MODUL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN (C++)

Disusun oleh :

Ultach Enri, S.Kom., M.Kom

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SINGAPERBANGSA KARAWANG
2018

DAFTAR ISI

PERTEMUAN 1

PENGENALAN C++

C++ dikenalkan oleh **Bjarne Stroustrup** pada tahun 1980, dikembangkan dari bahasa C yang telah dikenalkan oleh **Dennis Ritchie** pada tahun 1972. Nama C digunakan sebagai penerus bahasa B yang dikenalkan **Ken Thompson** yang merupakan penerus dari bahasa BPCL (*Basic Combined Programming Language*) yang dikenalkan oleh **Martin Richard** pada tahun 1967. Nama C++ menunjukkan adanya penambahan, yaitu class, tetapi pada awalnya dinamai dengan "*C with class*" yang dapat memfasilitasi pemrograman berorientasi objek.

Nama Bahasa	Penemu
Bcpl	Martin Richard
B	Ken Thompson
C	Dennis Ritchie
C++	Bjarne Strouptrup

Pada tahun 1998, C++ distandardisasikan oleh *International Standard Organization* (**ISO**) dan *American National Standards Institute* (**ANSI**). Standar baru ini memasukkan *Standard Template Library* (**STL**) yang aslinyadikembangkan oleh **Alexander Stepanov** pada tahun 1979. Terminologi "*C++ Standard*" merujuk ke versi standar dari bahasa ini.

Bahasa C++ bisa disebut bahasa pemrograman tingkat menengah (*middle level programming language*). Dalam hal ini, level yang di maksudkan adalah kemampuan mengakses fungsi-fungsi dan perintah-perintah dasar bahasa mesin / hardware (*machine basic instruction set*). Semakin tinggi tingkat bahasa pemrograman akan semakin mudah bahasa pemrograman tersebut dipahami oleh manusia, namun membawa pengaruh semakin berkurang kemampuan untuk mengakses langsung instruksi dasar bahasa mesin. Bahasa C++ bisa digolongkan dalam bahasa tingkat bahasa tingkat tinggi dalam perspektif mudahnya dipahami manusia, namun bahasa tersebut menyediakan kemampuan seperti yang ada pada bahasa tingkat rendah, misalnya tersedianya berbagai pemenuhan kebutuhan akan operasi bit, operasi byte, pengaksesan memori, dan sebagainya.

C diambil sebagai landasan dari C++ adalah karena keportabilitasan C yang memungkinkan diterapkannya ke berbagai mesin, dari PC hingga mainframe, serta pada pelbagai sistem operasi (DOS, UNIX, VMS dan sebagainya)

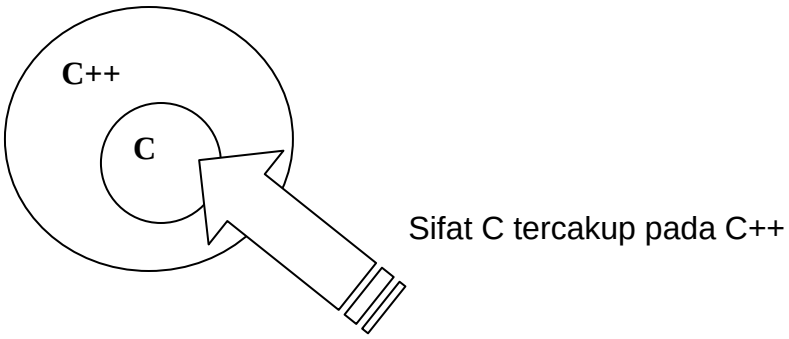
Keistimewaan yang sangat berarti pada C++ adalah karena bahasa ini mendukung pemrograman yang berorientasi obyek (Object Oriented Programming) tetapi sekali lagi C++ hanyalah bahasa yang bersifat hibrid, bukan bahasa murni yang berorientasi obyek.

Tujuan utama pembuatan C++ adalah untuk meningkatkan produktivitas pemrogram dalam membuat aplikasi. PBO dan C++ dapat mengurangi kekompleksitasan, terutama pada program yang besar yang terdiri dari 10.000 baris atau lebih.

Greg Perry pada tahun 1993 menyatakan C++ dapat meningkatkan produktivitas pemrogram lebih dari dua kali dibandingkan bahasa prosedural seperti C, PASCAL dan BASIC karena kode yang ditulis dengan C++ lebih mudah untuk digunakan kembali pada program-program lain

1.1. Struktur Program C++

Struktur program C++ sama seperti Struktur program C yang terdahulu. Struktur program C++ terdiri sejumlah blok fungsi, setiap fungsi terdiri dari satu atau beberapa pernyataan yang melaksanakan tugas tertentu.




```
#include <file_include>
main()
{
        pernyataan_program
}
```

Header

File include dasar input output :

#include <stdio.h> → *Standard Input Output*
Perintah Masukan stdio.h → scanf
Perintah keluaran stdio.h → printf & puts

#include <conio.h> → *Console Input Output Header*
Mendefinisikan keperluan untuk io pada mode console
Perintah masukan conio.h → getch
Perintah untuk membersihkan layar → clrscr

#include <iostream.h> → *Input Output Stream Header*
Mendefinisikan objek standar cin, cout dan endl
Perintah masukan iostream.h → cin
Perintah keluaran iostream.h → cout

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

main()
{
int u=8;
char t='H';
clrscr();

printf ("%c merupakan Abjad yang Ke - %d", t, u);
getch();
}
```

1.2. Pengenalan IDE Borland C++

IDE (*Integrated Development Environment*) merupakan lembar kerja terpadu untuk pengembangan program. IDE dari Borland C++, dapat digunakan untuk :

- Menulis Naskah Program
- Mengkompilasi Program (Compile)
- Melakukan Pengujian Program (Debugging)
- Mengaitkan Object dan Library ke Program (Linking)
- Menjalankan Program (Running)

IDE pada Borland C++, terbagi menjadi 4 bagian, yaitu :

a. Menu Utama (Menubar)
   Menu utama terdiri dari File, Edit, Search, Run, Compile, Debug, Project, Options, Window dan Help.
b. Jendela Editor
   Tempat untuk pengetikan program dan membuat program. Jika pertama kali anda membuat program, nama file jendela editor adalah NONAME01.cpp
c. Jendela Message
   Tempat untuk menampilkan pesan-pesan pada proses kompilasi dan link program.
d. Baris Status
   Baris dimana menampilkan keterangan-keterangan pada saat anda mengaktifkan menu bar dan sub menu.

1.3. Program C++

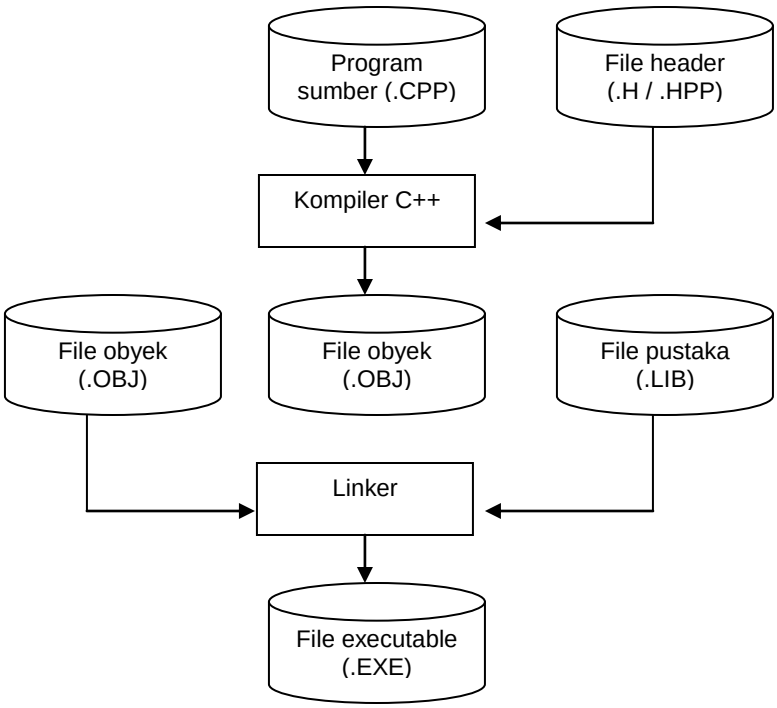
Program C++ dapat ditulis menggunakan sembarang editor teks, seperti EDIT (milik DOS), wordstar, SideKick ataupun menggunakan editor bawaan dari kompiler.
Contoh Program C++ :

```
#Include <iostream.h>

void main()
{
        cout<<"Hi How are you today?\n";
}
```

Program C++ biasa ditulis dengan nama ekstensi .CPP (dari kata C plus plus). Agar program ini bisa dijalankan (dieksekusi), program harus dikompilasi terlebih dahulu dengan menggunakan kompiler C++.

Pada saat pengkompilasian, program sumber (.CPP) bersama file-file header (berekstensi .H atau .HPP) akan di terjemahkan oleh kompiler C++ menjadi kode obyek (.OBJ). File obyek ini berupa file dalam format biner (berkode 0 dan 1).
Selanjutnya file obyek ini bersama-sama dengan file obyek lain serta file pustaka (.LIB) dikaitkan menjadi satu oleh linker. Hasilnya berupa file yang bersifat executable. File inilah yang bisa dijalankan dari sistem operasi secara langsung.



Latihan 1 :
Buatlah tampilan sbb:

```
        Identitas

Nama          : Ultach Enri
Alamat        : Jakarta
Jenis Kelamin : Perempuan
```

PERTEMUAN 2

Model Data, Fungsi Masukan dan Keluaran

2.1 Pengenal (*Identifier*)

Pengenal adalah suatu nama yang biasa dipakai dalam pemrograman untuk menyatakan :
* Variabel
* Konstanta bernama
* Tipe Data
* Fungsi
* Label
* Obyek

Serta hal-hal lain yang dideklarasikan atau didefinisikan oleh pemrogram.

2.2 Penamaan Pengenal

Suatu pengenal berupa satu atau beberapa karakter : Huruf, Digit, Garis-Bawah (_) dan berawalan dengan huruf atau garis-bawah.

Syarat-syarat pembuatan identifier :

- Tidak boleh diawali dengan angka (harus diawali dengan huruf)
- Tidak Boleh menggunakan character khusus
- Tidak boleh menggunakan spasi
- Tidak boleh menggunakan reserved words

Note : *Panjang maksimal nama pengenal pada C++ tergantung oleh kompiler yang digunakan. Misalnya, Borland C++ memperkenankan nama pengenal hingga 32 karakter (yang bersifat signifikans, kelebihannya akan diabaikan), sedangkan Turbo C++ menjamin nama yang signifikans hingga 31 karakter.*

2.3 Case Sensitive

Pada C++ huruf kecil dan huruf kapital pada suatu pengenal tidak dianggap sama. Sifat ini dikenal dengan istilah case sensitive.

2.4 *Reserved Words* (Kata Kunci)

Kata Kunci (*Keyword*) adalah pengenal system yang mempunyai makna khusus bagi kompiler. Kegunaan dari golongan ini tidak dapat diubah. Karena itu, kata kunci tak dapat digunakan sebagai pengenal yang di buat oleh pemrogram.

Reserved Words (Kata Kunci)

asm	auto	break	case	char	class	const	continue
default	delete	do	double	else	enum	extern	float
for	friend	goto	if	inline	int	long	new
operator	private	protected	public	register	return	short	signed
sezeof	static	struct	switch	template	this	typedef	union
unsigned	virtual	void	volatile	while			

2.5 Tipe Data

Tipe Data Dasar

Tipe Data	Ukuran Memori	Jangkauan Nilai	Jumlah Digit
Char	1 Byte	-128 s.d 127	
Int	2 Byte	-32768 s.d 32767	
Short	2 Byte	-32768 s.d 32767	
Long	4 Byte	-2,147,435,648 s.d 2,147,435,647	
Float	4 Byte	3.4 x 10-38 s.d 10+38	5-7
Double	8 Byte	1.7x 10-308 s.d 1.7 x 10+308	15-16
Long Double	10 Byte	3.4 x 10-4932 s.d 1.1 x 10+4932	19

Tipe Data Tambahan

Unsigned digunakan bila data yang digunakan hanya data yang positif saja.

Tipe Data	Jumlah Memori	Jangkauan Nilai
Unsigned Integer	2 Byte	0 – 65535
Unsigned Character	1 Byte	0 – 255
Unsigned Long Integer	4 Byte	0 – 4,294,967,295

2.6 Konstanta

Konstanta merupakan nilai data eksplisit yang di tulis oleh programmer, berupa nilai yang diketahui oleh kompiler pada saat kompilasi. Dengan kata lain konstanta adalah suatu nilai yang sifatnya tetap. Secara garis besar konstanta dapat dibagi menjadi 2 bagian, yaitu **konstanta bilangan** dan **konstanta teks**.

Bentuk Penulisannya : **const nama_konstanta = nilai_konstanta**

Contoh : const char nama[5] = “ute";

2.6.1. Konstanta Bilangan

a. Konstanta bilangan Bulat (*Integer*)
   Bilangan yang tidak mengandung nilai desimal
   Contoh : 1,2,3,…..dst
b. Konstanta Desimal Berpresisi Tunggal (Floating Point)
   Mempunyai bentuk penulisan :
   - Bentuk Desimal (contoh : 10,08)
   - Bentuk Eksponensial / Bilangan Berpangkat (contoh 4.22e3 $\rightarrow$ 4.22 x 10^3)
c. Konstanta Desimal Berpresisi Ganda (Double Precision)
   Pada prinsipnya sama seperti konstanta Floating Point, tetapi Konstanta Double Presicion mempunyai daya tampung data lebih besar.

2.6.2. Konstanta Teks

a. Data Karakter (*Character*)
   Konstanta karakter biasanya berisikan satu karakter dalam **tanda kutip tunggal**. Sebagai contoh, ‘u’ dan ‘$’ keduanya adalah konstanta karakter. Pada C, beberapa

karakter khusus. Terutama *nonprinting control characters*, di representasikan secara khusus yang disebut *escape characters*, yang masing-masing dimulai dengan karakter backslash (\).

Escape Characters

Escape Code	Karakter yang di wakili
\n	Baris Baru / (N)ew line
\t	Horizontal (t)ab, default 8 character
\b	(B)ackspace
\r	Carriage (R)eturn / Enter
\f	(F)orm feed (ganti halaman)
\\	Backslash
\’	Kutip Tunggal
\”	Kutip Ganda
\ddd	Karakter dengan nilai ascii ddd
\0	Null karakter (\000)
\v	Karakter tab vertical

b. Data Teks (*String*)
Konstanta string merupakan rangakain dari karakter yang diapit **tanda kutip ganda**. Panjang suatu string adalah jumlah karakter pada string tersebut.
Contoh : “Jakarta”, “Ultach Enri”, dll

2.7 Variabel

Adalah suatu tempat menampung data atau konstanta di memori yang mempunyai nilai atau data yang dapat berubah-ubah selama proses program.
Ketentuan pemberian nama variabel :

a. Tidak boleh ada spasi (contoh : nama mahasiswa) dan dapat menggunakan tanda garis-bawah (_) sebagai penghubung (contoh nama_mahasiswa)
b. Tidak boleh diawali oleh angka dan menggunakan operator aritmatika.

Variabel dibagi menjadi dua jenis kelompok :

a. Variabel Numerik
   - Bilangan Bulat atau Integer
   - Bilangan Desimal Berpresisi Tunggal atau Floating Point.
   - Bilangan Desimal Berpresisi Ganda atau Double precision.
b. Variabel Teks
   - Character (Karakter Tunggal)
   - String (untuk rangkaian karakter)

2.7.1. Deklarasi Variabel

Adalah proses memperkenalkan variabel kepada Borland C++ dan pendeklarasian tersebut bersifat mutlak karena jika tidak diperkenalkan terlebih dahulu maka Borland C++ tidak menerima variabel tersebut.
Setiap kali pendeklarasian variabel harus diakhiri oleh tanda titik koma (;)

Deklarasi Variabel

Tipe Variabel	Simbol Deklarasi
Integer	Int
Floating Point	Float
Double Precision	Double
Karakter	Char
Unsigned Integer	Unsigned int
Unsigned Character	Unsigned char
Long Integer	Long int
Unsigned Long Integer	Unsigned long int

Bentuk Penulisannya : **Type_data nama_variabel**

Contoh Deklarasi : int nilai;
Char grade;
Float rata_rata;

2.8 Komentar

Pada C++ suatu komentar diawali dengan dua tanda garis miring (//). Semua tulisan yang terletak sesudah tanda // hingga akhir baris dengan sendirinya akan di perlakukan sebagai keterangan. Bagi kompiler hal ini tidak berguna dan akan diabaikan pada saat kompilasi.

Selain menggunakan //, komentar pada C++ juga dapat ditulis dalam bentuk :

/* komentar */

2.9 Perintah Keluaran

Perintah standar output yang disediakan oleh Borland C++ , diantaranya adalah :

- *Printf()*

Fungsi printf() merupakan fungsi keluaran yang paling umum digunakan untuk menampilkan informasi ke layar.

Bentuk Penulisan :

```
Printf (“string-kontrol”,argumen1, argumen2, , )
```

String kontrol dapat berupa keterangan yang akan ditampilkan pada layar beserta penentu format. Penentu format dipakai untuk memberitahu kompiler mengenai jenis data yang dipakai dan akan ditampilkan. Argumen ini dapat berupa variabel, konstanta dan ungkapan.

Penentu Format Printf()

Tipe Data	Penentu Format untuk printf()
Integer	%d
Floating Point	
Bentuk Desimal	%f
Bentuk Berpangkat	%e
Bentuk Desimal dan pangkat	%g
Double Precision	%lf
Character	%c
String	%s
Unsigned Integer	%u
Long Integer	%ld
Long Unsigned Integer	%lu
Unsigned Hexadecimal Integer	%x
Unsigned Octal Integer	%o

* Penggunaan Penentu Lebar Field

Bila ingin mencetak atau menampilkan data yang bertipe data Float atau pecahan, tampilan yang tampak biasanya kurang bagus. Hal tersebut dapat diatur lebar fieldnya dan jumlah desimal yang ingin dicetak.

Contoh 1 (Tidak menggunakan penentu lebar field):

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

main()
{
float a=7.50, b=243.21;
clrscr();
printf("Bilangan A = %f \n", a);
printf("Bilangan B = %f ", b);
}
```

```
Hasilnya :
Bilangan A = 7.500000
Bilangan B = 243.210007
```

Contoh 2 (Menggunakan penentu lebar field) :

```
#include <stdio.h>
#include <conio.h>
Main()
{
Float a=7.50, b=243.21;
Clrscr();
Printf ("Bilangan A = %4.1f \n", a);
Printf ("Bilangan B = %4.1f ",b);
}
```

```
Hasilnya :
Bilangan A =     7.5
Bilangan B = 243.2
```

- *Puts*()

Sama dengan printf(), yaitu digunakan untuk mencetak string ke layar. Puts() berasal dari kata PUT STRING.

Printf()	Puts()
Harus menggunakan tipe data untuk data string, yaitu %s	Tidak perlu penentu tipe data string, karena fungsi ini khusus untuk tipe data string
Untuk mencetak pindah baris, memerlukan notasi \n	Untuk mencetak pindah baris tidak perlu notasi \n, karena sudah diberikan secara otomatis

Contoh :

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

main()
{
        char a[6] = "Unsika";
        clrscr();
        puts("Saya kuliah di .  ");
        puts (a);
getch();
}
```

- *Putchar*()

Digunakan untuk menampilkan sebuah karakter ke layar. Penampilan karakter tidak diakhiri dengan pindah baris.

Contoh :

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

main()
{
        clrscr;
        putchar('A');
        putchar('B');
        putchar('C');
getch();
}
```

- *Cout*()

Fungsi cout() merupakan sebuah objek di dalam Borland C++ digunakan untuk menampilkan suatu data ke layar. Untuk menggunakan fungsi cout() ini, harus menyertakan file header **iostream.h**.

Contoh :

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

void main()
{
        cout<<"Pilihan Anda Salah !\a";
getch();
}
```

- *Manipulator*
  Manipulator pada umumnya digunakan untuk mengatur tampilan data.

Fungsi manipulator yang disediakan oleh Borland C++, antara lain :

Manipulator	Keterangan
Endl	Menyisipkan newline dan mengirimkan isi penyangga keluaran ke piranti keluaran. B U : cout<<information<<endl;
Ends	Menyisipkan karakter null. B U : cout<<information<<ends;
Flush	Mengirimkan isi penyangga keluaran ke piranti keluaran. B U : cout<<information<<flush;
Dec	Mengkonversikan ke bilangan basis 10 B U : cout<<dec<<information<<endl;
Hex	Mengkonversikan ke bilangan basis 16 B U : cout<<hex<<information<<endl;
Oct	Mengkonversikan ke bilangan basis 8 B U : cout<<oct<<information<<endl;
Setbase(int n)	Mengkonversikan ke bilangan basis n (n=8, 10 atau 16) B U : cout<<setbase(n)<<information<<endl;
Setw(int n)	Mengatur lebar field untuk suatu nilai sebesar n karakter. B U : cout<<setw(n)<<information<<endl;
Setfill(int c)	Menyetel karakter pemenuh berupa c B U : cout<<setfill(c);
Setprecision(int n)	Menyetel presisi bilangan pecahan sebesar n digit. B U : cout<<setiosflags (ios::fixed); cout<<setprecision(n)<<information<endl;
Setiosflags(long f)	Menyetel format yang ditentukan oleh *f*. *f* adalah tanda format. B U : cout<<setiosflags(f);
Resetiosflags(long f)	Menghapus format yang ditentukan oleh *f*. *f* adalah adalah tanda format. B U : cout<<resetiosflags(f);

Note : *Jika menggunakan manipulator selain* ***dec, hex, oct, endl, flush*** *, file header bernama* ***iomanip.h*** *perlu disertakan.*

Format yang dipakai dalam *setiosflags*() dan *resetiosflags*() :

- **ios::left**
  Berguna untuk menyetel rata kiri terhadap lebar field yang diatur melalui setw().
- **ios::right**
  Berguna untuk menyetel rata kanan terhadap lebar field yang diatur melalui setw().
- **ios::showpos**
  Berguna untuk menampilakn tanda + pada bilangan positif
- **ios::scientific**
  Berguna untuk memformat keluaran dalam bentuk notasi eksponensial.
- **ios::fixed**
  Berguna untuk memformat keluaran dalam bentuk notasi desimal
- **ios::showpoint**
  Berguna untuk menampilkan titik desimal pada bilangan yang tidak memiliki bagian pecahan
- **ios::showbase**
  Berguna untuk menampilkan awalan 0x untuk bilangan heksa atau 0 untuk bilangan oktal
- **ios::oct**
  Berguna untuk memformat keluaran dalam basis 8
- **ios::dec**
  Berguna untuk memformat keluaran dalam basis 10
- **ios::hex**
  Berguna untuk memformat keluaran dalam basis 16
- **ios::uppercase**
  Berguna untuk memformat huruf pada notasi heksa dalam bentuk huruf kapital

1. **Manipulator endl**
   Manipulator ini digunakan untuk menyisipkan baris baru (*newline*) yang identik dengan \n.
   Contoh :

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main()
{
 int a=10, b=100, c=1000;
 clrscr();
 cout <<"Nilai a = " << a <<endl;
 cout <<"Nilai b = " << b <<endl;
 cout <<"Nilai c = " << c <<endl;
}
```

   Hasil :

```
Nilai a = 10
Nilai b = 100
Nilai c = 1000
```

2. **Manipulator setw()**
   Bermanfaat untuk mengatur lebar dari suatu tampilan data sehingga dapat diatur rapat kanan. Apabila nilai parameter lebar pada setw() lebih kecil dari jumlah karakter maka tidak akan memberikan efek apa-apa.
   Contoh :

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main()
{
 int a=10, b=100, c=1000;
 clrscr();
 cout <<"Nilai a = " << setw(4) <<a <<endl;
 cout <<"Nilai b = " << setw(4) <<b <<endl;
 cout <<"Nilai c = " << setw(4) <<c <<endl;
}
```

   Hasil :

```
Nilai a =   10
Nilai b =  100
Nilai c = 1000
```

3. **Manipulator setfill()**
   Manipulator ini digunakan untuk mengatur karakter yang dipakai memenuhi bagian fied yang ditentukan setw(), yang tidak dipakai untuk menampilkan data.
   Contoh :

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main()
{
 int a=10, b=100, c=1000;
 clrscr();
 cout <<setfill ('*');
 cout <<"Nilai a = " << setw(4) <<a <<endl;
 cout <<"Nilai b = " << setw(4) <<b <<endl;
 cout <<"Nilai c = " << setw(4) <<c <<endl;
}
```

   Hasil :

```
Nilai a = **10
Nilai b = *100
Nilai c = 1000
```

4. **Manipulator dec, oct dan hex**
   Manipulator ini digunakan untuk menampilkan suatu data dalam bentuk desimal (bilangan basis 10), oktal (bilangan basis 8) dan heksadesimal (bilangan basis 16).
   Contoh :

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

```
#include <iomanip.h>
void main()
{
 int nilai=250;
 clrscr();
 cout <<"Nilai bilangan = " <<nilai <<endl;
 cout <<"Konversi ke desimal = " << dec << nilai <<endl;
 cout <<"Konversi ke oktal = " << oct << nilai <<endl;
 cout <<"Konversi ke heksadesimal = " << hex << nilai <<endl;
}
```

Hasil :

```
Nilai bilangan = 250
Konversi ke desimal = 250
Konversi ke oktal = 372
Konversi ke heksadesimal = fa
```

5. **Manipulator setbase()**

Digunakan untuk menampilkan suatu data dalam bentuk desimal (bilangan basis 10), oktal (bilangan basis 8) dan heksadesimal (bilangan basis 16).

Contoh :

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main()
{
 int nilai=212;
 clrscr();
 cout <<"Nilai bilangan = " <<nilai <<endl;
 cout <<"Konversi ke desimal = " << setbase(10) << nilai <<endl;
 cout <<"Konversi ke oktal = " << setbase(8) << nilai <<endl;
 cout <<"Konversi ke heksadesimal = " << setbase(16) << nilai <<endl;
}
```

Hasil :

```
Nilai bilangan = 212
Konversi ke desimal = 212
Konversi ke oktal = 324
Konversi ke heksadesimal = d4
```

6. **Manipulator flush**

Digunakan agar data yang dikirimkan ke cout langsung ditransfer ke *standard output* tanpa menggunakan suatu penyangga (*buffer*). Hal ini dilakukan untuk mengefisiensikan pengiriman ke *standard output.* Namun jika Anda menggunakan *endl*, sebenarnya manipulator ini identik dengan *newline* diikuti dengan *flush.* Berikut ini adalah pernyataan yang mempunyai makna yang sama :

```
cout <<”C++\n” <<flush;
cout <<”C++” <<endl;
cout <<”C++ <<’\n’ <<flush;
```

7. **Manipulator ends**
   Berfungsi untuk menambahkan karakter *null* (ASCII nol) ke deretan suatu karakter. Hal ini diperlukan misalnya jika ingin mengirim sejumlah karakter ke file disk atau modem dan mengakhirinya dengan karakter *null*.
   Misal : cout << ‘a’ <<’b’ <<’c’ <<ends;
   Pernyataan di atas mengirimkan tiga buah karakter a, b dan c serta sebuah karakter *null* .

8. **Manipulator setiosflags()**
   Merupakan manipulator yag dapat dipakai untuk mengontrol sejumlah tanda format, misalnya :
   - ios::showpos
     Digunakan untuk menampilkan tanda plus pada bilangan positif
     Contoh :

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main()
{
 int x=4, y=-44;
 clrscr();
 cout <<"Nilai sebelum showpos"<<endl;
 cout <<"Nilai x = " <<x <<endl;
 cout <<"Nilai y = " <<y <<endl;
 cout << setiosflags(ios::showpos);
 cout <<"Nilai sesudah showpos"<<endl;
 cout <<"Nilai x = " <<x <<endl;
 cout <<"Nilai y = " <<y <<endl;
}
```

     Hasil :

```
Nilai sebelum showpos
Nilai x = 4
Nilai y = -44
Nilai sesudah showpos
Nilai x = +4
Nilai y = -44
```

   - ios::left dan ios::right
     Digunakan untuk mengatur rata kiri (left) dan rata kanan (right) dalam setw()
     Contoh :

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main()
{
 clrscr();
```

```
 cout <<setiosflags(ios::left) << setw(25) <<"N A M A"
                <<setiosflags(ios::right) << setw(8) <<"G A J I" <<endl;
 cout <<setiosflags(ios::left) << setw(25) <<"BUDI"
                <<setiosflags(ios::right) << setw(8) <<3000000 <<endl;
 cout <<setiosflags(ios::left) << setw(25) <<"SUSI"
                <<setiosflags(ios::right) << setw(8) <<950000 << endl;
 cout <<setiosflags(ios::left) << setw(25) <<"ANDI"
                <<setiosflags(ios::right) << setw(8) <<2000000 << endl;
}
```

- ios::scientific dan ios::fixed
  Digunakan untuk menampilkan bilangan dalam notasi eksponensial atau (scientific) atau dalam notas biasa (fixed)
  Contoh :

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main()
{
 clrscr();
 cout <<"Nilai x = 123.45" <<endl;
 cout <<"Bentuk scientific : " << setiosflags(ios::scientific) <<123.45 <<endl;
 cout <<"Bentuk fixed : " << setiosflags(ios::fixed) <<123.45 <<endl;
}
```

  Hasil :

```
Nilai x = 123.45
Bentuk scientific : 1.234500e+02
Bentuk fixed : 123.450000
```

9. **Manipulator resetiosflag()**
   Digunakan untuk mengembalikan format ke keadaan semula
   Contoh :

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main()
{
 clrscr();
 cout <<setiosflags(ios::showpos);
 cout <<"Nilai x =" << 44 << endl;
 cout <<resetiosflags(ios::showpos);
 cout <<"Nilai x setelah di resetiosflags() = " <<44 <<endl;
}
```

   Hasil :

```
Nilai x =+44
Nilai x setelah di resetiosflags() = 44
```

10. **Manipulator setprecision()**

Digunakan untuk mengatur jumlah digit pecahan yang ingin ditampilkan.

Contoh :

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main()
{
 float x=123.45;
 clrscr();
 cout <<setiosflags(ios::fixed);
 cout <<"Nilai awal x = 123.45" <<endl;
 cout <<setprecision(0) <<"Nilai x presisi 0 = " <<x << endl;
 cout <<setprecision(1) <<"Nilai x presisi 1 = " <<x << endl;
 cout <<setprecision(2) <<"Nilai x presisi 2 = " <<x << endl;
 cout <<setprecision(3) <<"Nilai x presisi 3 = " <<x << endl;
 cout <<setprecision(4) <<"Nilai x presisi 4 = " <<x << endl;
 cout <<setprecision(5) <<"Nilai x presisi 5 = " <<x << endl;
}
```

Hasil :

```
Nilai awal x = 123.45
Nilai x presisi 0 = 123
Nilai x presisi 1 = 123.4
Nilai x presisi 2 = 123.45
Nilai x presisi 3 = 123.450
Nilai x presisi 4 = 123.4500
Nilai x presisi 5 = 123.45000
```

2.10 Perintah Masukan

Perintah Standar input yang disediakan oleh Borland C++, diantaranya :

- Scanf()

Digunakan untuk memasukkan berbagi jenis data.

Bentuk Umum : `scanf("Penentu format",&nama_variabel);`

Note : *Simbol **&** merupakan pointer yang digunakan untuk menunjuk kealamat variabel memori yang dituju.*

Penentu Format Scanf()

Tipe Data	Penentu Format Untuk Scanf()
Integer	%d
Floating Point	
Bentuk Desimal	%e atau %f
Bentuk Berpangkat	%e atau %f
Double Precision	%lf
Character	%c
String	%s

Unsigned Integer	%u
Long Integer	%ld
Long Unsigned Integer	%lu
Unsigned Hexadecimal Integer	%x
Unsigned Octal Integer	%o

Contoh :

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

main()
{
int a, b, c=0;
clrscr();
printf("Masukkan Nilai A = "); scanf("%d, &a);
printf("Masukkan Nilai B = ");scanf("%d,&b);

c=a+b;
printf("Hasil Penjumlahan = %d",c);
}
```

- *Gets*()

  Digunakan untuk memasukkan data string.

Perbedaan scanf() dengan gets()

Scanf()	Gets()
Tidak dapat menerima string yang mengandung spasi atau tab dan dianggap sebagai data terpisah	Dapat menerima string yang mengandung spasi atau tab dan masing masing dianggap sebagai satu kesatuan data.

- *Cin()*

Fungsi cin() merupakan sebuah objeck di dalam C++ digunakan untuk memasukkan suatu data. Untuk menggunakan fungsi cin() ini harus menyertakan file header **iostream.h**

- *Getch()*

Berguna untuk membaca sebuah karakter tanpa perlu menekan enter, tidak menampilkan karakter dari tombol yang di tekan. File header yang harus di sertakan adalah **conio.h**

Contoh :

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

main()
{
        char kar;
        clrscr();
        printf("Masukan sebuah karakter bebas = ");
```

```
        kar = getch();
        printf("\nTadi anda memasukan karakter %c",kar);
        getch();
}
```

- *Getche()*

Berguna untuk membaca sebuah karakter tanpa perlu menekan enter, serta menampilkan karakter dari tombol yang di tekan.

Contoh :

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

main()
{
        char kar;
        clrscr();
        printf("Masukan sebuah karakter bebas = ");
        kar = getche();
        printf("\nTadi anda memasukan karakter %c",kar);
        getch();
}
```

TUGAS

1. Buat program untuk menampilkan tulisan sebagai berikut :
   "Anda diundang rapat hari Jum'at / pukul 14.00 WIB"

2. Buat program untuk mencari luas segitiga jika diketahui alas=10 dan tinggi=20
3. Buat program dengan konstanta untuk mencari luas dan keliling lingkaran jika diketahui jari-jari lingkaran=100

PERTEMUAN 3

OPERATOR

Operator adalah fungsi yang mengambil satu atau lebih ekspresi sebagai input dan mengembalikan ekspresi yang menggunakan symbol infix khusus, bukannya notasi fungsional biasa. Nilai yang dioperasikan operator disebut *operand*.

Sifat operator

Sifat	Keterangan	Contoh
Unary	Operator ini hanya melibatkan sebuah operand	-8
Binary	Operator ini melibatkan dua buah operand	10 + 1
Ternary	Operator ini melibatkan tiga buah operator	(10 * 2) – 2 + 4

3.1 Operator Aritmatika

Operator untuk operasi aritmatika tergolong sebagai operator *binary*.

Operator UNARY Aritmatika		
Operator	**Keterangan**	**Contoh**
-	Operator Minus	- a
+	Operator Plus	+ a
Operator BINARY Aritmatika		
Operator	**Keterangan**	**Contoh**
*	Operator Perkalian	a * b
/	Operator Pembagian	a / b
%	Sisa Pembagian (Modulus)	a % b
+	Operator Penjumlahan	a + b
-	Operator Pengurangan	a - b

3.2 Operator Penugasan

Operator penugasan yang berupa simbol sama dengan (=) berguna untuk memberikan suatu nilai ke suatu variabel. Operator ini dapat dikenakan sebagai ungkapan atau berdiri sebagai pernyataan.

Operator	Contoh
Penugasan Sederhana	a=1
Penugasan ungkapan	a=1+(b=1)
Penugasan berganda	a=b=1

3.3 Operator Decrement – Increment

Berkaitan dengan operasi aritmatika, C menyediakan operator yang disebut sebagai operator penaikan (*increment*) dan operator penurunan (*decrement*). Kedua operator ini digunakan pada operand bertipe bilangan bulat.

Operator	Keterangan	Contoh	
++	Operator Penaikan	--a	a--
--	Operator Penurunan	++b	b++

Operator penaikan digunakan untuk menaikkan nilai variabel sebesar satu. Sedangkan operator penurunan dipakai untuk menurunkan nilai variabel sebesar satu. Penempatan operator terhadap variabel dapat dilakukan di muka atau di belakangnya.

3.4 Operator Bitwise (Manipulasi Bit)

Merupakan operator yang digunakan untuk manipulasi data dalam bentuk bit. Operator bitwise hanya bisa dikenakan pada operand bertipe integer atau karakter.

Operator	Keterangan	Contoh
~	Operator bitwise NOT (Komplemen) *Mempunyai sifat membalik (invers).* B U : **~operand**	~ a
<<	Operator Left Shift (geser bit ke kiri) *Mempunyai efek seperti perkalian.* B U : **nilai << jumlah bit digeser ke kiri**	a<<b
>>	Operator Right Shift (geser bit ke kanan) *Mempunyai efek seperti pembagian.* B U : **nilai >> jumlah bit digeser kekanan**	a>>b
&	Operator bitwise AND B U : **operand1 & operand2**	a&b
|	Operator bitwise OR B U : **operand1 | operand2**	a|b
^	Operator bitwise XOR B U : **operand1 ^ operand2**	a^b

Tabel Kebenaran BITWISE					
a	**B**	**~a**	**a&b**	**a|b**	**a^b**
1	1	0	1	1	0
1	0	0	0	1	1
0	1	1	0	1	1
0	0	1	0	0	0

Note :

 # Operator bitwise mempunyai prioritas lebih rendah dibandingkan operator aritmatika.

 # Bit terkanan dalam penyajian bilangan biner disebut sebagai bit 0

3.5 Operator Majemuk

Operator majemuk merupakan operator yang digunakan untuk menyingkat suatu penulisan dari operator lain.

Operator	Contoh	Kependekan dari
+=	a+=b	a=a+b
-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
%=	a%=b	a=a%b
<<=	a<<=b	a=a<<b
>>=	a>>=b	a=a>>b
&=	a&=b	a=a&b
|=	a|=b	a=a|b
^=	a^=b	a=a^b

3.6 Ungkapan Kondisi

Ungkapan kondisi merupakan ungkapan yang menjadi dasar bagi pernyataan berkondisi. Ungkapan ini memberikan nilai Betul (1) dan salah (0).
Adapun elemen yang membentuk ungkapan ini adalah :

☺ Operator Relasi

Operator relasi biasa digunakan untuk membandingkan dua buah nilai.

Operator	Keterangan
= =	Operator sama dengan
!=	Operator tidak sama dengan
>	Operator lebih dari
<	Operator kurang dari
>=	Operator lebih dari atau sama dengan
<=	Operator kurang dari atau sama dengan

☺ Operator logika

Operator ini biasa digunakan untuk menghubungkan dua buah ungkapan kondisi menjadi sebuah ungkapan kondisi.

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

Note :
Pada bentuk pemakaian || atau && biasanya ungkapan1 dan ungkapan2 ditulis didalam tanda kurung. Contoh : (a = = c)&&(b = = d). Operator = = mempunyai prioritas lebih tinggi di bandingkan &&.

☺ Operator Kondisi

Operator kondisi biasa dipakai untuk mendapatkan sebuah nilai dari dua buah kemungkinan, berdasarkan suatu kondisi.

Bentuk umumnya : `ungkapan1 ? ungkapan2 : ungkapan3`

Ada 3 ungkapan yang dilibatkan. Oleh karena itu operator ?: tergolong sebagai operator *ternary*.

☺ Operator Koma

Operator ini berguna untuk meletakkan dua buah ungkapan pada suatu kaidah yang memerlukan sebuah ungkapan saja.

Bentuk umumnya : `ungkapan1 , ungkapan2`

Tanda koma adalah symbol operator koma. Nilai yang menggunakan operator koma sesuai dengan nilai ungkapan yang terletak di kanannya (ungkapan2). Umumnya operator koma terdapat pada pernyataan **for**.

3.7 Prioritas Operator

Simbol	Nama	Prioritas	Urutan Pengerjaan
::	Resolusi lingkup	Tertinggi	Kiri ke kanan
++ -- () [] -> **.**	Post-increment Post-decrement Pemanggilan fungsi Elemen Array Pointer ke anggota struktur atau kelas Anggota struktur, union atau kelas		Kiri ke kanan
++ -- ! ~ - + & * New	Pre-increment Pre-decrement Logika bukan (**NOT**) Bitwise komplemen Unary minus Unary plus Alamat (*address*) Indirection Pengalokasian memori		Kanan ke kiri

Delete Sizeof (type) Type()	Dealokasi memori Ukuran type data *Type casting* *Type casting*		
.* ->* ()	Dereferensi C++ Dereferensi C++ Kurung untuk ungkapan		Kiri ke kanan
* / %	Perkalian Pembagian Sisa pembagian (modulus)		Kiri ke kanan
+ -	Penjumlahan Pengurangan		Kiri ke kanan
<< >>	Geser kiri Geser kanan		Kiri ke kanan
< > <= >=	Kurang dari Lebih dari Kurang dari atau sama dengan Lebih dari atau sama dengan		Kiri ke kanan
= = !=	Sama dengan Tidak sama dengan		Kiri ke kanan
&	Bitwise dan (AND)		Kiri ke kanan
^	Bitwise exclusive OR (XOR)		Kiri ke kanan
|	Bitwise atau (OR)		Kiri ke kanan
&&	Logika dan (AND)		Kiri ke kanan
||	Logika atau (OR)		Kiri ke kanan
?:	Operator kondisi		Kiri ke kanan
= *=, /=, %=, +=, -=, <<=, >>=, &=, ^=, |=	Penugasan Operator majemuk		Kanan ke kiri
,	Operator koma	Terendah	Kiri ke kanan

TUGAS

1. Buat program untuk menyelesaikan rumus :
   $Y = bx^2 + 0,5x - c$, dimana nilai b = 15, x=5, c=10

2. Nilai akhir dari pelajaran Bahasa C ditentukan oleh tiga nilai yaitu :
   - Nilai Praktek bobot 20%
   - Nilai UTS bobot 30%
   - Nilai UAS bobot 50%

   Buatlah program untuk menghitung nilai akhir, jika diketahui Nilai Praktek=70, Nilai UTS=80 dan Nilai UAS= 75!

3. Buatlah program untuk menghitung Keliling, Luas Permukaan dan Isi dari sebuah bola dengan rumus :
   - Keliling = $2 \pi r$
   - Luas Permukaan = $4/3 \pi r^3$
   - Isi = $4 \pi r^2$

   Jika diketahui panjang jari-jarinya = 10 !

PERTEMUAN 4

Pernyataan Dasar

Pernyataan (*statement*) digunakan untuk melakukan suatu tindakan. Macam-macam pernyataan diantaranya :

- **Pernyataan ungkapan**
  Pernyataan ungkapan merupakan bentuk pernyataan yang paling umum dipakai, terdiri dari sebuah ungkapan dan diakhiri dengan tanda titik koma(**;**).

  Bentuk Umum : *ungkapan;*

  Biasanya pernyataan ungkapan berupa penugasan nilai terhadap variabel atau pemanggilan fungsi. Contoh : bil = 8;
  x--;

- **Pernyataan deklarasi/definisi**
  Bertujuan untuk memperkenalkan nama pengenal beserta tipe datanya.

  Bentuk Umum : *type_data nama_variabel;*

  Contoh : int gaber;

- **Pernyataan nol**
  Pernyataan yang berisi tanda titik koma saja (**;**). Perintah ini tidak melaksanakan apa-apa. Tetapi kehadirannya kadang-kadang diperlukan.

- **Pernyataan majemuk**
  Pernyataan majemuk merupakan sejumlah pernyataan yang berada di dalam tanda kurung kurawal. Seringkali disebut dengan istilah *blok*.

  Bentuk Umum : *{*
  pernyataan;
  }

  Pernyataan majemuk banyak dijumpai pada pernyataan seperti **if**, **while** ataupun **for**.

- Pernyataan **goto**
  Adalah pernyataan yang mengarahkan eksekusi ke pernyataan yang diawali dengan suatu nama label dan tanda titik dua(**:**)

  Bentuk Umum : *goto nama_label;*
  Nama_label:
  {

pernyataan;
}

- Pernyataan berkondisi : **if** dan **switch**
  - **If**

    Pernyataan yang dipakai untuk mengambil sebuah keputusan yang berdasarkan suatu kondisi.

    a. Pernyataan if sederhana

    B U : ***if****(kondisi)*
    Pernyataan;

    Contoh:

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

void main()
{
    int usia;
clrscr();
cout<<"Berapa Usia anda?";cin>>usia;
if(usia<17)
    cout<<"Anda tidak diperkenankan menonton"<<endl;
getch();
}
```

    Jika pernyataan yang mengikuti if berupa pernyataan majemuk , maka bentuknya sbb:

```
if(kondisi)
{
 pernyataan_1;
 pernyataan_2;
 pernyataan_n;
}
```

    Pernyataan-pernyataan di dalam {} akan dilaksanakan apabila kondisinya bernilai benar.

    b. Pernyataan if – else

    Bentuk penulisan :

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

    Pernyataan1 akan dilaksanakan selama kondisi bernilai benar sedangkan pernyataan2 akan dilaksanakan selama kondisi bernilaisalah. Untuk pernyataan if yang diikuti dengan pernyataan majemuk, pernyataan if diapit dengan tanda { dan }.

    Contoh :

```
#include <iostream.h>
```

```
#include <conio.h>
void main()
{
 int usia;
 clrscr();
 cout <<"Masukkan usia Anda : ";
 cin >>usia;
 if (usia < 17)
 {
  cout <<"Usia Anda dibawah 17 tahun " <<endl;
  cout <<"Anda dilarang masuk...";
 }
 else
 {
  cout <<"Usia Anda 17 tahun atau lebih" <<endl;
  cout <<"Silahkan masuk...";
 }
 }
```

c. Pernyataan nested if

Pernyataan if yang terletak di dalam if disebut *nested if* atau if bersarang.

Bentuk penulisan :

```
if (kondisi-1)
  pernyataan-1;
        else if (kondisi-2)
  pernyataan-2;
        else if (kondisi-m)
  pernyataan-m;
        else
  pernyataan-n;
```

Penyeleksian akan dilakukan secara bertingkat, begitu ada kondisi yang bernilai benar maka pernyataan yang bersesuaian dengan kondisi tersebut akan dilaksanakan sedangkan jika tidak ada kondisi yang memenuhi maka pernyataan-n yang akan dijalankan.

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int kode;
 clrscr();
 cout <<"Tujuan Wisata :" <<endl;
 cout <<"1: Bandung   2: Yogyakarta   3: Bali   4: Lombok" <<endl<<endl;
 cout <<"Masukkan kode tujuan : ";
 cin >>kode;
 if(kode==1)
     cout <<"Bandung" <<endl;
 else if(kode==2)
      cout <<"Yoyakarta" <<endl;
 else if(kode==3)
```

```
        cout <<"Bali" <<endl;
    else if(kode==4)
        cout <<"Lombok" <<endl;
    else
        cout <<"Pilihan Anda salah...";
   }
```

Hasil :

```
Tujuan Wisata :
1: Bandung   2: Yogyakarta   3: Bali   4: Lombok

Masukkan kode tujuan : 5
Pilihan Anda salah...
```

Pemakaian Operator Logika

Pemakaian operator logika terkadang dapat menyederhanakan penggunaan if.

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int x,y;
 char ulang;
 mulai :
  clrscr();
  cout <<"Masukkan nilai x : "; cin >>x;
  cout <<"Masukkan nilai y : "; cin >>y;
  if ((x>=0) && (y>=0))
                cout <<"x dan y bernilai POSITIF" <<endl;
  else
                cout <<"x atau y ada yang bernilai NEGATIF" <<endl;
                cout <<"Ulang lagi [y/t] ? "; cin >>ulang;
    if ((ulang=='y') || (ulang=='Y'))
        goto mulai;
}
```

Hasil :

```
Masukkan nilai x : 4
Masukkan nilai y : -44
x atau y ada yang bernilai NEGATIF
Ulang lagi [y/t] ? t
```

Operator Kondisi

Bahasa C++ menyediakan operator yang tergolong sebagai operator *ternary*, yakni operator yang memiliki tiga buah operand dengan menggunakan simbol **?:**.

Bentuk penulisan operator ini adalah : *kondisi ? ungkapan-1 : ungkapan-2*

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

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int x,y, maks;
 clrscr();
 cout <<"Masukkan nilai-1 : "; cin >>x;
 cout <<"Masukkan nilai-2 : "; cin >>y;
 maks=(x>y) ? x : y;
 cout <<"Nilai terbesar adalah : " <<maks;
}
```

Hasil :

```
Masukkan nilai-1 : 44
Masukkan nilai-2 : 22
Nilai terbesar adalah : 44
```

❖ **Switch**

Di dalam pernyataan switch, sebuah variabel secara berturut-turut diuji oleh daftar konstanta bilangan bulat atau konstanta karakter. Jika sesuai dengan sebuah konstanta, pernyataan yang mengikuti konstanta akan dikerjakan.

Bentuk penulisan :

```
switch (variabel)
{
  case konstanta1;
      pernyataan;
      break;
  case konstanta2;
     pernyataan;
     break;
  default
  pernyataan;
}
```

Jika sebuah konstanta sesuai dengan isi variabel, pernyataan-pernyataan setelah case akan dikerjakan sampai ditemukan pernyataan *break*. Pernyataan setelah default akan dikerjakan jika tidak ada konstanta yang sesuai. Pernyataan default bersifat optional. Jika tidak ada default dan tidak ada konstanta yang sesuai, tidak ada yang dikerjakan.

Ada tiga hal penting dalam pernyataan switch :

1. Switch hanya dapat memeriksa persamaan dengan sebuah konstanta, sedangkan if dapat memeriksa syarat-syarat lain (lebih besar, lebih kecil, tidak sama dengan dan sebagainya)
2. Tidak ada dua konstanta yang sama didalam sebuah switch
3. Pernyataan switch lebih baik daripada tangga if-else

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int kelas;
 float harga,pajak;
```

```
clrscr();
cout <<"Harga Barang              : Rp. ";
cin >>harga;
cout <<"Masukkan Jenis Kelas (1-3) : ";
cin >>kelas;
switch (kelas)
{
        case 1 :
                cout <<"Jenis Barang : Makanan" << endl;
                cout <<"Harga Barang : Rp. " <<harga <<endl;
                pajak = 0.1 * harga;
                cout <<"Pajak        : Rp. " <<pajak <<endl;
                break;
        case 2 :
                cout <<"Jenis Barang : Pakaian, Sepatu" << endl;
                cout <<"Harga Barang : Rp. " <<harga <<endl;
                pajak = 0.15 * harga;
                cout <<"Pajak        : Rp. " <<pajak <<endl;
                break;
        case 3 :
                cout <<"Jenis Barang : Mesin-mesin" << endl;`
                cout <<"Harga Barang : Rp. " <<harga <<endl;
                pajak = 0.175 * harga;
                cout <<"Pajak        : Rp. " <<pajak <<endl;
                break;
        default :
        cout <<"Pilihan Kelas salah...!";
 }
}
```

Hasil :

```
Harga Barang               : Rp. 100000
Masukkan Jenis Kelas (1-3) : 2
Jenis Barang : Pakaian, Sepatu
Harga Barang : Rp. 100000
Pajak        : Rp. 15000
```

Pernyataan break di dalam switch bersifat optional. Break dipakai untuk menghentikan pelaksanaan pernyataan-pernyataan yang mengikuti sebuah konstanta. Jika break tidak ada, pernyataan pada case berikutnya akan dilaksanakan sampai ditemukan break atau akhir dari switch.

TUGAS

1. Buat program untuk menenetukan apakah suatu bilangan bersifat GENAP atau GANJIL
2. Buat program untuk menghasilkan output sebagai berikut :

```
    Menu Restaurant Mc'Yahud
=====================================
1. Nasi Goreng Informatika     Rp. 5.000,-
2. Nasi Soto Ayam Internet     Rp. 7.000,-
3. Gado-gado Disket            Rp. 4.500,-
4. Bubur Ayam LAN              Rp. 4.000,-
=====================================

Masukkan Pilihan Anda... :1

Pilihan No.1 Nasi Goreng Informatika Rp.5.000,-
```

3. Buat program dengan input untuk mencari bilangan terkecil dari 4 buah bilangan.
4. Perusahaan Susu ABC ingin membuat sistem penjualan susu dengan tampilan sebagai berikut :

```
Masukkan Kode Susu (1-3) : 2
Masukkan Jumlah Pembelian : 5
Masukkan Ukuran (B/S/K) : S

Susu Indomilk
Harga Susu Rp.    4000.00
Jumlah Pembelian  Rp.   20000.00
```

Untuk daftar harga produk susu dapat dilihat pada tabel di bawah ini :

Kode Susu	**Nama Produk**	**Ukuran**	**Harga**
1	Dancow	B = Besar S = Sedang K = Kecil	Rp. 10.000,- Rp. 4.250,- Rp. 2.100,-
2	Indomilk	B = Besar S = Sedang K = Kecil	Rp. 8. 500,- Rp. 4.000,- Rp. 2.025,-
3	Sustacal	B = Besar S = Sedang K = Kecil	Rp. 17.000,- Rp. 14.500,- Rp. 8.300,-

Buat program dengan menggunakan nested switch !

PERTEMUAN 5

Perulangan (*Looping*)

PERNYATAAN WHILE

Pernyataan while merupakan salah satu pernyataan yang berguna untuk memproses suatu pernyataan atau beberapa pernyataan beberapa kali.

Bentuk penulisan : *while (ungkapan)*
pernyataan;

Bagian pernyataan yang mengikuti while akan dieksekusi selama ungkapan bernilai benar. Perlu diketahui, pengujian terhadap ungkapan pada while dilakukan sebelum bagian pernyataan. Oleh karena itu ada kemungkinan bagian pernyataan pada while tidak dijalankan sama sekali, yaitu jika kondisi yang pertama kali bernilai salah.

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int i=1;
 clrscr();
 while (i<=5)
 {
  cout <<"Putaran ke-" <<i <<endl;
  i++;
 }
}
```

Hasil :

```
Putaran ke-1
Putaran ke-2
Putaran ke-3
Putaran ke-4
Putaran ke-5
```

Pemakaian while dapat digunakan untuk mengatur agar pemakai menekan tombol pilihan yang absah.

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 char kode;
 clrscr();
 cout <<"Pilih salah satu kode [a, b, c] ?";
 kode=getch();
 while (!((kode =='a') || (kode=='b') || (kode=='c')))
 {
```

```
 kode=getch();
 }
 cout <<"\nPilihan Anda : " <<kode;
}
```

Hasil :

```
Pilih salah satu kode [a, b, c] ?
Pilihan Anda : b
```

Pernyataan while juga dapat digunakan untuk menangani pemasukan data, menjumlahkannya dan mencari rata-rata.

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int i=0;
 float nilai, total, rata=0;
 clrscr();
 cout <<"Mencari Nilai Total dan Rata-rata" <<endl;
 cout <<"Masukkan nol untuk keluar..." <<endl;

 while (!(nilai == 0))
 {
  i++;
  cout <<"Nilai ke-" <<i <<" = ";
  cin >>nilai;
  total+=nilai;
 }
 cout <<"Jumlah total nilai = " <<total <<endl;
 cout <<"Rata-rata = " <<(total/(i-1));
}
```

Hasil :

```
Mencari Nilai Total dan Rata-rata
Masukkan nol untuk keluar...
Nilai ke-1 = 6
Nilai ke-2 = 7
Nilai ke-3 = 8
Nilai ke-4 = 0
Jumlah total nilai = 21
Rata-rata = 7
```

PERNYATAAN DO WHILE

Pernyataan do while juga berguna untuk mengulang proses dan akan dijalankan minimal satu kali.

Bentuk penulisan :

```
do
{
  pernyataan;
} while (ungkapan)
```

Bagian pernyataan akan dijalankan secara berulang sampai ungkapan bernilai salah dan pengujian ungkapan akan dilakukan di belakang setelah pernyataan.
Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int x,y, maks;
 char lagi;
 do
 {
  clrscr();
  cout <<"Masukkan nilai-1 : "; cin >>x;
  cout <<"Masukkan nilai-2 : "; cin >>y;
  maks=(x<y) ? x : y;
  cout <<"Nilai terkecil adalah : " <<maks;
  cout <<"\n\nUlang lagi [Y/T] ? "; cin >>lagi;
 }while ((lagi=='y') || (lagi=='Y'));
}
```

Hasil :

```
Masukkan nilai-1 : 4
Masukkan nilai-2 : 44
Nilai terkecil adalah : 4

Ulang lagi [Y/T] ? t
```

PERNYATAAN FOR

Pernyataan for juga berguna untuk mengulang pengeksekusian terhadap satu atau sejumlah pernyataan.
Bentuk penulisan :

```
for (ungkapan1; ungkapan2; ungkapan3)
{
  pernyataan;
}
```

Dimana :

- Ungkapan1 merupakan pernyataan inisialisasi sebelum masuk ke for
- Ungkapan2 sebagai kondisi yang menentukan pengulangan terhadap pernyataan
- Ungkapan3 digunakan sebagai pengatur variabel yang digunakan dalam ungkapan1

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 char huruf;
 clrscr();
 for(huruf='A'; huruf < 'Z'; huruf++)
      cout <<huruf <<" ";
}
```

Hasil :

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y
```

Pernyataan for juga dapat digunakan untuk pengendalian isi variabel yang menurun.
Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int x;
 clrscr();
 for(x=20; x>=1; x--)
     cout <<x <<" ";
}
```

Hasil :

```
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
```

VARIASI FOR

- Menghilangkan sebuah bagian ungkapan
  Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 char huruf;
 clrscr();
 cout <<"Ketikkan karakter-karakter (0 = stop) : ";
 for (huruf=' '; huruf!='0'; )
     huruf=getche();
}
```

- **Loop tak hingga**
  Loop tak hingga dibuat dengan menghilangkan bagian ungkapan. Program tersebut meminta Anda mengetikkan sebuah huruf dan tidak akan berhenti. Untuk menghentikannya tekan tombol *CTRL + BREAK*
  Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 char huruf;
 clrscr();
 cout <<"Ketikkan karakter-karakter (CTRL+BREAK = stop) : ";
 for ( ; ; )
     huruf=getche();
}
```

FOR BERSARANG

Pada aplikasi tertentu, terkadang kita menggunakan pernyataan for yang juga berada di dalam pernyataan for.

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int i,j;
 clrscr();
 for (i=1 ; i <= 3 ; i++)
 {
 for (j=1 ; j <= 5 ; j++ ) cout <<i;
 cout <<"\n";
 }
 getch();
}
```

Hasil :

```
11111
22222
33333
```

Dari contoh tersebut dapat kita simpulkan bahwa variabel i menyatakan baris dan variabel j menyatakan kolom.

PERNYATAAN BREAK

Pernyataan break digunakan untuk memaksa keluar dari loop

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int i;
 clrscr();
 for (i=1; i<=25; i++)
 {
  cout <<i <<" ";
  if (i == 15) break;
 }
 cout <<"\nSelesai...!";
}
```

Hasil :

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Selesai...!
```

PERNYATAAN CONTINUE

Digunakan untuk menuju ke iterasi (putaran) berikutnya pada pernyataan yang terkait dengan perulangan

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int i;
 clrscr();
 for (i=1; i<=15; i++)
{
  if (i >=5 && i <=10)
  continue;
  cout <<i <<" ";
        }
 }
```

Hasil :

```
1 2 3 4 11 12 13 14 15
```

Tampak bahwa angka 5 sampai 10 tidak ditampilkan disebabkan oleh perintah CONTINUE

TUGAS

1. Buat program untuk menghasilkan deret bilangan Genap dan Ganjil antara 0 sampai dengan 50
2. Buat program untuk membuat tabel suhu CELCIUS, FAHRENHEIT dan REAMUR dengan menggunakan perintah DO-WHILE
   Ketentuan :
   Nilai Celcius 100 menurun ke 0
   Fahrenheit = 1,8 x Celcius + 32
   Reamur = 0,8 x Celcius

3. Buat program untuk menampilkan :
   *****
   ******
   ****
   ****

PERTEMUAN 6

FUNGSI

Sebuah fungsi berisi sejumlah pernyataan yang dikemas dalam sebuah nama. Nama ini selanjutnya dapat dipanggil beberapa kali di beberapa tempat dalam program.

Tujuan pembuatan fungsi adalah :
1. Memudahkan dalam mengembangkan program.
   Hal ini merupakan kunci dalam pembuatan program yang terstruktur dimana program dibagi menjadi beberapa modul yang kecil
2. Menghemat ukuran program.
   Manfaat ini terasakan kalau ada beberapa deretan instruksi yang sama digunakan pada beberapa tempat di dalam program.

Bentuk umum :

```
tipe nama_fungsi (deklarasi parameter)
{
  pernyataan;
  pernyataan;
}
```

- tipe
  Tipe nilai yang dihasilkan oleh fungsi. Jika tidak dinyatakan, hasil fungsi dianggap bilangan bulat (int)

- deklarasi parameter
  Daftar tipe dan nama variabel yang akan menerima nilai pada saat fungsi tersebut dipanggil. Setiap parameter dipisahkan oleh tanda koma. Jika fungsi tidak mempunyai parameter daftar ini akan kosong. Jadi hanya tanda kurung saja.

  Deklarasi parameter agak berbeda dengan deklarasi variabel. Pada deklarasi variabel, Anda dapat menyatakan sebuah tipe untuk beberapa variabel yang tipenya sama. Contoh : int a,b,c;

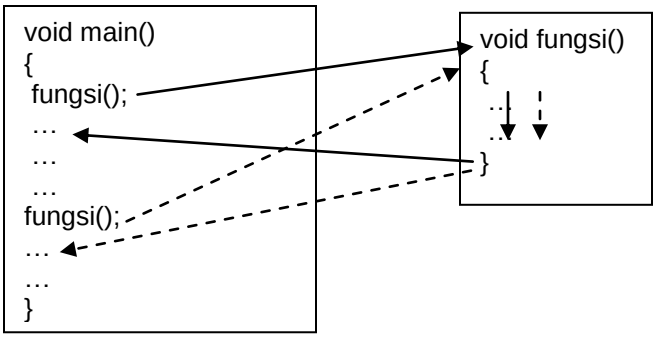
  Tetapi pada deklarasi parameter Anda harus menyatakan setiap tipe dari parameter.

  Bentuk umum :

  f (tipe nama_var1, tipe nama_var2, …, tipe nama_varn);

  Contoh : f (int a, int b, int c);

Bagan pemanggilan fungsi :



Ada dua cara untuk kembali ke program pemanggil :

1. Pada saat pernyataan terakhir dari function dijumpai (dijumpai tanda akhir fungsi "*}*")
   Contoh :

```
#include<iostream.h>
#include<conio.h>
void garis();
void main()
{
 clrscr();
 garis();
 cout <<"BIODATA" <<endl;
 garis();
 cout <<"NPM  : 12345" <<endl;
 cout <<"Nama : BUDI" <<endl;
 garis();
}
void garis()
{
 cout <<"--------------------------" <<endl;
}
```

   Prototipe fungsi garis()

   Pada contoh di atas, fungsi garis() digunakan untuk menampilkan karakter garis. Fungsi ini dipanggil tiga kali pada fungsi main(). Sebuah fungsi tidak dapat dipanggil kecuali sudah dideklarasikan. Manfaat dari pototipe fungsi adalah untuk menjamin tipe argumen yang dilewatkan pada pemanggilan fungsi benar-benar sesuai. Fungsi garis() tidak memiliki argumen dan nilai baliknya tidak ada (void).

2. Dengan menggunakan pernyataan return. Pernyataan return juga dapat dipakai tanpa menghasilkan sebuah nilai.
   Pernyataan return menyatakan dua hal :
   1. Return mengakhiri jalannya function dan kembali ke program pemanggil
   2. Menghasilkan sebuah nilai

Contoh :

```
#include<iostream.h>
#include<conio.h>
void hai();
void main()
{
 clrscr();
 hai();
}
void hai()
{
 cout <<"Hai.. Apa kabar..?" <<endl;
 return;
 cout <<"Baik-baik saja kannn..?" <<endl;
}
```

Hasil :

```
Hai.. Apa kabar..?
```

DEFINISI FUNGSI

Setiap fungsi yang dipanggil di dalam program harus didefinisikan. Letaknya bisa dimana saja. Khusus untuk fungsi yang disediakan sistem, definisinya sebenarnya ada dalam pustaka, yang akan digabungkan dengan progam sewaktu proses *linking.*

Contoh :

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void garis();
long kuadrat(long x);
void main()
{
 clrscr();
 cout <<"Nilai X" <<setw(10) <<"Kuadrat" <<endl;
 garis();
 for(long bil=1; bil <=10; bil++)
  cout <<bil <<setw(13) <<kuadrat(bil) <<endl;
 garis();
}

void garis()
```

```
{
 cout <<"------------------" <<endl;
}

long kuadrat(long x)
{
 return(x*x);
}
```

Hasil :

```
Nilai X   Kuadrat
------------------
1            1
2            4
3            9
4           16
5           25
6           36
7           49
8           64
9           81
10         100
------------------
```

Pernyataan return didalam fungsi digunakan untuk memberikan nilai balik fungsi.

LINGKUP VARIABEL

Lingkup variabel menentukan keberadaan suatu variabel tertentu dalam fungsi,
Jenis-jenis lingkup variabel yaitu :

1. Variabel Otomatis
   Yaitu variabel yang didefinisikan di dalam suatu fungsi, berlaku sebagai variabel lokal bagi fungsi, artinya variabel tersebut hanya dikenal di dalam fungsi tempat variabel didefinisikan. Pendeklarasian variabel dapat ditulis dengan awalan *auto.*
   Contoh :

```
#include<iostream.h>
#include<conio.h>
void alpha();
void main()
{
 int x=22;
 double y=44.5;
 clrscr();
 cout <<"Nilai pada fungsi main  : ";
 cout <<"X= " <<x <<" dan Y= " <<y <<endl;
 alpha();
 cout <<"Nilai pada fungsi main  : ";
 cout <<"X= " <<x <<" dan Y= " <<y <<endl;
}
void alpha()
{
 auto int x=11;
```

```
 auto double y=22.5;
 cout <<"Nilai pada fungsi alpha : ";
 cout <<"X= " <<x <<" dan Y= " <<y <<endl;
}
```

Hasil :

```
Nilai pada fungsi main  : X= 22 dan Y= 44.5
Nilai pada fungsi alpha : X= 11 dan Y= 22.5
Nilai pada fungsi main  : X= 22 dan Y= 44.5
```

2. Variabel Eksternal

   Adalah variabel yang didefinisikan di luar fungsi manapun. Variabel ini dikenal juga sebagai variabel *global*, karena variabel ini dikenal di semua fungsi. Pendeklarasian variabel dapat ditulis dengan awalan *extern.*

   Contoh :

```
#include<iostream.h>
#include<conio.h>
int x=1;
void tambah();
void main()
{
 clrscr();
 cout <<"Nilai awal X = " <<x <<endl;
 tambah();
 tambah();
 tambah();
 cout <<"Nilai X setelah fungsi = " <<x <<endl;
}
void tambah()
{
 extern x;
 x++;
}
```

   Hasil :

```
Nilai awal X = 1
Nilai X setelah fungsi = 4
```

3. Variabel Statis

   Baik variabel eksternal maupun otomatis dapat berkedudukan sebagai variabel statis. Variabel statis ditulis dengan awalan *static.* Suatu variabel statis mempunyai sifat :

   a. Jika variabel lokal berdiri sebagai variabel statis, maka :
      - Variabel tetap hanya dapat diakses pada fungsi yang mendefinisikannya
      - Variabel tidak hilang saat eksekusi fungsi berakhir
      - Inisialisasi akan dilakukan sekali saja, jika tidak ada maka variabel diisi dengan nol

   b. Jika variabel ekternal dijadikan variabel statis maka variabel ini dapat diakses oleh semua file yang didefinisikan.

Contoh :

```
#include<iostream.h>
#include<conio.h>
void tambah();
void main()
{
 int x=100;
 clrscr();
 cout <<"Nilai awal X di fungsi main = " <<x <<endl;
 tambah();
 tambah();
 tambah();
 cout <<"Nilai X di fungsi main = " <<x <<endl;
}
void tambah()
{
 static int x=44;
 x++;
 cout <<"Nilai X di fungsi tambah = " <<x <<endl;
}
```

Hasil :

```
Nilai awal X di fungsi main = 100
Nilai X di fungsi tambah = 45
Nilai X di fungsi tambah = 46
Nilai X di fungsi tambah = 47
Nilai X di fungsi main = 100
```

Jika perintah *static* pada fungsi tambah dihapus, maka akan menghasilkan :

```
Nilai awal X di fungsi main = 100
Nilai X di fungsi tambah = 45
Nilai X di fungsi tambah = 45
Nilai X di fungsi tambah = 45
Nilai X di fungsi main = 100
```

OPERATOR RESOLUSI LINGKUP

Operator ini digunakan untuk mengakses variabel yang didefinisikan di luar suatu fungsi dengan nama yang sama. Operator ini menggunakan dua buah tanda titik dua (::).

Contoh :

```
#include<iostream.h>
#include<conio.h>
int x=44;
void main()
{
 double x;
 clrscr();
 x=123.456;
 cout <<"Nilai x = "<<x <<" dan ::x = " <<::x <<endl;
 ::x=75;
 cout <<"Nilai x = "<<x <<" dan ::x = " <<::x <<endl;
 {
```

```
  char x='Y';
  ::x=11;
  cout <<"Nilai x = "<<x <<" dan ::x = " <<::x <<endl;
 }
}
```

Hasil :

```
Nilai x = 123.456 dan ::x = 44
Nilai x = 123.456 dan ::x = 75
Nilai x = Y dan ::x = 11
```

REFERENSI

Digunakan untuk memberikan nama alias dari variabel.
Bentuk deklarasi : *int &ref = nama_variabel;*
Contoh:

```
#include<iostream.h>
#include<conio.h>
void main()
{
 int i;
 int &r = i;
 clrscr();
 i=22;
 cout <<"Nilai i = "<<i <<" dan r = " <<r <<endl;

 i=44;
 cout <<"Nilai i = "<<i <<" dan r = " <<r <<endl;
}
```

Hasil :

```
Nilai i = 22 dan r = 22
Nilai i = 44 dan r = 44
```

Tampak bahwa pengubahan nilai terhadap i maupun r akan memberikan efek yang sama.

FUNGSI REKURSI

Suatu fungsi dapat memanggil fungsi yang merupakan dirinya sendiri. Rekursi jarang dipakai, diantaranya disebabkan :

- Membuat fungsi sulit untuk dipahami
- Hanya cocok untuk persoalan tertentu saja
- Memerlukan *stack* dengan ukuran yang lebih besar.

Contoh :

```
#include<iostream.h>
#include<conio.h>
long faktorial(int m);
void main()
{
 int x;
```

```
 clrscr();
 cout <<"Mencari Nilai Faktorial" <<endl;
 cout <<"Masukkan sebuah bilangan bulat positif : "; cin >>x;
 cout <<"Nilai faktorial dari " <<x <<" adalah " <<faktorial(x);
}

long faktorial(int m)
{
 if (m==0)
        return(1);
 else
        return(m*faktorial(m-1));
}
```

Hasil :

```
Mencari Nilai Faktorial
Masukkan sebuah bilangan bulat positif : 4
Nilai faktorial dari 4 adalah 24
```

Buat program dengan fungsi dan input untuk mencari nilai terbesar dari dua bilangan.
Jawab :

```
#include<iostream.h>
#include<conio.h>
int proses (int a, int b);
void main()
{
 int x, y;
 clrscr();
 cout <<"Mencari Nilai Terbesar" <<endl;
 cout <<"Masukkan nilai x : "; cin >>x;
 cout <<"Masukkan nilai y : "; cin >>y;
 cout <<"Nilai terbesar adalah : " << proses(x,y);
}

int proses (int a, int b)
{
 int hasil;
 hasil = (a > b) ?  a : b;
 return hasil;
}
```

Hasil :

```
Mencari Nilai Terbesar
Masukkan nilai x : 22
Masukkan nilai y : 44
Nilai terbesar adalah : 44
```

TUGAS

1. Buat program untuk menghitung gaji harian PT. XYZ dengan ketentuan :
   a. Gaji pokok karyawan Rp. 2000/jam
   b. Bila karyawan bekerja lebih dari 7 jam/hari maka kelebihannya dihitung lembur yang besarnya 1.5 dari gaji pokok
   c. Untuk karyawan yang bekerja 8 jam/hari atau lebih akan mendapat tambahan uang makan sebesar Rp. 3500
   d. Karyawan yang bekerja 9 jam/hari atau lebih akan mendapat uang transport lembur sebesar Rp. 4000

   Program ini akan terdiri dari 5 buah fungsi : main(), pokok(), lembur(), makan() dan jasa()
   Input : NIP, Nama, Jumlah jam kerja
   Output : NIP, Nama, Gaji pokok, Lembur, Uang makan, Transport lembur

PERTEMUAN 7

FUNGSI MATEMATIKA DAN STRING

FUNGSI MATEMATIKA

- abs()
  Kegunaan : memperoleh nilai absolut (nilai mutlak) suatu bilangan
  Contoh :

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
void main()
{
 int x;
 clrscr();
 cout <<"Masukkan sebuah bilangan negatif : ";
 cin >> x;
 cout <<"Nilai absolut = " <<abs(x);
}
```

  Hasil :

```
Masukkan sebuah bilangan negatif : -44
Nilai absolut = 44
```

- ceil()
  Kegunaan : memperoleh nilai pembulatan ke atas.

- floor()
  Kegunaan : memperoleh nilai pembulatan ke bawah
  Contoh :

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
void main()
{
 clrscr();
 cout <<"Pembulatan dengan ceil()" <<endl;
 cout <<"Pembulatan 4.1 = " <<ceil(4.1) <<endl;
 cout <<"Pembulatan 4.5 = " <<ceil(4.5) <<endl;
 cout <<"Pembulatan 0.4 = " <<ceil(0.4) <<endl;
 cout <<"Pembulatan dengan floor()" <<endl;
 cout <<"Pembulatan 4.1 = " <<floor(4.1) <<endl;
 cout <<"Pembulatan 4.5 = " <<floor(4.5) <<endl;
 cout <<"Pembulatan 0.4 = " <<floor(0.4) <<endl;
}
```

Hasil :

```
Pembulatan dengan ceil()
Pembulatan 4.1 = 5
Pembulatan 4.5 = 5
Pembulatan 0.4 = 1
Pembulatan dengan floor()
Pembulatan 4.1 = 4
Pembulatan 4.5 = 4
Pembulatan 0.4 = 0
```

- cos(), sin(), tan()
  Kegunaan : memperoleh nilai cosinus, sinus dan tangen dalam bentuk radian.

  Catatan :
  $360^0 = 2\pi$ radian

  $1^0 = \frac{2\pi}{360}$ radian $= \frac{2 * 3{,}14159}{360}$ radian

  $= \frac{1}{57{,}2958}$ radian

  Contoh :

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
void main()
{
 float x;
 clrscr();
 cout <<"Masukkan sebuah bilangan : "; cin >>x;
 cout <<"Nilai cosinus = " <<cos(x/57.2958) <<endl;
 cout <<"Nilai sinus = " <<sin(x/57.2958) <<endl;
 cout <<"Nilai tangen = " <<tan(x/57.2958) <<endl;
}
```

  Hasil :

```
Masukkan sebuah bilangan : 30
Nilai cosinus = 0.866025
Nilai sinus = 0.5
Nilai tangen = 0.57735
```

- exp()
  Kegunaan : memperoleh nilai eksponensial dari suatu bilangan

- log() dan log10(x)
  Kegunaan : memperoleh nilai logaritma alami dan logaritma basis10 dari suatu bilangan

- sqrt()
  Kegunaan : menghasilkan akar dari suatu bilangan
  Contoh :

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

```
#include<math.h>
void main()
{
 float x;
 clrscr();
 cout <<"Masukkan sebuah bilangan : "; cin >>x;
 cout <<"Nilai exp(x) = " <<exp(x) <<endl;
 cout <<"Nilai log(x) = " <<log(x) <<endl;
 cout <<"Nilai log10(x) = " <<log10(x) <<endl;
 cout <<"Nilai sqrt(x) = " <<sqrt(x) <<endl;
}
```

Hasil :

```
Masukkan sebuah bilangan : 100
Nilai exp(x) = 2.68812e+43
Nilai log(x) = 4.60517
Nilai log10(x) = 2
Nilai sqrt(x) = 10
```

- hypot()
  Kegunaan : memperoleh sisi miring segitiga siku-siku
  Contoh :

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
void main()
{
 float x, y;
 clrscr();
 cout <<"Masukkan alas segitiga : "; cin >>x;
 cout <<"Masukkan tinggi segitiga : "; cin >>y;
 cout <<"Panjang sisi miring = " <<hypot(x,y);
}
```

  Hasil :

```
Masukkan alas segitiga : 3
Masukkan tinggi segitiga : 4
Panjang sisi miring = 5
```

- max()
  Kegunaan : memperoleh nilai terbesar dari dua bilangan.

- min()
  Kegunaan : memperoleh nilai terkecil dari dua bilangan.

  Catatan : fungsi max() dan min() menggunakan file header *stdlib.h*
  Contoh :

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

```
void main()
{
 int x,y;
 clrscr();
 cout <<"Masukkan nilai x : "; cin >>x;
 cout <<"Masukkan nilai y : "; cin >>y;
 cout <<"Nilai terbesar adalah " <<max(x,y) <<endl;
 cout <<"Nilai terkecil adalah " <<min(x,y);
}
```

Hasil :

```
Masukkan nilai x : 44
Masukkan nilai y : 33
Nilai terbesar adalah 44
Nilai terkecil adalah 33
```

- pow(x,y)
  Kegunaan : memperoleh nilai x pangkat y
  Contoh :

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
void main()
{
 float x,y;
 clrscr();
 cout <<"Masukkan nilai x : "; cin >>x;
 cout <<"Masukkan nilai y : "; cin >>y;
 cout <<"Nilai x pangkat y = " <<pow(x,y);
}
```

Hasil :

```
Masukkan nilai x : 4
Masukkan nilai y : 3
Nilai x pangkat y = 64
```

- srand()
  Kegunaan : menghasilkan bilangan bulat secara acak dengan nilai tetap (menggunakan file header *stdlib.h*)
  Contoh :

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
void main()
{
 clrscr();
 srand(3);
 for(int i=0; i<4; i++)
 cout <<random(1000) <<endl;
}
```

Hasil :

```
31
998
997
429
```

- randomize()
  Kegunaan : menghasilkan bilangan bulat secara acak yang berubah-ubah setiap kali program dijalankan (menggunakan file header *stdlib.h*)
  Contoh :

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
void main()
{
 clrscr();
 randomize();
 for(int i=0; i<4; i++)
 cout <<random(1000) <<endl;
}
```

FUNGSI STRING

- strlen()
  Kegunaan : menghitung panjang suatu string (menggunakan file header *string.h*)

- strupr()
  Kegunaan : merubah teks menjadi huruf kapital

- strlwr()
  Kegunaan : merubah teks menjadi huruf kecil
  Contoh :

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
void main()
{
 char teks[50];
 clrscr();
 cout <<"Ketik suatu teks : "; cin.getline(teks, sizeof(teks));
 cout <<"Panjang teks = " <<strlen(teks) <<endl;
 cout <<"Teks dengan huruf kapital : " <<strupr(teks) <<endl;
 cout <<"Teks dengan huruf kecil: " <<strlwr(teks) <<endl;
}
```

- strcat ()
  Kegunaan : menggabungkan string
  Contoh :

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

```
#include<string.h>
void main()
{
 char teks1[50];
 char teks2[50];
 clrscr();
 cout <<"Ketik teks1 : "; cin.getline(teks1, sizeof(teks1));
 cout <<"Ketik teks2 : "; cin.getline(teks2, sizeof(teks2));
 cout <<"Digabung Menjadi : " <<strcat(teks1, teks2);
}
```

- strstr()
  Kegunaan : mencari suatu karakter tertentu dalam suatu string
  Contoh :

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
void main()
{
 char teks1[50];
 char teks2[50];
 clrscr();
 cout <<"Ketik teks1 : "; cin.getline(teks1, sizeof(teks1));
 cout <<"Ketik teks2 : "; cin.getline(teks2, sizeof(teks2));
 if (strstr(teks1, teks2))
 cout <<teks2 <<" terdapat pada " <<teks1;
 else
 cout <<teks2 <<" tidak ada pada " <<teks1;
}
```

- strrev()
  Kegunaan : membalik suatu string
  Contoh :

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
void main()
{
 char teks1[50];
 clrscr();
 cout <<"Ketik teks : "; cin.getline(teks1, sizeof(teks1));
 cout <<"Jika dibalik menjadi : " <<strrev(teks1);
}
```

- strcpy(teks1, teks2)
  Kegunaan : menyalin isi teks2 ke teks1
  Contoh :

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

```
void main()
{
 char t1[50];
 char t2[50];
 int i;
 clrscr();
 cout <<"Masukkan teks1 : "; cin.getline(t1, sizeof(t1));
 cout <<"Masukkan teks2 : "; cin.getline(t2, sizeof(t2));
 cout <<"Isi teks1 = " <<t1 <<endl;
 cout <<"Isi teks2 = " <<t2 <<endl;
 cout <<"Ditukar dengan strcpy(t1,t2)" <<endl;
 strcpy(t1,t2);
 cout <<"Isi teks1 = " <<t1 <<endl;
 cout <<"Isi teks2 = " <<t2 <<endl;
}
```

TUGAS

1. Buat program untuk menghitung jarak peluru yang ditembakkan dari suatu lokasi dengan sudut penembakan ($\propto$) dan kecepatan (V_0) dimasukan melalui keyboard dan gravitasi (g) = 9,8. Rumus untuk menghitung jarak adalah :

   Jarak peluru = $\frac{2 * V_0^2 * \text{Sin}(\propto) * \text{Cos}(\propto)}{g}$

2. Buat program untuk mencari kata apakah POLINDROM atau tidak. Misalnya :
   TAAT = TAAT : POLINDROM
   KASUR = RUSAK : bukan POLINDROM

PERTEMUAN 8

ARRAY

Array adalah deretan variabel yang berjenis sama dan mempunyai nama sama. Pada bahasa C, array mempunyai lokasi yang bersebelahan. Alamat terkecil menunjuk ke elemen pertama dan alamat terbesar menunjuk ke alamat terakhir. Sebuah elemen pada array diakses melalui *subscript* atau indeks array yang dimulai dari nol.

ARRAY BERDIMENSI SATU

Bentuk umum deklarasi array berdimensi satu :

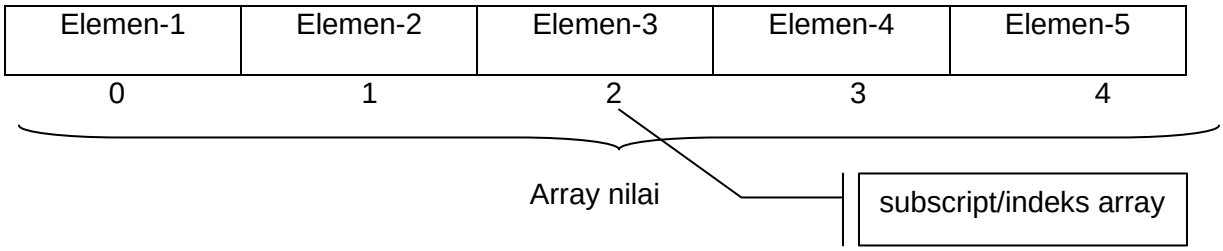
type nama_array[ukuran];

type : menyatakan tipe dasar array
ukuran : menyatakan banyaknya elemen pada array

misal :

int nilai[5];

Hal ini menyatakan array nilai dapat menyimpan lima buah data bertipe *int* dan dapat digambarkan sebagai berikut :



Setelah array didefinisikan, elemen array dapat diakses dengan bentuk :

nama_array[subscript];

Sebagai contoh jika ingin mengambil elemen3 pada gambar diatas, dapat diakses dengan menyebutkan nilai[2].

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int nilai[5];
 clrscr();
 nilai[0]=11;
 nilai[1]=22;
 nilai[2]=33;
 nilai[3]=44;
 nilai[4]=45;
 cout <<"Data Array ke-1 = " <<nilai[0] <<endl;
 cout <<"Data Array ke-2 = " <<nilai[1] <<endl;
```

```
 cout <<"Data Array ke-3 = " <<nilai[2] <<endl;
 cout <<"Data Array ke-4 = " <<nilai[3] <<endl;
 cout <<"Data Array ke-5 = " <<nilai[4] <<endl;
}
```

Hasil :

```
Data Array ke-1 = 11
Data Array ke-2 = 22
Data Array ke-3 = 33
Data Array ke-4 = 44
Data Array ke-5 = 45
```

Untuk menyingkat program di atas, Anda dapat menggunakan perintah perulangan *for* untuk memasukkan dan mencetak data.

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int nilai[5];
 int i;
 clrscr();
 for(i=0; i<5; i++)
 {
  cout <<"Input Data ke-" <<i+1 <<" = "; cin >>nilai[i];
 }
 cout <<"-- Hasil Input Data --" <<endl;
 for(i=0; i<5; i++)
 {
  cout <<"Data Array ke-" <<i+1 <<" = " <<nilai[i] <<endl;
 }
}
```

Hasil :

```
Input Data ke-1 = 11
Input Data ke-2 = 22
Input Data ke-3 = 33
Input Data ke-4 = 44
Input Data ke-5 = 55
-- Hasil Input Data --
Data Array ke-1 = 11
Data Array ke-2 = 22
Data Array ke-3 = 33
Data Array ke-4 = 44
Data Array ke-5 = 55
```

Program di atas akan meminta pemakai untuk memasukkan 5 buah data dari keyboard yang akan disimpan dalam array nilai. Selanjutnya data yang ada pada array tersebut ditampilkan ke layar.

Program di atas disebut juga dengan array berdimensi satu. Harus dipastikan bahwa tidak ada pengaksesan elemen data diluar 0 sampai dengan 4, karena dapat menyebabkan data dari variabel berubah.

MEMBERIKAN NILAI AWAL TERHADAP ARRAY

Array juga dapat diberi nilai awal (inisialisasi) pada saat didefinisikan.
Bentuk deklarasi :

type nama_array[ukuran] ={nilai-1, nilai-2 ,..., nilai-n};

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int i;
 int data[7]={11,22,33,44,55,66,77};
 clrscr();
 for(i=0; i<7; i++)
 {
  cout <<"Isi elemen ke-" <<i+1 <<" = " <<data[i] <<endl;
 }
}
```

Hasil :

```
Isi elemen ke-1 = 11
Isi elemen ke-2 = 22
Isi elemen ke-3 = 33
Isi elemen ke-4 = 44
Isi elemen ke-5 = 55
Isi elemen ke-6 = 66
Isi elemen ke-7 = 77
```

ARRAY STRING

Pada array jenis ini, indeks pertama menunjukkan banyaknya string dan indeks kedua menunjukkan panjang maksimum string.
Bentuk penulisan :

char nama_array[jumlah array] [panjang string];

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 char nama[50][20];
 int i, data;
 clrscr();
 cout <<"Banyaknya data : "; cin >>data;
 for (i=0; i<data; i++)
 {
  cout <<"Nama ke-" <<i+1 <<" = ";
  cin >>nama[i];
```

```
 }
 cout <<"-- Hasil Input Data --" <<endl;
 for (i=0; i<data; i++)
 {
  cout <<nama[i] <<" ";
 }
}
```

:

INISIALISASI ARRAY STRING

Array string juga dapat diberi nilai awal (inisialisasi) pada saat didefinisikan.
Bentuk deklarasi :

char nama_array [jumlah array] [panjang string]={"nilai-1","nilai-2" ,...,"nilai-n"};

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 char hari[7][20]={"Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu",
"Minggu"};
 int x;
 clrscr();
 mulai :
 cout <<"Masukkan kode hari [1-7] : "; cin >>x;
 if ((x>=1 && x<=7))
        cout <<"Kode = " <<x <<" adalah hari " <<hari[x-1];
 else
        {
         cout <<"Salah Kode...!" <<endl;
         goto mulai;
        }
}
```

Hasil :

```
Masukkan kode hari [1-7] : 8
Salah Kode...!
Masukkan kode hari [1-7] : 5
Kode = 5 adalah hari Jumat
```

ARRAY BERDIMENSI DUA

Bahasa C++ mengijinkan array dengan beberapa dimensi, contoh array dua dimensi. Array dua dimensi adalah sebuah daftar yang terdiri atas array-array 1 dimensi. Misalnya kita akan membuat array 2 dimensi dengan ukuran 2x3, bertipe int, maka deklarasinya adalah : *type nama_array [baris] [kolom];*

Deklarasi array pada bahasa C++ tidak sama dengan bahasa-bahasa lain. Biasanya setiap dimensi dipisahkan oleh tanda koma, tetapi bahasa C++ meletakkan setiap dimensi di dalam *kurung siku*.

Misalnya array dua dimensi dengan deklarasi *nilai [2][3];* dapat digambarkan sebagai berikut :

Nilai[0][0]	Nilai[0][1]	Nilai[0][2]
Nilai[1][0]	Nilai[1][1]	Nilai[1][2]

Array dua dimensi disimpan di dalam matriks yang tersusun menjadi baris dan kolom. Index pertama menunjukkan barisnya dan index kedua menunjukkan kolomnya, misal :

nilai [2][3] = $\begin{bmatrix} A\,B\,C \\ D\,E\,F \end{bmatrix}$

Untuk memasukkan data array adalah dengan menyebutkan lokasi baris dan kolom dari suatu array.

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int nilai[2][3];
 int x,y;
 clrscr();
 nilai[0][0] = 11;
 nilai[0][1] = 22;
 nilai[0][2] = 33;
 nilai[1][0] = 44;
 nilai[1][1] = 55;
 nilai[1][2] = 66;
 cout <<"Masukkan index baris [1-2] : ";
 cin >>x;
 cout <<"Masukkan index kolom [1-3] : ";
 cin >>y;
 cout <<"Data array baris " <<x <<" kolom " <<y <<" adalah " <<nilai[x-1][y-1];
}
```

Hasil :

```
Masukkan index baris [1-2] : 2
Masukkan index kolom [1-3] : 1
Data array baris 2 kolom 1 adalah 44
```

Untuk memasukkan data array dua dimensi, Anda dapat menggunakan perulangan *for* untuk memasukkan dan mencetak data array.

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int nilai[2][3];
 int i,j,x,y;
```

```
 clrscr();
 x=1;
 for(i=0; i<2; i++)
 {
  y=1;
  for(j=0; j<3; j++)
  {
        cout <<"Nilai baris-"<<x <<" kolom-"<<y <<" = ";
        cin >>nilai[i][j];
        y++;
  }
  x++;
  cout <<endl;
 }

 cout <<"Hasil Input :" <<endl;
 for(i=0; i<2; i++)
 {
  for(j=0; j<3; j++)
  {
        cout <<nilai[i][j] <<" ";
  }
  cout <<endl;
 }
}
```

Hasil :

```
Nilai baris-1 kolom-1 = 1
Nilai baris-1 kolom-2 = 2
Nilai baris-1 kolom-3 = 3

Nilai baris-2 kolom-1 = 4
Nilai baris-2 kolom-2 = 5
Nilai baris-2 kolom-3 = 6

Hasil Input :
1 2 3
4 5 6
```

INISIALISASI ARRAY BERDIMENSI DUA

Pendefinisian array dan inisialisasi terhadap array berdimensi dua dari program diatas dapat ditulis menjadi :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int i,j;
 int nilai[2][3]=
 {
  {1,2,3},
  {4,5,6}
 };
```

```
 clrscr();
 cout <<"-- Data Array --" <<endl;
 for(i=0; i<2; i++)
 {
  for(j=0; j<3; j++)
  {
        cout <<nilai[i][j] <<" ";
  }
  cout <<endl;
 }
}
```

Hasil :

```
-- Data Array --
1 2 3
4 5 6
```

ARRAY BERDIMENSI TIGA

Bentuk umum pendefinisian array yang berdimensi tiga adalah :

type nama_array[subscript1] [subscript2] [subscript3];

misal :

```
int nilai [2] [3] [3];
```

Seperti halnya array berdimensi satu atau dua, array berdimensi tiga juga bisa diinisialisasi.

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int i,j,k;
 int nilai[2][3][3]=
 {
  {{1,2,3},
        {4,5,6},
        {7,8,9}},
  {{10,11,12},
        {13,14,15},
        {16,17,18}},
 };

 clrscr();
 cout <<"-- Data Array --" <<endl;
 for(i=0; i<2; i++)
 {
  for(j=0; j<3; j++)
  {
        for (k=0; k<3; k++)
        {
         cout <<nilai[i][j][k] <<" ";
```

```
            }
            cout <<endl;
        }
       }
      }
```

Hasil :

```
-- Data Array --
1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
16 17 18
```

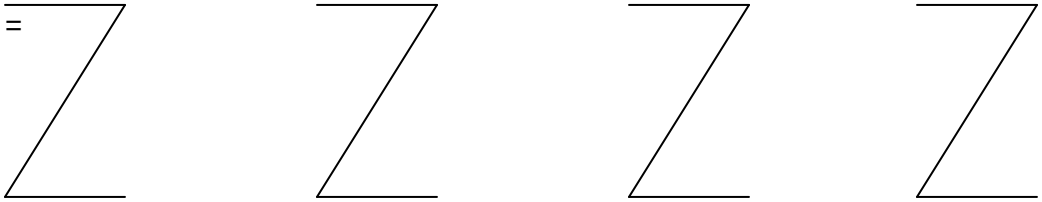
TUGAS

1. Buat program dengan array untuk memasukkan data nilai Bahasa C++.
   Input
   Jumlah Data : _

   NPM : _
   Nama : _
   Nilai UTS : _
   Nilai UAS : _
   Nilai Absen : _
   Nilai Tugas : _

   Proses
   NA = ((30%*UTS)+(40%*UAS)+(10%*ABSEN)+(20%*TUGAS))

   Output
   LAPORAN NILAI PEMROGRAMAN C++
   UNIVERSITAS ABC JAKARTA
   ==========
   =
   NO NPM NAMA UTS UAS ABSEN TUGAS AKHIR
   ==========
   =

   ==========
   =
   Dibuat Oleh : <NAMA ANDA>

2. Buat program untuk mengurutkan data pada array dengan metode *bubble sort*
3. Buat program untuk menjumlah dua buah matriks ber-ordo 2x2

PERTEMUAN 9

POINTER

Setiap byte di dalam memori komputer memiliki sebuah alamat. Alamat memori dimulai dari 0. Pada komputer yang memiliki memori 640Kb, alamat memori tertinggi yaitu 655.359. Didalam memori inilah variabel disimpan. Tetapi tentu saja pemogram tidak perlu menyebutkan alamat dari suat variabel secara eksplisit. Pada saat program dimuat di dalam memori, variabel akan diletakkan dengan sendirinya pada alamat tertentu.

Pointer banyak dilibatkan dalam program C++, misalnya untuk melewatkan string dari suatu fungsi ke fungsi yang lain. Penerapan pointer yang paling umum yaitu untuk menciptakan variabel dinamis yang memungkinkan untuk memakai memori bebas (memori yang belum dipakai) selama eksekusi program.

Variabel pointer sering dikatakan sebagai variabel yang menunjuk ke obyek lain. Pada kenyataan yang sebenarnya, variabel pointer berisi alamat dari suatu objek lain yaitu objek yang dikatakan ditunjuk oleh pointer. Sebagai contoh, px adalah pointer dan x adalah variabel yang ditunjuk oleh px. Jika px berada pada alamat memori (alamat awal) 1000, maka px akan berisi 1000.

Agar suatu pointer menunjuk ke variabel lain, mula-mula pointer harus diisi dengan alamat dari variabel yang akan ditunjuk. Untuk menyatakan alamat dari suatu variabel, operator **&** (operator alamat, yang bersifat unary) bisa digunakan dengan cara menempatkan operator di depan nama variabel.
Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int alpha=11;
 int beta=22;
 int charlie=33;
 clrscr();
 cout <<"Isi variabel :" <<endl;
 cout <<"alpha   = " <<alpha <<endl;
 cout <<"beta    = " <<beta <<endl;
 cout <<"charlie = " <<charlie <<endl;
 cout <<"Alamat variabel :" <<endl;
 cout <<"alpha   = " <<&alpha <<endl;
 cout <<"beta    = " <<&beta <<endl;
```

```
 cout <<"charlie = " <<&charlie <<endl;
}
```

Hasil :

```
Isi variabel :
alpha   = 11
beta    = 22
charlie = 33
Alamat variabel :
alpha   = 0x23d7244c
beta    = 0x23d7244a
charlie = 0x23d72448
```

Suatu variabel pointer dideklarasikan dengan bentuk :

*tipe_data *nama_variabel;*

- tipe_data berupa sembarang tipe
- nama_variabel adalah nama dari variabel pointer.

Jika suatu variabel sudah ditunjuk pointer, variabel tersebut dapat diakses melalui variabel itu sendiri (dikatakan sebagai pengaksesan langsung) ataupun melalui pointer (dikatakan sebagai pengaksesan tidak langsung). Pengaksesan tak langsung dilakukan dengan menggunakan operator indirection berupa simbol ***** (bersifat unary).

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int x, y;
 int *px;   // px adalah pointer yang menunjuk objek bertipe int
 clrscr();

 x=44;
 px=&x;
 y=*px;
 cout <<"Nilai  x = " <<x <<endl;
 cout <<"Nilai &x = " <<&x <<endl;
 cout <<"Nilai px=&x adalah " <<px <<endl;
 cout <<"Isi *px  = " <<*px <<endl;
 cout <<"Nilai y=*px adalah " <<y;
}
```

Hasil :

```
Nilai  x = 44
Nilai &x = 0x24072434
Nilai px=&x adalah 0x24072434
Isi *px  = 44
Nilai y=*px adalah 44
```

Antara tipe pointer dan tipe objek yang akan ditunjuk oleh pointer haruslah sejenis.

Contoh :

```
/* Pemakaian pointer yang salah */
#include <iostream.h>
#include <conio.h>
void main()
{
 float *pu;
 float nu;
 int u = 1234;
 clrscr();
 pu=&u; //pernyataan ini salah karena tipe pu dan u berbeda
 nu = *pu;
 cout <<"Nilai  u = " <<u <<endl;
 cout <<"Nilai nu = " <<nu;
}
```

Program di bawah ini memberikan gambaran tentang pengubahan isi suatu variabel secara tak langsung.

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 float d, *pd;
 clrscr();
 d=54.5;
 cout <<"Isi d semula = " <<d <<endl;
 pd=&d;
 *pd = *pd + 10;  // identik dengan d = d + 10
 cout <<"Isi d kini = " <<d;
}
```

Hasil :

```
Isi d semula = 54.5
Isi d kini = 64.5
```

POINTER dan ARRAY

Hubungan antara pointer dan array pada C++ sangatlah erat, sebab sesunguhnya array secara internal akan diterjemahkan dalam bentuk pointer.
Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 static int tgl_lahir[]={24, 12, 1970};
 int i, *ptgl;
 clrscr();

 ptgl = tgl_lahir;  // ptgl berisi alamat array
 for (i=0; i<3; i++)
        cout <<"Elemen subscript ke-" <<i <<" adalah " <<*(ptgl+i) <<endl;
}
```

Hasil :

```
Elemen subscript ke-0 adalah 24
Elemen subscript ke-1 adalah 12
Elemen subscript ke-2 adalah 1970
```

POINTER dan STRING

Program di bawah ini menggambarkan pertukaran dua string yang dilakukan melalui pertukaran isi array melalui pointer. Perhatikan bahwa dengan pointer, penyalinan isi arrray tidak perlu dilakukan. Sebagai efeknya bisa menghemat waktu eksekusi.
Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 char *nama1 = "CAT WOMAN";
 char *nama2 = "BATMAN";
 char *namax;
 clrscr();

 cout <<"Kata Semula :" <<endl;
 cout <<"Nama-1 = " <<nama1 <<endl;
 cout <<"Nama-2 = " <<nama2 <<endl;

 namax=nama1; /* penukaran string dengan pointer */
 nama1=nama2;
 nama2=namax;
```

```
 cout <<"Setelah ditukar :" <<endl;
 cout <<"Nama-1 = " <<nama1 <<endl;
 cout <<"Nama-2 = " <<nama2;
}
```

Hasil :

```
Kata Semula :
Nama-1 = CAT WOMAN
Nama-2 = BATMAN
Setelah ditukar :
Nama-1 = BATMAN
Nama-2 = CAT WOMAN
```

POINTER MENUNJUK POINTER

Suatu pointer bisa saja menunjuk ke pointer lain. Definisi untuk membentuk rantai pointer adalah sebagai berikut :

```
int x;
int *px1;
int **px2;
```

dimana:

- x adalah variabel bertipe int
- px1 adalah variabel pointer yang menunjuk ke x
- px2 adalah variabel pointer yang menunjuk ke pointer px1 dengan menggunakan dua tanda *

Agar px1 menunjuk ke variabel x digunakan perintah : px1 = &x;
Agar px2 menunjuk ke px1 digunakan perintah : px2=&px1;
Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int x = 44;
 int *px1;
 int **px2;
 px1=&x;
 px2=&px1;
 cout <<"Nilai x=44" <<endl;
 cout <<"Nilai &x = " <<&x <<endl;
 cout <<"Nilai &px1 = " <<&px1 <<endl;
 cout <<"Nilai *px1 = " <<*px1 <<endl;
 cout <<"Nilai **px2 adalah : " <<**px2 <<endl;
}
```

Hasil :

```
Nilai x=44
Nilai &x = 0x1bd72430
Nilai &px1 =0x1bd7242c
Nilai *px1 = 44
Nilai **px2 adalah : 44
```

POINTER dan FUNGSI

Pointer dan kaitannya dengan fungsi meliputi :

- Pointer sebagai parameter fungsi
  Penerapan pointer sebagai parameter yaitu jika diinginkan agar nilai suatu variabel internal dapat diubah oleh fungsi yang dipanggil.
  Contoh :

```
#include <iostream.h>
#include <conio.h>
void rubah(int *x, int *y);
void main()
{
 int a=3; int b=7;
 clrscr();
 cout <<"Nilai semula : " <<endl;
 cout <<"a = " <<a <<endl;
 cout <<"b = " <<b <<endl;
 rubah(&a,&b);
 cout <<"Nilai setelah fungsi : " <<endl;
 cout <<"a = " <<a <<endl;
 cout <<"b = " <<b <<endl;
}

void rubah(int *x, int *y)
{
 *x = *x + 2;
 *y = *y + 2;
}
```

  Hasil :

```
Nilai semula :
a = 3
b = 7
Nilai setelah fungsi :
a = 5
b = 9
```

- Pointer sebagai keluaran fungsi
  Suatu fungsi dapat dibuat agar keluarannya berupa pointer. Misalnya suatu fungsi menghasilkan keluaran berupa pointer yang menunjuk ke string.
  Contoh :

```
#include <iostream.h>
```

```
#include <conio.h>
char *nama_bulan(int n);
void main()
{
 int bl;
 char lagi;

 do
 {
  clrscr();
  cout <<"Masukkan Kode Bulan [1..12] : ";
  cin >> bl;
  cout <<"Kode bulan : " <<bl <<", nama bulan = " <<nama_bulan(bl) <<endl;
  cout <<"Input lagi [y/t] : ";
  lagi=getch();
 }
 while (lagi == 'y' || lagi == 'Y');
}

char *nama_bulan(int n)
{
 static char *bulan[] =
   {
            "Kode bulan salah", "Januari", "Februari", "Maret", "April",
            "Mei", "Juni", "Juli", "Agustus", "September", "Oktober",
            "November", "Desember"
    };
 return ((n<1 || n>12) ? bulan[0] : bulan[n]);
}
```

Hasil :

```
Masukkan Kode Bulan [1..12] : 4
Kode bulan : 4, nama bulan = April
Input lagi [y/t] :
```

LATIHAN

1. Buat program dengan menggunakan pointer sehingga menghasilkan keluaran sebagai berikut :
   D
   ND
   AND
   LAND
   RLAND
   ORLAND
   BORLAND

PERTEMUAN 10

STRUKTUR

Struktur adalah koleksi dari variabel yang dinyatakan dengan sebuah nama dengan sifat setiap variabel dapat memiliki tipe yang berlainan. Struktur biasa dipakai untuk mengelompokkan beberapa informasi yang berkaitan menjadi sebuah kesatuan.

Struktur bermanfaat untuk mengelompokkan sejumlah data dengan tipe yang berlainan. Masing –masing tipe elemen struktur dapat berlainan. Adapun variabel struktur menyatakan bahwa variabel struktur yang dideklarasikan bisa lebih dari satu dan dipisahkan dengan tanda titik koma.

Deklarasi struktur :

```
struct nama_tipe_struktur
{ tipe field-1;
  tipe field-2;
  tipe field-n;
} variabel_struktur1, ..., variabel struktur-n;
```

Misalnya :

```
 struct data_tanggal
{ int tanggal;
  int bulan;
  int tahun;
};
data_tanggal tgl_lahir;
```

ATAU

```
struct data_tanggal
        { int tanggal;
          int bulan;
          int tahun;
        } tgl_lahir;
```

MENGAKSES ELEMEN STRUKTUR

Elemen struktur dapat diakses dengan menggunakan bentuk : *variabel_struktur.nama_field* (antara variabel struktur dan nama field dipisahkan dengan tanda TITIK)

Misalnya :

```
tgl_lahir.tanggal = 30;
cin >> tgl_lahir.tanggal;
cout <<tgl_lahir.tanggal;
```

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 char nama[25];
 struct data_tanggal
 { int tanggal;
        int bulan;
        int tahun;
 }tgl_lahir;
```

```
 clrscr();
 cout <<"Nama Anda     : "; cin.getline(nama, sizeof(nama));
 cout <<"Tanggal Lahir : "; cin >>tgl_lahir.tanggal;
 cout <<"Bulan   Lahir : "; cin >>tgl_lahir.bulan;
 cout <<"Tahun   Lahir : "; cin >>tgl_lahir.tahun;
 clrscr();
 cout <<"Nama lengkap  : " <<nama <<endl;
 cout <<"Tanggal Lahir : " <<tgl_lahir.tanggal <<"-"
                           <<tgl_lahir.bulan <<"-"
                           <<tgl_lahir.tahun;
}
```

Dalam suatu struktur, elemen yang terkandung di dalamnya bisa juga berupa struktur, misalnya :

```
struct data_tanggal
{ int tanggal;
  int bulan;
  int tahun;
} tgl_lahir;

struct data_rekan
{ char nama[25];
   struct data_tanggal tgl_lahir;
 } info_rekan;
```

Contoh :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int i,n;
 struct data_tanggal
 { int tanggal;
        int bulan;
        int tahun;
 };

 struct data_rekan
 { char nama[30];
        struct data_tanggal tgl_lahir;
 }info_rekan;

 cout <<"Nama Lengkap : "; cin.getline (info_rekan.nama,
sizeof(info_rekan.nama));
 cout <<"Tanggal Lahir : "; cin >>info_rekan.tgl_lahir.tanggal;
 cout <<"Bulan   Lahir : "; cin >>info_rekan.tgl_lahir.bulan;
 cout <<"Tahun   Lahir : "; cin >>info_rekan.tgl_lahir.tahun;
 clrscr();
 cout <<"Nama lengkap  : " <<info_rekan.nama <<endl;
 cout <<"Tanggal Lahir : " <<info_rekan.tgl_lahir.tanggal <<"-"
```

```
            <<info_rekan.tgl_lahir.bulan <<"-"

            <<info_rekan.tgl_lahir.tahun;
    }
```

ARRAY dengan STRUKTUR

Penggunaan struktur sering dikaitkan dengan array. Array struktur adalah array yang dipakai untuk menyimpan data.

Contoh :

```
    #include <iostream.h>
    #include <conio.h>
    void main()
    {
     struct data_tanggal
      { int tanggal;
            int bulan;
            int tahun;
      };

     struct data_rekan
     {
            char nama[21];
            struct data_tanggal tgl_lahir;
     };

     struct data_rekan info[100];
     char tombol;
     int i, jumlah = 0;
     clrscr();
     cout <<"DATA REKAN-REKAN : " <<endl;
     do
     {
            cout <<"Nama  : "; cin >>info[jumlah].nama;
            cout <<"Tanggal Lahir : "; cin >>info[jumlah].tgl_lahir.tanggal;
            cout <<"Bulan   Lahir : "; cin >>info[jumlah].tgl_lahir.bulan;
            cout <<"Tahun   Lahir : "; cin >>info[jumlah].tgl_lahir.tahun;

            jumlah ++;
            cout <<"Mau memasukkan data lagi [y/t] ? ";
            tombol = getch(); cout <<endl;
           } while (tombol== 'y');
           clrscr();
           cout <<"Data Rekan : NAMA - TANGGAL LAHIR" <<endl;
           for(i=0; i<jumlah; i++)
           cout <<info[i].nama <<" - " <<info[i].tgl_lahir.tanggal <<"-"
                                       <<info[i].tgl_lahir.bulan <<"-"
                                       <<info[i].tgl_lahir.tahun <<endl;
    }
```

STRUKTUR dan FUNGSI

Program menentukan bulan dengan menggunakan struktur dan fungsi

Contoh:

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

void cetak_tanggal(int tg, int bl, int th);
void main()
{
 struct data_tanggal
 { int tanggal;
        int bulan;
        int tahun;
 } lahir;
 cout <<"Tanggal Lahir : "; cin >>lahir.tanggal;
 cout <<"Bulan   Lahir : "; cin >>lahir.bulan;
 cout <<"Tahun   Lahir : "; cin >>lahir.tahun;
 cetak_tanggal(lahir.tanggal, lahir.bulan, lahir.tahun);
 }

void cetak_tanggal(int tg, int bl, int th)
{
        static char *nama_bulan[]=
        { "Kode bulan salah...!", "Januari", "Februari", "Maret",
          "April", "Mei", "Juni", "Juli", "Agustus", "September",
          "Oktober", "November", "Desember" };
        clrscr();
        if (bl<1 || bl>12)
        cout <<"Kode bulan salah ...!";
        else
        cout <<tg <<"-" <<nama_bulan[bl] <<"-" <<th;
}
```

Hasil :

```
Tanggal Lahir : 04
Bulan   Lahir : 04
Tahun   Lahir : 1991
4-April-1991
```

LATIHAN

1. Buat program untuk menentukan zodiak kelahiran dengan menggunakan Array dengan Struktur.

   Bentuk keluaran :

   Tanggal Lahir Anda [tgl-bln-tahun] : 24-12-1970
   Zodiak Anda adalah : CAPRICORN

2. Buat program untuk menyimpan data mahasiswa dengan menggunakan struktur.
   Bentuk tampilan masukan :

   NPM : 1233456
   NAMA : Muh. REZA
   TGL LAHIR : 09-19-1983
   ALAMAT : Pamulang II
   TELEPON : 021876889
   Mau memasukkan data lagi [y/t] ?

Bentuk tampilan keluaran :

1233456 Muh. REZA 09-09-1983 Pamulang II/3 021876889
XXXXXX XXXXXXXXXX XXXXXXXX

Daftar Pustaka

1. Algoritma dan Pemrograman dengan Pascal dan C edisi Revisi, Rinaldi Munir, Bandung: Informatika, 2011.
2. Introduction to Algorithm, Thomas H .Cormen,et al., McGraw-Hill Company, Masachusetts, London, 2001.
3. Algoritma (Algoritma dan Struktur Data 1) dengan C, C++ dan Java,Moh Sjukani, Mitra Wacana Media, 2011
4. Pemrograman C++, Abdul Kadir, Andi Offset, 2003