

MODUL 1

Nama Percobaan : INTRODUCTION TO C++

Tujuan : Pengenalan terhadap c++ sebagai salah satu bahasa pemrograman terstruktur

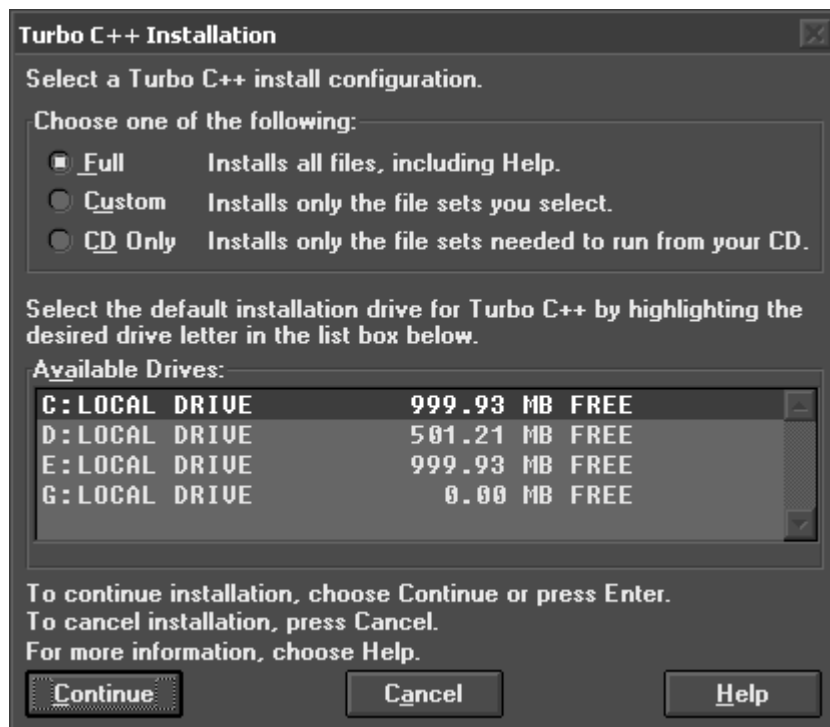
Materi:

Bahasa pemrograman c++ merupakan pengembangan dari bahasa c, dimana bahasa c merupakan bahasa yang biasa dipakai untuk keperluan pemrograman sistem, antara lain untuk membuat : assembler, interpreter, program paket, sistem operasi, editor, kompiler, utility, bahkan bahasa pemrograman populer sejenis PHP dan Java menggunakan sintaks dasar yang mirip bahasa c.

Instalasi Turbo C++

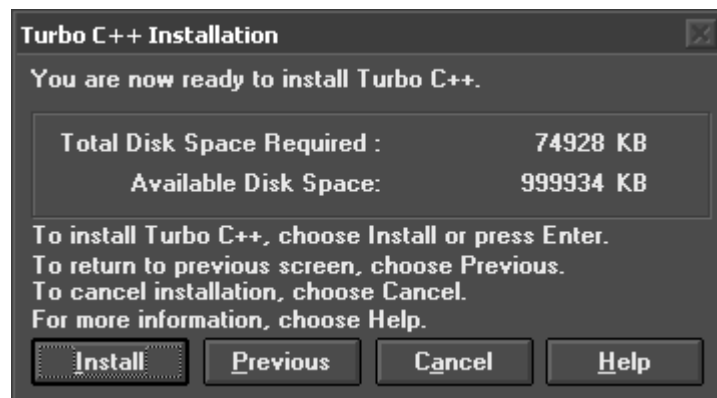
Pada pertemuan kali ini, digunakan Turbo C++ 4.5 for Windows. Untuk melakukan Instalasi Turbo C++ 4.5 ikuti langkah-langkah sebagai berikut:

1. Masukkan CD Installer Turbo C++ 4.5 for Windows ke dalam CD-ROM drive. Double klik file Install.exe.
2. Akan muncul window Configuration Notes, tekan tombol Continue atau skip untuk melanjutkan instalasi, maka akan ditampilkan dialog seperti pada gambar 1.1.



Gambar 1.1 Dialog Install Configuration

3. Pilih tipe dan drive tujuan instalasi. Tekan **Continue** untuk melanjutkan instalasi. Selanjutnya akan tampil window baru, tekan **Continue** lagi untuk melanjutkan instalasi. Setelah itu akan tampil dialog seperti berikut:



Gambar 1.2 Dialog Ready Installation

4. Tekan **Install** untuk memulai instalasi atau **previous** untuk mengulang konfigurasi instalasi. Tunggu Proses Instalasi sampai selesai.

Memulai Program Turbo C++ 4.5

Jalankan Turbo C++ melalui start menu > All Programs > Turbo C++ 4.5 > Turbo C++.



Gambar 1.3 Tampilan Turbo C++ 4.5

Turbo C++ 4.5 terdiri dari beberapa menu aplikasi, antara lain:

- **File**, terdiri dari:
 1. **New**, untuk memulai program baru
 2. **Open**, untuk mengambil atau membuka program yang telah dibuat
 3. **Save**, untuk menyimpan file/program dengan ekstensi *.CPP
 4. **Save as**, untuk menyimpan file/program dengan nama lain atau di drive lain
 5. **Save all**, untuk menyimpan seluruh file/program
 6. **Print**, untuk mencetak editor/listing program ke printer
 7. **Printer setup**, mengatur setting printer
 8. **Exit**, keluar dari program Turbo C++
- **Edit**, terdiri dari:
 1. **Undo**, untuk membatalkan pengeditan terakhir

2. **Redo**, kembali ke pengeditan terakhir yang telah di Undo
 3. **Cut**, untuk memotong bagian tertentu dari program
 4. **Copy**, untuk menduplikasi bagian program
 5. **Paste**, untuk meletakkan bagian program yang di Cut atau di Copy
 6. **Clear**, untuk menghapus bagian tertentu dari program
 7. **Clear all**, untuk menghapus seluruh editor program
 8. **Select all**, untuk select semua bagian program
 9. **Buffer list**, browse daftar buffer
- **Search**, terdiri dari:
 1. **Find**, untuk mencari text/string dalam program
 2. **Replace**, untuk mengganti text awal dengan text baru
 3. **Search again**, melakukan pencarian ulang
 4. **Browse symbol....**
 5. **Locate function**, mencari letak suatu fungsi dalam program
 6. **Previous message....**
 7. **Next message....**
 - **View**, terdiri dari:
 1. **Class Expert...**
 2. **Project...**
 3. **Message**, untuk menampilkan pesan error/kesalahan program
 4. **Dll.**
 - **Project**, diantaranya untuk untuk membuat project baru, open project yang telah ada, serta untuk mengcompile project (Alt + F9)
 - **Debug**, salah satu fungsi di dalamnya adalah untuk menjalankan program yang telah selesai.
 - **Tool**, terdiri dari Resource Workshop, Grep, WinSight, WinSpector, Key Map Compiler.
 - **Options**, untuk konfigurasi-konfigurasi tentang project, sheet, target, environment, dan lain-lain.
 - **Window**, fungsinya sama seperti menu window pada umumnya.
 - **Help**, sebagai referensi bantuan bila terdapat kesulitan dalam memprogram.

MODUL 2

Nama Percobaan : STRUKTUR DASAR C++

Tujuan : Mampu mengetahui struktur dasar dari c++ sebagai bahasa pemrograman terstruktur.

Materi:

Tipe Data

Tipe data merupakan bagian program yang paling karena tipe data mempengaruhi setiap instruksi yang akan dilaksanakan oleh komputer. Misalnya, 5 dibagi 2 bisa saja akan menghasilkan hasil yang berbeda tergantung tipe datanya. Jika keduanya bertipe integer maka akan menghasilkan nilai 2, namun jika keduanya bertipe float maka akan menghasilkan nilai 2.5000000. Pemilihan tipe data yang tepat akan membuat proses operasi data menjadi lebih efisien dan efektif.

Dalam bahasa C++ tipe data yang digunakan sama dengan tipe data pada bahasa C, yaitu:

No	Tipe Data	Ukuran	Range(Jangkauan)	Format
1	Char	8	-127 - 127	%c
2	Unsigned char	8	0 - 255	%c
3	Signed char	8	-127 - 127	%c
4	Int	16 / 32	-32,767 - 32,767	%d
5	Unsigned int	16 / 32	0 - 65,535	%u
6	Signed int	16 / 32	-32,767 - 32,767	%d
7	Short int	16	-32,767 - 32,767	%hd
8	Unsigned short int	16	0 - 65,535	%hu
9	Signed short int	16	-32,767 - 32,767	%hd
10	Long int	32	-2,147,483,647 - 2,147,483,647	%ld
11	Signed long int	32	-2,147,483,647 - 2,147,483,647	%ld
12	Unsigned long int	32	0 - 4,294,967,295	%lu

13	Float	32	3.4E-38 - 3.4E+38	%f
14	Double	64	1.7E-308 - 1.7E+308	%lf
15	Long double	80	3.4E-4932 - 3.4E+4932	%Lf
16	Void	0	-	

Konstanta

Konstanta adalah suatu nilai yang tidak berubah selama proses dari program. Konstanta harus didefinisikan terlebih dahulu di awal program. Konstanta dapat bernilai integer, pecahan, karakter dan string. Contoh konstanta: 50; 13; 3,14; 4.50005; 'A'; dll. Selain itu C++ juga menyediakan karakter khusus yang disebut karakter escape seperti pada bahasa C, antara lain:

- o \a : untuk bunyi bell (alert)
- o \b : mundur satu spasi (backspace)
- o \f : ganti halaman (form feed)
- o \n : ganti baris baru (new line)
- o \r : ke kolom pertama, baris yang sama (carriage return)
- o \t : tabulasi horizontal
- o \v : tabulasi vertikal
- o \0 : nilai kosong (null)
- o \' : karakter petik tunggal
- o \" : karakter petik ganda
- o \\ : garis backslash

Penggunaan karakter ini dengan menggunakan fungsi printf().

Variabel

Variabel adalah suatu pengenal yang digunakan untuk mewakili suatu nilai tertentu di dalam proses program. Berbeda dengan konstanta yang nilainya selalu tetap, nilai dari suatu variabel bisa diubah-ubah sesuai kebutuhan. Nama dari suatu variabel dapat ditentukan sendiri oleh pemrogram dengan aturan sebagai berikut:

1. Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf. Huruf kecil dan besar dianggap berbeda.
2. Tidak boleh mengandung spasi dan simbol-simbol khusus kecuali garis bawah.
3. Panjangnya bebas, tetapi hanya 32 karakter pertama yang terpakai

Deklarasi variabel adalah:

Tipe data nama variabel;

variabel ada beberapa macam, antara lain:

1. variabel lokal

yaitu variabel yang nama dan nilainya hanya dikenal di suatu blok statemen tertentu saja atau di dalam suatu fungsi. Variabel ini dideklarasikan di dalam blok bersangkutan, variabel ini akan dihapus dari memori jika proses sudah meninggalkan blok statemen letak variabel lokalnya.

Contoh:

```
Void main(void)
{
    Int x;  —————> Variabel lokal bagi fungsi main( )
    :
    :
}
```

2. Variabel Global

Yaitu variabel yang dikenal di semua bagian-bagian tertentu dari program. Variabel global dapat dibuat dengan mendeklarasikannya di luar suatu blok statemen atau di luar fungsi-fungsi yang menggunakannya.

Contoh:

```
#include<stdio.h>

Float tambah(float a, float b);

Float c;  —————> Variabel global
Main()
{
    :
}
Float tambah(float a, float b);
{
    :
}
```

Variabel *c* merupakan variabel global, sehingga dikenal dan dianggap sama di kedua fungsi. Perubahan nilai *c* pada salah satu fungsi tersebut berarti juga merubah nilai variabel *c* di fungsi yang lainnya.

3. Variabel Statik

Berbeda dengan variabel lokal, variabel statik akan tetap ada dan nilainya akan tetap dipertahankan walaupun proses telah keluar dari bloknya. Dengan variabel statik, suatu proses yang telah keluar dari suatu blok yang mendeklarasikannya dan kemudian masuk kembali ke blok fungsi tersebut, maka nilai terakhir dari variabel static masih tetap digunakan. Dengan demikian variabel static bersifat tetap, yaitu tidak dihapus variabel dan nilainya selama proses dari program. Variabel statik dibuat dengan dideklarasikan menggunakan pengubah **static**.

Contoh:

```
Void main(void)
{
    Static int x;  → variabel statik
    :
}
```

Untuk lebih jelasnya tentang variabel statik, akan dibahas pada pembahasan fungsi.

Operator

o Operator Penugasan

Operator penugasan dalam C++ berupa tanda sama dengan (“=”)

o Operator Aritmatika

- * : untuk perkalian
- / : untuk pembagian
- % : untuk sisa pembagian (modulus)
- + : untuk penambahan
- - : untuk pengurangan

o Operator Hubungan (Perbandingan)

- < : kurang dari
- <= : kurang dari sama dengan
- > : lebih dari
- >= : lebih dari sama dengan

- == : sama dengan / sejajar
- != : tidak sama dengan
- **Operator Logika**
 - && : Logika And (Dan)
 - || : Logika OR (Atau)
 - ! : Logika NOT (Ingkaran)
- **Operator Bitwise**
 - << : pergeseran bit ke kiri
 - >> : pergeseran bit ke kanan
 - & : bitwise AND
 - ^ : bitwise XOR (exclusive OR)
 - | : bitwise OR
 - ~ : bitwise NOT
- **Operator Unary**
 - - : unary minus
 - ++ : peningkatan dengan penambahan nilai 1
 - -- : penurunan dengan pengurangan nilai 1
 - sizeof : Ukuran dari operand dalam byte
 - ! : unary NOT
 - & : menghasilkan alamat memory operand
 - * : menghasilkan nilai dari pointer

MODUL 3

Nama Percobaan : OPERASI INPUT & OUTPUT

Tujuan : Mampu mengetahui struktur operasi input-output pada c++

Materi:

Menampilkan Data / Informasi Ke Layar

Untuk menampilkan data / informasi, fungsi yang digunakan adalah sama dengan dengan turbo c, yaitu *printf()*, *puts()*, dan *putchar()*.

1. **Printf()**

Bentuk umum pernyataan printf adalah:

Printf("string kontrol", argumen1,argumen2,...);

String kontrol dapat berupa keterangan yang akan ditampilkan beserta penentu format (seperti %d, %f, dll). Argumen adalah data yang akan ditampilkan ke layar yang dapat berupa variabel, konstanta, bahkan ungkapan.

Misal:

Printf("%d",20); // argumen berupa konstanta

Printf("%d",a); // argumen berupa variabel

Printf("%d",a+20); // argumen berupa ungkapan

Untuk menampilkan data bilangan, penentu format yang dapat dipakai berupa:

%u	Untuk menampilkan data bilangan tak bertanda dalam bentuk desimal
%d, %i	Untuk menampilkan bilangan integer bertanda dalam bentuk desimal
%o	Untuk menampilkan bilangan octal
%x, %X	Untuk menampilkan bilangan bulat dalam bentuk heksadesimal
%f	Untuk menampilkan bilangan real dalam notasi : dddd.dddddd
%e, %E	Untuk menampilkan bilangan real dalam notasi eksponential
%g, %G	Untuk menampilkan bilangan real dalam bentuk notasi seperti %f, atau %e, tergantung oleh kepresisian data. (Digit 0 yang tak berarti tak akan ditampilkan)

l, L	Awalan yang digunakan untuk menyatakan long double
H	Awalan yang digunakan untuk menyatakan short int

Contoh program berikut memberikan gambaran penggunaan beberapa penentu format. Buka turbo c++, buat program baru dengan memilih menu file, pilih new. Ketik code berikut pada layar editor.

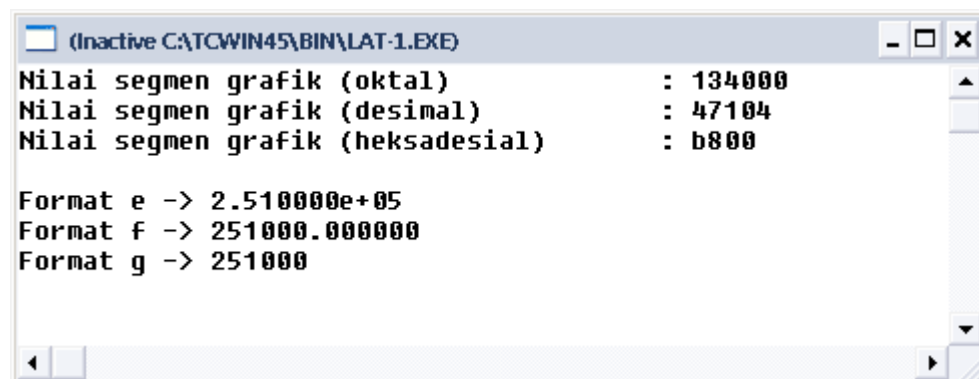
```
#include<stdio.h>

main()
{
    unsigned int segmen_grafik=0xB800;
    float x=251000.0;

    printf("Nilai segmen grafik (oktal)           : %o\n",segmen_grafik);
    printf("Nilai segmen grafik (desimal)        : %u\n",segmen_grafik);
    printf("Nilai segmen grafik (heksadesial)     : %x\n",segmen_grafik);

    printf("\nFormat e -> %e\n",x);
    printf("Format f -> %f\n",x);
    printf("Format g -> %g\n",x);
}
```

Simpan program misalnya dengan nama Lat-1.cpp, lalu jalankan program dengan menekan Ctrl+F9. Hasil eksekusi program adalah sebagai berikut:



```
(Inactive C:\TCWIN45\BIN\LAT-1.EXE)
Nilai segmen grafik (oktal)           : 134000
Nilai segmen grafik (desimal)        : 47104
Nilai segmen grafik (heksadesial)     : b800

Format e -> 2.510000e+05
Format f -> 251000.000000
Format g -> 251000
```

2. Puts() dan Putchar()

Fungsi puts() digunakan untuk menampilkan data string ke layar. Sifat fungsi ini secara otomatis akan diakhiri dengan \n (pindah baris). Berbeda dengan fungsi

Putchar() yang khusus untuk menampilkan sebuah karakter dan tidak diakhiri dengan perpindahan baris.

Contoh penggunaan:

```
Puts("Belajar Turbo C++);
```

```
Putchar('A');
```

Memasukkan Data Dari Keyboard

Untuk input data digunakan fungsi *scanf()*, *getch()* dan *getche()*.

1. Scanf()

Bentuk umum scanf() adalah:

Scanf("string kontrol", daftar argumen);

Untuk penentu format, bentuknya sama seperti pada printf(). Sedangkan untuk daftar argumen, haruslah berupa alamat yang ditambahkan tanda & (operator alamat). Ketik code berikut pada layar editor, buat aplikasi baru pada turbo c++ 4.5.

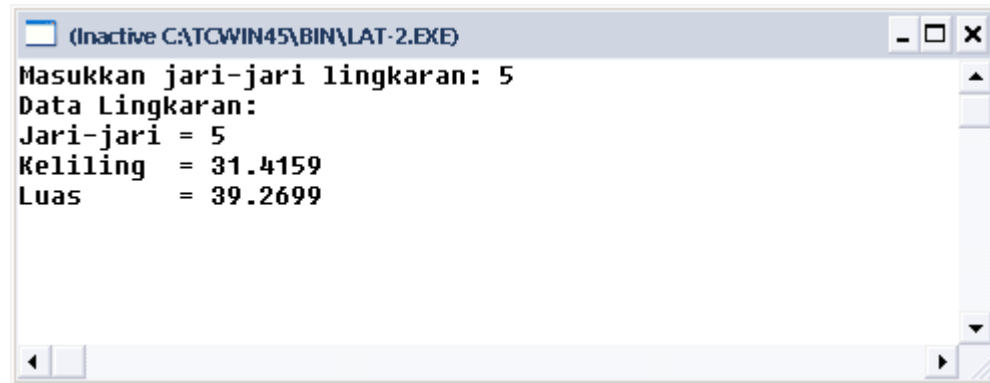
```
#include<stdio.h>
#define pi 3.141593

main()
{
    float radius,keliling,luas;

    printf("Masukkan jari-jari lingkaran: ");scanf("%f",&radius);
    keliling = 2*pi*radius;
    luas = 0.5*pi*radius*radius;

    puts("Data Lingkaran: ");
    printf("Jari-jari = %g\n",radius);
    printf("Keliling = %g\n",keliling);
    printf("Luas    = %g\n",luas);
}
```

Simpan program lalu jalankan program. Pertama program akan minta input radius, setelah di Enter maka akan ditampilkan luas dan keliling. Seperti pada tampilan berikut:



2. Getch() dan Getche()

Fungsi `getch()` digunakan untuk membaca sebuah karakter, dengan sifat karakter yang dimasukkan tak perlu diakhiri dengan ENTER. Disamping itu, karakter yang dimasukkan tak akan ditampilkan pada layar.

Sedangkan fungsi `getche()`, karakter yang dimasukkan akan ditampilkan pada layar. Berikut ini contoh penggunaan fungsi `getch()` dan `getche()`:

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

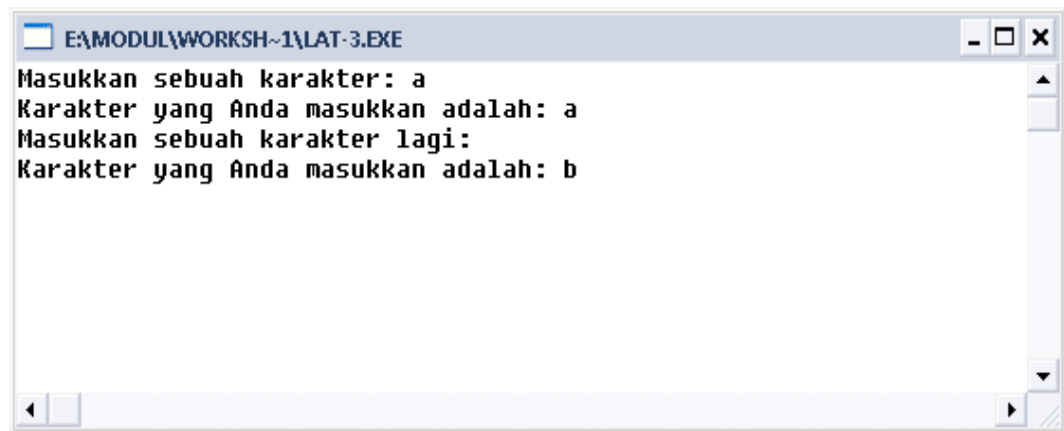
main()
{
    char huruf1,huruf2;

    printf("Masukkan sebuah karakter: ");
    huruf1=getche();    // karakter yang dimasukkan akan terlihat di layar
    printf("\nKarakter yang Anda masukkan adalah: %c",huruf1);

    printf("\nMasukkan sebuah karakter lagi: ");
    huruf2=getch();     // karakter yang dimasukkan tidak terlihat di layar
    printf("\nKarakter yang Anda masukkan adalah: %c",huruf2);

    getch();
}
```

Simpan program lalu jalankan. Setelah dieksekusi, maka hasilnya adalah sebagai berikut:



Latihan

1. Buat suatu program untuk menghitung luas segitiga
2. Buat program konversi dari jam ke detik dan dari detik ke jam

MODUL 4

Nama Percobaan : PENYELEKSIAN KONDISI

Tujuan : Mampu mengetahui dan memahami proses penyeleksian kondisi

Materi:

Struktur Kondisi “If...”

Struktur if dibentuk dari pernyataan if dan sering digunakan untuk menyeleksi suatu kondisi tunggal. Bila proses yang diseleksi terpenuhi atau bernilai benar, maka pernyataan yang ada di dalam blok if akan diproses dan dikerjakan. Bentuk umum struktur kondisi if adalah:

If(kondisi)

Pernyataan;

Buat program baru dan ketik code berikut:

```
#include<stdio.h>

main()
{
    double tot_pemb,korting;

    printf("Total Pembelian : Rp ");
    scanf("%lf",&tot_pemb);

    korting=0;    // Tentukan nilai awal variabel korting

    if(tot_pemb>=50000)
        korting=0.5*tot_pemb;

    printf("Besarnya Korting : Rp %.2lf\n",korting);
}
```

Simpan file misal dengan nama If_1.cpp kemudian jalankan, maka akan ditampilkan:



Struktur Kondisi “If ...Else...”

Dalam struktur kondisi if...else minimal terdapat dua pernyataan. Jika kondisi yang diperiksa bernilai benar atau terpenuhi maka pernyataan pertama yang dilaksanakan dan jika kondisi yang diperiksa bernilai salah maka pernyataan yang kedua yang dilaksanakan. Bentuk umumnya adalah sebagai berikut:

```
If(kondisi)
    Pernyataan-1
Else
    Pernyataan-2
```

Sebagai contoh penggunaan if...else yaitu untuk menyelesaikan penentuan besarnya bonus yang di depan dilakukan dengan if saja. Buat program baru dan simpan dengan nama If_2.cpp. Berikut source codenya:

```
#include<stdio.h>

main()
{
    double tot_pemb,korting;

    printf("Total Pembelian : Rp ");
    scanf("%lf",&tot_pemb);

    korting=0;

    if(tot_pemb>=50000)
        korting=0.5*tot_pemb;
    else
```



```
korting=0;

printf("Besarnya Korting : Rp %.2lf\n",korting);
}
```

Dalam program ditentukan jika total dari pembelian lebih besar atau sama dengan Rp 50.000,- maka akan diberikan korting sebesar 5 % dari total pembelian. Namun jika kurang dari itu maka tidak ada korting, sehingga jumlah korting yang ditampilkan adalah 0. hasil eksekusi program adalah sebagai berikut:



Berikut adalah contoh pernyataan if...else yang melibatkan pernyataan majemuk. Program ini digunakan untuk menentukan sebuah bilangan bulat termasuk ganjil atau genap. Penentuan genap atau ganjil dilakukan berdasarkan logika “suatu bilangan bulat jika habis dibagi dengan dua disebut bilangan genap, sedangkan kalau tidak habis dibagi dua berarti bilangan ganjil”.

```
#include<stdio.h>

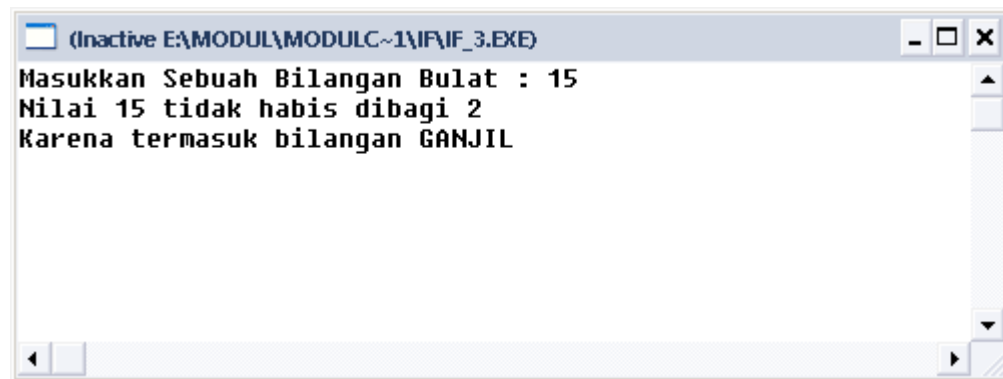
main()
{
    int bil;

    printf("Masukkan Sebuah Bilangan Bulat : ");
    scanf("%d",&bil);

    if(bil % 2==0)
    {
        printf("Nilai %d tidak habis dibagi 2\n",bil);
        puts("Karena termasuk bilangan GANJIL");
    }
    else
```

```
    {  
        printf("Nilai %d habis dibagi 2\n",bil);  
        puts("Karena termasuk bilangan GENAP");  
    }  
}
```

Simpan file dengan nama If_3.cpp, kemudian jalankan program. Berikut tampilan program tersebut:



Berikut ini adalah contoh program pemakaian if-else secara bertingkat. Program ini menggunakan if-else bertingkat yang digabung dengan pemakaian operator logika untuk menentukan grade dari suatu nilai.

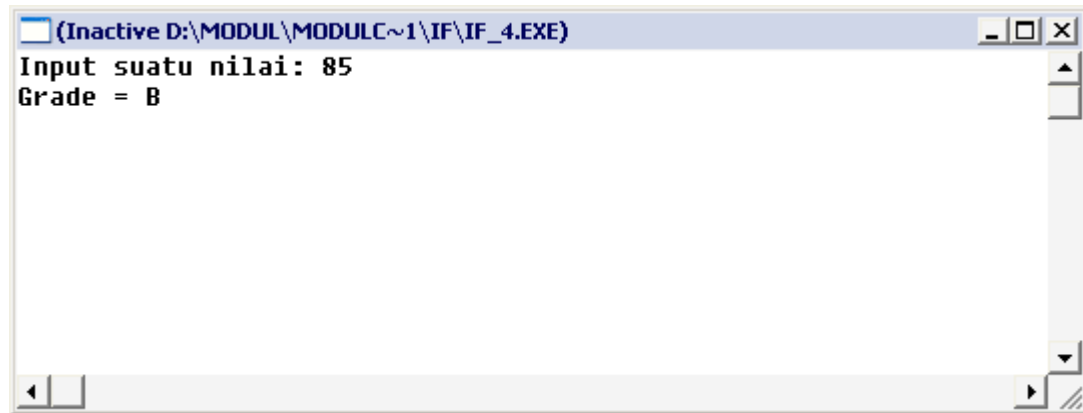
```
#include<stdio.h>  
#include<conio.h>  
  
main()  
{  
    float nil;  
    char grade;  
  
    clrscr();        // bersihkan layar  
    printf("Input suatu nilai: ");scanf("%f",&nil);  
  
    if(nil>85)  
        grade='A';  
    else  
        if((nil>=76)&&(nil<=85))  
            grade='B';  
        else  
            if((nil>=60)&&(nil<=75))  
                grade='C';
```

```
        else
            if((nil>=45)&&(nil<=59))
                grade='D';
            else
                if((nil>=0)&&(nil<45))
                    grade='E';
                else
                    puts("Input salah!!!");

        printf("Grade = %c",grade);    // Tampilkan Grade

    }
```

Simpan dengan nama If-4.cpp. setelah dijalankan maka akan ditampilkan:



Operator Kondisi

Operator kondisi memiliki tiga buah operand. Operator tersebut dinamakan sebagai operator kondisi dengan simbol `?`... Bentuk ungkapan yang menggunakan operator ini:

Kondisi ? ungkapan-1 : ungkapan-2

Maksud dari ungkapan kondisi:

- Jika kondisi bernilai benar, maka nilai ungkapan kondisi berupa nilai ungkapan-1.
- Jika kondisi bernilai salah, maka nilai ungkapan kondisi berupa nilai ungkapan-2.

Contoh penggunaan misalnya untuk memperoleh nilai terbesar(nilai maks) antara dua buah bilangan. Buat program baru dan ketikkan code berikut, simpan dengan nama `opr_kond.cpp`:

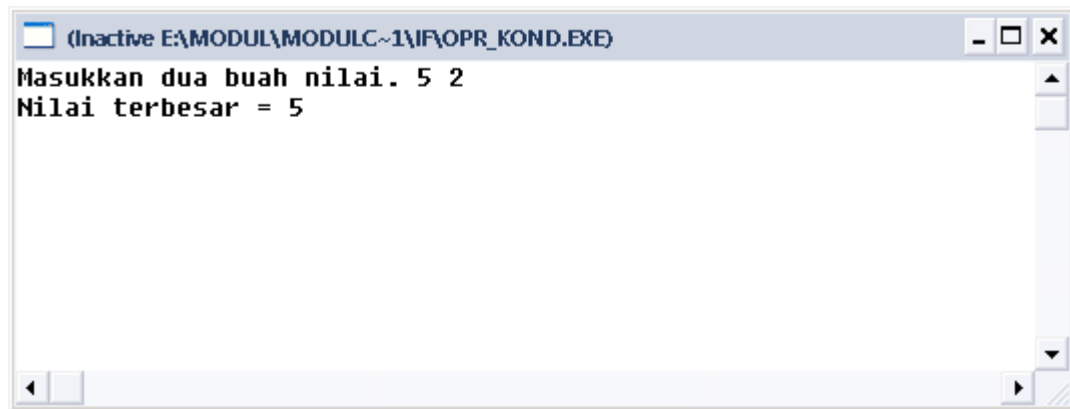
```
#include<stdio.h>

main()
{
    float nilai1,nilai2,maks;

    printf("Masukkan dua buah nilai. ");
    scanf("%f %f",&nilai1,&nilai2);

    // Tentukan maks
    maks=(nilai1>nilai2)?nilai1:nilai2;
    printf("Nilai terbesar = %g\n",maks);
}
```

Tampilan programnya adalah:



Struktur Kondisi “Switch...Case...Default...”

Struktur ini digunakan untuk penyelesaian kondisi yang memiliki sejumlah alternatif. Diantaranya untuk menggantikan pernyataan *if* bertingkat di atas. bentuk umum dari struktur kondisi ini adalah:

```
Switch(kondisi)
{
    Case konstanta-1: pernyataan-1
                      Break;
    Case konstanta-2: pernyataan-2
                      Break;
    ....
    ....
    Case konstanta-n: pernyataan-n
                      Break;
    Default: pernyataan -x
}
```

Contoh pemakaian struktur ini yaitu untuk menentukan nama hari berdasarkan kode hari yang diinput oleh user. Berikut contoh pemakaian struktur case:

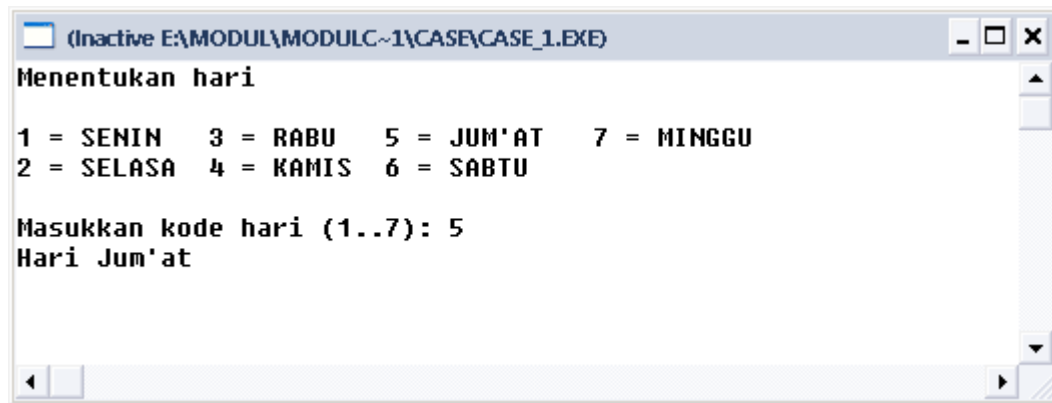
```
#include<stdio.h>

main()
{
    int kd_hari;

    // Tampilkan Menu Pilihan dan input kode hari
    puts("Menentukan hari\n");
    puts("1 = SENIN    3 = RABU    5 = JUM'AT    7 = MINGGU");
    puts("2 = SELASA    4 = KAMIS    6 = SABTU");
    printf("\nMasukkan kode hari (1..7): "); scanf("%d",&kd_hari);
    switch(kd_hari)
    {
        case 1:puts("Hari Senin");
                break;
        case 2:puts("Hari Selasa");
                break;
        case 3:puts("Hari Rabu");
                break;
        case 4:puts("Hari Kamis");
                break;
        case 5:puts("Hari Jum'at");
                break;
        case 6:puts("Hari Sabtu");
                break;
        case 7:puts("Hari Minggu");
                break;
        default:
```

```
        puts("Kode yang anda masukkan salah!!");  
    }  
}
```

Simpan program dengan nama case_1.cpp kemudian jalankan. Maka akan ditampilkan:



```
(Inactive E:\MODUL\MODULC~1\CASE\CASE_1.EXE)  
Menentukan hari  
  
1 = SENIN    3 = RABU    5 = JUM'AT    7 = MINGGU  
2 = SELASA   4 = KAMIS    6 = SABTU  
  
Masukkan kode hari (1..7): 5  
Hari Jum'at
```

Terlihat bahwa dengan menggunakan pernyataan **switch**, program menjadi lebih jelas dan mudah dipahami. Pernyataan **break** dimaksudkan agar setelah menampilkan nama hari, eksekusi dilanjutkan ke akhir switch. Jika tidak diberi **break** maka eksekusi akan dilanjutkan ke pernyataan yang terletak pada case berikutnya, walaupun nilai / kondisi case bernilai benar.

Latihan

1. Buat program untuk menentukan nilai maksimum dari 3 bilangan
2. Buat program untuk menentukan tahun kabisat
3. Buat program untuk mencetak nama bulan berdasarkan nomornya

MODUL 5

Nama Percobaan : PENGULANGAN PROSES (LOOPING)

Tujuan : Mampu mengetahui dan memahami proses pengulangan / looping dalam pemrograman seperti while dan for

Materi:

Struktur “While”

Perulangan WHILE banyak digunakan pada program yang terstruktur. Perulangan ini banyak digunakan bila jumlah perulangannya belum diketahui. Proses perulangan akan terus berlanjut selama kondisinya bernilai benar (true) dan akan berhenti bila kondisinya bernilai salah.

Contoh berikut menggunakan struktur WHILE. Program digunakan untuk menghitung banyaknya karakter dari kalimat yang dimasukkan melalui keyboard. Disamping itu juga menghitung banyaknya spasi. Untuk mengakhiri entri kalimat, tombol ENTER harus ditekan. Karena itu, tombol ENTER inilah yang dijadikan sebagai kondisi perhitungan jumlah spasi maupun karakter seluruhnya. Ketik code berikut ini pada editor program, simpan lalu jalankan.

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

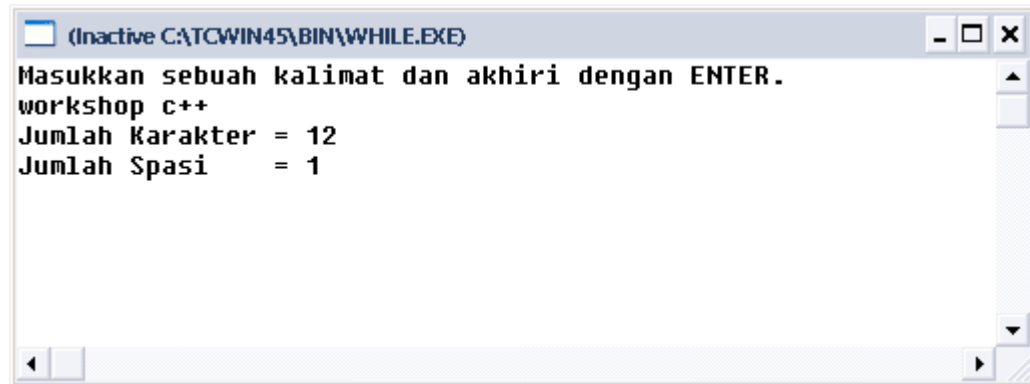
#define ENTER '\r'
#define SPASI ' '

main()
{
    char kar;
    int jumkar=0, jumspasi=0;

    puts("Masukkan sebuah kalimat dan akhiri dengan ENTER.");
    while((kar=getche())!=ENTER)
    {
        jumkar++;
        if(kar==SPASI) jumspasi++;
    }
}
```

```
printf("\nJumlah Karakter = %d",jumkar);  
printf("\nJumlah Spasi    = %d",jumspasi);  
}
```

Setelah dijalankan maka akan ditampilkan:



Struktur “DO...WHILE...”

Pada dasarnya perulangan do...while sama saja dengan struktur while, hanya saja pada proses pertulangan dengan while, seleksi berada di while yang letaknya di atas sementara pada perulangan do...while, seleksi while berada di bawah batas pertulangan. Jadi dengan menggunakan struktur do..while sekurang-kurangnya akan terjadi satu kali perulangan.

Program berikut adalah contoh penggunaan struktur do...while... Program tersebut adalah program dalam transaksi penjualan. Dengan input nama barang, harga serta jumlah barang yang dibeli, maka jumlah total akan diketahui setelah dikurangi dengan diskon jika ada. Proses perulangan akan berakhir jika ditekan tombol ESC. Program ini menggunakan suatu fungsi gotoxy(x , y), fungsi ini berguna untuk mencetak suatu output pada layar dalam koordinat xy. Jadi, x sebagai parameter kolom, sedangkan y sebagai parameter baris, sehingga letak dari suatu output bisa diatur sesuai kehendak pemrogram.

Ketik codenya, simpan lalu jalankan.


```

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

main()
{
    int i=1,y=5;
    char brg[25];
    float jml,tot1,tot2=0;
    float harga,diskon,disc;

    gotoxy(4,2);printf("=====");
    gotoxy(4,3);printf("| No. | Nama Barang | Harga | Jml | Total Harga |");
    gotoxy(4,4);printf("-----");
    do
    {
        gotoxy(4,y);printf("| | | | |");
        gotoxy(6,y);printf("%d.",i);
        gotoxy(11,y);scanf("%s",&brg);fflush(stdin);
        gotoxy(31,y);scanf("%f",&harga);
        gotoxy(43,y);scanf("%f",&jml);
        tot1=jml*harga;
        gotoxy(50,y);printf("%.2f",tot1);
        tot2+=tot1;
        y++;
        i++;
    }
    while(getch()!=27); // Sampai penekanan tombol ESC

    gotoxy(4,y);printf("-----");
    gotoxy(35,y+1);printf("Total : Rp ");
    gotoxy(50,y+1);printf("%.2lf",tot2);
    gotoxy(35,y+2);printf("Diskon : ");scanf("%f",&disc);
    diskon=(disc/100)*tot2;
    gotoxy(17,y+4);printf("Total Yang Harus dibayar : Rp ");
    gotoxy(50,y+4);printf("%.2lf",tot2-diskon);
    gotoxy(44,y+5);printf("=====");
}

```

Hasil eksekusi dari program tersebut adalah:

No.	Nama Barang	Harga	Jml	Total Harga
1.	Rinso	500	3	1500.00
2.	Pepsodent	5500	1	5500.00
3.	Lifebuoy	1750	2	3500.00

Total : Rp 10500.00
Diskon : 0

Total Yang Harus dibayar : Rp 10500.00

Struktur For

Struktur ini biasa digunakan untuk mengulang proses yang telah diketahui jumlah perulangannya.

Contoh program:

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

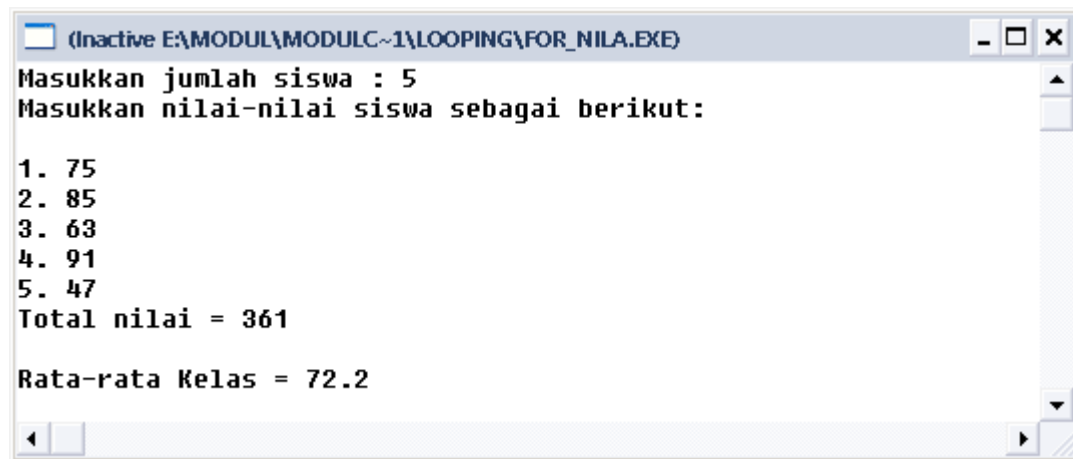
main()
{
    int i,jsis;
    float nil,tot,rata;

    clrscr();
    tot=0;

    printf("Masukkan jumlah siswa : ");scanf("%d",&jsis);
    printf("Masukkan nilai-nilai siswa sebagai berikut:\n\n");
    for(i=1;i<=jsis;i++)
    {
        printf("%d. ",i);scanf("%f",&nil);
        tot=tot+nil;
    }
    rata=tot/jsis;
    printf("Total nilai = %g\n",tot);
    printf("\nRata-rata Kelas = %g",rata);

    getch();
}
```

Hasil eksekusi program adalah:



```
(Inactive E:\MODUL\MODULC~1\LOOPING\FOR_NILA.EXE)
Masukkan jumlah siswa : 5
Masukkan nilai-nilai siswa sebagai berikut:

1. 75
2. 85
3. 63
4. 91
5. 47
Total nilai = 361

Rata-rata Kelas = 72.2
```

Loop di dalam Loop

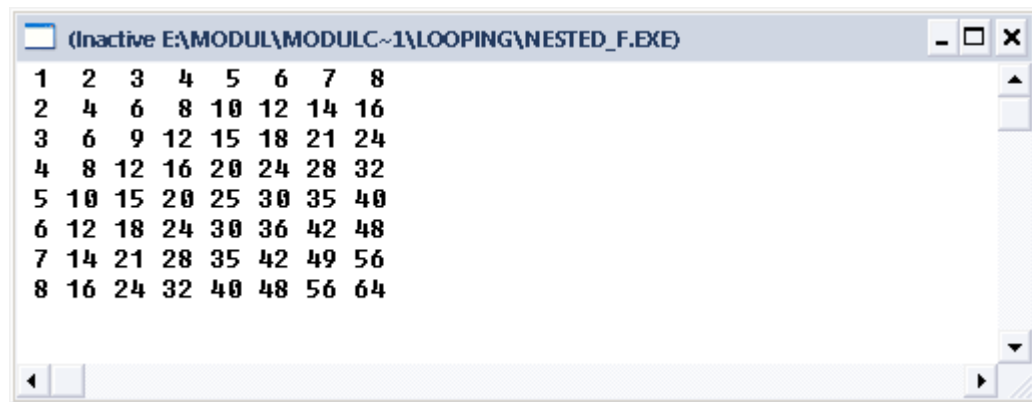
Dalam suatu loop bisa terkandung loop yang lain. Loop yang terletak di dalam loop biasa disebut dengan *Nested Loop*. Salah satu contoh nested loop misalnya pada permasalahan untuk membuat tabel perkalian. Loop luar digunakan untuk mengatur baris, sedangkan loop dalam untuk mencetak satu deret hasil perkalian dalam satu baris. Berikut source codenya:

```
#include<stdio.h>
#define MAKS      8

main()
{
    int brs,kol,h_kali;

    for(brs=1;brs<=MAKS;brs++)
    {
        for(kol=1;kol<=MAKS;kol++)
        {
            h_kali=brs*kol;
            printf("%2d ",h_kali);
        }
        printf("\n");
    }
}
```

Hasil eksekusi program adalah:



```
(Inactive E:\MODUL\MODULC~1\LOOPING\NESTED_F.EXE)
1  2  3  4  5  6  7  8
2  4  6  8 10 12 14 16
3  6  9 12 15 18 21 24
4  8 12 16 20 24 28 32
5 10 15 20 25 30 35 40
6 12 18 24 30 36 42 48
7 14 21 28 35 42 49 56
8 16 24 32 40 48 56 64
```

Latihan

1. Buat program untuk mencari total dan rata-rata dari sejumlah bilangan
2. Buat program untuk menghitung nilai faktorial

MODUL 6

Nama Percobaan : FUNGSI

Tujuan : Mampu membuat dan mengaplikasikan suatu fungsi ke dalam program

Materi:

Fungsi merupakan blok dari kode yang dirancang untuk melaksanakan tugas khusus. Tujuan fungsi adalah agar program menjadi terstruktur, sehingga mudah dipahami dan mudah dikembangkan. Selain itu fungsi dapat mengurangi pengulangan kode.

Bentuk umum definisi sebuah fungsi adalah sebagai berikut:

```
Tipe_fungsi nama_fungsi(parameter_fungsi)
{
    Statement-statement (body)
}
```

Hal-hal yang perlu diperhatikan dalam penggunaan fungsi:

1. Kalau tipe fungsi tidak disebutkan, maka akan dianggap fungsi dengan nilai output integer, maka jika output bukan integer diperlukan pendefinisian penentu tipe fungsi.
2. Untuk fungsi yang tidak mempunyai nilai keluaran maka dimasukkan ke dalam tipe void
3. Pernyataan akhir fungsi berupa pernyataan “*return*”.

Contoh program 1:

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

void info_program();

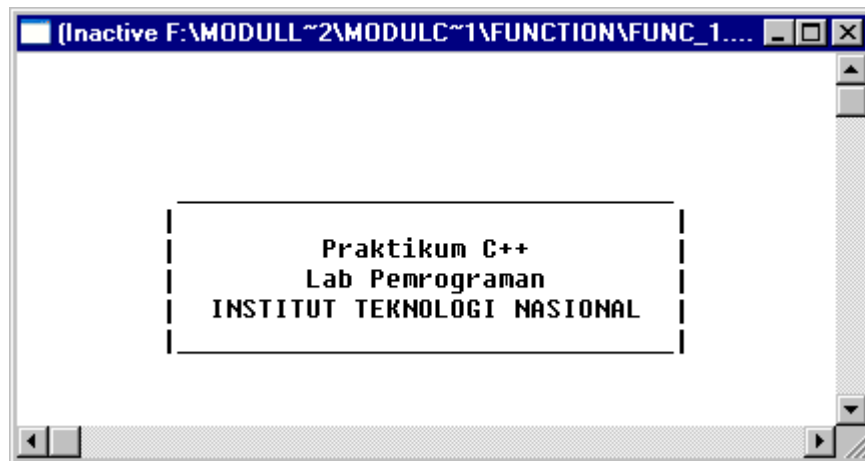
main()
{
    info_program();
}
```

```
void info_program()
{
    long i;
    int n,y=0;

    for(n=1;n<=20;n++)
    {
        clrscr();
        y++;
        gotoxy(25,y), printf("_____");
        gotoxy(25,y+1);printf("|");
        gotoxy(25,y+2);printf("|          Praktikum C++      |");
        gotoxy(25,y+3);printf("|          Lab Pemrograman    |");
        gotoxy(25,y+4);printf("| INSTITUT TEKNOLOGI NASIONAL |");
        gotoxy(25,y+5);printf("|_____");
        y++;

        // proses penundaan
        for(i=0;i<=2000000;i++)
        ;          // tunda sebentar
    }
}
```

Dalam program tersebut, `main()` akan memanggil fungsi `info_program()` untuk menampilkan border dengan tulisan “Worksop C++....” sebanyak 20 kali. Di dalam fungsi `info_program()`, agar output tidak menumpuk maka dipakai fungsi `clrscr()` di dalam looping, sehingga sebelum output dicetak ke layar maka layar akan dibersihkan terlebih dahulu. Selain itu dibuat proses penundaan atau delay, dimana variabel “i” akan terus diproses selama masih kurang dari 2000000, setelah itu program akan mencetak tampilan border kembali. Sehingga akan tampak bahwa border bergerak dari atas ke bawah. Di sini fungsi tidak memberikan pernyataan *return* karena fungsi `info_program()` tidak memberikan output kepada `main()`. Berikut hasil eksekusi program:



Itu adalah jika hanya memanggil satu fungsi, bagaimana jika banyak fungsi yang dipanggil? Berikut ini adalah contoh program yang terdiri dari beberapa fungsi dan memiliki nilai output:

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

// Define golongan
#define a      1500000.00
#define b      900000.00
#define c      750000.00

// Inisialisasi fungsi
float tunj_kel(float gaji);
float pjk(float gaji);
float gaji_bersih(float gaji,float pjk,float tunj);

main()
{
    char nip[10],nama[25],gol;
    int status;
    float gj,tunj,pajak,gj_bersih;

    printf("NIP    : ");gets(nip);fflush(stdin);
    printf("Nama  : ");gets(nama);fflush(stdin);
    printf("Golongan (a,b,atau c) :");scanf("%c",&gol);

    if((gol=='A')||(gol=='a'))
        gj=a;
    else
```

```
        if((gol=='B')||(gol=='b'))
            gj=b;
        else
            if((gol=='C')||(gol=='c'))
                gj=b;
            else
                {   printf("Input Tidak Terdefinisi");
                    return 0;
                }

    printf("Status :\\n"
    "\\t1]Single\\n"
    "\\t2]Menikah\\n"
    "\\t3]Duda/Janda\\n");scanf("%d",&status);

    if(status==2)
        tunj=tunj_kel(gj);

    pajak=pjk(gj);
    gj_bersih=gaj_bersih(gj,pajak,tunj);

    printf("\\n-----\\n");
    printf("\\nGaji Pokok      : Rp %.2f\\n",gj);
    printf("Tunjangan Keluarga  : Rp %.2f\\n",tunj);
    printf("Pajak                : Rp %.2f\\n",pajak);
    printf("Gaji Bersih          : Rp %.2f\\n",gj_bersih);
    printf("\\n-----\\n");
    getch();
    return 0;
}

float tunj_kel(float gaji)
{
    return(0.1*gaji);
}

float pjk(float gaji)
{
    return(0.25*gaji);
}

float gj_bersih(float gaji,float pjk,float tunj)
{
    return((gaji+tunj)-pjk);
}
```


Pada program di atas, fungsi utama / main() memanggil beberapa fungsi. Yaitu fungsi

- float tunj_kel(float gaji);
- float pjk(float gaji);
- float gaji_bersih(float gaji,float pjk,float tunj);

fungsi akan dieksekusi berdasarkan data yang diinput oleh user. Untuk menentukan suatu parameter, maka tipe data antara variabel lokal dan parameter fungsi harus sama.

Begitu juga dengan hasil eksekusi atau output, tipenya harus sama dengan variabel lokal yang digunakan untuk memanggil fungsi tersebut. Seperti misalnya variabel pajak untuk memanggil fungsi pjk() tipenya sama-sama bertipe float.

Hasil eksekusi program adalah:

```

E:\MODUL\MODULC~1\FUNCTION\GAPOK.EXE
NIP      : 10065788122
Nama     : Ahmad Subejo
Golongan (a,b,atau c) :b
Status   :
          1]Single
          2]Menikah
          3]Duda/Janda
2

-----

Gaji Pokok           : Rp 900000.00
Tunjangan Keluarga   : Rp 90000.00
Pajak                : Rp 225000.00
Gaji Bersih          : Rp 765000.00
  
```

Sedangkan contoh program berikut ini adalah fungsi dengan menggunakan suatu variabel static dimana variabel static ini nilainya bersifat tetap.

```

#include<stdio.h>
long int faktorial(long int x);

main()
{
    int i,n;
    long int fakt;

    printf("Berapa Faktorial ? ");scanf("%d",&n);
  
```

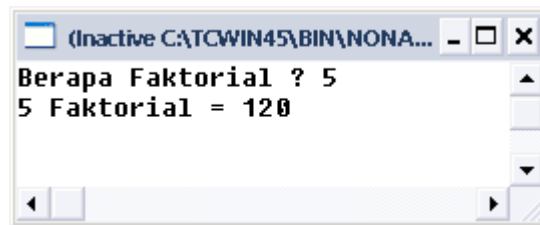
```
        if(n<=0)fakt=0;
        else
            for(i=1;i<=n;i++)fakt=faktorial(i);

        printf("%d Faktorial = %ld\n",n,fakt);
    }

    long int faktorial(long int x)
    {
        static long int F=1;

        F=F*x;
        return(F);
    }
```

Hasil eksekusinya adalah:



Pada contoh ini, variabel F di fungsi **faktorial** () merupakan variabel lokal yang bersifat static. Variabel F akan dikalikan dengan x untuk mendapatkan nilai F yang baru yang akan disimpan di memori sebagai nilai tetap. Di fungsi utama, fungsi **faktorial** () dipanggil sebanyak n kali. Sehingga untuk mempertahankan nilai F yang lama, maka variabel F harus bersifat static. Jika variabel F tidak bersifat static maka setiap kali fungsi **faktorial** () dipanggil, nilai variabel F akan mempunyai nilai awal 1.

Latihan

1. Buat program mencari luas segitiga dengan menggunakan fungsi
2. Buat fungsi untuk menentukan bilangan genap atau ganjil

MODUL 7

Nama Percobaan : ARRAY

Tujuan : Mampu memahami pembentukan dan pemrosesan suatu array

Materi:

Array merupakan kumpulan dari nilai-nilai data yang bertipe sama dalam urutan tertentu yang menggunakan nama yang sama. Letak atau posisi dari elemen array ditunjukkan oleh suatu index. Array ada 3 macam, yaitu:

Array Berdimensi Satu

Bentuk umum dari array berdimensi satu adalah:

Tipe nama_var[ukuran];

Dengan tipe adalah jenis data elemen array, sedangkan ukuran untuk menyatakan jumlah maksimal elemen array. Contoh program:

```
#include<stdio.h>
#include<conio.h>
#define MAKS      5

main()
{
    int i;
        float tot_nil,rata,maks,min;
        float nilai[MAKS];
        char jwb;
        do
        {
            clrscr();
            for(i=0;i<MAKS;i++)
            {
                printf("Nilai ke-%d : ",i+1);
                scanf("%f",&nilai[i]);
            }

            tot_nil=0;
```

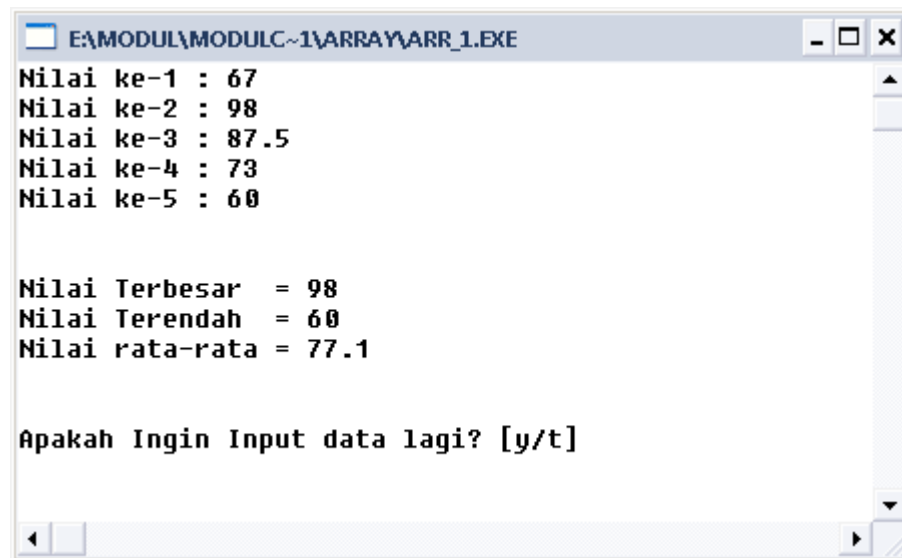
```
maks=nilai[0];
min=nilai[0];
for(i=0;i<MAKS;i++)
{
    tot_nil+=nilai[i];
    if(nilai[i]>maks)
        maks=nilai[i];

    if(nilai[i]<min)
        min=nilai[i];
}
rata = tot_nil/MAKS;

printf("\n\nNilai Terbesar = %g\n"
       "Nilai Terendah = %g\n"
       "Nilai rata-rata = %g\n",maks,min,rata);

printf("\n\nApakah Ingin Input data lagi? [y/t] ");
jwb=getche();
}
while(jwb=='Y'||jwb=='y');
```

Dalam program tersebut, variabel nilai adalah sebuah array dengan jumlah 8 elemen. Indeks array dimulai dari 0-7. sehingga proses input dimulai dari nilai[0]. Dengan array, proses pencarian nilai maksimum atau minimum akan lebih mudah seperti pada program ini. Nilai akan dibandingkan mulai nilai ke-0 sampai nilai ke-7, setelah akhir pengulangan maka nilai tersebut akan diketahui. Begitu juga untuk mencari total nilai. Hasil eksekusi program akan ditampilkan:



```
E:\MODUL\MODULC~1\ARRAY\ARR_1.EXE
Nilai ke-1 : 67
Nilai ke-2 : 98
Nilai ke-3 : 87.5
Nilai ke-4 : 73
Nilai ke-5 : 60

Nilai Terbesar = 98
Nilai Terendah = 60
Nilai rata-rata = 77.1

Apakah Ingin Input data lagi? [y/t]
```

Bagaimana dengan array karakter? Array dari suatu karakter identik dengan string, seperti nama, jurusan, dan lain-lain. Contoh berikut akan menjelaskan tentang array karakter.

```
#include"stdio.h"
#include"conio.h"
#include"string.h"

main()
{
    char jur[25],jenjang[10],nim[7],nama[25];

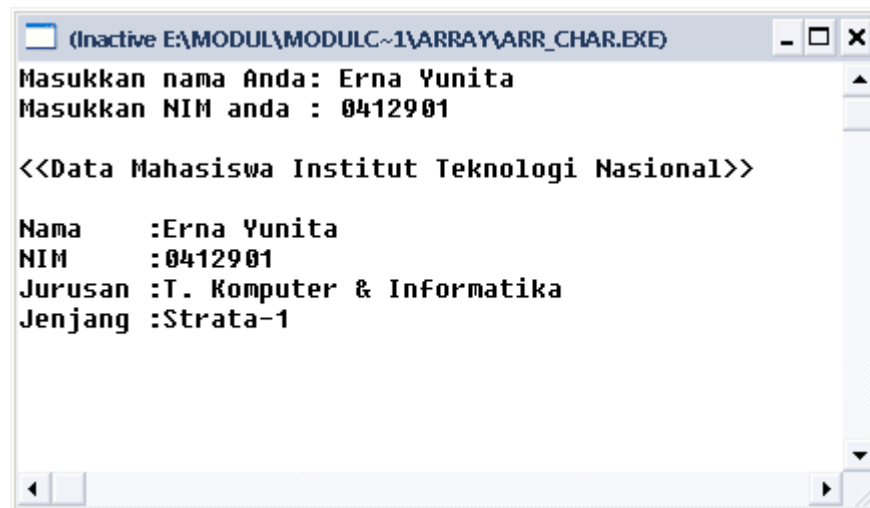
    printf("Masukkan nama Anda: ");gets(nama);
    printf("Masukkan NIM anda : ");gets(nim);

    switch(nim[3])
    {
        case '0':strcpy(jur,"Elektronika");
            break;
        case '1':strcpy(jur,"Sistem Tenaga");
            break;
        case '2':strcpy(jur,"T. Komputer & Informatika");
            break;
        case '3':strcpy(jur,"D3 Elektronika");
            break;
        default: printf("\nAnda salah memasukkan NIM. Coba periksa lagi!!!");
            break;
    }
}
```

```
if(nim[2]=='1')
    strcpy(jenjang,"Strata-1");
else
    if(nim[3]=='3')
        strcpy(jenjang,"Diploma-3");
    else
        printf("\nAnda salah memasukkan NIM. Coba periksa lagi!!");

printf("\n<<Data Mahasiswa Institut Teknologi Nasional>>\n\n");
printf("Nama   :%s\n",nama);
printf("NIM    :%s\n",nim);
printf("Jurusan   :%s\n",jur);
printf("Jenjang   :%s\n",jenjang);
getch();
}
```

Hasil eksekusi adalah:



```
(Inactive E:\MODUL\MODULC~1\ARRAY\ARR_CHAR.EXE)
Masukkan nama Anda: Erna Yunita
Masukkan NIM anda : 0412901

<<Data Mahasiswa Institut Teknologi Nasional>>

Nama      :Erna Yunita
NIM       :0412901
Jurusan   :T. Komputer & Informatika
Jenjang   :Strata-1
```

Array Berdimensi Dua

Array dua dimensi merupakan array yang terdiri dari m buah baris dan n buah kolom. Bentuknya dapat berupa matriks atau tabel. Misal ada variabel `x[3][4]`, berarti variabel `x` memiliki jumlah data maksimal 3 dan panjang karakter sebanyak 4 atau 4 kolom pada matriks. Contoh berikut ini penerapan array dua dimensi pada suatu string.

```
#include<stdio.h>
#include<string.h>
#define maks      5
#define panjang   20

main()
{
    static char nama[maks][panjang];    //mendeklarasikan var bersifat statis

    int i,j,jum_dt;
    char temp[panjang];

    for(i=0;i<maks;i++)
    {
        printf("Nama ke-%d: ",i+1);
        gets(nama[i]);
    }
    for(i=0;i<maks-1;i++)
        for(j=i+1;j<maks;j++)
            if(strcmp(nama[i],nama[j])>0)
            {
                strcpy(temp,nama[i]);
                strcpy(nama[i],nama[j]);
                strcpy(nama[j],temp);
            }

    puts("\nData Setelah Diurutkan: \n");

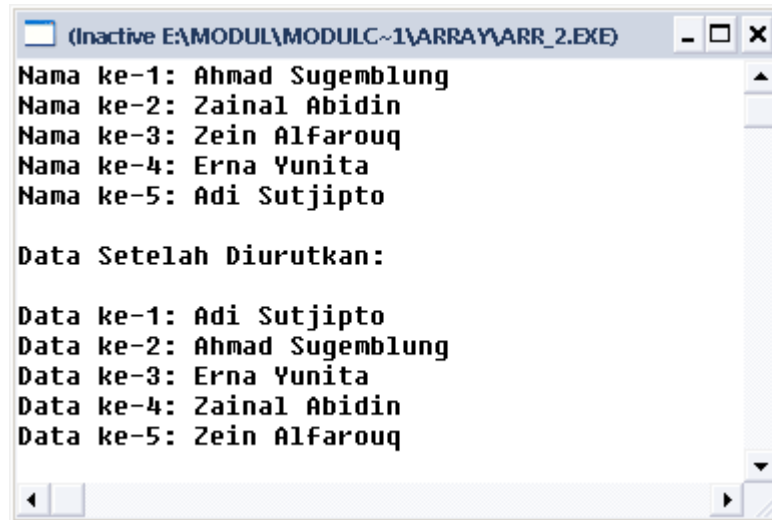
    for(i=0;i<maks;i++)
        printf("Data ke-%d: %s\n",i+1,nama[i]);
}
```

Mula-mula variabel nama sebagai suatu array dua dimensi dengan jumlah 5 dan panjang 20. perlu diingat bahwa indeks array pada bahasa C++ dimulai dari 0, maka untuk input data dimulai dari data ke 0 sampai 4 dimana panjang karakter maksimal 20. setelah itu, data akan dibandingkan antara data pertama dengan data berikutnya dengan menggunakan fungsi string `strcmp()`.

Hasil perbandingan bertipe integer. Jika negatif berarti string pertama kurang dari string kedua. Jika nol maka string pertama = string kedua. Tetapi jika hasilnya positif maka string pertama lebih besar dari string kedua. Pada program di atas nama akan

diurutkan secara ascending, maka jika hasilnya positif atau string pertama > string kedua maka lakukan proses pertukaran string menggunakan fungsi strcpy(). Fungsinya untuk menyalin suatu string asal ke variabel tujuan.

Hasil dari eksekusi program tersebut akan ditampilkan:



Array Multi Dimensi

Array multi dimensi merupakan array yang mempunyai ukuran lebih dari dua. Bentuk pendeklarasian array sama saja dengan array dimensi satu maupun dua. Misalnya variabel: char nama[5][3][15]. Berarti ada suatu data sebanyak 5 buah. Dalam tiap data tersebut ada 3 data lagi yang panjangnya 15.

Contoh program misalnya untuk menentukan beberapa nama dalam beberapa kelas.

```
#include <stdio.h>
main()
{
    int i,j;
    char nama[5][3][15]={
        "Adi","Budi","Candra",
        "Erni","Fatimah","Goedono",
        "Henri","Ipung","John",
        "Kiki","Lukas","Mike",
        "Nono","Ogut","Puspa"
    };

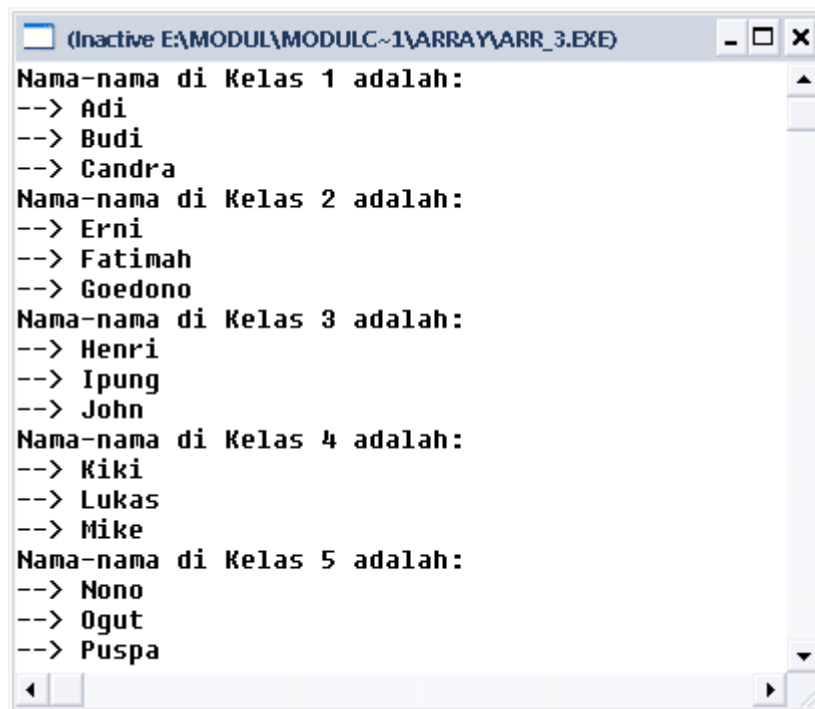
    for(i=0;i<5;i++)
    {
```



```
        printf("Nama-nama di Kelas %d adalah:\n",i+1);

        for(j=0;j<3;j++)
            printf("--> %s\n",nama[i][j]);
    }
}
```

Hasil eksekusi program:



```
(Inactive E:\MODUL\MODULC~1\ARRAY\ARR_3.EXE)
Nama-nama di Kelas 1 adalah:
--> Adi
--> Budi
--> Candra
Nama-nama di Kelas 2 adalah:
--> Erni
--> Fatimah
--> Goedono
Nama-nama di Kelas 3 adalah:
--> Henri
--> Ipung
--> John
Nama-nama di Kelas 4 adalah:
--> Kiki
--> Lukas
--> Mike
Nama-nama di Kelas 5 adalah:
--> Nono
--> Ogut
--> Puspa
```

Latihan

1. Buat program untuk mencari nilai x dalam suatu array, program akan menampilkan nilai indeks dari x tersebut. Misalkan dalam suatu array 4,6,1,3,9..jika ingin dicari nilai 1 maka outputnya adalah 2 karena angka 1 terletak pada indeks ke-2, tetapi jika tidak ditemukan maka tampilkan pesan kesalahan

MODUL 8

Nama Percobaan : POINTER

Tujuan : Mampu mengetahui dan memahami tentang suatu pointer

Materi:

Konsep Dasar

Suatu pointer(variabel penunjuk) adalah suatu variabel yang berisi dengan alamat lokasi suatu memori tertentu. Jadi suatu pointer bukan berisi dengan suatu nilai data, tetapi berisi dengan satu alamat. Misalnya **X** adalah suatu variabel yang berisi dengan nilai '**j**'. **X** bukan variabel penunjuk. Nilai dari **X** ini oleh compiler akan diletakkan di suatu lokasi memori tertentu. Nilai ini dapat diakses jika diketahui alamat memorinya. Alamat dari nilai **X** ini dapat diketahui dari ungkapan **&X**. Misalnya alamat dari nilai **X** ini akan disimpan di suatu variabel, maka dapat dituliskan sebagai **Alamat_X=&X**. **Alamat_X** adalah variabel pointer yang menunjuk ke lokasi dimana nilai **X** disimpan.

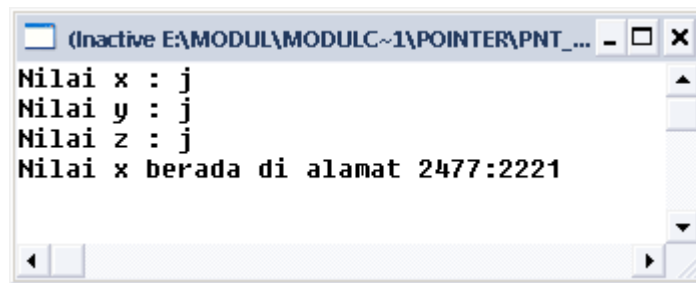
Contoh:

```
#include<stdio.h>
main()
{
    char *alamat_x,x,y,z;

    x='j';           // x berisi 'j'
    alamat_x=&x;      // alamat_x berisi alamat dari nilai x
    y=x;             // y berisi data yang sama dengan x
    z=*alamat_x;      // z berisi nilai yang alamatnya
                    // ditunjuk oleh alamat_x

    printf("Nilai x : %c\n",x);
    printf("Nilai y : %c\n",y);
    printf("Nilai z : %c\n",z);
    printf("Nilai x berada di alamat %p\n",alamat_x);
}
```

Jika program ini dijalankan akan didapatkan hasil:



Operasi Pointer

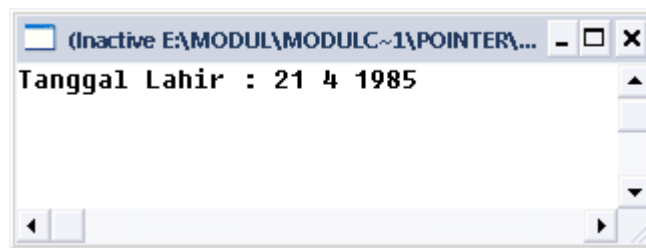
Pembahasan kali ini, kita langsung ke operasi pointer pada suatu array.

Contoh program:

```
#include<stdio.h>
main()
{
    int tglahir[3],*tgl;
    int i;

    tgl=tglahir;           // pointer tgl diisi alamat array
    tglahir[0]=21;          // tglahir[0] diisi 21
    tglahir[1]=*tgl-17; // tglahir diisi dengan nilai tglahir[0]-17 → tglahir[1] = 4
    tglahir[2]=*tgl+1964; // tglahir diisi dengan tglahir[0]+1964 →
                        // tglahir[2] = 1984
    printf("Tanggal Lahir : ");
    for(i=0;i<3;i++)
        printf("%d ",*(tgl+i));
}
```

Mula-mula variabel tglahir diisi dengan tanggal, bulan, tahun melalui pengaksesan variabel pointer yang menunjuk alamat array tglahir. Untuk menampilkan tglahir[] melalui variabel pointer digunakan cara `*(tgl + i)` atau bisa juga dengan `*tgl++`. Setelah dijalankan maka hasil eksekusi akan ditampilkan:



Berikut ini adalah contoh suatu pointer sebagai output dari fungsi:

```
#include<stdio.h>

char *nama_bulan(int n);
main()
{
    int bl;

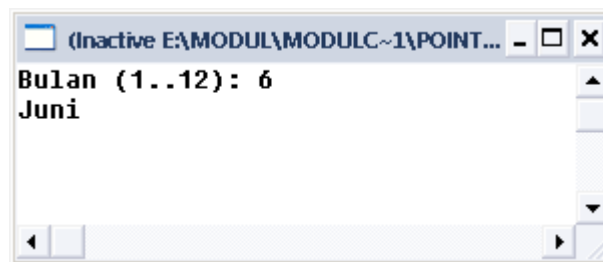
    printf("Bulan (1..12): ");scanf("%d",&bl);
    printf("%s\n",nama_bulan(bl));
}

char *nama_bulan(int n)
{
    static char *bulan[]={
        "Kode Bulan Salah",
        "Januari","Februari","Maret","April","Mei","Juni","Juli","Agustus",
        "September","Oktober","Nopember","Desember" };

    return((n<1 || n>12)? bulan[0]:bulan[n]);
}
```

Pada program di atas, output fungsi berupa pointer yang menunjuk ke obyek bertipe char. Di dalam fungsi, array dideklarasikan dan sekaligus diinisialisasi agar menunjuk sejumlah string yang menyatakan nama bulan.

Hasilnya adalah sebagai berikut:



Latihan

Buatlah program untuk menghitung total dari sebuah nilai yang bertipe pointer, dimana nilai adalah sebagai pengganti dari sebuah array.

MODUL 9

Nama Percobaan : STRUKTUR

Tujuan : Mampu mengetahui dan memahami tentang suatu struktur

Materi:

Struktur yaitu pengelompokan dari variabel-variabel yang bernaung dalam satu nama yang sama. Struktur biasa dipakai untuk mengelompokkan beberapa informasi yang berkaitan dengan sebuah kesatuan, atau biasanya disebut dengan *record*.

Mendeklarasikan Struktur

Cara mendeklarasikan struktur adalah dengan menggunakan kata kunci **struct**. Contoh:

```
struct struktur_psd{  
    char kode[5];  
    char nama[30];  
    int unit;  
    float harga;  
};
```

```
Struct struktur_psd persediaan_pusat, persediaan_cabang;
```

Struktur ini diberi nama `struktur_psd` yang mempunyai lima buah elemen, yaitu `kode[5]` dan `nama[30]` bertipe `char`, sedangkan `unit` bertipe `int` dan `harga` bertipe `float`. `struktur_psd` merupakan nama tipe data struktur dari lima elemen tersebut, bukan nama dari suatu variabel struktur. Sedangkan **`persediaan_pusat`** dan **`persediaan_cabang`** merupakan variabel-variabel yang mempunyai tipe data `struktur_psd`. Cara lain untuk mendeklarasikan struktur sebagai berikut:

```
struct struktur_psd{  
    char kode[5];  
    char nama[30];  
    int unit;  
    float harga; }persediaan_pusat, persediaan_cabang;
```

Untuk deklarasi seperti di atas, sebenarnya kata struktur_psd dapat juga tidak dituliskan. Ada juga penulisan struktur seperti di bawah ini:

```
typedef struct{
    char kode[5];
    char nama[20];
    int jml;
    float harga;
}psd;

psd persediaan_barang;    // pendefinisian nama variabel baru
```

Dengan menggunakan “typedef”, struktur psd didefinisikan kembali dengan nama baru tanpa menggunakan kata “struct” di awal nama baru tersebut.

Mengakses Elemen-elemen Struktur

Untuk mengakses elemen-elemen struktur secara individual dengan menyebutkan nama variabel strukturnya diikti oleh operator titik (‘.’) dan nama dari elemen strukturnya. Berikut contoh program untuk mengakses elemen struktur:

```
#include<stdio.h>
main()
{
    struct{
        float jari;
        float keliling;
        float luas;
    }lingkaran;    // nama variabel struktur

    printf("Masukkan Jari-jari Lingkaran : ");scanf("%f",&lingkaran.jari);

    lingkaran.keliling = 2 * 3.14259 * lingkaran.jari;
    lingkaran.luas     = 3.14259 * lingkaran.jari * lingkaran.jari;

    printf("Keliling Lingkaran = %g\n",lingkaran.keliling);
    printf("Luas LIngkaran    = %g\n",lingkaran.luas);
}
```

Hasil eksekusi program adalah:



Suatu struktur dapat juga berisi struktur lain, cara mengaksesnya juga berbeda. Berikut contoh struktur yang berisi struktur lain serta cara mengaksesnya.

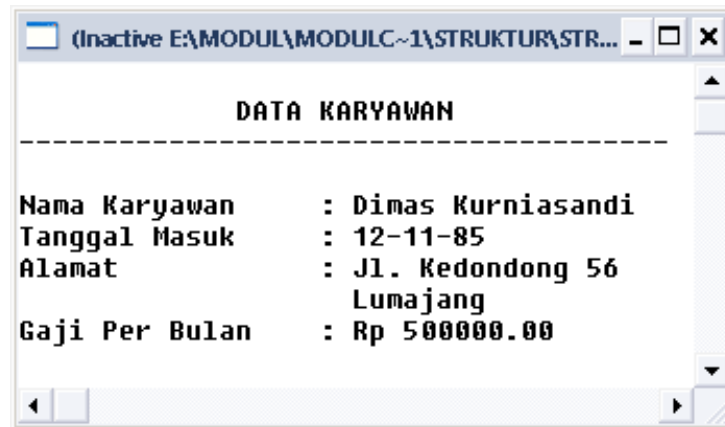
```
#include<stdio.h>
main()
{
    struct tgl{
        unsigned int hari;
        unsigned int bulan;
        unsigned int tahun;
    };

    struct alamat{
        char jalan[30];
        char kota[20];
    };

    struct{
        char nama[40];
        struct tgl masuk;
        struct alamat tinggal;
        float gaji;
    }karyawan={
        "Dimas Kurniasandi",        // nama karyawan
        12,                          // hari tgl masuk
        11,                          // bulan tgl masuk
        85,                          // tahun tgl masuk
        "Jl. Kedondong 56",          // jalan alamat tinggal
        "Lumajang",                  // kota alamat tinggal
        500000                        // gaji karyawan
    };

    printf("\n\n      DATA KARYAWAN      \n");
    printf("\n-----\n\n");
    printf("Nama Karyawan   : %s\n",karyawan.nama);
    printf("Tanggal Masuk   : %2d-%2d-%2d\n",karyawan.masuk.hari,
        karyawan.masuk.bulan,karyawan.masuk.tahun);
    printf("Alamat           : %s\n",karyawan.tinggal.jalan);
    printf("                  : %s\n",karyawan.tinggal.kota);
    printf("Gaji Per Bulan   : Rp %.2f\n",karyawan.gaji);
}
```

Pada program tersebut, variable karyawan merupakan struktur yang berisi struktur lain. Yaitu struktur tgl dan struktur alamat. Jadi, elemen lahir pada karyawan memiliki 3 elemen, yaitu elemen dari struktur tgl (hari, bulan, tahun). Sedangkan elemen tinggal memiliki 2 elemen yaitu elemen dari struktur alamat (jalan dan kota). Hasilnya dapat dilihat pada tampilan di bawah ini:



Latihan

Buat suatu program yang mengaplikasi suatu struktur mahasiswa. Struktur mahasiswa berupa array terdiri dari nim, nama, dan ip. Semua data adalah input dari user minimal 5 mahasiswa, lalu tampilkan dalam bentuk tabel sederhana.