MODUL PRAKTIKUM
BAHASA PEMROGRAMAN
(CIO220)

Disusun Oleh :
Yunita Fauzia Achmad, S.Kom., M.Kom

Program Studi Teknik Informatika
Fakultas Ilmu Komputer
Universitas Esa Unggul
2018

MODUL 1
Konsep Bahasa Pemrogaman C++ dan Instalasi Dev C++

Tujuan Pembelajaran

1. Praktikan dapat melakukan instalasi dan setting Dev C++
2. Praktikan dapat menggunakan text editor pada Dev C++
3. Praktikan dapat menjalankan (eksekusi) program C++ sederhana

Pengantar C++

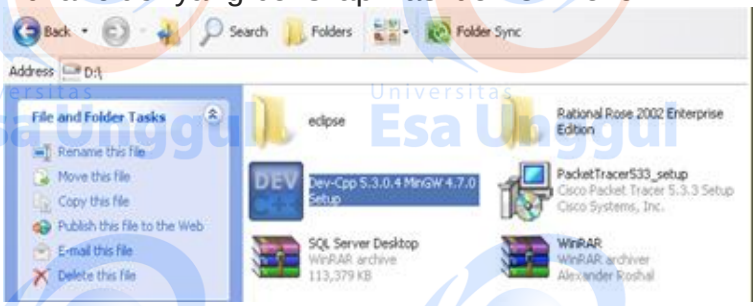
C++ diciptakan oleh Bjarne Stroustrup di laboratorium Bell pada awal tahun 1980-an, sebagai pengembangan dari bahasa C dan Simula. Saat ini, C++ merupakan salah satu bahasa yang paling populer untuk pengembangan software berbasis OOP. Kompiler untuk C++ telah banyak beredar di pasaran. Software developer yang paling diminati adalah Borland Inc. dan Microsoft Corp. Produk dari Borland untuk kompiler C++ adalah Turbo C++, Borland C++, Borland C++ Builder. Sedangkan dari Microsoft adalah Ms. Visual C++. Walaupun banyak kompiler yang tersedia, namun pada intinya bahasa pemrograman yang dipakai adalah C++. Sebelum mulai melakukan kode program, sebaiknya diingat bahwa C++ bersifat "*case sensitive*", yang artinya huruf besar dan huruf kecil dibedakan.

Perkembangan C++

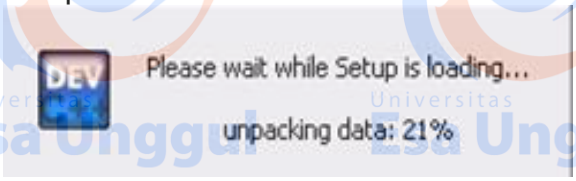
Bahasa C merupakan pengembangan dari bahasa B yang ditulis oleh Ken Thompson pada tahun 1970. Bahasa C untuk pertama kali ditulis oleh Brian W. Kernighan dan Denies M. Ricthie pada tahun 1972. Bahasa C, pada awalnya dioperasikan diatas sistem operasi UNIX. Bahasa C adalah merupakan bahasa pemrograman tingkat menengah yaitu diantara bahasa tinggat rendah dan tingkat tinggi yang biasa disebut dengan Bahasa Tingkat Tinggi dengan Perintah Assambly. Bahasa C mempunyai banyak kemampuan yang sering digunakan diantaranya kemampuan untuk membuat perangkat lunak, misalnya dBASE, Word Star dan lain-lain. Pada tahun 1980 seorang ahli yang bernama Bjarne Stroustrup mengembangkan beberapa hal dari bahasa C yang dinamakan "C with Classes" yang berganti nama pada tahun 1983 menjadi C++. Penambahan yang terdapat pada C++ ini adalah Object Oriented Programming (OOP), yang mempunyai tujuan utamanya adalah membantu membuat dan mengelola program yang besar dan kompleks.

Cara menginstal Dev C++

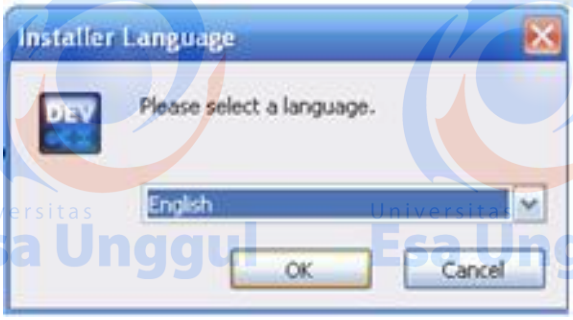
1. Buka folder yang berisi aplikasi dev C++.exe



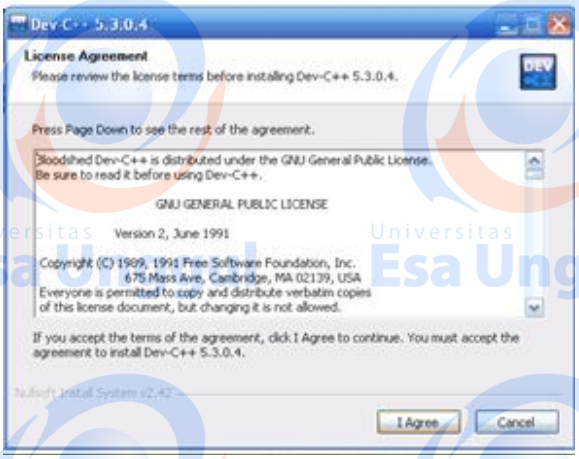
2. Double klik pada file .exe nya dan akan tampil layar seperti dibawah dan tunggu sampai 100%



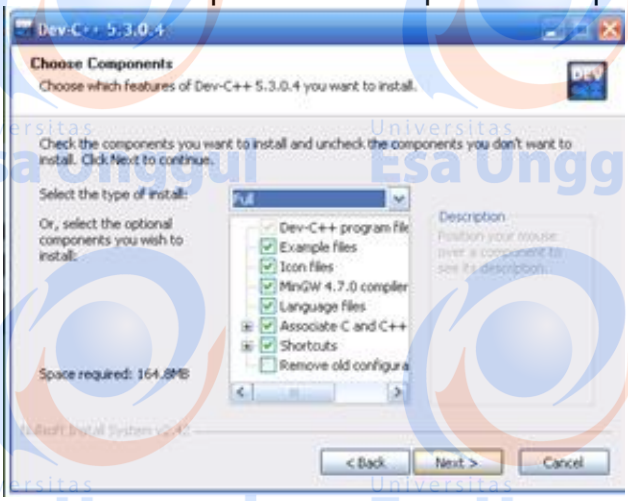
3. Akan tampil installer language pilih english lalu tekan OK



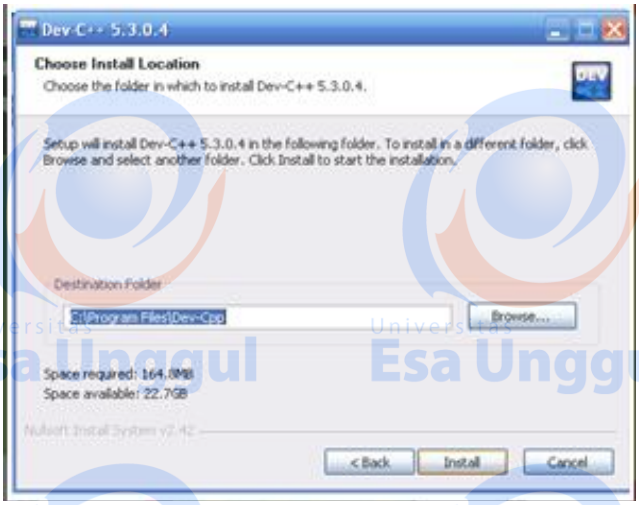
4. Kemudian tampil license agreement dan pilih yes untuk menyetujuinya



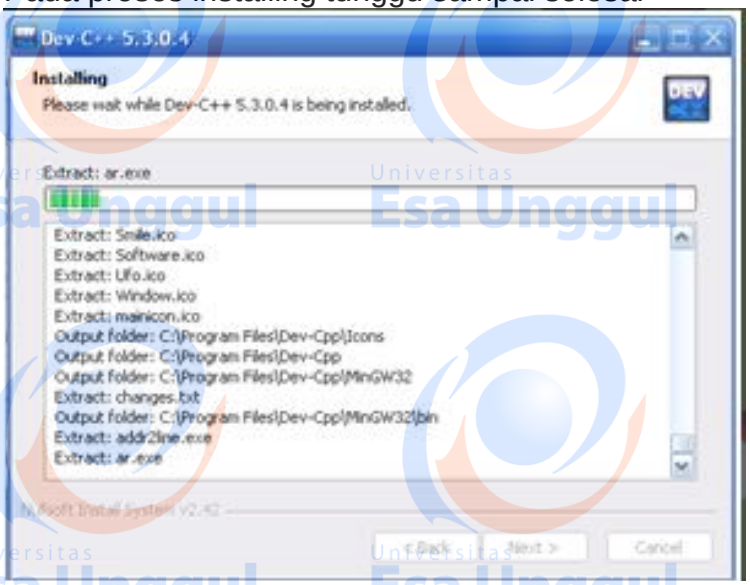
5. Kemudian tampil choose component dan pilih next



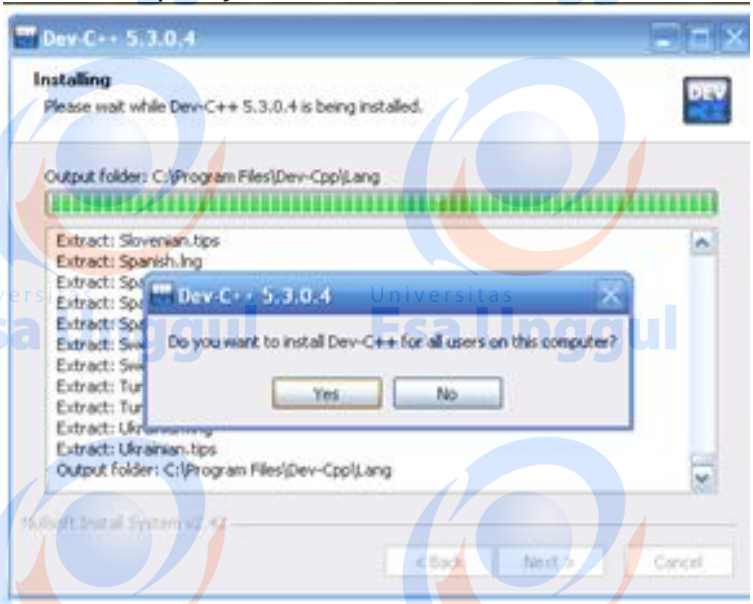
6. Selanjutnya tampil choose install location, kemudian pilih install



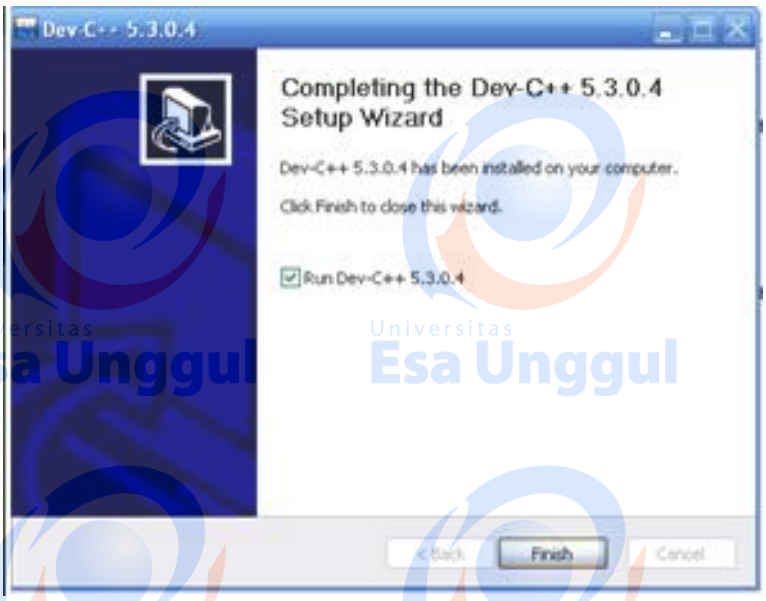
7. Pada proses installing tunggu sampai selesai



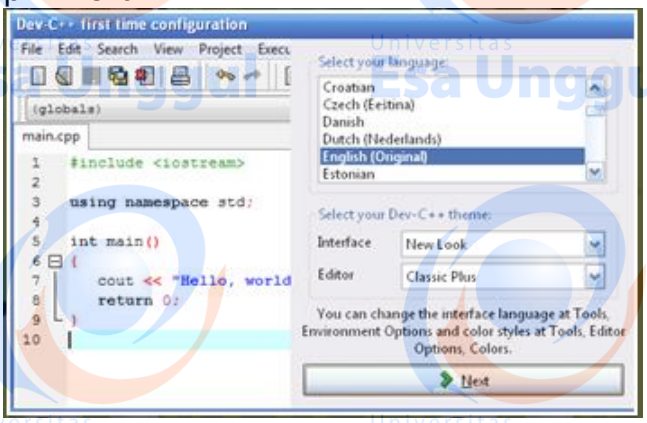
8. Kemudian pilih yes



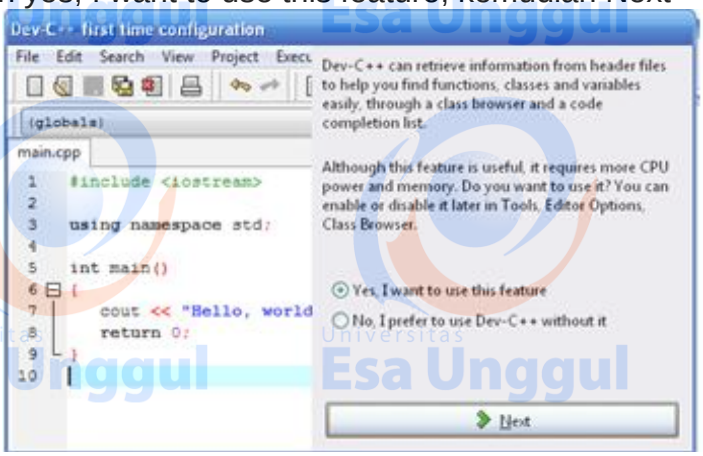
9. Lalu pilih finisih



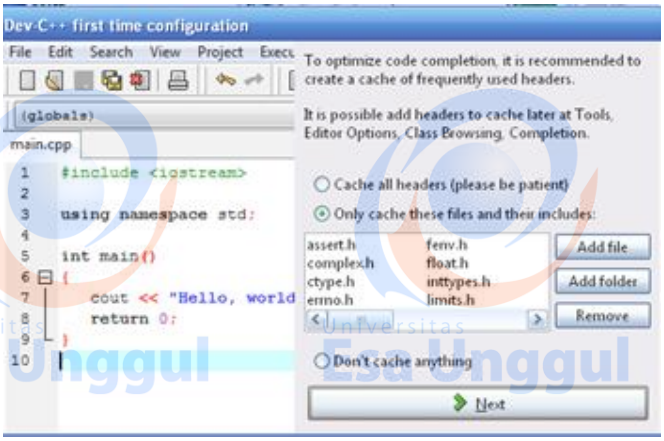
10. Kemudian tampil configuration atau konfigurasi pertama pilih english (original) lalu pilih next



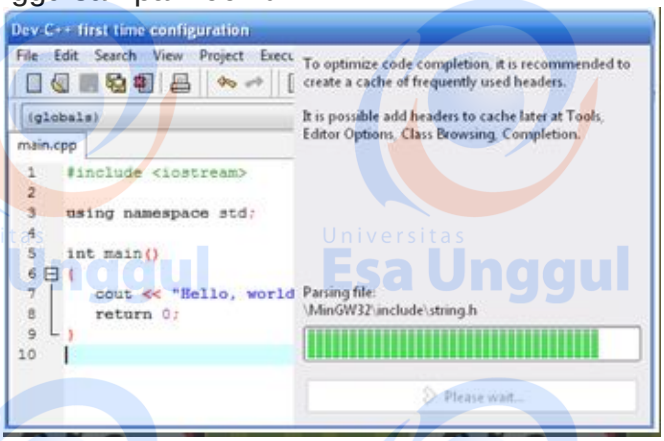
11. Pilih yes, I want to use this feature, kemudian Next



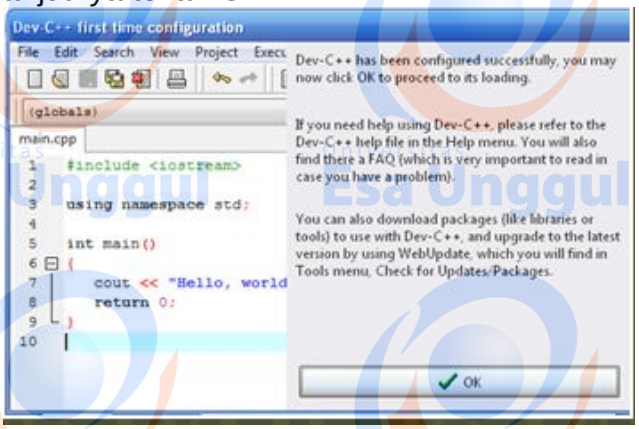
12. Pilih only canche, kemudian tekan next



13. Tunggu sampai 100 %

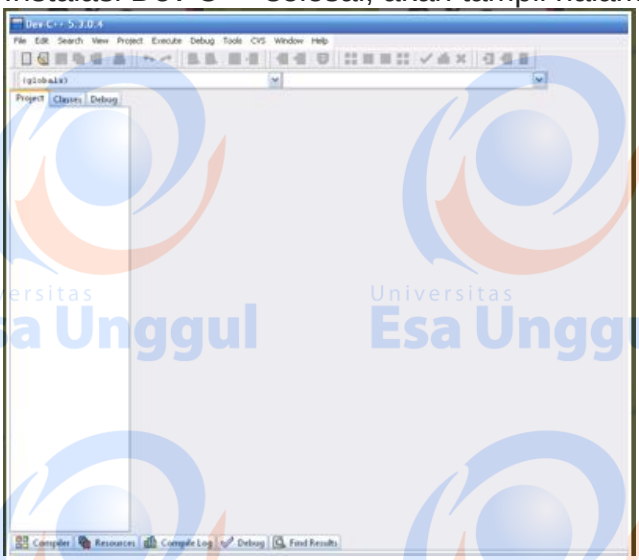


14. Selanjutnya tekan OK



15. Kemudian tampil tip of the day, lalu pilih close

16. Instalasi Dev C++ selesai, akan tampil halaman editor



Struktur Dasar 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>
Void main ()
{
      Pernyataan;
}
```

kode program sederhana C++ :

```
//program pertamaku
#include <iostream.h>
int main ()
{
      cout<<"Selamat Belajar C++";
      return 0;
}
```

Output dari kode program diatas:

program di atas, disimpan dengan**ektensi .cpp** dengan nama misalkan:**latih1.cpp**. Cara untuk menyimpan dan mengkompile program berbeda-beda, tergantung kompiler yang dipakai. Ketika di-run, maka di layar akan muncul sebuah tulisan **"Selamat Belajar C++"**. Contoh di atas, adalah sebuah contoh program sederhana menggunakan C++. Namun, penggalan program tersebut telah menyertakan sintak-sintak dasar bahasa C++. Sintak dasar tersebut, akan kita bahas satu per satu.

Dua buah tanda slash (//) pada program diatas merupakan sebuah baris komentar dan tidak akan berpengaruh pada hasil. Biasanya, baris komentar dipakai oleh programmer untuk memberikan penjelasan tentang program.

Terdapat dua tanda yang digunakan untuk komentar yaitu:

- // baris komentar
- /* blok komentar */

Bagian – bagian pada struktur pemrograman C++, terdiri dari :

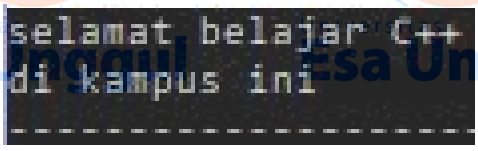
#include <iostream.h> pernyataan yang diawali dengan tanda (#) merupakan pernyataan untuk menyertakan preprocessor. Pernyataan ini bukan untuk dieksekusi. #include <iostream.h> berarti memerintahkan kompiler untuk menyertakan file header iostream.h. Dalam file header ini, terdapat beberapa fungsi standar yang dipakai dalam proses input dan output. Seperti misalnya perintah cout yang dipakai dalam program utama.
int main () Baris ini menandai dimulainya kompiler akan mengeksekusi program. Atau dengan kata lain, pernyataan main sebagai penanda program utama. Adalah suatu keharusan, dimana sebuah program yang ditulis dalam bahasa C++ memiliki sebuah main.
main diikuti oleh sebuah tanda kurung () karena main merupakan sebuah fungsi. Dalam bahasa C++ sebuah fungsi harus diikuti dengan tanda (), yang nantinya dapat berisi argumen. Dan sintak formalnya, sebuah fungsi dimulai dengan tanda {}, seperti dalam contoh program.
cout<< "Selamat Belajar C++"; perintah ini merupakan hal yang akan dieksekusi oleh compiler dan merupakan perintah yang akan dikerjakan. cout termasuk dalam file iostream. cout merupakan perintah untuk menampilkan ke layer.
Perlu diingat, bahwa setiap pernyataan dalam C++ harus diakhiri dengan tanda semicolon (;) untuk memisahkan antara pernyataan satu dengan pernyataan lainnya.
return 0; pernyataan return akan menyebabkan fungsi main() menghentikan program dan mengembalikan nilai kepada main. Dalam hal ini, yang dikembalikan adalah nilai 0. Mengenai pengembalian nilai, akan dijelaskan nanti mengenai Fungsi dalam C++.

Latihan

Buatlah program sederhana dengan hasil output seperti dibawah ini :

```
//program pertamaku
#include <iostream.h>
int main ()
{
      cout<<"Selamat Belajar C++";
      cout<<"di kampus ini";
      return 0; }
```

Tampilan Layar :



MODUL 2
Tipe Data dan Perintah Keluar / Masukan

Tujuan Pembelajaran:

a. Praktikan dapat membuat variabel dengan benar.
b. Praktikan mampu menggunakan berbagai tipe data dalam berbagai kepentingan.
c. Praktikan mampu menggunakan berbagai tipe data dan mengimplementasikannya dalam pemrograman.

Pengenalan Tipe Data

Dalam bahasa pemrograman C++ terdapat 7 tipe data dasar, yaitu diantaranya:

Tabel 1 tipe data

Nama Tipe Data	Ukuran memori	Jangkauan Nilai
Char	1 byte	–128 s.d 127
Short Integer	2 byte	–32768 s.d 32767
Integer	2 byte	–2147483648 s.d 2147483647
Long Integer	4 byte	–223372036854775808 s.d 223372036854775807
Float	4 byte	3.4 x 10-38 s.d 3.4 x 10 + 38
Double	8 byte	1.7 x 10-308 s.d 1.7 x 10+38
Long Double	10 byte	3.4 x 10-4932 s.d 1.1 x 10+4932

Pada bahasa pemrograman C++ terdapat tipe data tambahan, diantaranya:

Tabel 2 Tipe DataTambahan

Nama Tipe Data	Ukuran 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

Variabel

Variabel adalah suatu tempat menampung data atau konstanta dimemori yang mempunyai nilai atau data yang dapat berubah – ubah selama proses program.

Pada pemberian nama variabel, terdapat beberapa ketentuan – ketentuan, diantaranya:

a. Tidak boleh menggunakan spasi (cth : gaji bersih) dan jika ingin lebih dari dua kata menggunakan tanda garis (_) sebagai penghubung (cth : gaji_bersih)
b. Tidak boleh diawali oleh angka dan menggunakan operator aritmatika

Variabel dibagi menjadi dua bagian, yaitu :

a. Variabel Numerik
   Variabel numerik ini dibagi menjadi menjadi 3 (tiga) macam :
   1. Bilangan Bulat
      Adalah bilangan yang tidak mengandung titik desimal
      Contoh : 1, 2, 3
   2. Bilangan Desimal Berpresisi Tunggal atau Floating Point.
      Floating point, mempunyai bentuk penulisan, yaitu :
      - Bentuk desimal (cth : 6.56)
      - Bentuk eksponensial (cth : 4.22e8542)
   3. Bilangan Desimal Berpresisi Ganda atau Double Precision.

Double precision pada prinsipnya sama dengan floating point, tetapi double precision memiliki daya tampung data lebih besar.

b. Variabel Text

Variabel text ini dibagi menjadi menjadi 2(dua) macam :

1. Character (karakter tunggal)
   Character hanya terdiri dari sebuah karakter saja yang diapit oleh tanda kutip tunggal (‘ ‘). Character dapat berupa abjad (huruf besar atau kecil), angka, ataupun simbol.
   Contoh: ‘A’, ‘a’, ‘8’, ‘&’ dan lain-lain
2. String (rangkaian karakter)
   String merupakan rangkaian dari beberapa karakter yang diapit oleh tanda kutip ganda (“ “)
   Contoh : “Jakarta”, “Esa Unggul”, dan lain-lain

Deklarasi Variabel

Deklarasi Variabel adalah proses memperkenalkan variabel kepada java dan pendeklarasian tersebut bersifat mutlak karena jika tidak diperkenalkan terlebih dulu maka java tidak menerima variabel tersebut.

Deklarasi Variabel ini meliputi tipe variabel, seperti: integer atau character dan nama variabel itu sendiri. Setiap kali pendeklarasian variabel harus diakhiri oleh tanda titik koma (;).

Bentuk penulisannya :

```
Tipe_data nama_variabel;
```

Contoh Deklarasi :

```
String nama_mahasiswa;
Char grade ;
Float rata – rata;
Int nilai 1, nilai2;
```

Menempatkan Nilai kedalam Variabel

Setelah pendeklarasian Variabel dilaksanakan, selanjutnya variabel tadi bisa anda masukan nilai kedalam variabel. Berikut cara yang mudah untuk menempatkan nilai kedalam variabel.

Bentuk penulisannya :

```
Nama_variabel = nilai;
```

Contoh Penempatan Nilai kedalam Variabel :

```
nama_mahasiswa = "Rahayu ningsih";
grade = 'A';
rata_rata = “94
nilai1 = “95”; nilai2 = “93”;
```

Perintah masukan

Dalam bahasa pemrograman c++ terdapat perintah standar input, diantaranya adalah:

1. **Perintah scanf()**
   Fungsi perintah **scanf()** digunakan untuk memasukan berbagai jenis data
   Bentuk umum perintah **scanf():**

```
Scanf(“penentuan format”, &nama-variabel);
```

Keterangan:
Simbol & merupakan pointer yang digunakan untuk menunjukkan ke alamat variabel memori yang dituju.
Berikut ini merupakan penentu format pada perintah **scanf()**, diantaranya:

Tabel 3 Penentu Format **scanf()**

Nama Tipe Data	Ukuran memori
Integer	%d
Floating Point Bentuk decimal Bentuk berpangkat	 %e atau %f % 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

Kode program C++ dengan masukan perintah scanf():

```
# include <stdio.h>
# include <conio.h>
void main()
{
   int a, b, c;

   printf("Masukan Nilai A = "); scanf("%d",&a);
   printf("Masukan Nilai B = "); scanf("%d",&b);

   c = a + b;

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

Output yang dihasilkan:

```
Masukan Nilai A = 20
Masukan Nilai B = 20
Hasil Penjumlahan = 40
```

2. **Perintah gets()**
   Fungsi perintah **gets()** digunakan untuk memasukkan data string.

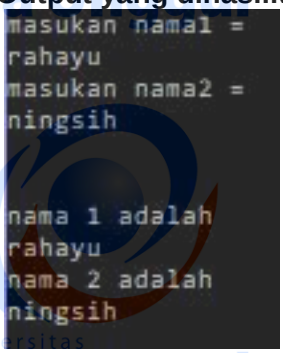
   Bentuk Umum dari fungsi ini adalah :

```
gets(nama-variabel-array);
```

Kode program C++dengan masukan perintah gets():

```
# include <stdio.h>
# include <conio.h>
void main()
{
      char nm1[20];
      char nm2[20];

      puts("Masukan nama ke - 1 = ");
      gets(nm1);
      puts("Masukan nama ke - 2 = ");
      gets(nm2);

      printf("\n\n");

      puts("Senang Berkenalan Dengan Anda ..");
      puts(nm1);
      puts("Senang Berkenalan Dangan Anda ..");
      puts(nm2);
 }
```

Output yang dihasilkan:



3. **Perintah cin()**

perintah cin() merupakan sebuah objeck didalam C++ digunakan untuk memasukkan suatu data. Untuk menggunakan perintah cin() ini, harus menyertakan file header **iostream.h .**

kode program C++dengan masukan perintah cin():

```
  # include <stdio.h>
  # include <conio.h>
  # include <iostream.h>
  using namespace std;

  void main()
  {
   float a, b, c;

      cout<<"Masukan Nilai A : ";cin>>a;
      cout<<"Masukan Nilai B : ";cin>>b;

       c = a + b;

      cout<<"Nilai C : "<<c<<endl;
  }
```

Output dari kode program diatas adalah :

```
Masukan Nilai A : 20
Masukan Nilai B : 30
Nilai C : 50
```

4. **Perintah getch()**

Perintah **getch()** (*get character and echo*) dipakai untuk membaca sebuah karakter dengan sifat karakter yang dimasukkan tidak perlu diakhiri dengan menekan tombol ENTER, dan karakter yang dimasukan tidak akan ditampilkan di layar. File header yang harus disertakan adalah **conio.h**

Contoh program C++dengan masukan perintah getch():

```
# include <stdio.h>
# include <conio.h>
main()
{
  char kar;

  printf("Masukan Sebuah Karakter A/B/C = ");
  kar = getch();

  printf("\nTadi Anda Memasukan karakter %c", kar);
  getch();
}
```

Output dari kode program diatas adalah :

```
masukan karakter bebas =
 tampilkan karakter U
```

5. **Perintah getche()**

Fungsi **getche()**dipakai untuk membaca sebuah karakter dengan sifat karakter yang dimasukkan tidak perlu diakhiri dengan menekan tombol ENTER, dan karakter yang dimasukan ditampilkan di layar. File header yang harus disertakan adalah **conio.h**

kodeprogram C++dengan masukan perintah getche():

```
# include <stdio.h>
# include <conio.h>
main()
{
  char kar;

  printf("Masukan Sebuah Karakter A/B/C = ");
  kar = getche();

  printf("\nTadi Anda Memasukan karakter %c", kar);
  getch();
}
```

Output yang dihasilkan

```
masukan karakter bebas = D
 tampilkan karakter D
```

Perintah Keluaran

Perintah standar output yang pada bahasa pemrograman C++, diantaranya adalah :

1. Perintah printf()

Perintah**printf()** merupakan fungsi keluaran yang paling umum digunakan untuk menampilkan informasi kelayar.

Bentuk penulisan perintahprintf() :

```
Printf("string control", argumen1, argumen2,…);
```

Keterangan:
String-Kontrol dapat berupa keterangan yang akan ditampilkan pada layar beserta penentu format. Penentu format dipakai untuk memberi tahu kompiler mengenai jenis data yang dipakai dan akan ditampilkan. Argumen ini dapat berupa variabel, konstanta dan ungkapan.

Contoh program C++dengan keluaran perintah printf():

```
# include <stdio.h>
# include <conio.h>
void main()
{
   int a
   char b;

    a = 5;
    b = 'E';

   printf("%c merupakan abjad ke %d", b, a);
}
```

Output dari yang dihasilkan

```
E merupakan abjad ke 5
```

2. Perintah puts()

Perintah puts() (*put string*) sebenarnya sama dengan printf(), yaitu digunakan untuk mencetak string ke layar.

Perbedaan antara puts() dan printf() :

Tabel perbedaan puts() dan printf()

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

Kode program C++dengan keluaran perintah puts():

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

void main()
{
  char a[4] = "UEU";
  puts("Saya Kuliah di. ");
  puts(a);
}
```

Output dari kode program diatas adalah :

3. Perintah putchar()

Perintah **putchar()** digunakan untuk menampilkan sebuah karakter ke layar. Penampilan karakter tidak diakhiri dengan pindah baris.

Contoh program C++dengan keluaran perintah putchar():

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

void main()
{
  putchar('U');
  putchar('E');
  putchar('U');
}
```

Output dari kode program diatas adalah :

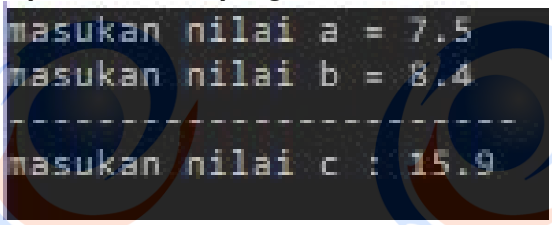
4. Perintah cout()

Perintah **cout()** merupakan sebuah objeck didalam C++ digunakan untuk menampilkan suatu data kelayar. Untuk menggunakan perintah cout() ini, harus menyertakan file header **iostream.h** dan perintah **using namespace std** di letakan diatas **main().**

Kode program C++dengan keluaran perintah cout():

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
Using namespace std;

void main()
{
   float a, b, c;

   a=7.5; b=8.4; c=0;

   cout<<"Masukan Nilai A : "<<a;
   cout<<"Masukan Nilai B : "<<b;

    c = a + b;

   cout<<"Masukan Nilai C : "<<c;
   getch()
}
```

Output dari kode program diatas adalah :



Latihan :

Buatlah kode program dengan tampilan outputnya seperti dibawah ini !

Layar Masukan

```
PROGRAM HITUNG NILAI RATA – RATA TUGAS
Nama Mahasiswa : … <diinput>
Nilai Tugas 1 : … <diinput>
Nilai Tugas 2 : … <diinput
Nilai Tugas 3 : … <diinput>
Nilai tugas 4 : … <diinput>
Nilai tugas 5 : … <diinput>
```

Layar Keluaran

```
Mahasiswa yang bernama … <tampil data>
Memperoleh nilai rata – rata tugas … <hasil proses>
```

MODUL 3
OPERATOR

Tujuan Pembelajaran:

Praktikan mampu menggunakan berbagai Operator pada bahasa pemrograman C++dan mengimplementasikannya dalam program sederhana

Pengenalan Operator

Operator adalah simbol atau karakter yang biasa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi, seprti penjumlahan, pengurangan dan lain-lain.

Sifat – sifat Operator

Dalam bahasa pemrograman C++ terdapat beberapa sifat yang dimiliki oleh operator, diantaranya:

1. Unary
   Sifat Unary pada operator adalah hanya melibatkan sebuah operand pada suatu operasi aritmatika
   Contoh : -5
2. Binary
   Sifat Binary pada operator adalah melibatkan dua buah operand pada suatu operasi aritmatika
   Contoh : 5 + 4
3. Ternary
   Sifat Tenary pada operator adalah melibatkan tiga atau lebih operand pada suatu operasi aritmatika
   Contoh : (15 / 3) + 5 + 8

1. Operator Aritmatika

Operator aritmatikaOperator untuk operasi aritmatika yang tergolong sebagai operator binary. Berikut ini table Operator Aritmatika, diantaranya:

Tabel 4 Operator Aritmatika

Operasi	Operator	Ekspresi
Perkalian	*	4 * 5
Pembagian	/	7 / 4
Sisa Pembagian	%	5 % 2
Penjumlahan	+	7 + 3
Pengurangan	–	6 – 4

Untuk sifat operator yaitu unary, terdapat beberapa operator unary, diantaranya:

Tabel 4 Operator Unary

Operasi	Operator	Ekspresi
+	Plus	+4
-	Minus	-6

Deklarasi Operator Aritmatika
Berikut Bentuk penulisan ekspresi aritmatika dikaitkan dengan pernyataan pemberi nilai.
Bentuk Umum :

```
Variable = ekspresi aritmatika
```

kode program penggunaan operator aritmatika sifat binary

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
Using namespace std;

void main()
{
   int a, b, c, d, e, f, g;

   a = 10;
   b = 5;

   cout<<"Masukan Nilai A : "<<a;
   cout<<"Masukan Nilai B : "<<b;

    c = a + b;
    d = b – a;
    e = a * b;
    f = a % b;
    g = a / b;
   cout<<"Masukan Nilai C : "<<c;
   cout<<"Masukan Nilai d : "<<d;
   cout<<"Masukan Nilai e : "<<e;
   cout<<"Masukan Nilai f : "<<f;
   cout<<"Masukan Nilai g : "<<g;

   getch();
}
```

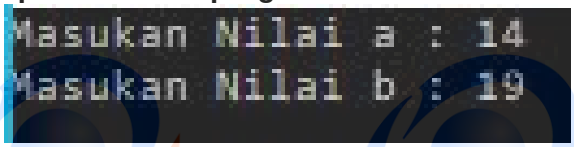
Output dari kode program diatas adalah :

```
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2
10 % 5 = 0
```

Kode program penggunaan operator aritmatika sifat ternary

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
Using namespace std;

void main()
{
   int a, b;

   a = 10 + 3 * 6 /4;
   b = (10 + 3) * 6 / 4;

   cout<<"Masukan Nilai a : "<<a;
   cout<<"Masukan Nilai b : "<<b;

   getch();
 }
```

Output dari kode program diatas adalah :



2. Operator Pemberi Nilai

Sebelumnya kita telah mengenal operator pemberi nilai (*assignment operator*) yaitu tanda "=".

Sebagai contoh penggunaan operator pemberi nilai : A = A + 1

bahasa pemrograman C++ dapat menyederhanakan menjadi: A += 1, karenanotasi " += " ini dikenal dengan operator pemberi nilai aritmatika

berikut ini operator pemberi nilai

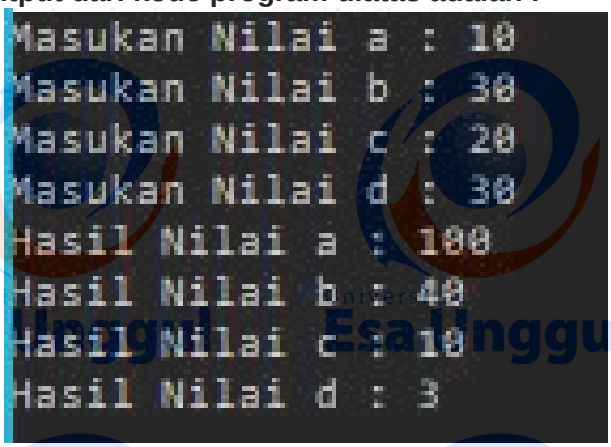
Tabel 5 Operator Pemberi Nilai

Operasi	Operator Pemberi Nilai	Contoh Ekspresi	Penggunaan Operator Pemberi Nilai
Perkalian	*=	A = A * 5	A *= 5
Pembagian	/=	A = A / 5	A /= 5
Sisa Pembagian	%=	A = A % 2	A %= 2
Penjumlahan	+=	A = A + 1	A+= 1
Pengurangan	–=	A = A – 4	A –= 4

Kode program penggunaan operator pemberi nilai

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
Using namespace std;

main()
{
   int a, b, c, d;

   cout<<"Masukan Nilai a : "; cin>>a;
   cout<<"Masukan Nilai b : "; cin>>b;
   cout<<"Masukan Nilai c : "; cin>>c;
   cout<<"Masukan Nilai d : "; cin>>d;

   a *= 10;
   b += 10;
   c -= 10;
   d /= 10;

   cout<<"Hasil Nilai a : "<<a<<endl;
   cout<<"Hasil Nilai b : "<<b<<endl;
   cout<<"Hasil Nilai c : "<<c<<endl;
   cout<<"Hasil Nilai d : "<<d<<endl;

   getch();
}
```

Output dari kode program diatas adalah :



3. Operator Penambah dan Pengurang

Penambahan adalah suatu penambahan nilai yang terjadi pada sebuah variabel. Adapun operator yang digunakan untuk melakukan increment adalah operator ++. Operator ini akan menambahkan nilai dari suatu variabel dengan nilai 1. Sedangkan, **Pengurangan** adalah kebalikan dari proses increment, yaitu menurunkan / mengurangi nilai dari suatu variabel

Tabel 6 Operator Penambahan dan Pengurangan

Operator	Keterangan
++	Penambahan
– –	Pengurangan

A = A + 1 atau A = A - 1; disederhanakan menjadi :

A += 1 atau A -= 1; masih dapat disederhanakan menjadi A ++ atau A-- Notasi " ++ " atau
"-- " dapat diletakan didepan atau di belakang variabel.

A ++ atau ++A / A-- atau –

Perbedaan bentuk penulisan notasi penambahan dan pengurangan ini mempunyai arti yang berbeda, diantaranya :

- **Jika diletakan didepan variabel**, maka proses penambahan atau pengurangan akan dilakukan sesaat sebelum atau langsung pada saat menjumpai ekspresi ini, sehingga nilai variabel tadi akan langsung berubah begitu ekspresi ini ditemukan.
- **Jika diletakan dibelakang variabel**, maka proses penambahan atau pengurangan akan dilakukan setelah ekspresi ini dijumpai atau nilai variabel akan tetap pada saat ekspresi ini ditemukan.

Kode program penggunaan operator penambahan dan pengurangan jika diletakan di depan variable

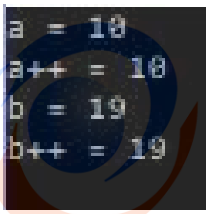
```
#include<stdio.h>
#include<conio.h>
#include<iostream>
Using namespace std;

void main()
{
   int a, b;

   a = 10 + ++a;
   b = 5 - --b;

   cout<<"Nilai a = " <<a<<endl;
   cout<<"Nilai ++a = "<<a<<endl;
   cout<<"Nilai b = "<<b<<endl;
   cout<<"Nilai --b = "<<b<<endl;

   getch();
}
```

Output dari kode program diatas adalah :

kodeprogram penggunaan operator penambahan dan pengurangan jika diletakan di belakang variabel

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
Using namespace std;

void main()
{
   int a, b;

   a = 10;
   b = 5;

   cout<<"Nilai a = " <<a<<endl;
   cout<<"Nilai a++ = "<<a<<endl;
   cout<<"Nilai b = "<<b<<endl;
   cout<<"Nilai b-- = "<<b<<endl;

   getch();
}
```

Output dari kode program diatas adalah :



4. Operator Pembanding

Operator pembanding digunakan untuk membandingkan dua buah nilai. Hasil dari operator ini menghasilkan nilai numeric 1 yang berarti true dan 0 yang berarti false.

Berikut ini adalah operator pembanding dalam bahasa pemrograman C++

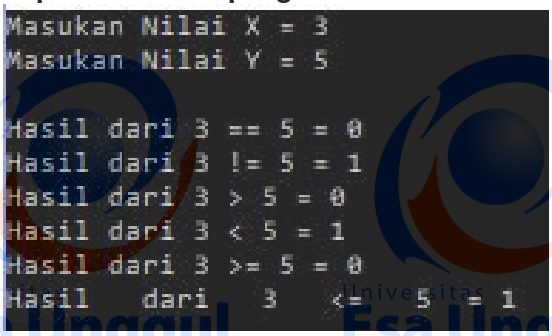
Tabel 7 Operator Pembanding

Operator	Keterangan
==	Sama Dengan (bukan pemberi nilai)
!=	Tidak Sama dengan
>	Lebih Dari
<	Kurang Dari
>=	Lebih Dari sama dengan
<=	Kurang Dari sama dengan

Kode program sederhana dari penggunaan operator pembanding

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
Using namespace std;
main()
{
float a, b, c, d, e, f, x, y;

cout<<"Masukan Nilai X = ";  cin>>x;
cout<<"Masukan Nilai Y = ";  cin>>y;

a = x == y;
 b = x != y;
c = x > y;
d = x < y;
e = x >= y;
f = x <= y;

cout<<endl;
cout<<"Hasil   dari   "<<x<<"   ==   "<<y<<"   =   "<<a<<endl;
cout<<"Hasil   dari   "<<x<<"   !=   "<<y<<"   =   "<<b<<endl;
cout<<"Hasil   dari   "<<x<<"   >   "<<y<<"   =   "<<c<<endl;
cout<<"Hasil   dari   "<<x<<"   <   "<<y<<"   =   "<<d<<endl;
cout<<"Hasil   dari   "<<x<<"   >=   "<<y<<"   =   "<<e<<endl;
cout<<"Hasil dari "<<x<<" <= "<<y<<" = "<<f<<endl;
getch();
}
```

Output dari kode program diatas adalah :



5. Operator Logika

Operator logika digunakan untuk menghubungan dua buah operasi relasi menjadi sebuah ungkapan kondisi. Hasil dari operator logika merupakan nilai numeric 1 (TRUE) dan 0 (FALSE)

Berikut table dari operator logika, diantaranya :

Tabel 8 Operator Logika

Operator	Keterangan
&&	Operator Logika AND
||	Operator Logika OR
!	Operator Logika NOT

a. Operator Logika AND

Operator logika AND digunakan untuk menghubungkan dua atau lebih ekspresi relasi, akan dianggap BENAR, bila semua ekspresi relasi yang dihubungkan bernilai BENAR.

contoh :
((5==5) && (3>6)) mengembalikan nilai false, karena (true && false)

b. Operator Logika OR

Operator logika OR digunakan untuk menghubungkan dua atau lebih ekspresi relasi, akan dianggap BENAR, bila salah satu ekspresi relasi yang dihubungkan bernilai BENAR dan bila semua ekspresi relasi yang dihubungkan bernilai SALAH, maka akan bernilai SALAH.

Contoh:
A+4 < 10 || B>A+5 || C–3 > 4 BENAR = 1

c. Operator Logika NOT

Operator logika NOT akan memberikan nilai kebalikkan dari ekspresi yang disebutkan. Jika nilai yang disebutkan bernilai BENAR maka akan menghasilkan nilai SALAH, begitu pula sebaliknya.

Contoh :
!(5==5) akan mengembalikan nilai false, karena !(true).

Kode program sederhana dari penggunaan operator logika

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
Using namespace std;
main()
{
float a, b, c, d, e, f, g, h, i;
cout<<"Masukan Nilai A =";cin>>a;
cout<<"Masukan Nilai B = ";cin>>b;
cout<<"Masukan Nilai C = "; cin>>c;

d = a + 4 < 10;
e = b > a + 5;
f = c - 3 >= 4;
g = d && e && f;
h = d || e || f;
I = !(d)

cout<<endl<<endl;
cout<<"Hasil dari d = a + 4 < 10 adalah " <<d<<endl;
cout<<"Hasil dari e = b > a + 5  adalah " <<e<<endl;
cout<<"Hasil dari f = c - 3 >= 4 adalah " <<f<<endl;

cout<<endl<<endl;

cout<<"Hasil dari g = d && e && f adalah " <<g<<endl;
cout<<"Hasil dari h = d || e || f adalah " <<h<<endl;
cout<<"Hasil dari I = !(d) adalah " <<I;

cout<<endl;
getch();
}
```

Output dari kode program diatas adalah :

```
Masukan Nilai A = 35
Masukan Nilai B = 44
Masukan Nilai C = 67


Hasil dari d = a + 4 < 10 adalah 0
Hasil dari e = b > a + 5  adalah 1
Hasil dari f = c - 3 >= 4 adalah 1

Hasil dari g = d && e && f adalah 0
Hasil dari h = d || e || f adalah 1
Hasil dari I = !(d) adalah 1
```

6. Operator Bitwise

Operator bitwise merupakan operator yang digunakan untuk memanipulasi data dalam bentuk bit

Berikut ini operasi – operasi yang terdapat pada operator bitwise, diantaranya:

Table 9 Operator Bitwise

Operator	Keterangan
~	Bitwise NOT
<<	Bitwise Shift Left
>>	Bitwise Shift Right
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR

a. Operator Bitwise ~ (NOT)

Operator Bitwise ~ (Not) digunakan membalik nilai bit dari suatu operand.

Contoh :

00001000 = 8

Maka kebalikannya :

11110111 = 247 = -9

Kode program C++ untuk operator Bitwise ~(NOT)

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
Using namespace std;

void main()
{
   int a, x;

   cout<<"masukan nilai x =  ";cin>>x;

   a = ~X;

   cout<<'\n';
   cout<<"hasil dari bitwise not"<<x<<"="<<a<<endl;

   getch();
 }
```

Output yang dihasilkan dari kode program diatas, adalah :

b. Operator Bitwise << (Shift Left)

Operator Bitwise Shift Left digunakan untuk menggeser sejumlah bit kekiri.

Contoh :

0000000011001001 = 201

Dilakukan pergeseran sebanyak 1 bit ke kiri:

0000000110010010 = 402

Kode program C++ untuk operator Bitwise << (Shift Left)

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
Using namespace std;

void main()
{
   int x;

   cout<<"masukan nilai x =  ";cin>>x;

   x = X<<1; /* 1 menunjukan pergeseran sebanyak 1 bit*/

   cout<<'\n';
   cout<<"hasil dari bitwise shift left"<<x<<endl;

   getch();
 }
```

Output yang dihasilkan dari kode program diatas, adalah :

```
Masukan Nilai X = 201
Hasil dari Geser 1 Bit Kekiri = 402
```

c. Operator Bitwise >> (Shift Right)

Operator Bitwise Shift Right digunakan untuk menggeser sejumlah bit kanan.
Contoh :

0000000011001001 = 201

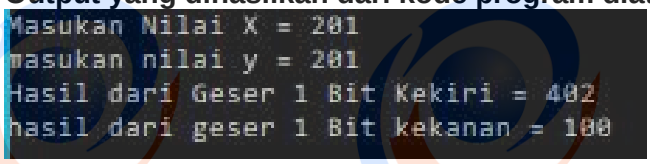
Dilakukan pergeseran sebanyak 1 bit ke kekanan:

0000001100100100 = 100

Kode program C++ untuk operator Bitwise >>(Shift Right)

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
Using namespace std;

void main()
{
   int x;

   cout<<"masukan nilai x =  ";cin>>x;

   x = X<<1; /* 1 menunjukkan pergeseran sebanyak 1 bit kekiri*/
   y = y>>1; /* 1 menunjukkan pergeseran sebanyak 1 bit ke kanan*/

   cout<<'\n';
   cout<<"hasil dari bitwise shift left"<<x<<endl;
   cout<<"hasil dari bitwise shift right"<<y<<endl;
   getch();
 }
```

Output yang dihasilkan dari kode program diatas, adalah :



d. Operator Bitwise & (AND)

Operator Bitwise & (And) digunakan untuk membandingkan bit dari dua operand. Akan bernilai benar (1) jika semua operand yang digabungkan bernilai benar (1).

Contoh :

Bit dari 201 = 11001001
Bit dari 100 = 01100100
———————————— AND
64 = 01000000

Kode program C++ untuk operator Bitwise &(AND)

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
Using namespace std;

void main()
{
   int a,x,y;

   cout<<"masukan nilai x =  ";cin>>x;
   cout<<"masukan nilai y =  ";cin>>y;

   a = x & y;

   cout<<'\n';

   cout<<"hail dari"<<x<<"&"<<y<<"="<<a<<endl;

   getch();}
```

Output yang dihasilkan dari kode program diatas, adalah :

e. Operator Bitwise ^ (XOR)

Operator Bitwise ^ (XOr) digunakan untuk membandingkan bit dari dua operand. Akan bernilai benar (1) jika dari dua bit yang dibadingkan hanya sebuah bernilai benar (1).

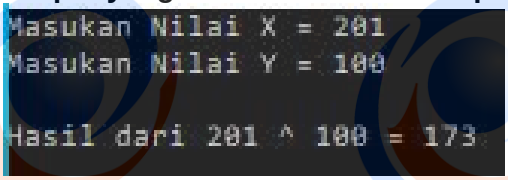
Contoh :

Bit dari 201 = 11001001
Bit dari 100 = 01100100
———————————— XOR
173 = 10101101

Kode program C++ untuk operator bitwise ^ (XOR)

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
Using namespace std;

void main()
{
   int a,x,y;

   cout<<"masukan nilai x =  ";cin>>x;
   cout<<”masukan nilai y =  “;cin>>y;

   a = x ^ y;

   cout<<’\n’;

   cout<<”hasil dari”<<x<<”|”<<y<<”=”<<a<<endl;

   getch();
}
```

Output yang dihasilkan dari kode program diatas adalah:



f. Operator Bitwise | (OR)

Operator Bitwise | (Or) digunakan untuk membandingkan bit dari dua operand. Akan bernilai benar jika ada salah satu operand yang digabungkan ada yang bernilai benar (1).

Contoh :

Bit dari 201 = 11001001
Bit dari 100 = 01100100
——————————— OR
237 = 11101101

Kode program C++ untuk operator bitwise | (OR)

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
Using namespace std;

void main()
{
   int a,x,y;

   cout<<"masukan nilai x =  ";cin>>x;
   cout<<”masukan nilai y =  “;cin>>y;

   a = x | y;

   cout<<’\n’;

   cout<<”hasil dari”<<x<<”|”<<y<<”=”<<a<<endl;

   getch();
}
```

Output yang dihasilkan dari kode program diatas adalah:

Latihan

1. Buatlah program kalkulator serhana yang mengimplementasikan beberapa operator yang sudah dijelaskan.
2. Buatlah sebuah program untuk yang menampilakan hasil operasi dasar logika AND OR dan NOT

MODUL 4
Kondisi

Tujuan Pembelajaran:

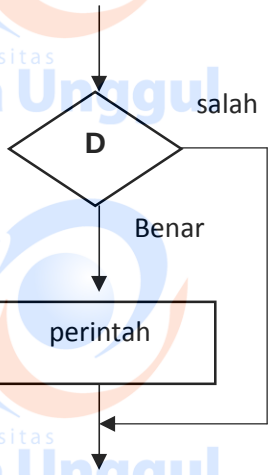
a. Praktikan mengenal beberapa perintah untuk seleksi kondisi
b. Praktikan mampu menggunakan berbagai conditional statement dalam berbagai kebutuhan.

Teori Singkat

Pernyataan percabangan digunakan untuk memecahkan persoalan untuk mengambil suatu keputusan diantara sekian banyak pernyataan yang ada. Terdapat beberapa perintah yang terdapat pada bahasa pemrograman C++, yaitu diantaranya:

1. Pernyataan IF

Pernyataan IF diartikan sebagai pernyataan yang "jika kondisi bernilai benar, maka perintah akan dikerjakan dan jika tidak memenuhi syarat maka akan diabaikan". Dari pengertian tersebut dapat digambarkan diagram alirnya, sebagai berikut :



Gambar Diagram alir IF

Bentuk umum pernyataan IF

```
If(kondisi)
Pernyataan;
……
```

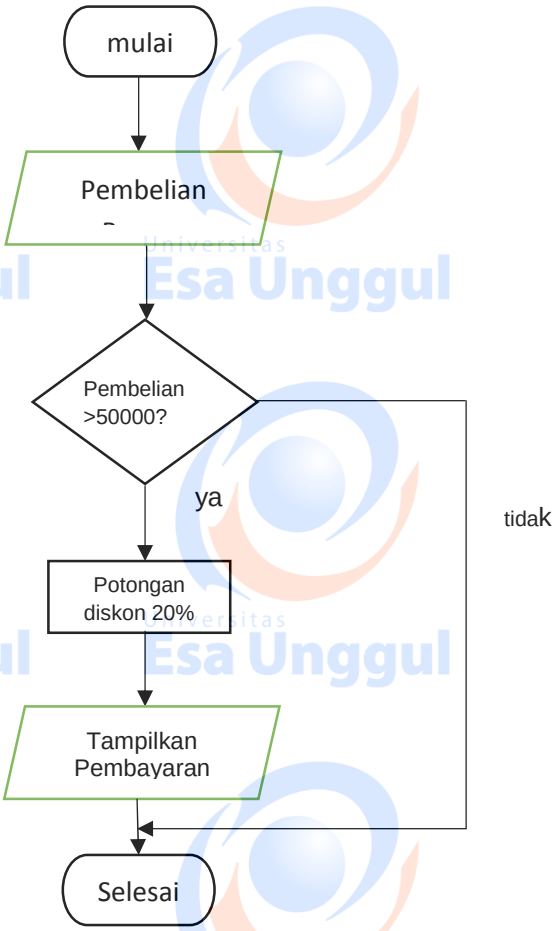
Bentuk umum pernyataan IF jika lebih

```
If(kondisi)
{
Pernyataan;
}
```

Contoh soal pernyataan IF

Menentukan besarnya potongan dari pembelian barang yang diberikan oleh seorang pembeli, dengan beberapa kriteria, yaitu :

1. Tidak ada potongan jika total pembelian kurang dari Rp 50.000,-
2. Jika total pembelian lebih dari atau sama dengan Rp 50.000,- maka pembeli mendapatkan potongan sebesar 20%



Gambar Diagram alir IF untuk contoh diatas

Kode Program C++ untuk contoh kasus diatas

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
Using namespace std;

main()
{
   double tot_beli, potongan=0, jum_bayar=0;

   cout<<"Total Pembelian Rp. ";cin>>tot_beli;

   if (tot_beli >= 50000)
   potongan = 0.2 * tot_beli;

   cout<<"Besarnya Potongan Rp. "<<potongan<<endl;

   jum_bayar = tot_beli - potongan;

   cout<<"Jumlah yang harus dibayarkan Rp. "<<jum_bayar;

   getch();
}
```

Output yang dihasilkan :

Output pernyataan berhasil

```
Total Pembelian Rp. 100000
besarnya potongan Rp. 20000
jumlah yang harus dibayarkan Rp. 80000
```

Output pernyataan tidak berhasil

```
Total Pembelian Rp. 45000
besarnya potongan Rp. 0
jumlah yang harus dibayarkan Rp. 45000
```

2. Pernyataan if-else

Pernyataan if-else digunakan untuk mengeksekusi pernyataan A jika suatu kondisi bernilai benar, dan sebaliknya akan mengeksekusi pernyataan B jika suatu kondisi bernilai salah.

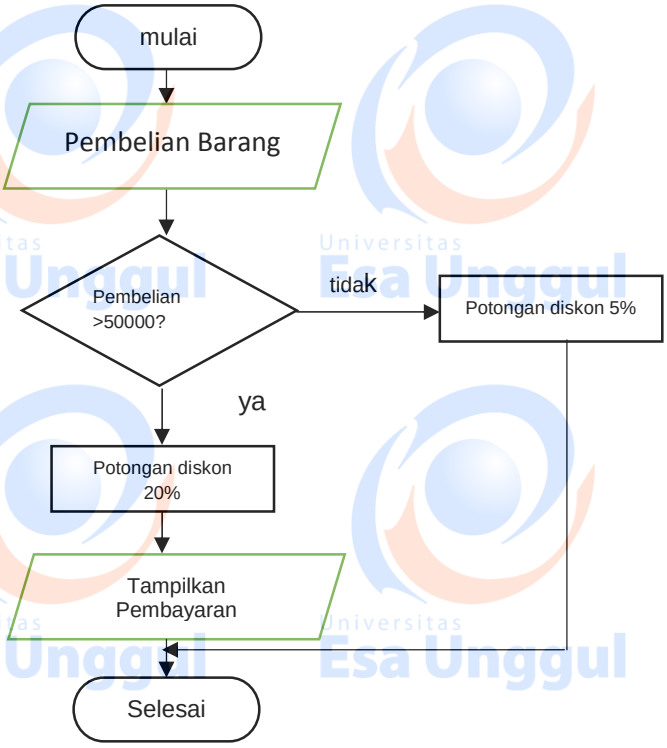
Bentuk umum pernyataan if - else

```
if(kondisi)
      Pernyataan;
else
      pernyataan;
```

Contoh soal pernyataan if-else

Menentukan besarnya potongan dari pembelian barang yang diberikan seorang pembeli, dengan kriteria :

- jika total pembelian kurang dari Rp. 50.000,- potongan yang diterima sebesar 5% dari total pembelian.
- Jika total pembelian lebih dari atau sama dengan Rp. 50.000,- potongan yang diterima sebesar 20% dari total pembelian.



Gambar diagram alir pernyataan if-else pada soal diatas

Kode program C++ untuk permasalahan diatas

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
Using namespace std;

main()
{
  double tot_beli, potongan=0, jum_bayar=0;

      cout<<"Total Pembelian Rp. ";cin>>tot_beli;

  if (tot_beli >= 50000)
      potongan = 0.2 * tot_beli;
  else
      potongan = 0.05 * tot_beli;

      cout<<"Besarnya Potongan Rp. "<<potongan<<endl;

      jum_bayar = tot_beli - potongan;

      cout<<"Jumlah yang harus dibayarkan Rp. "<<jum_bayar;

      getch();
}
```

Output yang dihasilkan dari permasalahan di atas adalah sebagai berikut :
Jika total pembelian > 50000, output yang dihasilkan seperti :

```
Total Pembelian Rp.100000
besarnya potongan Rp. 20000
jumlah yang harus dibayarkan Rp.80000
```

Jika total pembelian < 5000, output yang dihasilkan seperti :

```
Total Pembelian Rp.40000
besarnya potongan Rp. 2000
jumlah yang harus dibayarkan Rp.38000
```

3. Pernyataan nested if

Nested if merupakan pernyataan if yang berada di dalam pernyataan if yang lainnya.

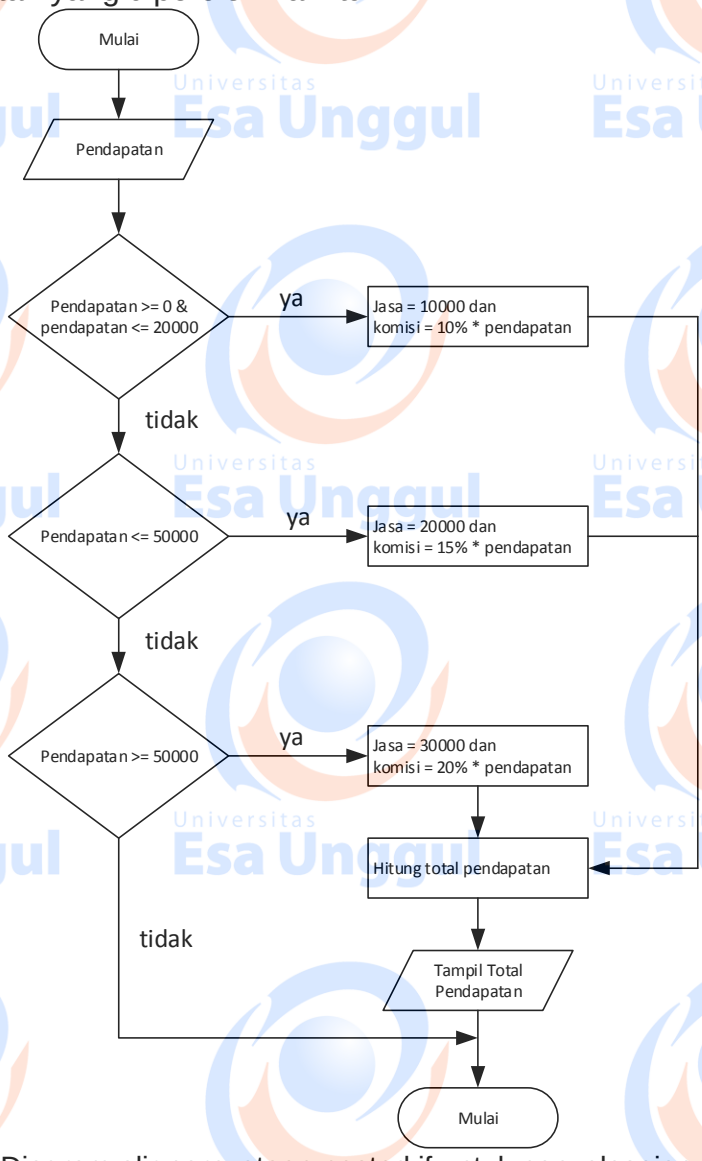
Berikut bentuk penulisan pernyataan nested if :

```
if(syarat)
if(syarat)
  … perintah;
else
  … perintah;
else
  if(syarat)
     … perintah;
  else
     … perintah;
```

Contoh kasus menggunakan pernyataan Nested if

Suatu perusahaan memberikan komisi kepada para selesman dengan ketentuan sebagai berikut:

- Bila salesman dapat menjual barang hingga Rp. 20.000 ,- , akan diberikan uang jasa sebesar Rp. 10.000 ditambah dengan uang komisi Rp. 10% dari pendapatan yang diperoleh hari itu.
- Bila salesman dapat menjual barang diatas Rp. 20.000 ,- , akan diberikan uang jasa sebesar Rp. 20.000 ditambah dengan uang komisi Rp. 15% dari pendapatan yang diperoleh hari itu.
- Bila salesman dapat menjual barang diatas Rp. 50.000 ,- , akan diberikan uang jasa sebesar Rp. 30.000 ditambah dengan uang komisi Rp. 20% dari pendapatan yang diperoleh hari itu.

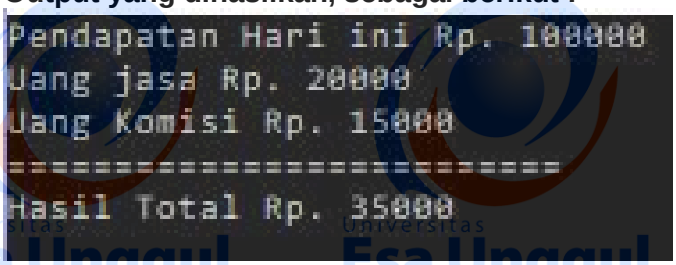


Gambar Diagram alir pernyataan nested if untuk penyelesaian soal diatas

Kode program C++ pada kasus di atas

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

main()
{
      float pendapatan, jasa = 0, komisi = 0, total = 0;

      cout<<"Pendapatan Hari ini Rp. ";cin>>pendapatan;

      if(pendapatan >= 0 && pendapatan <= 200000)
      {
            jasa = 10000;
            komisi = 0.1 * pendapatan;
      }
      else;
      {
            if(pendapatan <= 500000)
            {
                  jasa = 20000;
                  komisi = 0.15 * pendapatan;
            }
            else
            {
                  jasa = 30000;
                  komisi = 0.2 * pendapatan;
            }
      }
      /* menghitung total */

      total = komisi + jasa;

      cout<<"Uang jasa Rp. "<<jasa<<endl;
      cout<<"Uang Komisi Rp. "<<komisi<<endl;
      cout<<"=========================="<<endl;
      cout<<"total pendapatan Rp. "<<total<<endl;

      getch();
}
```

Output yang dihasilkan, sebagai berikut :



4. Pernyataan switch - case

Bentuk dari switch - case merupakan pernyataan yang dirancangan khusus untuk menangani pengambilan keputusan yang melibatkan sejumlah atau banyak alternatif. Pernyataan switch - case ini memiliki kegunaan sama seperti if – else bertingkat, tetapi penggunaannya untuk memeriksa data yang bertipe karakter atau integer.

Bentuk penulisan pernyataan switch – case

```
Switch (ekspresi integer atau character)
{
      Case 1:
            Perintah;
            Perintah;
            Break;
      Case 2:
            Perintah;
            Perintah;
            Break;
      ……
      ……
      Default:
            Perintah;
            Perintah;
            Break;
}
```

Keterangan:

Setiap cabang akan dijalankan, jika syarat nilai konstanta tersebut dipenuhi dan **default**akan dijalankan, jika semua cabang diatasnya tidak terpenuhi. Pernyataan **break**menunjukkan bahwa perintah siap keluar dari ***switch*****,** jika pernyataan ini tidak ada, maka program akan diteruskan kecabang – cabang yang lainnya.

Contoh soal pernyataan switch – case:

Terdapat beberapa kode barang , diantaranya :

a. Alat olah raga
b. Alat elektronik
c. Alat masak

Ketika ingin memilih salah satu dari kode barang tersebut, maka akan tampil kode barang dan keteranagn barang tersebut, sesuai dengan kode yang dipilih, tetapi jika tidak todak memilih salah satu dari ketiga kode barang tersebut, maka akan tampil “anda salah memasukan kode”

Kode program C++ dari kasus diatas:

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
using namespace std;

main()
{
    Char kode;

    Cout<<"masukan kode barang = ";cin>>kode;

    Switch(kode)
    {
        Case 'A':
            Cout<<"alat elektronik";
            Break;
        Case 'B':
            Cout<<"alat olah raga";
            Break;
        Case 'C':
            Cout<<"alat masak";
            Break;
        Default:
            Cout<<"anda salah memasukan kode";
            Break;
        }
    getch();
}
```

Output yang dihasilkan :

5. Operator ?:

Operator ?: disebut dengan Conditional Operator atau Operator Kondisi yang digunakan untuk menyeleksi nilai untuk mendapatkan hasil dari kondisi yang diseleksi. Operator ?: ini tergolong kedalam operator ternary.

Berikut bentuk penulisan dari operator ?:

```
Ekspresi Logika-OR ?Ekspresi : Ekspresi Kondisi
```

Kode program dari operator ?:

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
using namespace std;

main()
{
    Int x,y,z;

    X = 5;
    Y = 6;

    Z = (x < y> ? x : y;

    Cout<<"\nNilai bilangan x = "<<x;
    Cout<<"\nNilai bilangan y = "<<y<<endl;
    Cout<<"\nNilai yang lebih kecil adalah = "<<z;
    Getche();
}
```

Output yang hasilkan :

```
Nilai bilangan x = 5
Nilai bilangan y = 6

Nilai yang lebih kecil adalah = 5
```

Latihan

Buatlah program untuk menghitung nilai akhir seorang mahasiswa. dengan ketentuan sebagai berikut :

a. Nama Siswa, Nilai Tugas, Nilai UTS dan Nilai UAS diinput.

b. Proses yang dilakukan untuk mendapatkan nilai :

- Nilai Tugas = Nilai Tugas dikalikan dengan 30%.
- Nilai UTS = Nilai UTS dikalikan dengan 35%
- Nilai UAS = Nilai UAS dikalikan dengan 35%

c. Nilai Akhir adalah Nilai Tugas + Nilai UTS + Nilai UAS

d. Ketentuan untuk mendapatkan grade nilai :

- Nilai Akhir >= 80 mendapat Grade A
- Nilai Akhir >= 70 mendapat Grade B
- Nilai Akhir >= 60 mendapat Grade C
- Nilai Akhir >= 50 mendapat Grade D

- Nilai Akhir < 40 mendapat Grade E

MODUL 5
Perulangan

Tujuan Pembelajaran:

a. Praktikan mengenal beberapa perintah untuk melakukan perulangan
b. Praktikan mampu menggunakan berbagai bentuk perulangan dalamberbagai kebutuhan.

Perulangan

Perulangan atau yang biasa disebut dengan “looping” adalah proses yang dilakukan secara berulang – ulang sampai batas yang ditentukan. Biasanya, bila dalam perulangan tidak disertakan batasnya maka kode program akan error karena proses itu akan terjadi berulang terus tak terhingga sementara variabel dalam komputer sangatlah terbatas.

Dalam perulangan, umumnya terdiri dari 3 komponen, yaitu :

1. **Nilai awal / inisialisasi**, yaitu menentukan nilai awal dalam perulangan
2. **Syarat perulangan**, jika nilai memenuhi kondisi tertentu, perulangan akan dilanjutkan, jika tidak, maka perulangan akan dihentikan
3. **Perubahan nilai**, selama perulangan berlangsung nilai akan diubah secara kontiyu.

Berikut ini diagram perulangan secara umum:

Terdapat beberapa jenis perulangan pada bahasa pemrograman C++, diantaranya :

1. Perintah: for

Bentuk for digunakan untuk melakukan perulangan, dimanabanyaknya perulangan telah diketahui sebelumnya. Pernyataandengan for akan memiliki counter yang akan bergerak (naikatau turun) secara otomatis.

Bentuk umum perulangan for:

```
For (inisialisasi; syarat perulangan; pengubah nilai pencacah)
```

Bila pernyataan didalam for lebih dari satu, maka pernyataan – pernyataan tersebut diletakkan dalam tanda kurung

```
For (inisialisasi; syarat perulangan; pengubah nilai pencacah)
{
     Pernyataan;
     Pernyataan;
}
```

Contoh :
Buatlah program untuk mencetak bilangan 1 sampai 10 secara menaik, menurun dan menampilkan bilangan ganjil

Kode program untuk mencetak bilangan secara menaik

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main()
{
     int a;
     for(a = 1; a <= 10; ++a)

     cout<<a;

     getch();
}
```

Output yang dihasilkan

Kode program untuk mencetak bilangan secara menurun

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main()
{
     int a;
     for(a = 1; a >= 10; --a)

     cout<<a;

     getch();
}
```

Output yang dihasilkan

Kode program untuk mencetak bilangan ganjil

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main()
{
      int a;
      for(a = 1; a <= 10; a+=2)

      cout<<a;

      getch();
}
```

Output yang dihasilkan

```
13579
```

2. Perintah: while

Perulangan while berbeda dengan perulangan for, karena perulangan while digunakan dengan jumlah perulangan yang belum diketahui. Proses perulangan akan terus berlanjut selama kondisinya bernilai benar (TRUE) dan akan berhenti jika kondisinya bernilai salah (FALSE).

Bentuk umum perulangan while :

```
Inisialisasi;
while(syarat)

      Pernyataan;
```

Bentuk umum perulangan while, jika memiliki beberapa pernyataaan:

```
Inisialisasi;
while(syarat)
{
      Pernyataan;
      Pernyataan;
}
```

Contoh :

Buatlah program untuk mencetak bilangan 1 sampai 10 secara menaikdan menampilkan bilangan genap

Kode program untuk perulangan while dengan menampilkan nilai 1 - 10

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

main()
{
      int bilangan = 1;

      while(bilangan <= 10)
      {
            cout<<bilangan<<" ";
            ++bilangan;
      }
      getch();
}
```

Output yang dihasilkan

Kode program untuk perulangan while dengan menampilkan nilai genap

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

main()
{
      int bilangan = 1;

      while(bilangan <= 10)
      {
            cout<<bilangan<<" ";
            bilangan+=2;
      }
      getch();
}
```

Output yang dihasilkan

3. **Perintah: do-while**

Perulangan do-while adalah perulangan yang akan dilakukan minimal 1x terlebih dahulu, kemudiaan baru dilakukan pengecekan terhadap kondisi, jika kondisi benar, maka perulangan masih akan tetap dilakukan. Perulangan dengan do – while() akan dilakukan sampai kondisi false.

Bentuk umum pada perulangan do – while

```
Inisialisasi;
do {
      Pernyataan 1;
      Pernyataan N;
      Pernyataan nilai;
}
While(syarat perulangan);
```

.

Berikut Diagram dari bentuk umum do – while

Gambar diagram alir pada bentuk umum do-while

Contoh program perulangan do-while

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main()
{
      int bil=2;
do
      {
      cout<<bil<<" ";
      bil+=2;
      }
      while(bil<=10);
      getch();
}
```

Output yang dihasilkan:

```
2 4 6 8 10
```

Latihan

1. Buatlah sebuah program untuk menampilkan output berikut:
   30 9 28 27 26.....16 1 2 3 4 5 6 7 8 9.....15
2. Buatlah sebuah program untuk menampilkan sederatan angka genap dan ganjil beserta jumlahnya
   Contoh :
   2 3 5 7 9 = 25
   3 4 6 8 10 = 30

MODUL 6
Fungsi

Tujuan Pembelajaran :

1. Praktikan mengerti dan memahami fungsi – fungsi dari bahasa pemrograman C++
2. Praktikan mampu menggunakan fungsi dari C++ pada program sederhana

Teori Singkat

Dengan mempergunakan fungsi, maka struktur program akan terlihat lebih rapi. Fungsi merupakan sebuah blok instruksi yang dieksekusi dan dipanggil dari bagian lain bagian program.

Bentuk penulisan umum fungsi:

```
tipe nama(argumen1, argumen2,…)pernyataan;
```

Keterangan :
Tipe : berisi tipe data yang akan dikembalikan oleh fungsi
Nama :merupakan pengenal untuk memanggil fungsi
Argumen :(dapat dideklarasikan sesuai dengan kebutuhan). Tiap-tiap argument terdiri dari tipe-tipe data yang diikuti oleh pengenalnya. Sama seperti mendeklarasikan variable baru (contoh, int x).

Pernyataan : merupakan bagian tubuh fungsi. Dapat berupa pernyataan tunggal ataupun pernyataan majemuk.

Terdapat dua jenis variabel yang digunakan dalam fungsi :

1. Variabel lokal
   Variabel yang hanya dapat diakses di dalam tubuh fungsi itu saja. Berikut deklarasi variabel lokal

Kode program penggunaan fungsi variabel lokal

```
//contoh fungsi
#include <iostream.h>
#include <conio.h>

int penjumlahan(int a, int b)
{
      int r;
      r = a + b;
      return (r);
}
int main()
{
      int z;
      z = penjumlahan(5,3);
      cout<<"hasil penjumalah = "<<z;

      return 0;
}
```

Output yang dihasilkan

```
hasil penjumalah = 8
```

2. **Variabel global**
   Variabel yang dapat diakses dari mana saja. Dari dalam maupun dari luar tubuh fungsi. Berikut deklarasi variabel global.

Kode program C++ fungsi pada varibel global

```
#include<iostream.h>
#include<conio.h>
int kurang(int a, int b)
{
      int r;
      r=a-b;
      return (r);
}

int main()
{
      int x= 5, y=3, z;
      z=kurang(7,2);

      cout<<"Pertama : " << z<<endl;
      cout<<"Kedua : "<< kurang(7,2)<<endl;
      cout<<"Ketiga  :  "   <<   kurang   (x,y)
<<endl;

      z=4+kurang(x,y);

      cout<<"Keempat : "<<z<<endl;

      return 0;
      }
```

Output yang dihasilkan :

```
Pertama : 5
Kedua : 5
Ketiga : 2
Keempat : 6
```

Keterangan :

Pada program diatas, terdapat sebuah fungsi yang diberi nama**kurang**. fungsi ini diperintahkan untuk mengerjakan pengurangan nilai pada dua variabel dan kemudian mengembalikan hasilnya.

Fungsi tanpa tipe (menggunakan void)

Kadang-kadang terdapat fungsi yang tanpa memerlukan adanya pengembalian nilai. Misalkan, sebuah fungsi yang hanya bertugas mencetak kalimat ke layar monitor dan tanpa memerlukan adanya pertukaran parameter. Dalam kondisi seperti ini, maka dipergunakan kata kunci ***void*.**

Contoh kode program fungsi tanpa tipe

```
#include<iostream.h>
#include<conio.h>
void Ucapan(void)
{
cout<<"Selamat Belajar C++";
}
int main()
{
Ucapan();
return 0;
}
```

Output yang dihasilkan

```
Selamat Belajar C++
```

Keterangan :

Pemanggilan fungsi harus disertai dengan tanda ( ). Seperti kode program diatas fungsi ucapan walaupun dideklarasikan tanpa tipe data dan tanpa argument, tetap dapat dipanggil dalam fungsi main dengan format ucapan().

Terdapat dua cara melakukan panggilan argument, diantaranya :

1. Pemanggilan dengan nilai (*arguments passed by value*)
   Berikut ini fungsi – fungsi yang telah dibuat dengan menggunakan argumen berdasarkan nilai.
   Contoh :
   int x= 5, y=3, z;
   z=kurang(x,y);
   contoh diatas, terjadi pemanggilan terhadap fungsi kurang() dengan x dan y masing – masing bernilai 5 dan 3. Pada z = kurang(x,y) , maka pengisian nilai terhadap variabel x dan y yaitu x = 5 dan y = 3, ketika terjadi pengisian nilai seperti ini, maka nilai x dan y tidak akan mengalami perubahan apapun.

2. Pemanggilan dengan acuan (*arguments passed by reference*)
   Namun, kadangkala diinginkan sebuah pertukaran nilai yang mempengaruhi nilai variabel pemberinya. Untuk melakukannya, diperlukan sebuah fungsi dengan Pemanggilan dengan nilai.

Kode program C++

```
#include<iostream.h>
#include<conio.h>
void kali (int& a, int& b, int& c) {
     a *= 2;
     b *= 2;
     c *= 2;
}
  int main()
{
  int x=1, y=3, z=7;
  kali(x,y,z);
  cout<<  "x=  "<<x<<",  y=  "<<y<<",  z=
  "<<z;
  return 0;
}
```

Output yang dihasilkan

Keterangan :
Hasil didapat berbeda adalah cara pertukaran argumen menggunakan tanpa ampersand (&) yang artinya fungsi melayani pengisian berdasarkan referensi dengan pengisiian nilai seperti ini, maka apabila terjadi perubahan nilai pada variabel a, b, c, maka akan mempengaruhi nilai variabel x, y dan z.

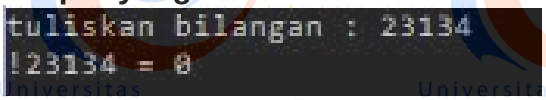
Fungsi Rekursif

Fungsi rekursif adalah suatu fungsi yang memanggil dirinya sendiri, artinya fungsi tersebut dipanggil di dalam tubuh fungsi itu sendiri. Fungsi rekursif sangat berguna bila diimplementasikan untuk pekerjaan pengurutan data, atau menghitung nilai factorial suatu bilangan.

Contoh kode program fungsi rekursif

```
#include<iostream.h>
#include<conio.h>
long factorial (long a)
{
      if (a>1)
      return (a* factorial (a-1));
      else
      return (1);
}
      int main()
{
      long l;
      cout<<"tuliskan bilangan : ";cin>>l;
      cout<<"!"<<l<<" = "<<factorial(l);
      return 0;
}
```

Output yang dihasilkan



Prototype Fungsi

Sampai saat ini, setiap dideklarasikan sebuah fungsi baru diletakkan di atas fungsi main(). Namun terdapat pula alternative lain dalam pendeklarasian fungsi baru, yaitu dideklarasikan di bawah fungsi main() dengan menggunakan prototype fungsi.

Bagi compiler, informasi dalam prototype akan dipakai untuk memeriksa validitas parameter dalam pemanggilan fungsi.

Keuntungan pemakaian prototype yaitu compiler akan melakukan konversi seandainya antara tipe parameter dalam definisi dan parameter saat pemanggilan fungsi tidak sama, atau akan menunjukkan kesalahan kalau jumlah parameter dalam definisi dan saat pemanggilan berbeda.

Bentuk umum penulisan prototype fungsi:

```
tipe nama(argumen1, argumen2,…)pernyataan;
```

Sama seperti pendeklarasian judul fungsi, kecuali :

1. tidak memiliki baris pernyataan (tubuh fungsi) yang ditandai dengan { dan }.
2. diakhiri dengan tanda ;
3. dalam pendeklarasian argumennya, cukup hanya dengan mendeklarasikan tipe datanya saja. Walaupun sangat dianjurkan untuk menyertakan argumen secara lengkap.

Contoh kode program prototype fungsi

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

void bagi (int a, int b);
int main()
{
     cout<<bagi(20,4);
     return 0;
}
     void bagi (int a, int b)
{
     int r;
     r=a/b;
     return (r);
}
```

untuk pendeklarasian prototype fungsi dapat berbentuk seperti berikut:

void bagi (int a, int b);

atau

void bagi (int , int);

Latihan :

1. Buatlah program menghitung luas dan keliling lingkaran dengan menggunakan fungsi. Fungsi yang harus dibuat *luas()* untuk menghitung luas lingkaran dan *keliling()* untuk menghitung luas lingkaran.
2. Buatlah program untuk menghitung besarnya diskon yang diberikan atas besanya sejumlah pembelian, dengan ketentuan sebagai berikut : - Jika belanja dibawah Rp. 1,000,000 , maka tidak mendapat diskon. - Jika belanja dimulai dari Rp. 1,000,000 , sampai dengan Rp. 5.000.000, maka mendapat diskon sebesar 20%. - Jika belanja diatas Rp. 5.000.000, maka mendapat diskon sebesar 35%.
   Fungsi yang harus dibuat potong() untuk menghitung besar potongan yang akan diberikan. Dengan tampilan yang diinginkan sebagai berikut :
   Program Hitung Potongan.
   Besar pembelian barang Rp. ……. <di input >
   Besar diskon yang diberikan Rp. ……< hasil proses >
   Besar harga yang harus dibayarkan Rp. ……< hasil proses >

MODUL 7
String

Tujuan Pembelajaran:

1. Praktikan mengerti dan memahami penggunaan string
2. Praktikan mampu menggunakan beberapa operator dan method yang menyertai penerapan string.

Teori singkat

String merupakan bentuk data yang biasa dipakai dalam bahasa pemrograman untuk keperluan menampung dan memanipulasi data teks, misalnya untuk menampung (menyimpan) suatu kalimat. Pada bahasa C++, string bukanlah merupakan tipe data tersendiri, melainkan hanyalah kumpulan dari nilai – nilai karakter yang berurutan dalam bentuk array berdimensi satu.

Konstanta string

Suatu konstanta string ditulis dengan diawali dan diakhiri tanda petik ganda
(“ “), contoh : “ABCDE”
Nilai string disimpan dalam memori secara berurutan dengan komposisi sebagai berikut :

Memori rendah ⟶ memori tinggi

A	B	C	D	E	0

Setiap karakter akan menempati memori sebesar 1 byte. Byte terakhir otomatis akan berisi karakter NULL (0). Mengetahui bahwa suatu string diakhiri nilai NULL, maka akhir dari nilai suatu string akan dapat dideteksi. Sebagai sebuah array karakter, karakter pertama dari nilai string mempunyai indeks ke-0, karakter kedua mempunyai indeks ke-1 dan seterusnya.

Variabel string

Variabel string adalah variabel yang dipakai untuk menyimpan nilai string
Contoh :

```
char name[10];
```

keterangan :

instruksi diatas merupakan deklarasi variabel string dimana panjang maksimum yang diberikan adalah 10 karakter (termasuk karakter NULL)

Inisialisasi String

Variabel string dapat dinisialisasi seperti halnya array yang lain. tetapi tetap saja elemen terakhirnya adalah karakter NULL
Contoh :

```
char name[ ] = { ‘R’, ‘A’, ‘H’, ‘A’, ‘Y’, ‘U’, ‘\0’}
```

akan menyatakan bahwa name adalah variabel string dengan nilai awal adalah “RAHAYU”. Pada bentuk NULL tidak perlu ditulis secara implisit akan tetapi cukup disisipkan oleh compiler.

Input dan Output Data String

Input Data String

Untuk menginput data string ke dalam suatu variabel dapat dilakukan dengan menggunakan fungsi **gets( )** dan **scanf( )**
Bentuk umum penulisan input data string

```
#include <stdio.h>
gets(nama_array);
```

atau

```
#include <stdio.h>
scanf(“%s”,nama_array);
```

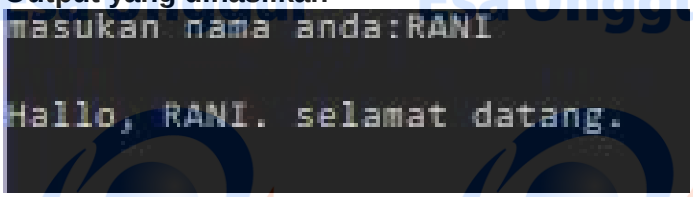
keterangan :

- Nama_array adalah variabel yang bertipe array of char yang digunakan untuk menyimpan string masukan
- Di depan nama_array tidak perlu ada operator & (operator alamat), karena nama_array tanpa kurung [ ] sudah menyatakan alamat yang ditempati oleh elemen pertama dari array
- Kalau memakai scanf( ), data string masukan tidak boleh mengandung spasi.

Contoh masukan string dengan scanf()

```
#include<stdio.h>
main()
{
      char name[10];

      printf("masukan nama anda:");
      scanf("%s",name);

      printf("\nHallo, %s. selamat datang. \n", name);
}
```

Output yang dihasilkan



Ruang yang disediakan pada kode program diatas / dideklarasikan : char name[10], sehingga ada 10 ruang yang kosong, seperti :

Setelah data dimasukan, maka menjadi

R	A	N	I	\0					

Byte yang di kosong tidak digunakan. Dapat diperhatikan bahwa nama array tanpa kurung siku / name menyatakan alamat dari lokasi data string. Dan dengan proses pemasukan data seperti di atas, lkasi memori yang terletak sesudah lokasi yang berisi 'N'akan secara otomatis terisi karakter NULL.
Sedangkan untuk fungsi gets( ) akan membaca seluruh karakter yang diketik sampai tombol ENTER ditekan.

Output String

Untuk menampilkan isi dari variabel string, fungsi yang digunakan adalah **puts()** atau **printf( )**

Bentuk umum penulisan output string :

```
#include <stdio.h>
puts(var_string);
```

atau

```
printf("%s", var_string);
```

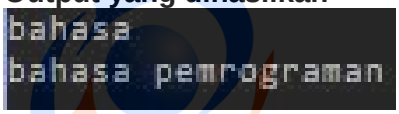
pada var_string adalah sebuah variabel yang berupa sebuah array of char. Sedangkan fungsi put() akan menampilkan isi dari var_string dan secara otomatis menambahkan karakter '\n' di akhir string. Fungsi prinf( ) akan menampilkan isi variabel string tanpa tambahan '\n'.

berikut contoh kode program dari dua fungsi output pada string:

kode program

```
#include<stdio.h>
void bentuk1(void);
void bentuk2(void);

main()
{
      bentuk1();
      bentuk2();
}
void bentuk1(void)
{
      char kompiler_c[] =
      {'b','a','h','a','s','a','\0'};

      puts(kompiler_c);
}
void bentuk2(void)
{
      char kompiler_c[] = "bahasa pemrograman";
      printf("%s\n", kompiler_c);
}
```

Output yang dihasilkan



Fungsi Manipulasi String

1. **Fungsi strcat()**

Fungsi strcat() digunakan untuk menggabungkan nilai string. File header yang disertakan **string.h** dan **ctype.h**

Bentuk umum penulisan fungsi **strcat()**

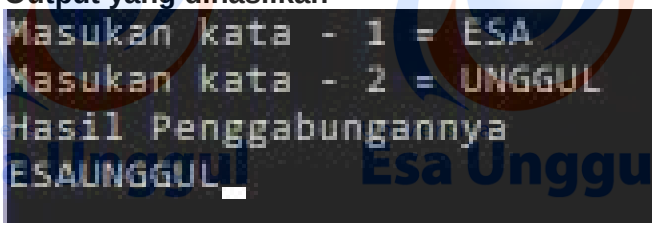
```
#include<string.h>
strcat(tujuan, sumber);
```

Mengabungkan dua buah nilai string tidak dapat dilakukan dengan operator '+', karena operator tersebut bukan untuk operator untuk string. Sehingga penggabungan dua buah nilai string dapat dilakukan dengan fungsi strcat( ) dengan menambahkan string **sumber** ke bagian akhir dari string **tujuan**. Keluaran dari fungsi **strcat** adalah string **tujuan**.

Contoh kode program C++ fungsi strcat()

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

main()
{
    char a1[20];
    char a2[20];

    cout<<"Masukkan Kata - 1= ";    cin>>a1;
    cout<<"Masukkan Kata - 2= ";    cin>>a2;
    strcat(a1, a2);
    cout<<"Hasil Penggabungannya "<<a1;
    getch();
}
```

Output yang dihasilkan



pada hasil dihasil dapat dilihat bahwa a1 (“ESA”) digabungkan dengan a2(“UNGGUL”) dengan hasilnya yang berada di a1(“ESAUNGGUL”).

2. **Fungsi strcmp()**

Fungsi strcmp() digunakan untuk membandingkan dua nilai string. Hasil dari fungsi strcmp() ini bertipe data integer (int). file header yang harus disertakan adalah **string.h**

Bentuk umum penulisan fungsi strcmp()

```
#include<string.h>
Var_int = strcmp(str1, str2);
```

Fungsi ini digambarkan dengan membandingkan string str1 dengan str2. Keluarannya bertipe int yang berupa nilai, diantaranya:

- -1, jika str1 kurang dari str2
- 0, jika str1 sama dengan str2
- 1, jika str1 lebih dari str2

Contoh kode program

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

main()
{
    char a1[] = "UEU";
    char a2[] = "UeU";
    char b1[] = "ueu";

    cout<<"Hasil Perbandingan "<<a1<<" dan "<<a2<<"->";
    cout<<strcmp(a1,a2)<<endl;
    cout<<"Hasil Perbandingan "<<a2<<" dan "<<a2<<"->";
    cout<<strcmp(a2,a1) <<endl;
    cout<<"Hasil Perbandingan "<<a1<<" dan "<<b1<<"->";
    cout<<strcmp(a1,b1) <<endl;

    getch();
}
```

Output yang dihasilkan

```
hasil perbandingan  UEU  dan  UeU ->-1
hasil perbandingan  UeU  dan  UeU ->1
hasil perbandingan  UEU  dan  ueu ->-1
```

3. **Fungsi strcpy()**

Fungsi strcpy() untuk menyalin nilai pada string asal ke variabel string tujuan, dengan syarat string tujuan harus mempunyai fipe data dan ukuran yang sama dengan string asal. File header yang harus disertakan adalah **string.h**

Bentuk umum penulisan strcpy()

```
#include<string.h>
strcpy(tujuan, asal);
```

Variabel tujuan haruslah mempunyai ukuran yang dapat digunakan untuk menampung seluruh karakter dari string asal.

Contoh kode program fungsi strcpy()

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

main()
{
      char huruf[20];
      char pindah[20];

      cout<<"masukan sembarang kata = ";
      gets(huruf);

      /*proses*/
      strcpy(pindah, huruf);

      cout<<"pemindahannya = "<<pindah;

      getch();
}
```

Output yang dihasilkan

```
masukan sembarang kata = esa unggul
pemindahannya = esa unggul
```

4. **Fungsi strlen()**

Fungsi strlen() digunakan untuk mengetahui panjang nilai string. File header yang harus disertakan adalah **string.h**

Bentuk umum penulisan strlen()

```
#include<string.h>
streln(var_string);
```

Keluaran dari fungsi ini adalah panjang dari var_string. Karakter NULL tidak dihitung.

Contoh kode program fungsi strlen()

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

main()
{
      char huruf[20];
      char pindah[20];

      cout<<"masukan sembarang kata = ";
      gets(huruf);

      cout<<"Panjang kata yang diinputkan = ";
      cout<<strlen(huruf);

      getch();
}
```

Output yang dihasilkan

```
masukan sembarang kata = Esa Unggul
Panjang kata yang diinputkan = 10
```

5. **Fungsi strrev()**

Fungsi strrev() ini digunakan untuk membalik letak urutan pada string. String urutan paling akhir dipindahkan keurutan paling depan dan seterusnya. File header yang harus disertakan adalah **string.h**

Bentuk penulisan fungsi strrev()

```
#include<string.h>
strrev(str);
```

Contoh kode program fungsi strrev()

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

main()
{
      char kata[20];

      cout<<"masukan sembarang kata = ";
      gets(kata);

      strrev(kata);
      cout<<"hasil perubahan = "<<kata;

      getch();
}
```

Output yang dihasilkan

```
masukan sembarang kata = Esa Unggul
hasil perubahan = luggnU asE
```

Fungsi konfersi string

1. **Fungsi atof()**
   Fungsi atof() digunakan untuk mengubah string (teks) angka menjadi bilangan numerik float. File header yang harus disertakan adalah **math.h**

   Bentuk umum penulisan fungsi atof()

```
#include<math.h>
Variabel_angka= atof(variabel_char);
```

   Contoh kode program fungsi atof()

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

main()
{
      char kata[20];
      float angka, a, b;

      cout<<"Masukan Sembarang angka = ";
      gets(kata);

      angka = atof(kata);

      a = angka + 5;
      cout<<"Hasil Perubahan ditambah dengan 5 = "<<a;
      getch();
      }
```

   Output yang dihasilkan

```
Masukan Sembarang Kata berupa angka = 454
Hasil Perubahan ditambah dengan 5 = 459
```

2. **Fungsi atoi()**
   Fungsi atoi digunakan untuk mengubah string (teks) angka menajdi bilangan numerik integer. File header yang disertakan adalah **stdlib.h**

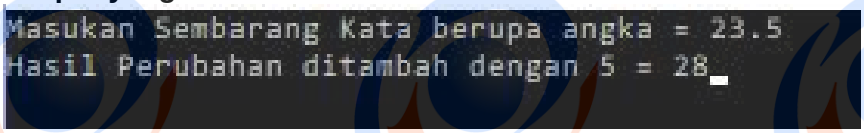
   Bentuk umumpenulisan fungsi atoi()

```
#include<stdlib.h>
Variabel_angka= atoi(variabel_char);
```

Contoh kode program fungsi atoi

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

main()
{
      char kata[20];
      float angka, a, b;

      cout<<"Masukan Sembarang Kata berupa angka = ";
      gets(kata);
      angka = atoi(kata);

      a = angka + 5;
      cout<<"Hasil Perubahan ditambah dengan 5 = "<<a;

      getch();
}
```

Output yang dihasilkan



3. **Fungsi atol()**

Fungsi atol() digunakan untuk mengubah string angka menjadi bilangan numerik long integer.file header yang disertakan adalah **stdlib.h**

Bentuk umumpenulisan fungsi atoi()

```
#include<stdlib.h>
Variabel_angka= atol(variabel_char);
```

Contoh kode program fungsi atol()

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

main()
{
char kata[20];
      float angka, a, b;

      cout<<"Masukan Sembarang Kata berupa angka = ";

      gets(kata);

      angka = atol(kata);
      a = angka + 5;

      cout<<"Hasil Perubahan ditambah dengan 5 = "<<a;

      getch();
}
```

Output yang dihasilkan

```
Masukan Sembarang Kata berupa angka = 23432532522
Hasil Perubahan ditambah dengan 5 = 1.9577e+009
```

4. **Fungsi strlwr()**

Fungsi strlwr() digunakan untuk mengubah setiap huruf kapital (huruf besar) dalam string menjadi huruf kecil. File header yang disertakan adalah **string.h**

Bentuk umumpenulisan fungsi strlwr()

```
#include<string.h>
Strlwr(str);
```

Contoh kode program fungsi strlwr()

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

main()
{
      char kata[20];

      cout<<"Masukan Sembarang Kata dengan Huruf Besar =
";
      gets(kata);

      strlwr(kata);

      cout<<"Hasil Perubahan = "<<kata;
      getch();
}
```

Output yang dihasilkan

```
Masukan Sembarang Kata dengan Huruf Besar =ESA UNGGUL
Hasil Perubahan = esa unggul
```

5. **Fungsi strupr()**

Fungsi strupr() digunakan untuk mengubah setiap huruf kecil menjadi huruf capital (huruf besar). File header yang disertakan adalah string.h

Bentuk umum penulisan fungsi strupr():

```
#include<string.h>
strupr(str);
```

Contoh kode program fungsi strupr()

```
Masukan Sembarang Kata dengan Huruf Kecil = esa unggul
Hasil Perubahan = ESA UNGGUL
```

Latihan

Buatlah beberapa program berikut ini:

1. Buatlah program untuk menghitung panjang karakter berikut ini :

```
Fakultas ilmu komputer esa unggul
```

2. Buatlah program untuk membalik kata – kata berikut ini :

```
   sistem informasi dan ilmu komputer
menjadi
 retupmok umli nad isamrofni metsis
```

3. Buatlah program untuk menggabungkan dua buah string berikut :

```
     Kata 1 : pasar
     Kata 2 : malam
menjadi
      pasarmalam
```

4. Masukan nama lengkap anda, ubah ke dalam huruf besar semua, balikkan urutan hurufnya dan tampilkan hasilnya ke layar

MODUL 8
Array

Tujuan Pembelajaran:

1. Praktikan mengerti dan memahami penggunaan array
2. Praktikan mampu menggunakan beberapa operator dan method yangmenyertai penerapan array

Teori Singkat

Array

Variabel Larik atau lebih dikenal dengan ARRAY adalah adalah Tipe terstruktur yang terdiri dari sejumlah komponen-komponen yang mempunyai tipe yang sama. Suatu Array mempunyai jumlah komponen yang banyaknya tetap. Banyaknya komponen dalam suatu larik ditunjukan oleh suatu indek untuk membedakan variabel yang satu dengan variabel yang lainnya.

Variabel array dapat digolongkan menjadi 3 buah dimensi, diantaranya :

1. Array dimensi 1
2. Array dimensi 2
3. Array dimensi 3

Deklarasi array

sama seperti variabel lainnya, array juga harus dideklarasikan dulu sebelum digunakan

bentuk umum array :

```
tipe_data nama_variabel_array[]
```

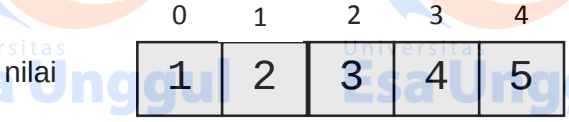
berdasarkan bentuk umum diatas, bahwa array terdiri dari dua bagian
`tipe_data`yang ada di dalam bahasa pemrograman c++ seperti int, float,dll
`nama_variabel_array[ ]` yang dibuat berdasarkan keperluan dan di dalam tanda siku dapat diisi dengan nilai int berapa jumlah maksimum elemen array yang akan dibuat.

Inisialisasi array

Jika nilai suatu variabel array dapat di inisialisasi secara langsung pada saat deklarasi, sebagai contoh :

```
Int nilai[5] = {1,2,3,4,5}
```

Maka, penyimpanan di dalam array dapat digambarkan sebagai berikut :

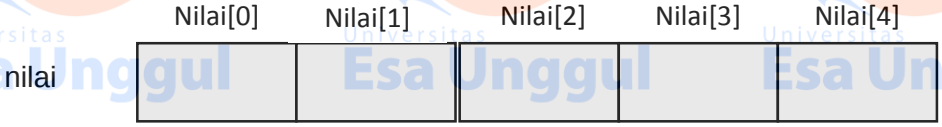


mengakses nilai array

untuk mengakses nilai yang terdapat dalam array, ditulis dalam bentuk, sebagai berikut :

```
Nama[index];
```

pada contoh diatas, variabel nilai memiliki 5 buah elemen yang masing – masing berisi data. Maka, pengaksesan tiap – tiap elemen data adalah :



mengakses array berdimensi satu
suatu array, dapat diakses dengan menggunakan subscript atau indexnya :
bentuk umum pengakses dengan subscript atau index:

```
Nama_array[subscript/index]
```

contoh kode program array dimensi satu

```
#include <conio.h>
#include <iostream.h>
using namespace std;
int main()
{
 //Deklarasi array 'ARnilai' dengan 5 buah elemen berisi int
int ARnilai[5];

// Mengisi nilai ke dalam elemen array
cout<<"== Mengisi Array ARnilai ==\n";
for(int i=0; i<5;i++){
cout<<"Isi indek ke ["<<i<<"] = ";
cin>>ARnilai[i];
     }

     // Menampilkan nilai atau isi dari array 'ARnilai'
for(int i=0; i<5;i++){
cout<<"\nTampil nilai indek ke ["<<i<<"] = "<<ARnilai[i];
     }

return 0;
}
```

Output yang dihasilkan

```
== Mengisi Array ARnilai ==
Isi indek ke [0] = 3
Isi indek ke [1] = 3
Isi indek ke [2] = 5
Isi indek ke [3] = 6
Isi indek ke [4] = 6

Tampil nilai indek ke [0] = 3
Tampil nilai indek ke [1] = 3
Tampil nilai indek ke [2] = 5
Tampil nilai indek ke [3] = 6
Tampil nilai indek ke [4] = 6
```

Array Dimensi Dua
Array dimensi dua tersusun dalam bentuk baris dan kolom, dimana indeks pertama menunjukkan baris dan indeks kedua menunjukan kolom. Array dimensi dua dapat digunakan seperti pendapatan penjualan, pendataan nilai dan lain sebagainya

Bentuk umum penulisan array dimensi dua :

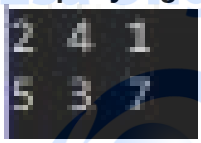
```
Tipe_data nama_variabel[index-1][index-2]
```

Keterangan :
Tipe data : untuk menyatakan tipe data yang digunakan
Index-1 : untuk menyatakan jumlah baris
Index-2 : untuk menyatakan jumlah kolom

Inisialisasi Array Dimensi dua
Contoh kode array dimensi dua

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

using namespace std;

int main()
{
      int i, j;
      int data[2][3] = {{2, 4, 1}, {5, 3, 7}}; //inisialisasi

            for(i=0;i<2;i++)
            {
                  for(j=0;j<3;j++)
                  {
                        cout<<data[i][j];
                        cout<<" ";
                  }
                  cout<<endl;
            }

}
```

Output yang dihasilkan



Pada kode program terdapat array dua dimensi seperti :

```
int data[2][3]
```

Jumlah kolom
Jumlah baris
Nama array
Tipe data elemen array

Contoh lain penggunaan array

Berikut tabel penjualan pada PT X

Data Penjualan Pertahun

NO	Tahun Penjualan		
	2001	2002	2003
1	150	160	230
2	100	125	150
3	210	125	156

Kode program C++ pada untuk soal diatas

```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
#include<iomanip>
using namespace std;
main()
{
      int i, j;
      int data_jual[4][4];

      for(i=1;i<=3;i++)
      {
      for(j=1;j<=3;j++)
      {
      cout<<"Data Ke - "<<i<<" "<<j<<endl;
      cout<<"Jumlah Penjulan : ";

      cin>>data_jual[i][j];
      } }
      cout<<"Data Penjualan Pertahun"<<endl;
      cout<<"----------------------"<<endl;
      cout<<"NO   2001   2002   2003"<<endl;
      cout<<"----------------------"<<endl;

      for(i=1;i<=3;i++)
      {
      cout<<setiosflags(ios::left)<<setw(5)<<i;
      for(j=1;j<=3;j++)
      {
      cout<<setiosflags(ios::right)<<setw(4);
      cout<<data_jual[i][j];
      cout<<"   ";
      }
      cout<<endl;
      }
      cout<<"----------------------"<<endl;

      getch();
}
```

Output yang dihasilkan

```
Data Ke - 1 2
Jumlah Penjulan    : 120
Data Ke - 1 3
Jumlah Penjulan    : 100
Data Ke - 2 1
Jumlah Penjulan    : 200
Data Ke - 2 2
Jumlah Penjulan    : 130
Data Ke - 2 3
Jumlah Penjulan    : 130
Data Ke - 3 1
Jumlah Penjulan    : 250
Data Ke - 3 2
Jumlah Penjulan    : 200
Data Ke - 3 3
Jumlah Penjulan    : 144
Data Penjualan Pertahun
----------------------
NO   2001   2002   2003
----------------------
1     150    120    100
2     200    130    130
3     250    200    144
----------------------
```

Array Dimensi tiga

Array dimensi dua tersusun dalam bentuk baris, kolom dan isi dari baris, dimana indeks pertama menunjukan baris, indeks kedua menunjukkan kolom dan indeks ketiga menunjukkan isi dari baris.

Bentuk umum penulisan array dimensi 3

```
Tipe_data Nama_variabel[index_1][index_2][index_3]
```

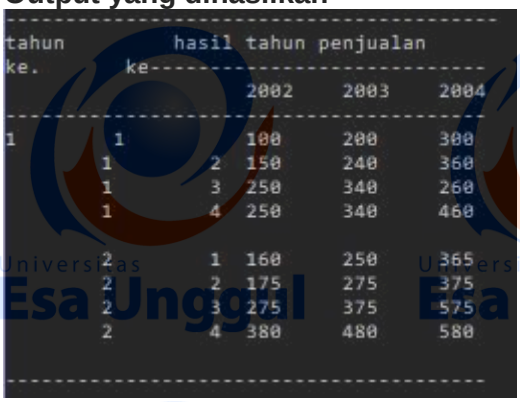
Keterangan :

Type data : untuk menyatakan type data yang digunakan
Index_1 : untuk menyatakan jumlah baris
Index_2 : untuk menyatakan jumlah isi dari baris
Index_3 : untuk menyatakan jumlah kolom

Inisialisasi array dimensi tiga

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
      int i, j, k;
      float data[2][4][3] = {
                              {{100,200,300},
                              {150,240,360},
                              {250,340,260},
                              {250,340,460}},
                              {{160,250,365},
                              {175,275,375},
                              {275,375,575},
                               {380,480,580}}
                              };
      cout<<"---------------------------------------";
      cout<<endl;
      cout<<"tahun         hasil tahun penjualan";
      cout<<endl;
      cout<<"ke.  ke----------------------------";
      cout<<endl;
      cout<<"               2002    2003    2004";
      cout<<endl;
      cout<<"---------------------------------------";
      cout<<endl;

      for(i=0;i<2;i++)
      {
            for(j=0;j<4;j++)
            {
                  cout<<setiosflags(ios::left)<<setw(9)<<i+1;
                  cout<<setiosflags(ios::left)<<setw(9)<<j+1;
                  for(k=0;k<3;k++)
                  {
                        cout<<setiosflags(ios::right)<<setw(5);
                        cout<<data[i][j][k];
                        cout<<"   ";
                  }
                  cout<<endl;
            }
            cout<<endl;
      }
            cout<<"---------------------------------------";
            cout<<endl;
            getch();
}
```

Output yang dihasilkan



```
-------------------------------------
tahun        hasil tahun penjualan
ke.      ke-----------------------------
                  2002    2003    2004
-------------------------------------
1        1        100     200     300
         1   2    150     240     360
         1   3    250     340     260
         1   4    250     340     460

         2   1    160     250     365
         2   2    175     275     375
         2   3    275     375     575
         2   4    380     480     580

-------------------------------------
```

Latihan

1. buatlah program dengan hasil output seperti dibawah ini

```
Layar masukan
Masukan lima data nilai:
Nilai 1 = ……..  <diinput>
Nilai 2 = ……..    <diinput>
Nilai 3 = ……..  <diinput>
Nilai 4 = ……..  <diinput>

Layar keluaran
Data nilai yang anda masukan
1    2    3    4    5
…    …    …    …    …  <ditampilkan>
```

2. Sebuah perusahaan ayam goreng dengan nama “GEROBAK FRIED CHICKEN” yang telah lumayan banyak pelanggannya, ingin dibantu dibuatkan program untuk membantu kelancaran usahaannya. “GEROBAK FRIED CHICKEN” mempunyai daftar harga ayam sebagai berikut :

Kode	Jenis	Harga
D	Dada	Rp. 2500
P	Paha	Rp. 2000
S	Sayap	Rp. 1500

Buatlah programnya dengan ketentuan :

- Setiap pembeli dikenakan pajak sebesar 10% dari pembayaran.
- Banyak Jenis, Jenis Potong dan Banyak Beli diinput.
- Tampilan yang diinginkan sebagai berikut :

```
Layar masukan
GEROBAK FRIED CHICKEN
---------------------
Kode    Jenis    Harga
-------------------------
D      Dada   Rp. 2500
P      Paha     Rp. 2000
S      Sayap   Rp. 1500
-------------------------
Banyak Jenis : ... <diinput>
Jenis Ke - ... <proses counter>
Jenis Potong [D/P/S] : ... <diinput>
Banyak Potong : ... <diinput>
Layar keluaran
GEROBAK FIRED CHICHEN
-----------------------------------------------
No.   Jenis       Harga     Bayak      Jumlah
       Potong      Satuan    Beli       Harga
-----------------------------------------------
...    ....          ....      ....     Rp.... .
...    ....          ....      ....      Rp ....
-----------------------------------------------
                        Jumlah Bayar Rp ....
                        Pajak 10%    Rp ....
                        Total Bayar  Rp ....
```

MODUL9
Pointer

1. Tujuan Pembelajaran:

a. Praktikan mampu memahamipointer
b. Praktikan dapar menggunakan pointer pada program – program sederhana

2. Teori Singkat

Pointer adalah variabel yang berisi (menyimpan alamat memori dari sebuah variabel lain atau pointer dapat diartikan sebagai variabel yang menunjuk ke sebuah alamat memori dari sebuah variabel lain. pointer digunakan dengan maksud untuk menunjukkan ke suatu alamat memori, sehingga dapat mengetahui dengan mudah alamat dari sebuah variabel. Pointer juga bisa diartikan sebagai tipe data yang nilainya mengarah pada nilai yang terdapat pada suatu alamat memori.

Terdapat dua macam operator pointer, diantaranya :

1. **Operator dereference (&)**
   Setiap variabel yang dideklarasikan disimpan dalam sebuah lokasi memori dan pengguna biasanya tidak mengetahui di alamat mana data tersebut disimpan. Dalam C++ untuk mengetahui alamat tempat penyimpanan data, dapat digunakan tanda (&) yang dapat diartikan “alamat”.
   Contoh :
   Bilangan_1 =& Bilangan_2;
   Dibaca : isi variabel Bilangan_1 sama dengan alamat Bilangan_2
2. **Operator reference (*)**
   Penggunaan operator ini, berarti mengakses nilai sebuah alamat yang ditunjuk oleh variable pointer.
   Contoh :
   Bilangan_1 =* Bilangan_2;
   Dibaca: Bilangan_1 sama dengan nilai yang ditunjuk oleh Bilanga_2

Deklarasi pointer pada konstanta

```
Tipe_data * const nama_konstanta;
```

Contoh Kode program C++

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main()
{
      char * const nama = "Borland C++";
      cout<<"Nama Program = "<<nama<<endl;
      nama = "Visual C++";
      cout<<"Nama Program  = "<<nama<<endl;
      getch();
}
```

Pada program diatas, terdapat kesalahan dan tidak dapat dijalankan, penyebabnya pada pernyataan nama = "Visual C++";. Karena variabel nama, merupakan merupakan pointer konstanta, yaitu tidak dapat diubah- ubah. Pesan Kesalahan Yang Tampil adalah : Cannot modify a const object.

Deklarasi pointer pada variabel

Bentuk penulisan :

```
Tipe_data * nama_konstanta;
```

Contoh Kode program

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main()
{
      int ilham, amir, *raka;

      ilham = 75;
      amir  = ilham;
      raka  =&ilham;

      cout<<"Nilai ILHAM = "<<ilham<<endl;
      cout<<"Nilai AMIR  = "<<amir<<endl;
      cout<<"Nilai RAKA  = "<<raka<<endl;

      getch();
}
```

Output yang dihasilkan

```
Nilai ILHAM = 75
Nilai AMIR  = 75
Nilai RAKA  = 0x6ffe2c
```

Contoh lain kode program

```
#include<conio.h>
#include<iostream.h>
main()
{
      int yofrie = 93;
      int *hadiansyah ;
      cout<<"Nilai awal yofrie = "<<yofrie<<endl;
      hadiansyah = &yofrie;

      cout<<"Nilai hadiansyah sekarang = ";
      cout<<*hadiansyah<<endl;

      *hadiansyah = 50;

      cout<<"Nilai hadiansyah sekarang = ";
      cout<<*hadiansyah<<endl;

      getch();
}
```

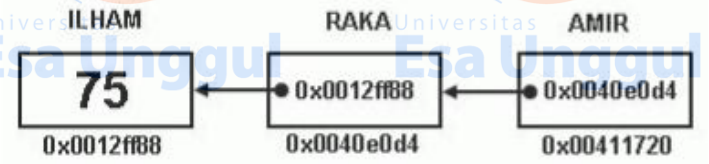
Output yang dihasilkan

```
Nilai awal yofrie = 93
Nilai hadiansyah sekarang = 93
Nilai hadiansyah sekarang = 50
```

Pointer pada Pointer

Tidak terbatas menunjuk alamat dari suatu variabel, pointer dapat pula menunjuk ke pointer lainnya. Di dalam pendeklarasianya, hanya menambahkan pointer reference (*) pada variabel yang akan ditunjuk. Sebagai contoh:

```
char ilham;
char *raka;  //pointer ke variabel
char **amir;  //pointer pada pointer

ilham = '75';
raka = &ilham;
amir = &raka;
```



Kode program

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
using namespace std;

main()
{
      int ilham;
      int *raka; //pointer ke variabel
      int **amir; // pointer pada pointer

      ilham = 75;

      cout<<"nilai ilham = "<<ilham<<endl;

      //penugasan ke alamat memori

      raka = &ilham;
      amir = &raka;

      cout<<"nilai raka hasil mengakses ilham = ";
      cout<<*raka<<endl;

      cout<<"nilai amir hasil mengakses ilham = ";
      cout<<**amir<<endl;

      getch();
}
```

Output yang dihasilkan

```
nilai ilham = 75
nilai raka hasil mengakses ilham = 75
nilai amir hasil mengakses ilham = 75
```

Pointer pada Array

Konsep Array diantaranya adalah banyak loncatan dari pointer satu ke pointer yang lain. karena secara internal array juga menyatakan alamat, yaitu pengenal array sama dengan alamat pada elemen pertama, pada array.

Contoh sederhana dapat dilihat pada kode program dibawah ini !

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main()
{
      int i;
      int nilai[5];
      int *ptrnilai;

      ptrnilai = nilai;

      for(i=1;i<=5;i++)
      {
            cout<<"Masukan Nilai Pertama = ";
            cin>>nilai[i];
      }
            cout<<endl;
            cout<<"Hasil Pengaksesan Elemen Array Lewat";
            cout<<"Pointer";
            cout<<endl<<endl;
      for(i=1;i<=5;i++)
      {
            cout<<"Elemen "<<i<<". Nilai "<<nilai[i];
            cout<<", Menempati Alamat Memori = ";
            cout<<&ptrnilai[i];
            cout<<endl;
      }

      getche();
}
```

Output yang dihasilkan

```
Masukan Nilai Pertama = 70
Masukan Nilai Pertama = 80
Masukan Nilai Pertama = 90
Masukan Nilai Pertama = 80
Masukan Nilai Pertama = 89

Hasil Pengaksesan Elemen Array LewatPointer

Elemen 1. Nilai 70, Menempati Alamat Memori = 0x6ffe14
Elemen 2. Nilai 80, Menempati Alamat Memori = 0x6ffe18
Elemen 3. Nilai 90, Menempati Alamat Memori = 0x6ffe1c
Elemen 4. Nilai 80, Menempati Alamat Memori = 0x6ffe20
Elemen 5. Nilai 89, Menempati Alamat Memori = 0x6ffe24
```

Pointer pada String

Pointer pada string dapat anda lihat pada contoh program berikut :

```
#include<iostream.h>
#include<conio.h>
main()
{
      char band_metal[] = "SEPULTURA";
      char *band_punk = "RANCID";

      cout<<"Nama Band Metal = "<<band_metal<<endl;
      cout<<"Nama Band Punk  = "<<band_punk<<endl;

      band_punk+=3;   //menambah nilai penunjuk / pointer
      cout<<"Nama Band Metal = "<<band_metal<<endl;
      cout<<"Nama Band Punk  = "<<band_punk;

      getche();
}
```

Output yang dihasilkan

```
Nama Band Metal = SEPULTURA
Nama Band Punk  = RANCID
Nama Band Metal = SEPULTURA
Nama Band Punk  = CID
```

Latihan

1. Buatlah program sederhana dengan output seperti dibawah ini !

```
Layar masukan
Masukan nilai 1 : ..... <diinput>
Masukan nilai 2 : ..... <diinput>
Masukan nilai 3 : ..... <diinput>
Masukan nilai 4 : ..... <diinput>
Masukan nilai 5 : ..... <diinput>

Layar keluaran
Nilai a[0] = ..... <tampil nilai 1>
Nilai a[1] = ..... <tampil nilai 2>
Nilai a[2] = ..... <tampil nilai 3>
Nilai a[3] = ..... <tampil nilai 4>
Nilai a[4] = ..... <tampil nilai 5>
```

Modul 8
Fungsi *call by value*danFungsi*call by reference*

Tujuan Pembelajaran:

a. Praktikan mampu menterjemahkan Fungsi ke dalam bentuk pemrograman.
b. Praktikan mampu menerapkan fungsi – fungsi tersebut ke dalam program sederhana

Teori Singkat

Fungsi (*function*) merupakan blok dari kode yang dirancang untuk melaksanakan tugas khusus. Kegunaan fungsi ini adalah untuk, diantaranya :

- Mengurangi pengulangan penulisan program yang berulang atau sama
- Program menjadi lebih terstruktur, sehingga mudah dipahami dan dapat lebih dikembangkan

Fungsi – fungsi yang sudah dikenal sebelumnya adalah fungsi main() merupakan fungsi yang bersifat mutlak, karena fungsi ini program akan dimulai, sebagai contoh yang lainnya printf(), cout() yang mempunyai tugas untuk menampilkan informasi atau data kelayar.

Bentuk struktur umum penulisan fungsi, sebagai berikut :

```
Nama_fungsi(argument)
{
      …… pernyataan / perintah;
      …… pernyataan / perintah;
      …… pernyataan / perintah;
      }
```

Keterangan :

- Nama fungsi, boleh dituliskan secara bebas dengan ketentuan, tidak menggunakan spasi dan nama-nama fungsi yang mempunyai arti sendiri.
- Argumen, diletakan diantara tanda kurung “( )” yang terletak dibelakang nama fungsi. Argumen boleh diisi dengan suatu data atau dibiarkan kosong.
- Pernyataan / perintah, diletakan diantara tanda kurung ‘{ }’.

Contoh fungsi sederhana :

Parameter fungsi

Terdapat dua macam parameter fungsi, diantaranya :

- Parameter formal adalah variabel yang terdapat pada daftar parameter yang berada di dalam definisi fungsi
- Parameter aktual adalah variabel yang digunakan pada pemanggilan suatu fungsi

Bentuk umum penulisan parameter forma dan parameter aktual

Ada dua cara untuk melewatkan parameter ke dalam fungsi, yaitu berupa :

1. **Pemanggilan dengan nilai (*call by value*)**
   Pemanggilan dengan nilai yaitu nilai dari parameter aktual akan dimasukkan ke parameter formal. Dengan cara ini nilai parameter aktual tidak bisa berubah, walaupun nilai dari parameter formal berubah.

Contoh kode program pemanggilan dengan nilai

```
#include<iostream>
using namespace std;
#include<conio.h>

void Vtukar(int bil1, int bil2)
{
      int temp;
      temp = bil1;
      bil1 = bil2;
      bil2 = temp;
      cout<<"nilai pada saat berada di fungsi Vtukar : "<<endl;
      cout<<"nilai bilangan 1 = "<<bil1<<"| alamat bilangan 1 =
"<<&bil1<<endl;
      cout<<"nilai bilangan 2 = "<<bil2<<"| alamat bilangan 2 =
"<<&bil2<<endl;
      cout<<endl;
}
int main()
{
      int bil1, bil2;
      bil1 = 7;
      bil2 = 6;

      cout<<"\tCALL BY VALUE"<<endl;
      cout<<"t--------------"<<endl;
      cout<<"\nNilai  pada  saat  berada  di  fungsi  Vtukar  :
"<<endl;
      cout<<"nilai bilangan 1 = "<<bil1<<"| alamat bilangan 1 =
"<<&bil1<<endl;
      cout<<"nilai bilangan 2 = "<<bil2<<"| alamat bilangan 2 =
"<<&bil2<<endl;
      cout<<endl;

      _getche();
      return 0;
}
```

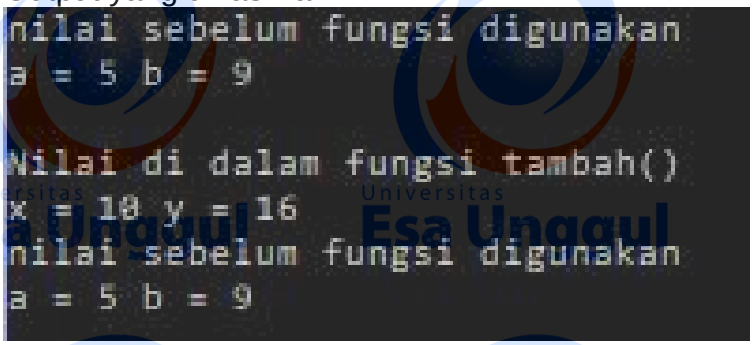
Output yang dihasilkan

```
        CALL BY VALUE
t--------------

Nilai pada saat berada di fungsi Vtukar :
nilai bilangan 1 = 7| alamat bilangan 1 = 0x6ffe3c
nilai bilangan 2 = 6| alamat bilangan 2 = 0x6ffe38
```

Contoh lain kode program

```
#include<conio.h>
#include<stdio.h>
#include<iostream>
using namespace std;

void tambah(int x, int y);
main()
{
      int a, b;

      a = 5;
      b = 9;

      cout<<"nilai sebelum fungsi digunakan ";
      cout<<"\na = "<<a<<" b = "<<b;

      tambah(a,b);

      cout<<"nilai sebelum fungsi digunakan ";
      cout<<"\na = "<<a<<" b = "<<b;
      getch();
}

void tambah(int x, int y)
{
      x += 5;
      y += 7;

      cout<<"\n\nNilai di dalam fungsi tambah()";
      cout<<"\nx = "<<x<<" y = "<<y;
      cout<<endl;
}
```

Output yang dihasilkan :

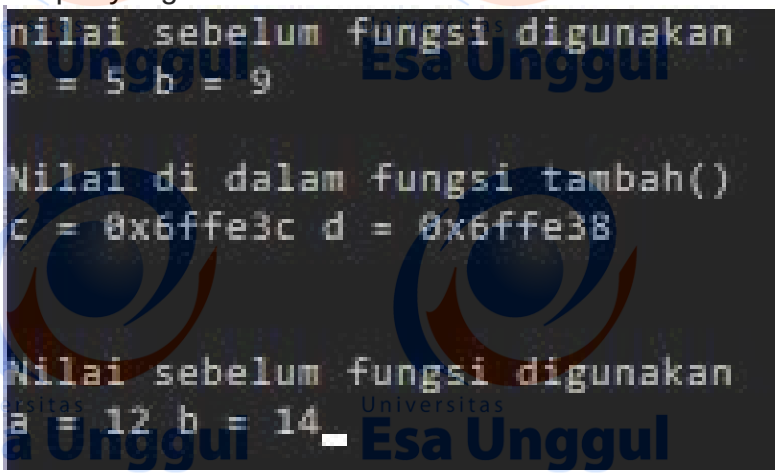


2. **Pemanggilan dengen referensi (*call by reference*)**
Pemanggilan dengan referensi merupakan pemanggilan alamat suatu variabel di dalam fungsi. Cara ini dapat dipakai untuk mengubah isi suatu variabel yang diluar dari fungsi dengan melaksanakan pengubahan nilai dari suatu variabel dilakukan di dalam fungsi

Contoh kode program referensi

```
#include<conio.h>
#include<stdio.h>
#include<iostream>
using namespace std;

void tambah(int *c, int *d);
main()
{
      int a, b;

      a = 5;
      b = 9;

      cout<<"nilai sebelum fungsi digunakan ";
      cout<<"\na = "<<a<<" b = "<<b;

      tambah(&a,&b);

      cout<<"nilai sebelum fungsi digunakan ";
      cout<<"\na = "<<a<<" b = "<<b;

      getch();
}

void tambah(int *c, int *d)
{
      *c += 7;
      *d += 5;

      cout<<"\n\nNilai di dalam fungsi tambah()";
      cout<<"\nc = "<<c<<" d = "<<d;
      cout<<endl;
}
```

Output yang dihasilkan :



Pernyataan *return*( )

Pernyataan return() digunakan untuk mengirimkan nilai atau nilai dari suatu fungsi kepada fungsi lain yang memanggilnya. Pernyataan return( ) diikuti oleh argument yang berupa nilai ynag akan dikirimkan

Contoh pemakain return()

```
#include<string>
#include<iostream>
using namespace std;
string hallo(string a=""){
     return "hallo "+a+"!";
}
int main(){
     cout<<hallo("belajar C++");
     return 0;
}
```

Output yang dihasilkan

```
hallo belajar C++!

--------------------------------
```

Contoh lain kode program pernyataan return

```
#include <iostream>
using namespace std;

double luaslingkaran(double r=0){
     if(!r){
          cout<<"anda tidak memasukan argumen"<<endl;
          return 0;
     }
     const double pi = 3.14;
     return pi*r*r;
}
int main(){
     cout<<luaslingkaran(3)<<endl;
     return 0;
}
```

Output yang dihasilkan

```
28.26
```

Pengiriman Data ke fungsi

Pengiriman data ke fungsi terdiri dari dua macam, yaitu :

1. **Pengiriman data konstanta ke fungsi**

   Mengirimkan suatu nilai data konstanta ke suatu fungsi yang lain dapat dilakukan dengan cara yang mudah.

   Berikut contoh program C++

```
#include<conio.h>
#include<stdio.h>
#include<iostream>
using namespace std;

float luas(float sisi=0);

main()
{
      float luas_bs;

      luas_bs = luas(4.25);
      cout<<"\nLuas Bujur Sangkar = "<<luas_bs;
      getch();
}
float luas(float sisi){
      return(sisi*sisi);
}
```

   Output yang dihasilkan

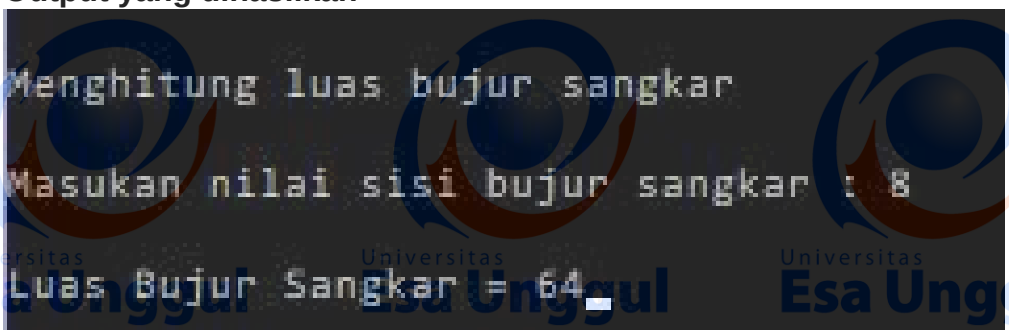
2. **Pengiriman Data Variabel ke Fungsi**

   Bentuk pengiriman data variabel, sama seperti halnya pengiriman suatu nilai data konstanta ke suatu fungsi. Hanya saja nilai yang dikirimkan sewaktu waktu dapat berubah – ubah.

   Contoh kode program data variabel ke fungsi

```
#include<conio.h>
#include<stdio.h>
#include<iostream>
using namespace std;

float luas(float sisi);

main()
{
      float luas_bs, sisi_bs;

      cout<<"\nMenghitung luas bujur sangkar"<<endl;
      cout<<"\nMasukan    nilai    sisi    bujur    sangkar    :
";cin>>sisi_bs;

      luas_bs = luas(sisi_bs);

      cout<<"\nLuas Bujur Sangkar = "<<luas_bs<<"cm";

      getch();
}
      float luas(float sisi)
      {
            return(sisi*sisi);
```

Output yang dihasilkan



Jenis – jenis Variabel

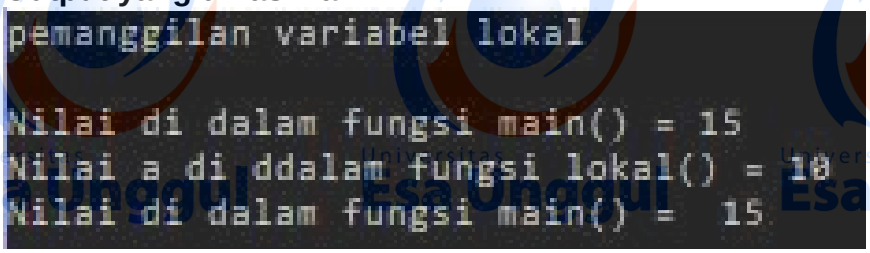
Jenis variabel pada C++ sangat berguna di dalam penulisan suatu fungsi agar penggunaan di dalam penggunaan suatu variabel tidak sala, Terdapat beberapa jenis variabel, yaitu :

a. **Variabel lokal**
Variabel lokal adalah variabel yang dideklarasikan di dalam fungsi dan hanya dikenal oleh fungsi yang bersangkutan. Variabel lokal biasanya disebut dengan variabel otomatis

Contoh kode program variabel lokal

```
#include<conio.h>
#include<stdio.h>
#include<iostream>
using namespace std;

void lokal();

main(){
      int a = 15;

      cout<<"pemanggilan variabel lokal"<<endl;
      cout<<"\nNilai di dalam fungsi main() = "<<a;

      lokal();

      cout<<"\nNilai di dalam fungsi main() =  "<<a;
      getch();
}
void lokal()
{
      int a= 10;
      cout<<"\nNilai a di ddalam fungsi lokal() = "<<a;
}
```

Output yang dihasilkan



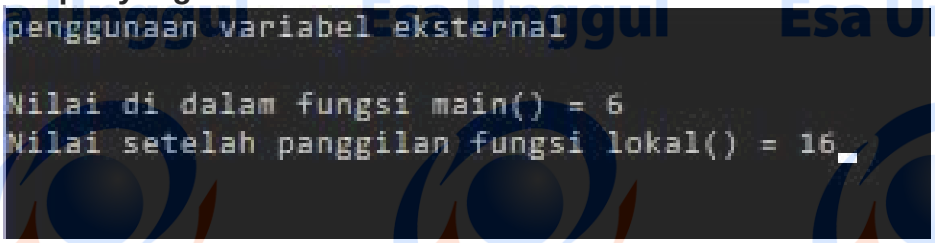
b. Variabel eksternal atau global

Variabel eksternal adalah variabel yang dideklarasikan di luar fungsi yang bersifat global yang artinya dapat digunakan bersama – sama tanpa harus dideklarasikan berulang – ulang

Berikut contoh program variabel eksternal

```
#include<conio.h>
#include<stdio.h>
#include<iostream>
using namespace std;

int a = 6; // deklarasi variabel eksternal

void lokal();

main(){
      cout<<"penggunaan variabel eksternal"<<endl;
      cout<<"\nNilai di dalam fungsi main() = "<<a;

      lokal(); //pemanggilan fungsi lokal

      cout<<"\nNilai setelah panggilan fungsi lokal() = ";
      cout<<a;
      getch();
}
void lokal()
{
      a+=10;
}
```

Output yang dihasilkan



c. Variabel statis

Variabel statis dapat berupa variabel lokal atau variabel eksternal. Terdapat beberapa sifat yang dimiliki oleh variabel statis, diantaranya :

a. Jika variabel statis bersifat lokal, maka variabel hanya dikenal oleh fungsi tempat variabel dideklarasikan
b. Jika variabel statis bersifar eksternal, maka variabel dapat dipergunakan oleh semua fungsi yang terletak pada file yang sama ditempat variabel statis dideklarasikan
c. Jika tidak ada inisialisasi oleh pemrograman secara otomatis akan diberikan nilai awal nol

Pada variabel statis diperoleh dengan menambahkan kata kunci statis di depan penentu tipe data variabel.

Berikut contoh program variabel statis

```
#include<conio.h>
#include<stdio.h>
#include<iostream>
using namespace std;

int hitung(); //prototype fungsi hitung

main()
{
      int K = 7;

      hitung();
      hitung();

      cout<<"\nNilai K di dalam fungsi main() = "<<K;
      getch();
}
int hitung()
{
      static int K; //deklarasi variabel statis
      K += 4;
      cout<<"\nNilai K di dalam fungsi() = "<<K;
}
```

Output yang dihasilkan

```
Nilai K di dalam fungsi() = 4
Nilai K di dalam fungsi() = 8
Nilai K di dalam fungsi main() = 7
```

Latihan:

Buatlah program untuk menghitung jumlah pembayaran pada perpustakaan X, dengan ketentuan :

Kode Jenis Buku	Jenis Buku	Tarif Buku
C	CerPen (Kumpulan Cerita Pendek)	500
K	Komik	700
N	Novel	1000

Petunjuk proses:

1. Buatlah fungsi tarif untuk menentuak tarif sewa
2. Gunakan pernyataan if-else dan tampilan output yang diingikan !

Nama Penyewa Buku : .... <diinput>
Kode Buku [C/K/N] : .... <diinput>
Banyak Pinjam : .... <diinput>

Tampilan Keluaran yang diinginkan :

Tarif Sewa Rp. .... <hasil proses>
Jenis Buku : ..... < hasil proses >
Penyewa dengan Nama ..... <hasil proses>
Jumlah Bayar Penyewaan Sebesar Rp. ..... <hasil proses>

Modul11
Fungsi Overloading dan Fungsi Inline

Tujuan Pembelajaran*:*

Praktikan mampu menerapkan operator inline dan overloading pada bahasa pemrograman C++

Teori Singkat

Fungsi overloading adalah suatu proses menggunakan nama yang sama untuk dua atau lebih fungsi. Setiap definisi ulang dari fungsi yang di overloading harus menggunakan tipe parameter, urutan parameter atau jumlah parameter yang berbeda. jumlah, tipe atau suatu fungsi disebut function signature. Jika memiliki sejumlah fungsi dengan nama yang sama compiler akan mengidentifikasi fungsi – fungsi tersebut berdasarkan parameter

Berikut operator yang dapat di overloading :

Operator	Nama	Tipe
,	comma	binary
!	logical NOT	unary
!=	inequality	binary
%	modulus	binary
%=	modulus assignment	binary
&	bitwise AND	binary
&	address of	unary
&&	logical AND	binary
&=	bitwise AND assignment	binary
( )	function all	-
( )	cast operator	unary
*	multiplication	binary
*	pointer dereference	unary
*=	multiplication assignment	binary
+	addition	binary
+	unary plus	unary
++	increment	unary
+=	addition assignment	binary
-	subtraction	binary
-	unary negation	unary
--	decrement	unary
-=	subtraction assignment	binary
->	member selection	binary
->*	pointer- to- member selection	binary
/	division	binary
/=	division assignment	binary
<	less than	binary
<<	left shift	binary
<=	less than or equal to	binary
=	assignment	binary
==	equality	binary
>	greater than	binary
>=	greater than equal to	binary
>>	right shift	binary
>>=	right shift assignment	binary
[ ]	array subscript	-

^	exclusive OR	binary
^=	exclusive OR assignment	binary
|	bitwise inclusive OR	binary
|=	bitwise inclusive OR assignment	bnary
||	logical OR	binary
~	one's complement	unary
delete	delete	-
new	new	-
conversion operators	conversion operators	unary

Berikut merupakan operator overloading yang tidak dapat di overload:

operator	keterangan
::	operator resolusi
?:	operator kondisi
.	operator pemilihan member
sizeof	operator size-of
.*	de-reference pointer untuk class member

Contoh kode program function overloading

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

int hitung(int b);
long hitung(long c);
float hitung(float d);
main()
{
cout<< "Hasilnya Fungsi overload -1 : ";
cout<<hitung(4)<<endl;
cout<< "Hasilnya Fungsi overload -2 : ";
cout<<hitung(2)<<endl;
cout<< "Hasilnya Fungsi overload -3 : ";
cout<<hitung(3)<<endl;

getch();
  }
int hitung(int b)
   {
return(b*b);
   }
long hitung(long c)
   {
return(c*c);
   }
double hitung(double d)
   {
return(d*d);
   }
```

Output yang dihasilkan

```
Hasilnya Fungsi overload -1 : 16
Hasilnya Fungsi overload -2 : 4
Hasilnya Fungsi overload -3 : 9
```

Contoh lain kode program fungsi overloading

```
#include <iostream>
using namespace std;

/* contoh fungsi overloading */
int persegi( int sisi) /* fungsi return value interger */
{
cout<< "Luas persegi yang sisinya " << sisi  << " adalah " ;
return sisi*sisi;
}
double persegi (double sisi) /* fungsi return value double */
{
cout  << "Luas sisi persegi yang sisinya " << sisi << " adalah "
;
return sisi*sisi;
}

int main(){
cout<< persegi (5) <<endl; // pemanggilan fungsi
cout<<persegi(5.5) << endl;
return 0;
}
```

Output yang dihasilkan

```
Luas persegi yang sisinya 5 adalah 25
Luas sisi persegi yang sisinya 5.5 adalah 30.25
```

Contoh 3 kode program fungsi overloading

```
#include <iostream>
using namespace std;

//fungsi prototype
int volumePersegi (int panjang=1, int  lebar=1 ,int tinggi=1);

int main(){
cout<<"Volume  persegi  dengan  panjang  10,  lebar  1  dan  tinggi  1
adalah"<<volumePersegi(10)<<endl;
cout<<"Volume persegi dengan panjang 10, lebar 5 dan tinggi 1 adalah"
<<volumePersegi(10,5)<<endl;
cout<<"Volume persegi dengan panjang 10, lebar 5 dan tinggi 6 adalah"
<<volumePersegi (10,5,6)<<endl;
return 0;
}

int volumePersegi (int panjang, int  lebar, int tinggi)
{
return panjang*lebar*tinggi;
}
```

Output yang dihasilkan

```
Volume persegi dengan panjang 10, lebar 1 dan tinggi 1 adalah  10
Volume persegi dengan panjang 10, lebar 5 dan tinggi 1 adalah  50
Volume persegi dengan panjang 10, lebar 5 dan tinggi 6 adalah  300
```

Fungsi inline

Fungsi inline digunakan untuk mengurangi lambatnya eksekusi program dan mempercepat eksekusi program terutama pada program yang sering mmenggunakan atau memanggil fungsi yang berlebih. Terutama program – program yang menggunakan pernyataan perulangan proses seperti for, while, dan do – while. fungsi inline dideklarasikan dengan menambahkan kata kunci inline di depan tipe data

Contoh kode program fungsi inline

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

inline int kali(int i, int j)
 {
      return(i * j);
}
main()
{
      int k;
      for(k = 1; k < 20; k++)
      cout<<kali(k, k*2)<<" ";

      getch();
      }
```

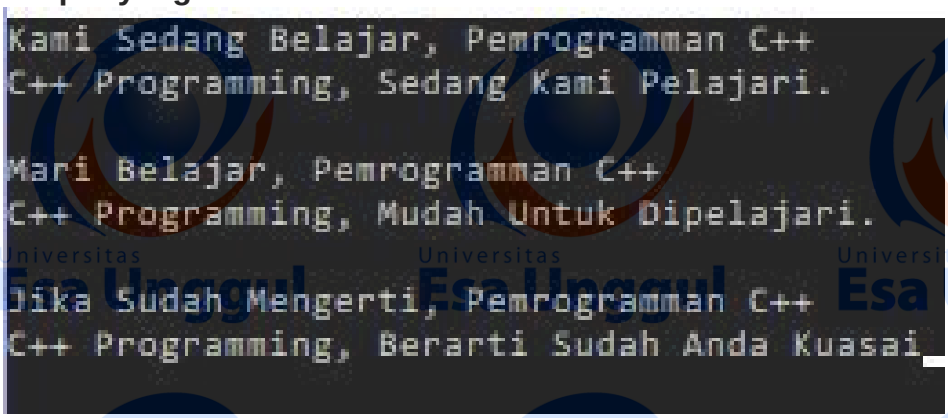
Output yang dihasilkan

```
2 8 18 32 50 72 98 128 162 200 242 288 338 392 450 512 578 648 722
--------------------------------
```

Contoh yang lain

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

inline static void cplusplus()
{
      cout<< "Pemrogramman C++\n";
      cout<< "C++ Programming, ";
}
      int main()
      {   {
      cout<< "Kami Sedang Belajar, ";
      cplusplus();
      cout<<"Sedang Kami Pelajari.\n\n";}
            {
             cout<< "Mari Belajar, ";
             cplusplus();
             cout<< "Mudah Untuk Dipelajari.\n\n";
              } {      cout << "Jika Sudah Mengerti, ";
             cplusplus();
             cout<< "Berarti Sudah Anda Kuasai";
              }
             getche();
      }
```

Output yang dihasilkan



Latihan :

Buatlah program untuk menghitung konversi dari derajat fahrenheit ke celcius dengan Petunjuk :

1. Gunakan Function Overloading.
2. Buatlah 3 (tiga) buah fungsi untuk dioverloading, dengan variabel untuk masing-masing fungsi berbeda-beda:
   - Untuk fungsi pertama variabel yang digunakan adalah double
   - Untuk fungsi pertama variabel yang digunakan adalah float
   - Untuk fungsi pertama variabel yang digunakan adalah integer
3. Rumus konversi yang digunakan adalah

   c = (f - 32.0) * 5 / 9;

Contoh :

Jika nilai Fahrenheit = 100
c = (100 - 32) * 5 / 9;
c = (68) * 5 / 9;
c = 37,7778

Hasil keluaran yang diinginkan :

```
Pemanggilan dengan tipe data double
Proses dengan tipe data double
100 sama dengan 37.7778

Pemanggilan dengan tipe data float
Proses dengan tipe data float
100 sama dengan 37.7778

Pemanggilan    dengan    tipe    data
integer
Proses dengan tipe data integer
```

Modul 12
Struktur pada C++

Tujuan Pembelajaran:

Praktikan mampu memahami konsep struktur pada bahasa pemrograman C++

Pengertian Struktur pada C++

Struktur C++ adalah koleksi variabel dibawah sebuah nama, variabel – variabel ini dapat berbentuk berbagai type, yaitu sebagai berikut :

- Int
- Float
- Char
- Dan lain-lain

Perbedaan utama antara struktur dan array adalah bahwa dalam array memiliki tipe data yang sama, sedangkan struktur adalah sebuah koleksi dari variabel – variabel dibawah nama yang sama, dimana setiap elemen dapat saja memiliki tipe yang berbeda.

Deklarasi struktur dalam C++

Struktur dalam C++ dideklarasikan menggunakan keyword struct diikuti dengan nama struktur atau sering disebut dengan tag. Variabel – variabel struktur dideklarasikan dalam kurung kurawal { }, setiap elemennya dipisahkan dengan tanda titik koma atau semi colom

Berikut bentuk penulisan umum struktur dalam c++

```
Struct nama_struct

{

Anggota_struktur;

};
```

Contoh deklarasi struktur

```
Struct data_tanggal
{
   int tahun;
   int bulan;
   int tanggal;
};
```

Contoh kode program menggunakan struktur

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
using namespace std;
main(){
      struct data_tanggal{
            int tanggal;
            int bulan;
            int tahun;
      };
      data_tanggal tgl1, tgl2;
      tgl1.tanggal = 1;
      tgl1.bulan = 9;
      tgl1.tahun = 2018;
      tgl2 = tgl1;
      //atau
      tgl2.tanggal = tgl1.tanggal;
      tgl2.bulan = tgl1.bulan;
      tgl2.tahun = tgl1.tahun;
      cout<<tgl1.tanggal<<'/'<<tgl1.bulan<<'/'<<tgl1.tahun<<endl;
      cout<<tgl2.tanggal<<'/'<<tgl2.bulan<<'/'<<tgl2.tahun<<endl;
}
```

Output yang dihasilkan

Struktur dalam fungsi

Suatu struktur juga dapat digunakan untuk argument / parameter suatu fungsi (*function*)

Contoh kode program

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
using namespace std;
struct data{
      int x;
      int y;
};
void tampilkan(data nilai);
main()
{
      data nilaiku;
      nilaiku.x = 10;
      nilaiku.y = 16;
      tampilkan(nilaiku);
}
```

```
void tampilkan(data nilai)
{
      cout<<"nilai x = "<<nilai.x<<endl;
      cout<<"nilai y = "<<nilai.y<<endl;
}
```

Output yang dihasilkan

```
nilai x = 10
nilai y = 16
```

Struktur dalam pointer

Contoh kode program struktur dalam pointer

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
using namespace std;

//deklarasi struktur
struct koordinat{
      int x, y;
};
void ubah_posisi(koordinat *posisi); //prototipe
main(){
      koordinat posisi; //definisi variabel struktur
      posisi.x = 10;
      posisi.y = 30;
      cout<<"isi semula : x = "<<posisi.x<<"y = "<<posisi.y<<endl;
      ubah_posisi(&posisi); //parameter berupa alamat
      cout<<"isi    sekarang    :    x    =    "<<posisi.x<<"y    =
"<<posisi.y<<endl;
}
//definisi fungsi
void ubah_posisi(koordinat *posisi){
      (*posisi).x = 100;
      (*posisi).y = 300;
}
```

Output yang dihasilkan

```
isi semula : x = 10y = 30
isi sekarang : x = 100y = 300
```

Latihan :

Buatlah program sederhana dengan layar masukan dan keluaran seperti dibawah ini

```
Layar masukan

Data Mahasiswa
Nim    : ... <input>
Nama   : ... <input>
Jurusan: ... <input>

Layar keluaran

Data Mahasiswa
Nim    : ... <tampil>
Nama   : ... <tampil>
Jurusan: ...<tampil>
```

Modul 13 dan 14
Pemrograman Berorientasi Objek

Tujuan Pembelajaran

- Praktikan mengerti konsep pewarisan(*inheritance*) dan deklarasi pewarisan(*inheritance*) dalam bahasa pemrograman C++
- Praktikan mengerti konsep polimorfisme dan deklarasi polimorfisme dalam bahasa pemrograman C++
- Pratikan mengerti konsep enkapsulasi dan deklarasi enkapsulasi dalam bahasa pemrograman C++

Teori Singkat

Pemrograman berorientasi objek (PBO) adalah metode pemrograman yang meniru cara memperlakukan sesuaitu (benda). Terdapat tiga karakteristik bahasa pemrograman berorientasi objek, yaitu :

1. Inheritance (pewarisan)

inheritance mendefinisikan suatu kelas dan kemudian menggunakannya untuk memabangun hirarki kelas turun yang mana masing – masing turunan mewarisi semua akses kode maupun data kelas dasarnya.

Konsep inheritance

Suatu kelas dapat diciptakan berdasarkan kelas lain, kelas baru ini mempunyai sifat – sifat yang sama dengan kelas pembentuknya dann ditambah dengan sifat – sifat khusus lainnya. Dengan inheritance (pewarisan) dapat menciptakan kelas – kelas baru yang mempunyai sifat yang sama dengan kelas lainnya tanpa harus menulis ulang bagian – bagian yang sama. Pewarisan merupakn unsure penting dalam OOP dan merupakan blok bangunan dasar pertama dengan penggunaan kode ulang (*code reuse*).

Sifat – sifat inheritance

Dalam bahasa pemrograman C++ terdapat tiga macam sifat inheritance, diantaranya:

1. Public
   Penentuan akses berbasis public menyebabkan anggota dari public sebuah kelas utama akan menjadi anggota public kelas turunan.
2. Private
   Penentuan akses berbasis private menyebabkan anggota public dari kelas utama akan menjadi anggota protect kelas turunan dan menyebabkan anggota kelas utama menjadi protect kelas turunan. Anggota kelas private tetap pada private kelas utama.
3. Protected
   Penentu akses berbasis protected menyebabkan anggota dari anggota protect dan public dari kelas utama akan menjadi anggota private dari kelas turunan. Anggota private dari kelas utama selalu menjadi anggota private kelas utama.

Manfaat penggunaan konsep pewarisan, antara lain:

1. Dapat mengunakan kembali kelas – kelas yang dibuat sebagai superkelas dan membuat kelas – kelas baru berdasarkan superkelas tersebut dengan karakteristik yang lebih khusus dari behaviour umum yang dimiliki superkelas
2. Dapat membuay superkelas yang hanya mendefinisikan behaviour namun tidak memberi implementasi dari metode – metode yang ada. Hal ini berguna jika kita ingin membuat semacam template kelas. Kelas semacam in disebut kelas abstrak, karena behaviournya masih abstrak dan belum diimplementasikan. Subkelas – subkelas dari kelas semacam ini, yang disebut kelas konkret, mengimplementasikan behaviour abstrak tersebut sesuai dengan kebutuhan masing – masing.

Ada tiga jenis – jenis pewarisan, diantaranya :

1. **inheritance Tunggal**

Ilustrasikan pewarisan

Keterangan :

- Kelas hewan sebagai kelas dasar mewariskan anggotanya yaitu indera, alat gerak, dll) kepada kelas burung dan kelas gajah yang selanjutnya disebut sebagai kelas turunan
- Anggota khusus merupakan anggota yang hanya dimiliki oleh kelas tersebut sehingga membedakan dengan kelas lain.
  Misalnya anggota khusus dari kelas burung adalah sayap terbang, sedangkan anggota khusus dari kelas gajah adalah belalai dan gading

Bentuk umum penulisan pewarisan (inheritance) :

```
Class nama_kelas_turun : penentu_pewarisan nama_kelas_dasar
{
    /*code untuk derived class*/
}
```

Tabel penentu warisan

penentu pewarisan	akses modifier di kelas dasar	akses modifier pada derived class
private	private protected public	tidak diwariskan private private
protected	private protected public	tidak diwariskan protected protected
public	private protected	tidak diwariskan protected

	public	public

Contoh kode program pewarisan (inheritance)

```
#include<iostream>
using namespace std;

class induk{
      int x;
      public:
            void SetX(int XX){
                  x = XX;
            }
            int GetX(){
                  return x;
            }
};

class turunan: public induk{
      int y;
      public:
            void SetY(int YY){
                  y = YY;
            }
            int GetY(){
                  return y;
            }
};

int main()
{
      induk A;

      A.SetX(12);

      cout<<"nilai z yang dipanggil dari kelas induk : ";
      cout<<A.GetX()<<endl;
      cout<<"\n";

      turunan B;

      B.SetX(40);

      cout<<"nilai Y yang terdapat pada kelas turunan : ";
      cout<<B.GetY()<<endl;
      cout<<"\n";

      B.SetX(35);

      cout<<"nilai X yang dipanggil dari kelas turunan : ";
      cout<<B.GetX()<<endl;

      return 0;
}
```

Output yang dihasilkan ;

```
nilai z yang dipanggil dari kelas induk : 12

nilai Y yang terdapat pada kelas turunan : 0

nilai X yang dipanggil dari kelas turunan : 35
```

Contoh lain kode program pewarisan

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

class kendaraan
{
private:
char nama [15];
public:
          kendaraan (char*nama_kendaraan=" xxx")
          {strcpy(nama,nama_kendaraan);
cout<<"hidupkan mesin kendaraan.......... !"<<endl;
}
~kendaraan ()
{    cout<<"matikan  mesin kendaraan ......... !" <<endl;}
void info_kendaraan ()
          {cout<<nama<<"sedang berjalan..... !" <<endl;}
};
class truk:public kendaraan
{
public:
truk (char*nama_truk) : kendaraan(nama_truk)
          {cout<<"hidupkan mesin truk.....!"<<endl;}
  ~truk()
   {cout<<"matikan mesin truk.. !"<<endl;}
};

main()
{
truk fuso("truk fuso");
fuso.info_kendaraan();
cout<<"akhir main()..... ."<<endl;

getch();
}
```

Output yang dihasilkan

```
hidupkan mesin kendaraan.......... !
hidupkan mesin truk.....!
truk fusosedang berjalan..... !
akhir main()..... .
```

Contoh lain kode program pewarisan

```
#include<iostream>
using namespace std;

class makhluk
{
      public:
      void berkembang();
};
class hewan : public makhluk
{
      public:
      void bergerak();
};
class kuda : public hewan
{
      public:
            void berlari();
};
main()
{
      makhluk mk; hewan hw; kuda kd;
      cout<<endl<<"sifat - sifat dari makhluk adalah :"<<endl;
      mk.berkembang();
      cout<<endl<<"sifat - sifat dari kuda adalah : "<<endl;
      mk.berkembang();
      hw.bergerak();
      cout<<endl<<"sifat - sifat dari kuda adalah :"<<endl;
      mk.berkembang();
      hw.bergerak();
      kd.berlari();
}
void makhluk::berkembang()
{
      cout<<"berkembang biak"<<endl;
}
void hewan::bergerak()
{
      cout<<"bergerak berpindah tempat"<<endl;
}
void kuda::berlari()
{
      cout<<"berlari sangat kencang seperti angin"<<endl;
}
```

Output yang dihasilkan

```
sifat - sifat dari makhluk adalah :
berkembang biak

sifat - sifat dari kuda adalah :
berkembang biak
bergerak berpindah tempat

sifat - sifat dari kuda adalah :
berkembang biak
bergerak berpindah tempat
berlari sangat kencang seperti angin
```

2. **Multiple inheritance**

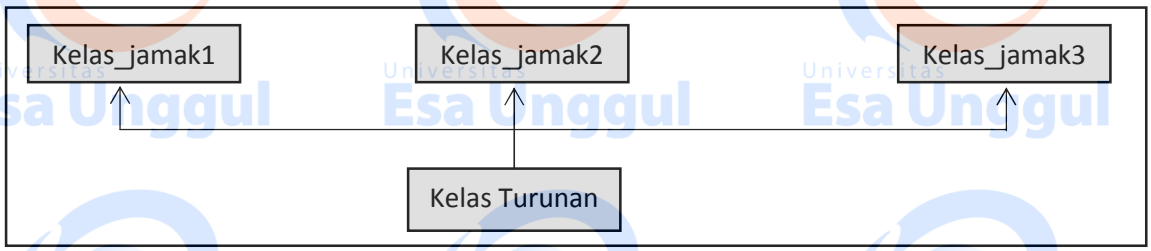
Multiple inheritance adalah pewarisan dimana satu kelas diturunkan lebih dari satu kelas yang berbeda. dalam pewarisan ini jumlah kelas dasarnya lebih dari satu kelas.

Contoh ilustrasi :

Jika memiliki kelas A dan kelas B yang masing – masing berdiri sendiri. Kemudian ingin membuat kelas C yang merupakan turunan dari kelas A dan B, dengan demikian kelas C tentu akan mewarisi sifat – sifat yang terdapat pada kelas A dan kelas B.

Dan sebenarnya kita bisa juga melakukan dengan proses yang berulang, yaitu dengan menurunkan kelas A menjadi kelas B, kemudian baru menurunkan kelas B menjadi kelas C. tetapi hal ini akan memakan waktu yang cukup lama dibandingkan menurunkannya dari beberapa induk secara langsung.

Berikut gambar ilustrasi



Bentuk umum penulisan multiple inheritance

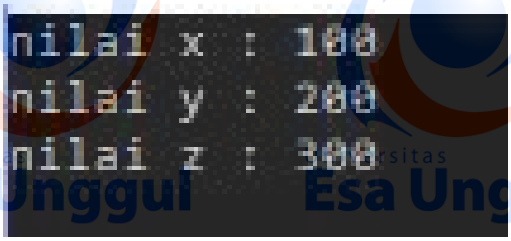
```
Class nama_kelas_turunan:
      penentu_pewarisan nama_kelas_dasar1;
      penentu_pewarisan nama_kelas_dasar2;
      penentu_pewarisan nama_kelas_dasar3;
{
      /*code untuk derived class*/
}
```

Contoh kode program multiple inheritance

```
#include<iostream>
using namespace std;

//kelas induk 1
class induk1{
      int x;
      public:
            void SetX(int XX){
                  x = XX;
            }
            int GetX(){
                  return x;
            }
};
```

```
//membuat kelas turunan yg merupakan trn
induk1 & induk2
class turunan: public induk1, public induk2{
      int z;
      public:
            void SetZ(int zz){
                  z = zz;
            }
            int GetZ(){
                  return z;
            }
};

//fungsi utama
int main()
```

Output yang dihasilkan



3. Virtual multiple inheritance

Virtual multiple inheritance merupakan pewarisan yang mana kelas dasarnya lebih dari satu dan beberapa di antara kelas dasar tersebut merupakan kelas turunan dari kelas dasar yang sama. Mekanisme pewarisan sifat suatu kelas dasar kepada kelas turunan sama dengan pewarisan yang lain.:

Berikut bentuk penulisan dari virtual multiple inheritance

```
Class A
{
   . . .
};
Class B: virtual public A
{
   . . .
};
Class C: virtual public A
{
   . . .
};
Class D: public B, public C
{
   . . . };
```

Kelas D merupakan turunan dari kelas B dan C, sedangkan kelas B dan C merupakan kelas turunan dari kelas dasar yang sama yaitu kelas A. biar berjalan pewarisan dari kelas A kepada kelas B maupun C harus secara virtual. Kelas virtual A pertama menurunkan kelas B, sedangkan kelas virtual A kedua menurukan kelas C

Berikut contoh kode program dalam pewarisan :

```
#include<iostream.h>
class hewan{
  public:
  void bergerak(){
  cout<<"#        bergerak        berpindah
tempat"<<endl;
  }
};
class kuda:virtual public hewan {
```

```
class pegasus:public kuda, public burung{
   public:
   void lariterbang(){
   cout<<"#   bersayap,   lari   dan   dapat   terbang   ke
angkasa"<<endl;
   }
};
main()
{
   pegasus pg;
   cout<<">>sifat dari pegasus<< "<<endl;
   cout<<"======================"<<endl;
   pg.bergerak();
   pg.berlari();
   pg.terbang();
   pg.lariterbang();
}
```

Output yang dihasilkan:

```
>>sifat dari pegasus<<
======================
# bergerak berpindah tempat
# berlarinya sangat cepat
# terbang menembus awan
# bersayap, lari dan dapat terbang ke angkasa
```

Membuat constructor pada proses penurunan kelas

Penulisan umum pada pembuatan constructor pada kelas adalah sebagai berikut :

```
class INDUK{
      int x;
public:
//constructor kelas INDUK
INDUK(int XX){
X = XX;
}
      // fungsi lain.. };
```

Apa bila ingin melakukan inisialisasi terhadap nilai X diatas dari sebuah kelas turunan, maka dapat memanggul constructor kelas INDUK pada saat membuat constructor kelas TURUNAN. Berikut bentuk umum penulisan constructor pada kelas turunan

```
Class TURUNAN:public INDUK(
      Int Y;
Public:
//constructor kelas TURUNAN
      TURUNAN(int XX, int YY){
      INDUK(XX);  //memanggil constructor kelas INDUK
      Y = YY;     //set nilai Y
}
      Fungsi lain…
};
```

Contoh kode program constructor

```
#include<iostream>
using namespace std;

class INDUK{
      int x;
      public:
      //constructor kelas INDUK
            INDUK(int XX){
                  x = XX;
                  cout<<"constructor kelas INDUK"<<endl;
            }
      //membuat fungsi GetX
            int GETX(){
                  return x;
            }
};
class TURUNAN:public INDUK{
      int y;
      public:
//constructor pada kelas TURUNAN
TURUNAN(int XX, int YY): INDUK(XX){

      cout<<"constructor kelas TURUNAN"<<endl;
            }
            //membuat fungsi GETY
            int GETY(){
                  return y;
            }
};
y = YY;
                  cout<<"constructor kelas TURUNAN"<<endl;
            }
      //membuat fungsi GETY
            int GETY(){
                  return y;
            }
};
//fungsi utama
int main(){
      //melakukan instansiasi terhadap kelas turunan
      TURUNAN A(10, 20);
      //melakukan pemanggilan fungsi melalui objek A
      cout<<"nilai x : "<<A.GETX()<<endl;
      cout<<"nilai y : "<<A.GETY()<<endl;

      return 0;
}
```

Output yang dihasilkan:

2. Enkapsulasi

Enkapsulasi adalah proses atau cara menyembunyikan informasi dari suatu class program, enkapsulasi akan melindungi program dari intervensi dari program lain yang dapat mempengaruhinya. Hal ini sangat membantu untuk menjaga keutuhan program.

Jenis enkapsulasi

1. Private : artinya semua yang berada di dalam private mulai dari variabel dan lain-lain tidak dapat diakses secara bebas, dapat diartikan semua yang berada dalam private sudah tersembunyi
2. Public : artinya semua yang berada di dalam public mulai dari variabel, class dan lain-lain dapat diakses secara bebas, artinya siapa saja dapat mengaksesnya.

Contoh kode progam enkapsulasi

```
#include<conio.h>
#include<iostream>
using namespace std;

class adder{
      public:
            //constructor
            adder(int i=0){
                  total = i;
            }
            //interface to outside world
            void addNum(int number){
                  total += number;
            }
            //interface to outside world
            int GetTotal(){
                  return total;
            }
            private:
            //hidden data form outside world
            int total;
};
int main(){
      adder a;

      a.addNum(10);
      a.addNum(20);
      a.addNum(30);

      cout<<"total "<<a.GetTotal()<<endl;
```

Output yang dihasilkan :

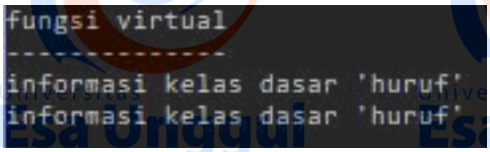
3. Polimorfisme

Polimorfisme merupakan ciri OOP (*object oriented programming*) yang keempat setelah abstraksi, pembungkus atau pengkapsulan dan pewarisan (inheritance).

Contoh kode program polimorfisme

```
#include<conio.h>
#include<iostream>
using namespace std;
//kelas dasar
class huruf{
	public:
		void info(){
			cout<<"informasi kelas dasar 'huruf'"<<endl;
		}
};
//kelas turunan
class A:public huruf{
	public:
		void info(){
			cout<<"informasi kelas turunan 'A'"<<endl;
		}
};
//kelas turunan
class B:public huruf{
	public:
		void info(){
			cout<<"informasi kelas turunan 'B'"<<endl;
		}
};

main(){
	//deklarasi objek
	huruf *obj_huruf; //pointer ke objek berkelas huruf
	A obj_a;
	B obj_b;

	cout<<"fungsi virtual"<<endl;
	cout<<"--------------"<<endl;

	//menunjuk ke objek dari kelas A
	obj_huruf = &obj_a;

	obj_huruf->info();

	obj_huruf = &obj_b;

	obj_huruf->info();

	_getche();
	return 0;
}
```

Output yang dihasilkan



Tetapi, hasil yang diinginkan :

```
Fungsi virtual
--------------
Informasi kelas dasar ‘A’
Informasi kelas dasar ‘B’
```

Untuk mendapatkan hasil yang diinginkan, maka ditambah dengan **fungsi virtual**. **Fungsi virtual** adalah fungsi yang mendukung adanya polymorphic function yang artinya fungsi tersebut dapat didefinisikan ulang pada kelas – kelas turunannya. Fungsi virtual ini biasanya terdapat pada kelas – kelas dasar. Tetapi dapat mendeklarasikan fungsi virtual pada kelas – kelas turunan yang akan dijadikan sebagai kelas dasar bagi kelas-kelas lainnya. Dalam C++ untuk mendefinisikan fungsi sebagai fungsi virtual adalah bagian menggunakan kata kunci virtual, dengan menempatkannya di depan pendeklarasian fungsi. Pendefinisian fungsi virtual yang terdapat pada kelas dasar .

berikut contoh kode program

```
#include<conio.h>
#include<iostream>
using namespace std;
//kelas dasar
class huruf{
	public:
		virtualvoid info(){
			cout<<"informasi kelas dasar 'huruf'"<<endl;
		}
};
//kelas turunan
class A:public huruf{
	public:
		void info(){
			cout<<"informasi kelas turunan 'A'"<<endl;
		}
};
//kelas turunan
class B:public huruf{
	public:
		void info(){
			cout<<"informasi kelas turunan 'B'"<<endl;
		}
};

main(){
	//deklarasi objek
	huruf *obj_huruf; //pointer ke objek berkelas huruf
	A obj_a;
	B obj_b;

	cout<<"fungsi virtual"<<endl;
	cout<<"--------------"<<endl;

	//menunjuk ke objek dari kelas A
	obj_huruf = &obj_a;

	obj_huruf->info();

	obj_huruf = &obj_b;
```

Output yang dihasilkan

```
fungsi virtual
--------------
informasi kelas turunan 'A'
informasi kelas turunan 'B'
```

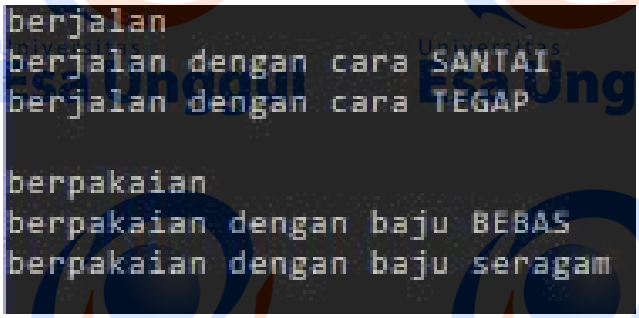
Fungsi virtual dapat didefinisikan ulang pada kelas turunannya, hal ini tentu memungkinkan untuk terjadinya proses pembaharuan dari definisi suatu fungsi dalam OOP proses pembaharuan tersebut disebut **override.**

Berikut contoh kode program

```
#include<iostream>
using namespace std;
class manusia{
      char* nama;
      int tinggi;
      int berat;
      public:
            void SetNama(char* N){
                  nama = N;
            }
            void SetTinggi(int T){
                  tinggi = T;
            }
            void SetBerat(int B){
                  berat = B;
            }
            char* GetNama(){
                  return nama;
            }
            int GetTinggi(){
                  return tinggi;
            }
            int GetBerat(){
                  return berat;
            }
            //membuat fungsi virtual
            virtual void berjalan(){
                  cout<<"berjalan"<<endl;
            }
            virtual void berpakaian(){
                  cout<<"berpakaian"<<endl;
            }
};
      class mahasiswa:public manusia{
      char* universitas;
      char* jurusan;
      int semester;
      public:
            void SetUniversitas(char* U){
```

```
            int GetSemester(){
                  return semester;
            }
            // melakukan overide terhadap fungsi berjalan
            void berjalan(){
                  cout<<"berjalan dengan cara SANTAI"<<endl;
            }
            //melakukan override terhadap fungsi berpakaian
            void berpakaian(){
                  cout<<"berpakaian dengan baju BEBAS"<<endl;
            }
};
class tentara:public manusia{
      char* pangkat;
      char* kesatuan;
      public:
            void SetPangkat(char* pkt){
                  pangkat = pkt;
            }
            void SetKesatuan(char* kstn){
                  kesatuan = kstn;
            }
            char* GetPangkat(){
                  return pangkat;
            }
            char* GetKesatuan(){
                  return kesatuan;
            }
            //melakukan override terhadap fungsi berjalan
            void berjalan(){
                  cout<<"berjalan dengan cara TEGAP"<<endl;
            }
            //melakukan override terhadap fungsi berpakaian
            void berpakaian(){
                  cout<<"berpakaian dengan baju seragam"<<endl;
            }
};
//fungsi utama
int main(){
      //melakukan instanisasi terhadap kelas manusia
      manusia m;
      //melakukan instanisasi terhadap kelas mahasiswa
      mahasiswa mhs;
      //melakukan instanisasi terhadap kelas tentara
      tentara ttr;
```

Output yang dihasilkan :



Berdasarkan hasil kode program dapat dilihat bahwa dengan mendefinisikan fungsi **berjalan()** sebagai fungsi virtual, dapat melakukan override terhadap fungsi tersebut. Di atas, fungsi **berjalan()** akan diimplementasikan berdasarkan kelas masing – masing, begitu juga dengan fungsi berpakaian().

Latihan :

1. Tuliskan sebuah program dengan hasil output yang ada dibawah ini !

```
Masukkan jumlah bus:12 <diinput>
Masukkan jumlah penumpang:120 <diinput>

Setiap bus memuat sejumlah 10 penumpang.
Akhir program.

Masukkan jumlah bus:-1 <diinput>

Tidak bisa memiliki nilai negatif pada Bus, tidak riil.
Akhir program.

Masukkan jumlah bus:12 <diinput>
Masukkan jumlah penumpang:-5 <diinput>

Tidak bisa memiliki nilai negatif pada Penumpang, tidak riil.
Akhir program.
```

2. Tuliskan sebuah program dengan hasil output yang ada di bawah ini !

```
Layar masukan

Masukkan pembilang:5<input>
Masukkan penyebut:20<input>

Layar output
5/20 = 0.25<tampi>
```

```
Layar masukan

Masukkan pembilang:12<input>
Masukkan penyebut:0<input>

Layar output
Error : pembagian nol! <tampi>
Program dibatalkan<tampil>
```