, *array* adalah kumpulan data yang memiliki tipe yang sama. *Array* dapat diibaratkan seperti sebuah tabel. *Array* biasa juga disebut larik. *Array* dapat digunakan untuk menyimpan data yang berjumlah banyak namun masih memiliki suatu hubungan atau terdapat kesamaan antar data tersebut. Sebagai contoh, *array* dapat digunakan untuk menyimpan data 117 nama orang yang semuanya merupakan pelanggan sebuah toko obat, atau 65 nama yang kesemuanya merupakan nama mahasiswa yang mengambil kelas pada semester pendek. Data yang terdapat dalam sebuah *array* dapat diidentifikasikan atau dibedakan dengan menggunakan index.

Penggunaan *array* dapat mengurangi kerumitan dalam proses penyimpanan data dalam jumlah yang besar. Jika kita hendak menyimpan variabel nama yang berjumlah sepuluh nama, kita tidak harus membuat 10 variabel untuk nama tersebut. Dengan menggunakan *array* kita tidak perlu membuat banyak variabel untuk data yang memiliki tipe yang sama. Selain itu, dalam alokasi memori penyimpanan, tipe data *array* melakukan pemesanan tempat terlebih dahulu sesuai dengan kebutuhan yang ada.

Terdapat 2 jenis *array*, yaitu *array* 1 dimensi dan *array* 2 dimensi. Perbedaan mendasar antara kedua jenis *array* ini adalah, *array* dengan 1 dimensi merupakan *array* yang dapat digambarkan sebagai sebuah baris. Selain itu, dalam *array* 1 dimensi, elemen yang ada di dalamnya dapat diakses hanya dengan menggunakan 1 indeks saja. Sedangkan *array* 2 dimensi merupakan *array* yang dapat digambarkan seperti sebuah matrik. Selain itu elemen yang ada dalam *array* 2 dimensi dapat diakses dengan menggunakan 2 indeks, yaitu indeks kolom dan juga indeks baris. Berikut format pendeklarasian *array*:

```
tipe_data nama_larik[jumlah_elemen]
```

Dalam hal ini *jumlah_elemen* harus berupa konstanta.

Contoh pendeklarasian larik :

Deklarasi	Keterangan
int cacah[4];	Larik *cacah* mempunyai 4 buah elemen bertipe **int** (bilangan bulat)
char vokal[5];	Larik *vokal* mempunyai 5 buah elemen bertipe **char** (karakter)
char kota[6][20]	Larik *kota* mempunyai 6 buah elemen bertipe string dengna panjang maksimal sebesar 19 karakter

Pengaksesan elemen larik dilakukan dengan menggunakan notasi
nama_array[indeks]

KEGIATAN PRAKTIKUM

Salinlah *coding* program di bawah ini:

1. **Pendeklarasian dan pengaksesan *array* 1 dimensi**

```
#include<stdio.h>

int main()
{
	char vokal[5];
	int i;

	vokal[0]='A';
	vokal[1]='I';
	vokal[2]='U';
	vokal[3]='E';
	vokal[4]='O';

	for(i=0;i<5;i++)
	{
		printf("%c\n", vokal[i]);
	}

	return 0;
}
```

Dengan melihat pendeklarasian *array* di atas maka diketahui bahwa, pendeklarasian *array* dilakukan dengan langsung mengisikan nilai dari tiap indeksnya. Lalu, bagaimana jika *user* sendiri yang diminta untuk mengisikan nilai dari tiap indeks yang ada?

2. Menghitung nilai rata-rata menggunakan *array* 1 dimensi

```
#include<stdio.h>
#include<conio.h>

int main()
{
        int NIM[5];
        float uts[5];
        float uas[5];
        float kuis[5];
        float tugas[5];
        float rata[5];
        int i, bnyk;

        printf("Masukkan jumlah mahasiswa : ");scanf("%d",&bnyk);

        for(i=0;i<bnyk;i++)
        {
                printf("\nData mahasiswa ke-%d",i+1);
                printf("\nMasukkan NIM mahasiswa : ");scanf("%d",&NIM[i]);
                printf("Masukkan nilai kuis      : ");scanf("%f",&kuis[i]);
                printf("Masukkan nilai tugas     : ");scanf("%f",&tugas[i]);
                printf("Masukkan nilai UTS       : ");scanf("%f",&uts[i]);
                printf("Masukkan nilai UAS       : ");scanf("%f",&uas[i]);

                rata[i]=(kuis[i]+tugas[i]+uts[i]+uas[i])/4;
        }

        printf("\n-----------------------------------------------------");
        for(i=0;i<bnyk;i++)
        {
                printf("\nNIM         : %d",NIM[i]);
                printf("\nNilai kuis  : %f",kuis[i]);
                printf("\nNilai tugas : %f",tugas[i]);
                printf("\nNilai UTS   : %f",uts[i]);
                printf("\nNilai UAS   : %f",uas[i]);

                printf("\nNilai rata  : %f",rata[i]);

                printf("\n");
        }

        return 0;


}
```

Apabila diperhatikan dengan seksama, maka dapat dilihat bahwa *coding* program di atas merupakan contoh dari adanya penggunaan lebih dari 1 *array* dalam 1 program. Selain itu bahwa *coding* program di atas juga menunjukkan bagaimana cara menempatkan nilai sesuai dengan indeks yang ada.

TUGAS

1. Buatlah sebuah program yang dapat mencetak bilangan prima yang ada dalam bilangan 1-20 dengan menggunakan *array*, dimana user menginputkan nilai dari 1-20!

MODUL 9
ARRAY 2 DIMENSI
(Pertemuan 10)

Tujuan:

- Mempraktekkan tentang *array* dan jenis-jenis *array*, serta penggunaan *array* pada pemrograman.
- Mempraktekkan tentang *array* 2 dimensi dalam pemrograman.

Tugas Pendahuluan:

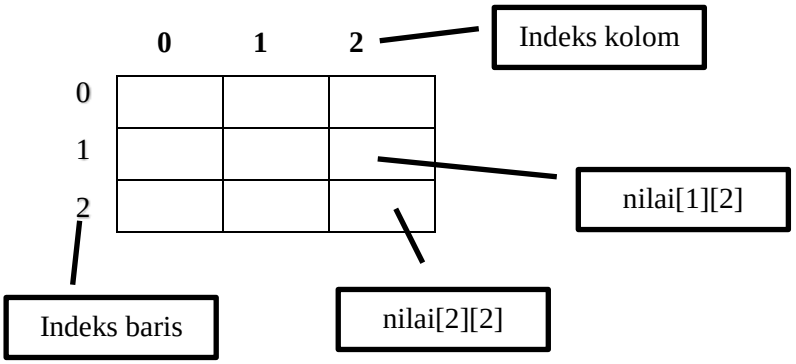
1. Apa yang Anda ketahui tentang array 2 dimensi?
2. Apakah perbedaan antara *array* 1 dimensi dan *array* 2 dimensi?

DASAR TEORI

Seperti yang telah dijelaskan pada pertemuan sebelumnya, a*rray* merupakan suatu tipe data yang terstruktur dan dapat digunakan untuk menyimpan data yang memiliki tipe data yang sama. Dengan kata lain, *array* adalah kumpulan data yang memiliki tipe yang sama. Penggunaan *array* dapat mengurangi kerumitan dalam proses penyimpanan data dalam jumlah yang besar. Dengan menggunakan *array* kita tidak perlu membuat banyak variabel untuk data yang memiliki tipe yang sama.

Terdapat 2 jenis *array*, yaitu *array* 1 dimensi dan *array* 2 dimensi. *Array* dengan 1 dimensi dapat digambarkan sebagai sebuah baris, sedangkan *array* 2 dimensi merupakan dapat digambarkan seperti sebuah matrik. Berikut format pendeklarasian *array* 2 dimensi:

```
Tipe_data nama_larik[indeks_baris][indeks_kolom]
```



KEGIATAN PRAKTIKUM

Salinlah *coding* program di bawah ini:

- **Pendeklarasian dan pengaksesan *array* 2 dimensi**

```
#include<stdio.h>
#include<conio.h>

int main()
{


	int matrik[3][3]={{5,6,7},{8,9,10},{1,2,3}};
	int i,j;



	printf("\nMatrik Ordo 3x3: \n");


	for(i=0;i<=2;i++)
	{
		for(j=0;j<=2;j++)
		{
			printf("[%d] ",matrik[i][j]);

		}
		printf("\n");
	}


	return 0;

}
```

Koding di atas merupakan contoh koding yang dapat menampilkan sebuah matrik dengan ordo 3 x 3 menggunakan *array* 2 dimensi.

TUGAS

1. Buatlah sebuah program untuk menjumlahkan dua buah matrik menggunakan *array* dua dimensi dengan ketentuan:
   - Syarat dari penjumlahan matrik adalah jumlah ordo matrik A = jumlah ordo matrik B.
   - *User* yang menginputkan ordo matrik.
   - *User* yang menginputkan nilai dari tiap elemen matrik yang ada.
2. Buat program perkalian antara 2 buah matrik dengan menggunakan *array* dimana nilai tiap elemennya di-inputkan oleh *user*! Selain itu, perhatikan syarat perkalian antar matrik, yaitu jumlah kolom pada matrik A = jumlah baris pada matrik B!
3. Transpose matrik adalah terjadinya pertukaran tempat antara elemen baris dan elemen kolom dan sebaliknya pada sebuah matrik. Buatlah sebuah program transpose matrik dengan ordo yang ditentukan oleh *user* sendiri!

MODUL 10
RECORD
(Pertemuan 11)

Tujuan:

1. Mahasiswa mengetahui definisi *record*.
2. Mahasiswa mengetahui cara penggunaan *record*.
3. Mahasiswa dapat mempraktekkan tentang *record* dan menggunakannya dalam pemrograman.

Tugas Pendahuluan:

1. Apa yang Anda ketahui tentang *record*?
2. Apakah perbedaan antara *array* dan *record*?

DASAR TEORI

Pada bahasa pemrograman C, terdapat salah satu tipe data yang bernama *record*. *Record* merupakan salah satu tipe data selain array yang dapat menampung lebih dari 1 data. Perbedaan antara array dan *record* adalah setiap data yang tersimpan dalam array harus memiliki tipe data yang sama. Sedangkan dalam *record*, tipe datanya dapat berbeda. Setiap elemen dalam *record* disebut dengan *field*. Berikut cara mendeklarasikan *record*:

```
typedef struct {
      tipe_data nama variabel1;
      tipe_data nama variabel1;
      .......
      ........
}nama_record;
```

Sebagai contoh:

```
typedef struct {
         char NIM[10];
         char nama[50];
         float ipk;
 }Mahasiswa;
```

KEGIATAN PRAKTIKUM

Salinlah *coding* program di bawah ini :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>

typedef struct {
         char NIM[12];
         char nama[50];
         float ipk;
 }Mahasiswa;



int main()
{

    Mahasiswa recMhs;
    printf("=====Input Data Mahasiswa=====\n");
    printf("Masukkan NIM  : ");scanf("%s",recMhs.NIM);
    printf("Masukkan Nama : ");scanf("%s",recMhs.nama);
    printf("Masukkan IPK  : ");scanf("%f",&recMhs.ipk);

    printf("\n=====Data Mahasiswa=====\n");
    printf("NIM       : %s\n",recMhs.NIM);
    printf("Nama      : %s\n",recMhs.nama);
    printf("IPK       : %f\n",recMhs.ipk);

    return 0;

}
```

Kompilasi projek dengan menekan *f9* dan jalankan dengan menekan *f10* pada *keyboard*. Potongan *coding* di atas menunjukkan cara mendeklarasikan dan menggunakan tipe data *record*. Berdasarkan *coding* di atas, Mahasiswa menjadi tipe data dan recMhs menjadi variabel yang memiliki tipe data bentukan yang diberi nama Mahasiswa. Jika memperhatikan *coding* di atas dengan seksama, maka dapat dilihat bahwa pemanggilan sebuah *record* dilakukan dengan mengetikkan variabel *record* dan indeks variabel *record* yang dimaksudkan. Sebagai contoh, untuk menampung nilai NIM pada variabel nim yang terdapat pada *record* Mahasiswa, maka dilakukan dengan cara mengetikkan **recMhs.NIM**.

1. Menggabungkan *array* dan *record*

```
#include <stdio.h>
#include <conio.h>

typedef struct {
        char NIM[10];
        char nama[50];
        char kota[20];
        char konsntrasi[30];
        float ipk;
 }Mahasiswa;

int main()
{
   Mahasiswa rechMhs[10];
   int i,byk;

   printf("Masuk banyaknya mahasiswa : ");scanf("%d",&byk);

   for(i=0;i<byk;i++)
   {
          printf("NIM Mahasiswa          : ");scanf("%s",rechMhs[i].NIM);
          printf("Nama Mahasiswa         : ");scanf("%s",rechMhs[i].nama);
          printf("Kota Mahasiswa         : ");scanf("%s",rechMhs[i].kota);
          printf("Konsentrasi Mahasiwa   : ");scanf("%s",rechMhs[i].konsntrasi);
          printf("IPK Mahasiwa           : ");scanf("%f",&rechMhs[i].ipk);
          printf("\n");
    }

    printf("\nData telah berhasil diinput");


   printf("\n------------Data Mahasiswa--------------");
   for(i=0;i<byk;i++)
   {
          printf("\nNIM Mahasiwa           : %s",rechMhs[i].NIM);
          printf("\nNama Mahasiwa          : %s",rechMhs[i].nama);
          printf("\nKota Mahasiwa          : %s",rechMhs[i].kota);
          printf("\nKonsentrasi Mahasiwa   : %s",rechMhs[i].konsntrasi);
          printf("\nIPK Mahasiwa           : %f",rechMhs[i].ipk);
          printf("\n");
    }



   return 0;

}
```

Coding program di atas merupakan *coding* yang sama dengan *coding* pada praktikum nomor 1, hanya saja dengan menambahkan beberapa baris kode *array*, maka *record* mahasiswa di atas dapat menampung lebih dari 1 mahasiswa. Berbeda dengan praktikum nomor 1 yang hanya dapat menampung data untuk 1 mahasiswa saja. *Coding* di atas merupakan contoh penggabungan antara *array* dan *record*.

TUGAS

Buatlah sebuah program yang dapat mengolah data mahasiswa dengan ketentuan:

- User menginputkan NIM, Nama, Nilai Kuis, Nilai UTS, dan Nilai UAS.
- Hasil atau output yang diinginkan: NIM, Nama, Nilai Kuis, Nilai UTS, Nilai UAS, dan Nilai Akhir.
- Nilai akhir = 20% dari nilai kuis + 30% dari nilai UTS + 50% dari nilai UAS.

MODUL 11
PROSEDUR
(Pertemuan 12)

Tujuan:

- Mahasiswa diharapkan dapat mengetahui struktur serta cara menggunakan dan memanggil prosedur.
- Mahasiswa dapat mempraktekkan penggunaan prosedur dalam kasus yang berhubungan dengan prosedur.

Tugas Pendahuluan:

1. Apa yang Anda ketahui mengenai prosedur?
2. Apakah fungsi atau kegunaan dari sebuah prosedur?

DASAR TEORI

Prosedur merupakan bagian dari program yang berupa serangkaian *statement* yang memiliki nama. Sebuah prosedur dapat terdiri atas satu atau lebih dari satu prosedur. Penggunaan prosedur dapat memberikan beberapa keuntungan, sebagai berikut:

1. Dapat membagi atau memecah program menjadi lebih sederhana.
2. Dapat membantu menghemat penggunaan memori penyimpanan karena dapat dapat memperkecil ukuran *file.*
3. Dapat digunakan pada program lain cukup dengan mengopi prosedur yang hendak digunakan.

Oleh karena itu, prosedur pada umumnya digunakan pada program yang kompleks dan memiliki subprogram. Dalam penggunaannya, sebuah prosedur dapat digunakan dengan memanggil atau mengetikkan nama prosedur tersebut.

Parameter

Pada prosedur dan fungsi (dibahas pada pertemuan berikutnya) akan melibatkan penggunaan parameter. Parameter adalah suatu variable yang berfungsi sebagai penampung nilai pada prosedur atau fungsi, yang diberikan oleh pemanggil prosedur atau fungsi. Parameter yang dikirmkan dari modul utama ke modul prosedur disebut dengan parameter nyata (*actual parameter*) dan parameter yang ada dan dituliskan pada judul prosedur disebut dengan parameter formal (*formal parameter*).

Proses pengiriman data lewat parameter nyata ke parameter formal disebut dengan parameter passing. Parameter nyata dan parameter formal harus dengan **tipe yang sama**.

a. Pass by Value

Parameter yang dikirimkan berupa nilai (value)nya saja. Jadi apabila terjadi pengubahan nilai pada prosedur ataupun fungsi tidak akan mempengaruhi nilai pada variabel yang dipassingkan, atau yang dikirimkan.

b. Pass by Reference

Parameter yang dikirimkan berupa acuan. Jadi apabila terjadi pengubahan nilai pada prosedur ataupun fungsi akan mempengaruhi nilai pada variabel yang dipassingkan, atau yang dikirimkan.

Terdapat dua jenis prosedur, yaitu prosedur dengan parameter dan prosedur tanpa parameter. Berikut format penulisan pendeklarasian prosedur yang memiliki parameter:

```
procedure nama_prosedur (tipe_data parameter1,tipe_data parameter2, ….)
{
       tipe_data variabel1;
       tipe_data variabel2;
       ......
       ......

              Prenyataan_1;
              Prenyataan_2;
              Prenyataan_3;
              .......
              .......
}
```

Selain prosedur dengan parameter, terdapat pula prosedur yang tidak memiliki parameter.Berikut format prosedur tanpa parameter:

```
procedure nama_prosedur()
{
       tipe_data variabel1;
       tipe_data variabel2;
       ......
       ......

              Prenyataan_1;
              Prenyataan_2;
              Prenyataan_3;
              .......
              .......
}
```

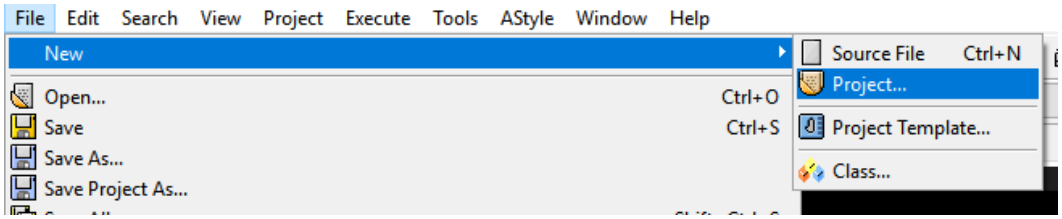
KEGIATAN PRAKTIKUM

Buatlah projek baru dengan nama ModProsedur, buat 3 file dalam projek tersebut :

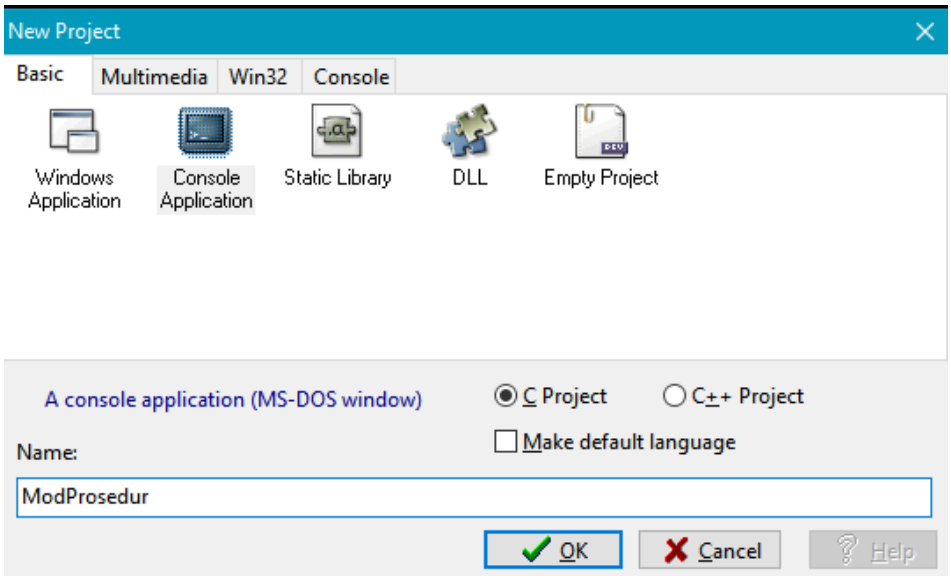
1. header.h
2. function.c
3. main.c

Langkah-langkah membuat projek aplikasi :

1. Membuat projek baru, pilih **file > new > project..**

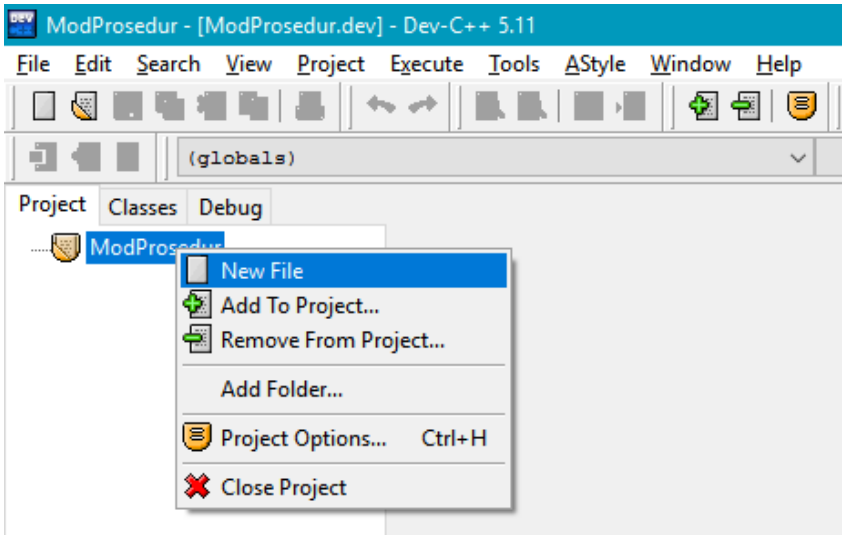


2. Setting new project

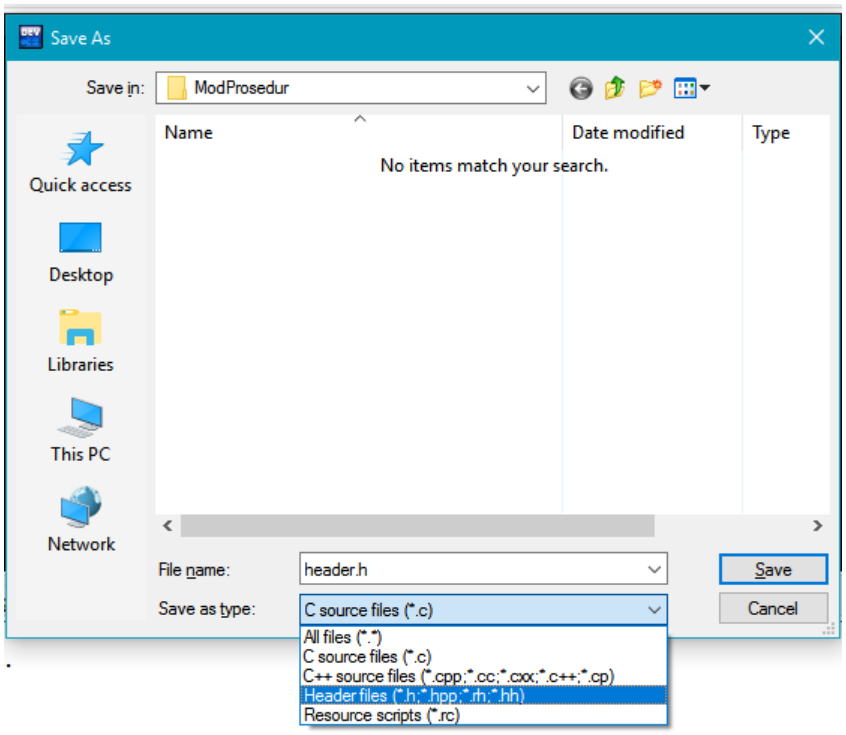


   - Basic > Pilih Consule Application
   - Selanjutnya gunakan bahasa C Project
   - Terakhir beri nama ModProsedur

3. Menambahkan file baru pada projek



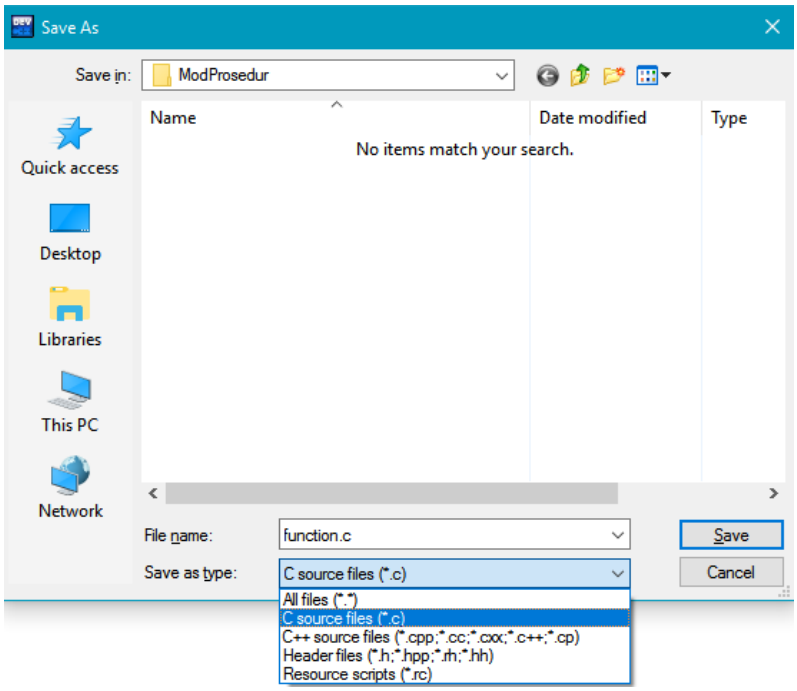
4. Simpan file pertama dengan nama header.h



Salinlah *coding* program di bawah ini ke dalam file header.h :

```
#include <stdio.h>
#include <stdlib.h>

void tampil();
```

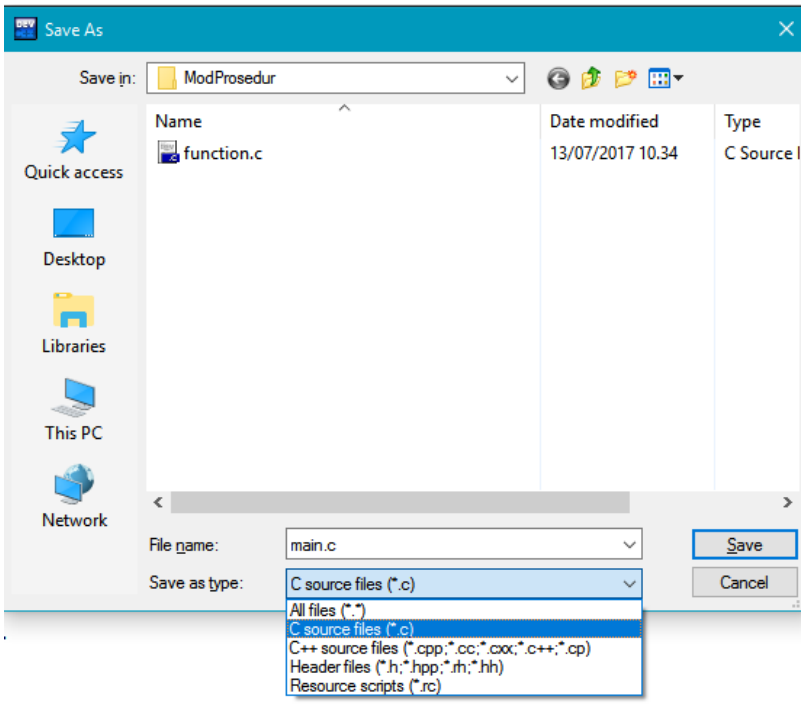
5. Simpan file kedua dengan nama function.c



Salinlah *coding* program di bawah ini ke dalam file function.c :

```
#include "header.h"

void tampil()
{
	printf("Hallo, ini percobaan pertama prosedur..");
}
```

6. Simpan file ketiga dengan nama main.c



Salinlah *coding* program di bawah ini ke dalam file main.c :

```
int main() {

	tampil();
	return 0;
}
```

Setelah pekerjaan tersimpan, lakukan kompilasi projek dengan menekan **F9** dan jalankan program dengan menekan **F10**. Amatilah hasilnya!

1. Prosedur dengan parameter

Buatlah projek baru dengan nama ModProsedur2, buat 3 file dalam projek tersebut :

1. header.h
2. function.c
3. main.c

Salinlah *coding* program di bawah ini ke dalam file header.h :

```
#include <stdio.h>
#include <stdlib.h>

void tulispagar(int n);
```

Salinlah *coding* program di bawah ini ke dalam file function.c :

```
#include "header.h"

void tulispagar(int n)
{
	int i;
	for(i=0;i<n;i++)
	{
		printf("#");
	}
}
```

Salinlah *coding* program di bawah ini ke dalam file main.c :

```
int main() {

	tulispagar(5);
	return 0;
}
```

2. Menggunakan Lebih dari Satu Prosedur

1. header.h

```
#include <stdio.h>
#include <stdlib.h>

void persegi(float sisi);
void persegi_panjang(float panjang,float lebar);
void jajar_genjang(float alas,float tinggi);
void layang_layang(float d1,float d2);
```

2. function.c

```
#include "header.h"

void persegi(float sisi)
{
    float luas;
    luas=sisi*sisi;
    printf("\nLuas persegi : %f", luas);
}

void persegi_panjang(float panjang,float lebar)
{
    float luas;
    luas=panjang*lebar;

    printf("\nLuas persegi panjang : %f", luas);
}

void jajar_genjang(float alas,float tinggi)
{
    float luas;
    luas=alas*tinggi;

    printf("\nLuas jajar genjang : %f", luas);
}

void layang_layang(float d1,float d2)
{
    float luas;
    luas=0.5*d1*d2;

    printf("\nLuas layang layang : %f", luas);
}
```

3. main.c

```
#include "header.h"
int main() {

    int pil;
    float sisi, panjang, lebar, alas, tinggi,diag1,diag2;

    printf("======Menu aplikasi========\n");
    printf("1. Menghitung luas persegi.\n");
    printf("2. Menghitung luas persegi panjang.\n");
    printf("3. Menghitung luas jajar genjang.\n");
    printf("4. Menghitung luas layang-layang.\n");
    printf("\nMasukkan pilihan anda : ");scanf("%d",&pil);

    switch(pil)
    {
           case 1:

                  printf("Masukkan nilai sisi :
");scanf("%f",&sisi);
                  persegi(sisi);
                  break;
           case 2:
                  printf("Masukkan nilai panjang :
");scanf("%f",&panjang);
                  printf("Masukkan nilai lebar   :
");scanf("%f",&lebar);
                  persegi_panjang(panjang,lebar);
                  break;
           case 3:
                  printf("Masukkan nilai alas     :
");scanf("%f",&alas);
                  printf("Masukkan nilai tinggi   :
");scanf("%f",&tinggi);
                  jajar_genjang(alas,tinggi);
                  break;
           case 4:
                  printf("Masukkan nilai diagonal 1 :
");scanf("%f",&diag1);
                  printf("Masukkan nilai diagonal 2 :
");scanf("%f",&diag2);
                  layang_layang(diag1,diag2);
                  break;
    }
    return 0;
}
```

Dalam sebuah program, bisa saja terdapat lebih dari satu *procedure* di dalamnya. Contoh di atas, merupakan contoh penggunaan lebih dari satu *procedure.* Dimana *procedure* yang ada dideklarasikan terlebih dahulu, untuk kemudian dapat dipanggil oleh program utama.

TUGAS

1. Buatlah program untuk menghitung luas segi tiga dengan menggunakan prosedur yang memiliki parameter!
2. Buatlah program untuk menghitung luas persegi panjang dengan menggunakan prosedur tanpa parameter!
3. Buatlah program menggunakan lebih dari satu *procedure* dengan menu pilihan sebagai berikut:
    1) Menghitung Gaji
    2) *Booking* Kamar Hotel
    3) Keluar

    Buatlah berdasarkan ketentuan seperti di bawah ini:

    1) Menghitung Gaji

        Gaji pokok setiap pegawai dilihat berdasarkan golongan kepegawaiannya, yaitu:

        - Golongan 1: Rp 2.000.000,-
        - Golongan 2: Rp 3.000.000,-
        - Golongan 3: Rp 4.000.000,-

        Selain itu, terdapat tunjangan bagi setiap pegawai yang telah menikah dengan ketentuan:

        - Tunjangan Rumah Tangga: 50% dari gaji pokok
        - Tunjangan Anak: jumlah anak * 25% dari gaji pokok

        Program yang dibuat diharapkan dapat menghasilkan output:

        - Nama karyawan:
        - Status:
        - Tunjangan:
        - Total Gaji:

    2) *Booking* Kamar Hotel

        Harga kamar hotel per-malamnya ditentukan dari jenis kamar hotel yang ada yaitu:

        - *Single Room:* Rp 650.000,-/malam
        - *Double Room:* Rp 900.000,-/malam
        - *Deluxe Room:* Rp 1.200.000,-/malam
        - *Suite Room:* Rp 1.500.000,-/malam

        Petugas hotel selaku *user* nantinya akan bertugas untuk menginputkan nama tamu, jenis kamar yang diminta oleh tamu, serta lama menginap. Sehingga hasil atau *output* yang dihasilkan adalah jumlah biaya yang harus dibayarkan oleh tamu.

MODUL 12
FUNGSI
(Pertemuan 13)

Tujuan:

- Mahasiswa dapat mengetahui dan mempraktekkan tentang fungsi dan penggunaannya dalam pemrograman.
- Mahasiswa mengetahui struktur dan cara menggunakan fungsi.

Tugas Pendahuluan:

1. Apa yang Anda ketahui mengenai fungsi (*function*)?
2. Apakah kegunaan dari sebuah fungsi?
3. Apakah perbedaan antara fungsi dan prosedur?

DASAR TEORI

Pada dasarnya fungsi atau *function* maupun prosedur adalah sama. Fungsi maupun prosedur sama-sama merupakan subprogram. Selain itu, fungsi dan prosedur dapat digunakan atau dipanggil berkali-kali atau berulang-ulang dalam sebuah program. Namun, fungsi lebih cenderung digunakan untuk membuat perintah-perintah yang bersifat perhitungan. Selain itu, perbedaan antara fungsi dan prosedur adalah fungsi memiliki pengembalian nilai, sehingga pada saat dipanggil, sebuah fugnsi dapat langsung digunakan untuk mengisikan ekspresi yang ada. Dalam bahasa pemrograman C, fungsi didefinisikan dengan *function.* Berikut format penulisan untuk mendeklarasikan sebuah *function:*

```
return_type nama_fungsi(tipe_data parameter)
{
   Pernyataan_1;
   Pernyataan_2;

   return value;

}
```

Suatu fungsi secara umum terdiri dari dua buah komponen utama, yaitu **defenisi fungsi** dan **tubuh fungsi**. Defenisi fungsi berisi dengan tipe databalikan dari fungsi, nama dari fungsi dan argumen-argumennya jika digunakan. Tubuh dari fungsi berisi dafengan pernyataan-pernyataan yang melakukan tugas yang diberikan kepada fungsi yang bersangkutan, ditulis dalam tanda kurung kurawal buka dan tutup.

KEGIATAN PRAKTIKUM

Buatlah projek baru dengan nama ModFunngsi, buat 3 file dalam projek tersebut :

1. header.h
2. function.c
3. main.c

Salinlah *coding* program di bawah ini ke dalam file header.h :

```
#include <conio.h>
#include <stdio.h>

float kali(float a,float b);
```

Salinlah *coding* program di bawah ini ke dalam file function.c :

```
#include "header.h"

float kali(float a,float b)
{
      return a*b;
}
```

Salinlah *coding* program di bawah ini ke dalam file main.c :

```
#include "header.h"

int main() {

      printf("hasil perkalian : %f",kali(10,40));

      return 0;
}
```

Sebuah program dapat memiliki lebih dari satu buah fungsi. Pada program di atas fungsi dideklarasikan pada file header yang bertujuan agar kompiler dapat mendeteksi semua fungsi-fungsi yang digunakan dalam program tanpa diikuti oleh pernyataan-pernyataan yang digunakan dalam fungsi. File function.c berisi defenisi detail (operasi) dari masing masing fungsi dan file main.c berisi dari cara pemanggilan fungsi.

1. ***Function* dengan input dari *user***

Buat projek baru dengan nama fungsikali yang berisi 3 file yaitu header.h, function.c, dan main.c

File header.h

```
#include <stdio.h>
#include <stdlib.h>

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

File function.c

```
#include "header.h"

float perkalian(float a, float b)
{
	return a*b;
}
```

File main.c

```
#include "header.h"

/* run this program using the console pauser or add your own
getch, system("pause") or input loop */

int main() {

	float a,b;

	printf("Masukkan nilai pertama : ");scanf("%f",&a);
	printf("Masukkan nilai kedua   : ");scanf("%f",&b);

	printf("Hasil perkalian : %f",perkalian(a,b));

	return 0;
}
```

TUGAS

1. Buat sebuah program untuk menghitung luas belah ketupat dengan ketentuan input dilakukan oleh *user*!

2. Buatlah sebuah program yang dapat menentukan bilangan terbesar di antara 3 angka yang di-inputkan oleh user menggunakan *function*!

MODUL 13
Rekursif
(Pertemuan 14)

Tujuan

1. Mahasiswa mengetahui dan memahami tentang Rekursif.
2. Mahasiswa mampu mengimplementasikan Rekursif dalam bahasa pemrograman.
3. Mahasiswa dapat memecahkan berbagai kasus yang berkaitan dengan Rekursif.

Tugas Pendahuluan

1. Apakah yang anda ketahui tentang Rekursif?

Dasar Teori

Rekursif adalah kemampuan suatu subrutin untuk memanggil dirinya sendiri. Pada beberapa persoalan, kemampuan seperti ini sangat berguna karena mempermudah solusi. Namun demikian rekursif juga memiliki kelemahan yaitu memungkinkan terjadinya *overflow* pada *stack* (*stack* tidak mampu menangani permintaan pemanggilan subrutin karena kehabisan memori). Dalam Rekursif sebenarnya terkandung pengertian prosedur dan fungsi. Perbedaannya adalah bahwa rekursif bisa memanggil ke dirinya sendiri, tetapi prosedur dan fungsi harus dipanggil lewat pemanggil prosedur dan fungsi. Contoh paling sederhana dari proses rekursif ini adalah proses menghitung nilai faktorial.

Sub Program Rekursif adalah subprogram yang memanggil dirinya sendiri selama kondisi pemanggilan dipenuhi. Dengan melihat sifat sub program rekursif di atas maka sub program rekursif harus memiliki :

1. Kondisi yang menyebabkan pemanggilan dirinya berhenti (disebut kondisi khusus atau *special condition*)
2. Pemanggilan diri sub program (yaitu bila kondisi khusus tidak dipenuhi)

Secara umum bentuk dari sub program rekursif memiliki statemen kondisional :

```
if (kondisi khusus terpenuhi)
then
    lakukan instruksi yang akan dieksekusi bila kondisi khusus dipenuhi
else
    panggil diri-sendiri dengan parameter yang sesuai
```

Program :

```
#include <stdlib.h>
#include <conio.h>

long int faktorial(unsigned int n)
{
	if(n==0 || n==1)
		return 1;
	else
		return n*faktorial(n-1);
}

int main()
{
	int n;
	long int hasil;

	printf("masukkan nilai n = "); scanf("%d",&n);

	hasil = faktorial(n);

	printf("Hasil %d ! = %ld ", n, hasil);

	return 0;

}
```

Simpan coding di atas dengan nama **faktoria.c**. Kompilasi dan jalankan program tersebut dan amati hasilnya.

Kegiatan Praktikum

Salinlah coding berikut pada projek dengan nama **rekPangkat**:

file header

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

long int pangkat(unsigned int y, unsigned int n);
```

file function

```
#include "header.h"

long int pangkat(unsigned int y, unsigned int n)
{
	if(n==1)
		return y;
	else
		return y*pangkat(y, n-1);
}
```

file main

```
#include "header.h"


int main() {
	int y, n;
	long int hasil;

	printf("Masukkan nilai Y = ");scanf("%d",&y);
	printf("Masukkan nilai n = ");scanf("%d",&n);

	hasil=pangkat(y,n);

	printf("%d ^ %d = %ld",y,n,hasil);
	return 0;
}
```

Simpan projek dan kompilasi dengan menekan F9, jalankan projek dengan menekan F10.

TUGAS

Buatlah program untuk menyelesaikan persoalan fungsi Fibonacci dalam bentuk rekursif.