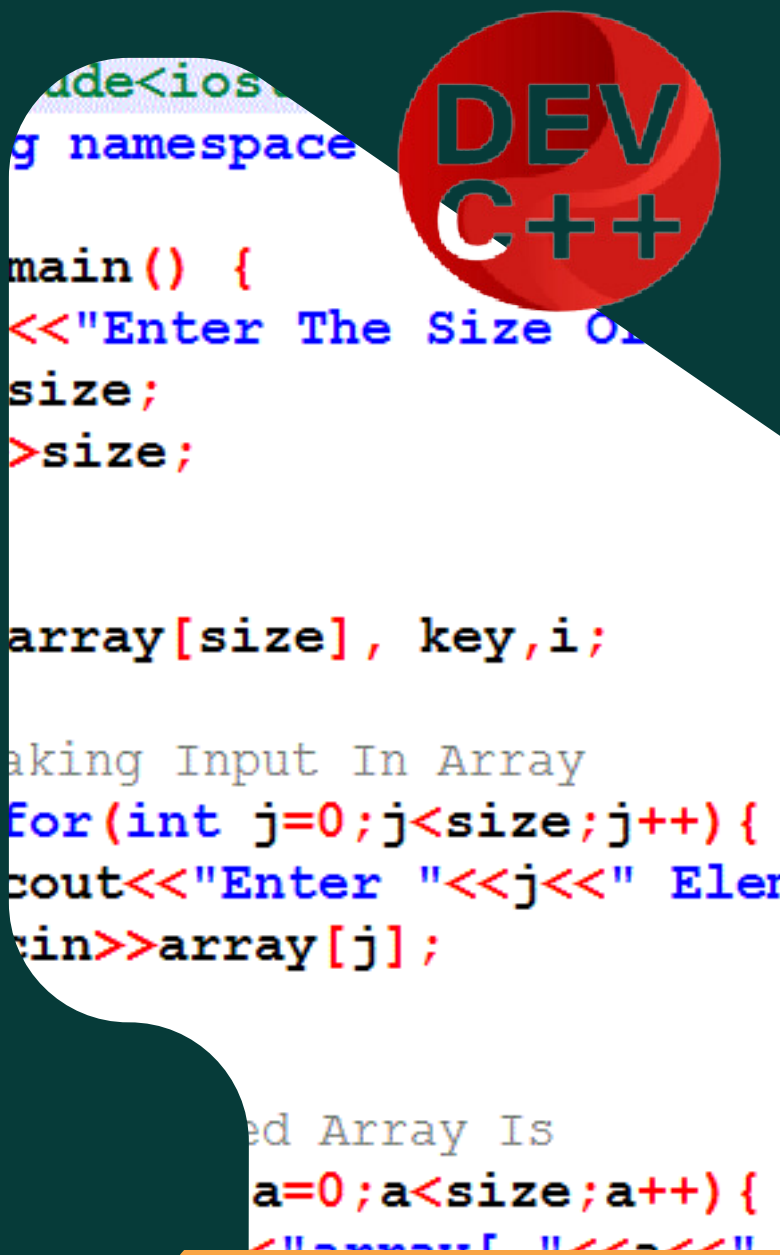


EKSPRES

Mudah | Cepat | Fokus

ASAS
Pengaturcaraan C++

Mudah digunakan untuk belajar dan membina kod pengaturcaraan menggunakan Dev C++

Ayu Hasnidah, Syarifah Fauziah

Edisi Pertama

ASAS Pengaturcaraan C++

https://kkpayabesar.mypolycc.edu.my
elearningkkpb@gmail.com

HAKCIPTA

Penulis:
Ayu Hasnidah binti Zainudin
Syarifah Fauziah binti Syed Mohamed

eISBN: 978-629-99371-0-4

Diterbitkan oleh:
Unit Pembelajaran Digital
Kolej Komuniti Paya Besar
Kementerian Pengajian Tinggi Malaysia
26300 Gambang, Pahang

Email : elearningkkpb@gmail.com
Website : https://kkpayabesar.mypolycc.edu.my

SINOPSIS

Buku yang sangat tepat untuk pembaca yang sedang atau akan mempelajari asas pengaturcaraan komputer. Buku ini mengandungi pembelajaran mengenai logik dan algoritma untuk menyelesaikan berbagai masalah yang dilaksanakan oleh komputer. Buku ini meliputi: Pengenalan kepada Asas Pengaturcaraan, Elemen Asas Pengaturcaraan, Struktur Kawalan Aturcara, Tatasusunan dan Penunjuk serta Fungsi. Setiap bahan dalam buku ini dilengkapi dengan berbagai contoh penyelesaian masalah beserta kod aturcara dalam bahasa C++. Bahasa yang digunakan dalam buku ini dirancang agar pembaca dapat mempelajarinya secara kendiri tanpa melibatkan bantuan orang lain. Diharapkan buku ini mampu memberikan ilmu kepada pembaca supaya dapat mempelajari asas pengaturcaraan dengan mengikuti aturan yang betul.

ISI KANDUNGAN

QUICK TIPS

Klik pada nombor mukasurat untuk terus ke halaman yang dikehendaki.

ISI KANDUNGAN

Klik pada nombor mukasurat untuk terus ke halaman yang dikehendaki.

TOPIK 1.0

1.0 Pengenalan kepada Asas Pengaturcaraan

Sejarah bahasa pengaturcaraan

Sejarah Ringkas C++

C++ adalah bahasa yang berasal dari C tetapi diperkemaskan lagi dengan keupayaan penggunaan teknologi berorientasikan objek.

Bahasa C telah dipopularkan oleh Kernighan dan Ritchie pada tahun 1978, serta telah diseragamkan oleh American National Standard Institute (ANSI) menjadi ANSI C.

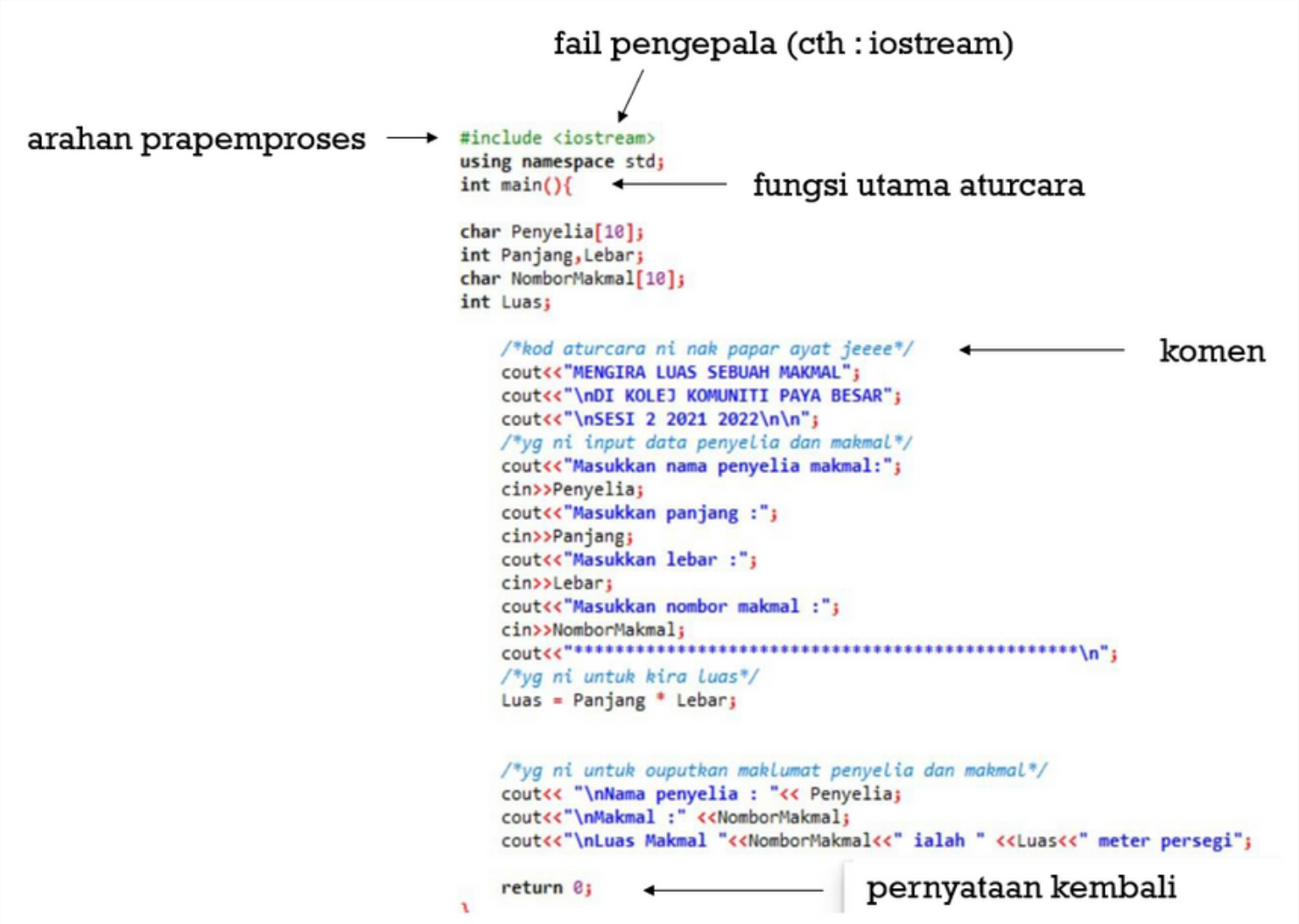
Stroustrup kemudiannya memperkenalkan bahasa C yang berorientasikan objek dan menamakannya C++. C++ mengandungi semua ciri-ciri C dengan tambahan ciri-ciri baru bertujuan untuk menghapuskan kekurangan yang ada pada bahasa C.

1.0 Pengenalan kepada Asas Pengaturcaraan

Struktur Asas Pengaturcaraan

Suatu aturcara C++ mempunyai struktur asas seperti berikut:

- Komen (*comment*)
- Arahan prapemproses (*preprocessor directives*)
- Fail pengepala (*header file*)
- Fungsi utama (*main functio*n)
- Penyataan kembali (*return statement*)




```
#include <iostream>
using namespace std;
int main(){

char Penyelia[10];
int Panjang,Lebar;
char NomborMakmal[10];
int Luas;

    /*kod aturcara ni nak papar ayat jeeee*/
    cout<<"MENGIRA LUAS SEBUAH MAKMAL";
    cout<<"\nDI KOLEJ KOMUNITI PAYA BESAR";
    cout<<"\nSESI 2 2021 2022\n\n";
    /*yg ni input data penyelia dan makmal*/
    cout<<"Masukkan nama penyelia makmal:";
    cin>>Penyelia;
    cout<<"Masukkan panjang :";
    cin>>Panjang;
    cout<<"Masukkan lebar :";
    cin>>Lebar;
    cout<<"Masukkan nombor makmal :";
    cin>>NomborMakmal;
    cout<<"**************************************************\n";
    /*yg ni untuk kira luas*/
    Luas = Panjang * Lebar;


    /*yg ni untuk ouputkan maklumat penyelia dan makmal*/
    cout<< "\nNama penyelia : "<< Penyelia;
    cout<<"\nMakmal :" <<NomborMakmal;
    cout<<"\nLuas Makmal "<<NomborMakmal<<" ialah " <<Luas<<" meter persegi";

    return 0;
}
```

1.0 Pengenalan kepada Asas Pengaturcaraan

Pengecam & jenis data

- Nama yang diberikan untuk pelbagai unsur di dalam aturcara.
- Cth: nama pembolehubah

```
char Penyelia[10];
int Panjang,Lebar;
char NomborMakmal[10];
int Luas;
```

- Perkataan ditakrifkan oleh pengguna
- Nama pengecam mesti unik

Peraturan untuk menamakan pengecam :

- Aksara pertama mesti huruf atau garis bawah
- Boleh terdiri daripada A-Z, a-z, 0-9 dan garis bawah _
- Tiada had panjang tetapi pengkompil akan mengenali 32 aksara pertama
- Tiada tempat kosong
- Kata kunci tidak boleh digunakan
- *Case sensitive*, cth: Jumlah tidak sama dengan jumlah

Peraturan	Contoh
Nombor dan huruf boleh digabungkan, tetapi mesti bermula dengan huruf	**nombor1, markah2**
Mesti bermula dengan huruf atau underscore	**ujian1, _kuiz**
Boleh terdiri daripada huruf besar, huruf kecil dan aksara '*underscore*'	**Markah_Ujian, nom_2**
Tidak boleh mengandungi operator aritmetik	**Jumlah*bersih, nom1+nom2**
Tidak boleh mengandungi '*punctuation marks*'	**#@$%&**
Tidak boleh mengandungi kata simpanan C	**struct; cout;**
Tidak boleh mengandungi ruang kosong	**Markah Ujian, hasil tambah**
Pengecam adalah kes sensitif	**Tax** tidak sama dengan **tax**

1.0 Pengenalan kepada Asas Pengaturcaraan

Pembolehubah adalah pengecam yang boleh mengubah nilai semasa pelaksanaan program.
Ia merupakan lokasi ingatan yang mana datanya boleh berubah.
Jenis-jenis data:

i. int - nombor integer : 1, 2, 4,….
ii. char - aksara : ‘a’, ‘b’, …
iii. float, double - nombor perpuluhan: 2.5, 4.96

Nilai pembolehubah boleh berubah semasa pelaksanaan aturcara
Pengisytiharan sesuatu pembolehubah melibatkan jenis data dan nama.
Contoh:

```
int nombor ;
jenis data      nama pembolehubah
```

Pemalar ialah suatu entiti yang terkandung di dalam aturcara dengan nilai tetap iaitu nilainya tidak akan berubah.
Sekiranya nilai pemalar diubah maka akan terdapat ralat dalam aturcara tersebut.
Pengisytiharan nilai pemalar dimulakan dengan perkataan **const**
Contoh:

```
const int saiz = 5 ;
```

1.0 Pengenalan kepada Asas Pengaturcaraan

Jenis-jenis data ialah seperti char, int, double, float, bool, string

i. **char**

Ø Juga dikenali sebagai string

Ø Contoh:

- Nombor: 0 - 9
- Huruf besar dan kecil: a - z dan A - Z
- Ruang kosong
- Aksara khas: , . ; ? “ / ( ) [ ] { } * & % ^ < > dan lain-lain

katakunci: char

Contoh kod:

```
char huruf;
huruf = 'U';
```

Contoh nilai:

‘B’ ‘d’ ‘4’ ‘?’ ‘*’

ii.**String**

ØJujukan aksara yang diwakili dalam *double quotes* (“ “)

ØContoh :

“Hello” “Year 2000” “1234”

ØString kosong(*null string*) tidak mengandungi sebarang aksara dan ditulis sebagai “ ”

1.0 Pengenalan kepada Asas Pengaturcaraan

iii.**int**

Ø Digunakan untuk isytihar pembolehubah jenis integer

Ø Semua nombor termasuk +ve dan -ve

Ø Katakunci: int

Ø Contoh :

```
int nombor;
nombor = 12;
```

Ø Contoh nilai nombor :

4578 -4578 0

iv. **double**

Ø Digunakan untuk isytihar pembolehubah jenis perpuluhan

Ø Nombor eksponen,+ve dan -ve

Ø Katakunci: double

Ø Contoh :

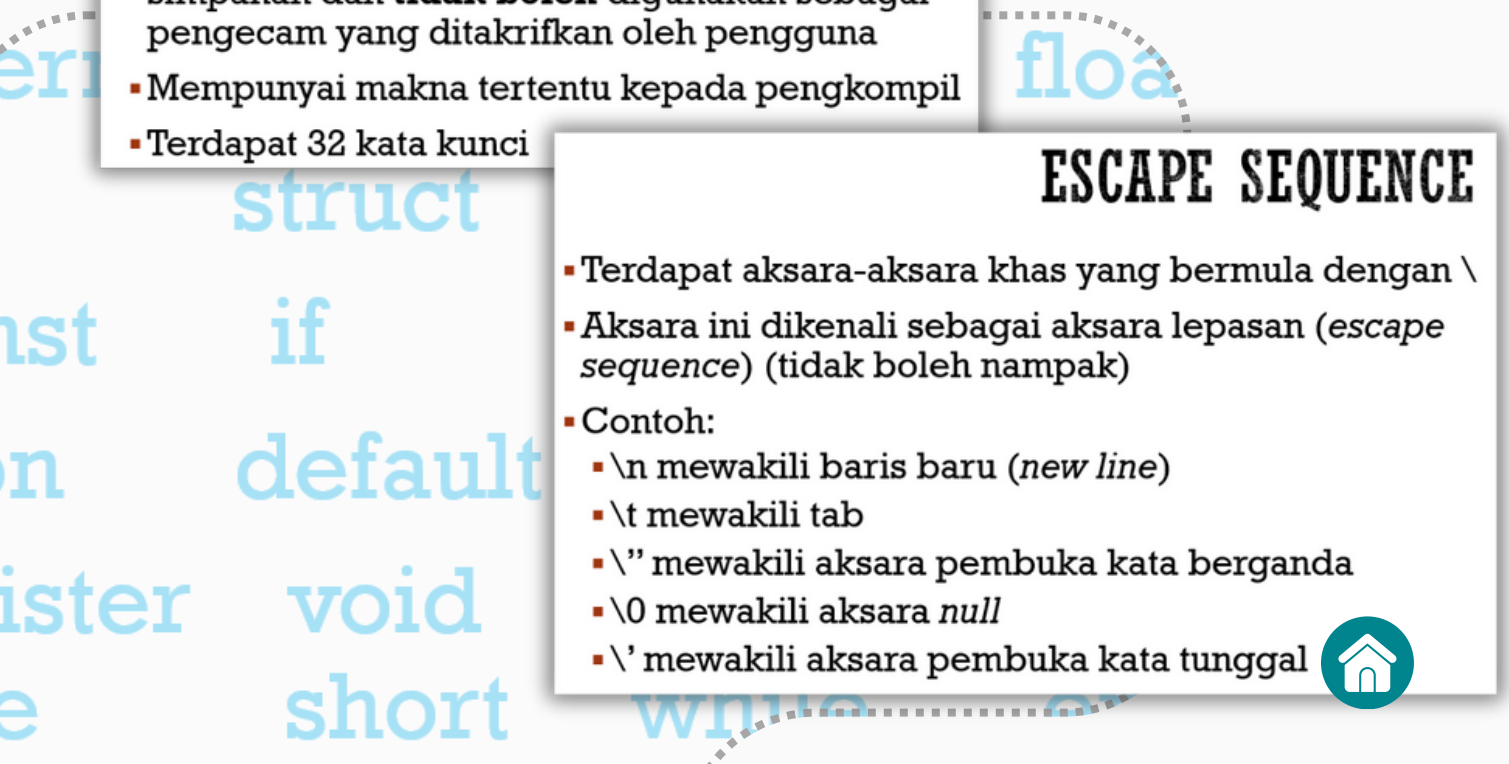
```
double purata;
purata = 12.4;
```

KATA KUNCI

- Kata kekunci merupakan perkataan-perkataan simpanan dan **tidak boleh** digunakan sebagai pengecam yang ditakrifkan oleh pengguna
- Mempunyai makna tertentu kepada pengkompil
- Terdapat 32 kata kunci

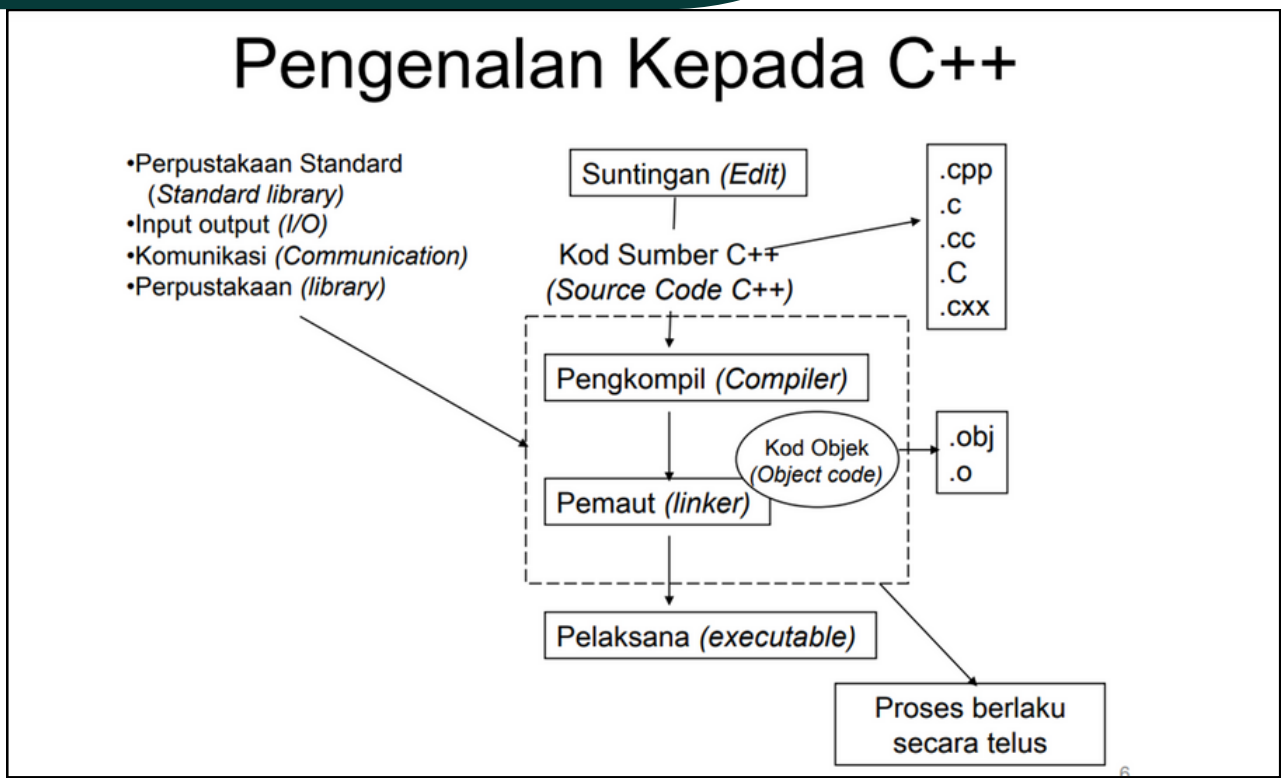
ESCAPE SEQUENCE

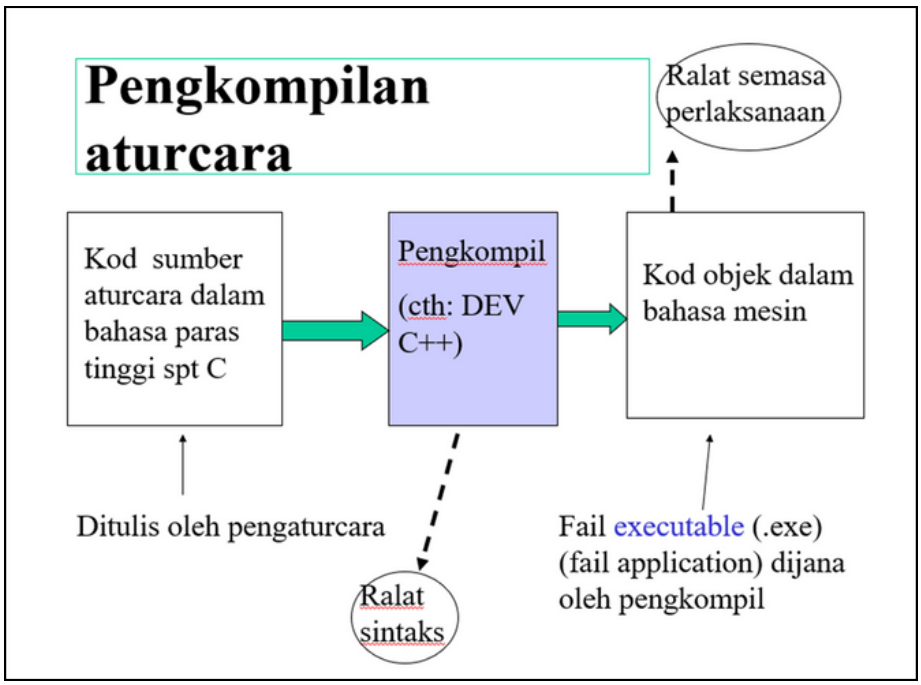
- Terdapat aksara-aksara khas yang bermula dengan \
- Aksara ini dikenali sebagai aksara lepasan (*escape sequence*) (tidak boleh nampak)
- Contoh:
  - \n mewakili baris baru (*new line*)
  - \t mewakili tab
  - \" mewakili aksara pembuka kata berganda
  - \0 mewakili aksara *null*
  - \' mewakili aksara pembuka kata tunggal



1.0 Pengenalan kepada Asas Pengaturcaraan

Penggunaan perisian DEV C++





1.0 Pengenalan kepada Asas Pengaturcaraan

Compile, debug dan ralat dalam pengaturcaraan

Kod sumber (*source code*)

- Adalah set arahan dan pernyataan yang ditulis oleh seorang pengaturcara menggunakan bahasa pengaturcaraan komputer.
- Kod ini kemudian diterjemahkan ke dalam bahasa mesin oleh pengkompil.
- Kod diterjemahkan disebut kod objek.

Pengkompil (*Compiler*)

- Adalah program perisian yang mengubah kod sumber peringkat tinggi yang ditulis oleh pengaturcara dalam bahasa pengaturcaraan peringkat tinggi ke kod objek tahap rendah (kod binari) dalam bahasa mesin, yang dapat difahami oleh pemproses.
- Proses menukarkan pengaturcaraan peringkat tinggi ke dalam bahasa mesin dikenali sebagai kompilasi.
- Contoh : DEV C++

Pemaut (*Linker*)

- Menukar kod objek ke bentuk fail jenis .exe
- Menggabungkan kesemua bahagian yang diperlukan (seperti fail perpustakaan) oleh aturcara untuk menghasilkan kod perlaksanaan terakhir dalam bentuk fail .exe yang dapat dilarikan(*run*).

Fail pelaksana (*Executable file*) .exe

- Fail yang dijana oleh pengkompil
- Merupakan gabungan fail-fail objek

1.0 Pengenalan kepada Asas Pengaturcaraan

RALAT

Ralat sintaks	Ralat masa larian	Ralat logik
1. Kesilapan bahasa pengaturcaraan. 2. Ada paparan mesej ralat.	1. Atur cara terhenti tiba-tiba. 2. Disebabkan memori yang tidak mencukupi. 3. Disebabkan jangkitan virus atau perisian hasad.	1. Output yang tidak diingini. 2. Kesilapan semasa pembinaan algoritma. 3. Sebab utama ialah kecuaian manusia. 4. Tiada paparan mesej ralat.

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main (){
6       int nombor1, nombor2, jumlah;
7
8       cout<<"MENGIRA HASIL TAMBAH 2 NOMBOR";
9
10      cout<<"\n\nSila masukkan nombor 1 : ";
11      cin>>nombor;
12
13      cout<<"\nSila masukkan nombor 2 : ";
14      cin>>nombor2;
15
16      jumlah = nombor1 + nombor2;
17
18      cout<<"\nJumlah hasil tambah 2 nombor adalah :"<<jumlah;
19
20
21  }
22
23
```

!!! Contoh **ralat sintaks** pada nama pembolehubah - nombor

1.0 Pengenalan kepada Asas Pengaturcaraan

Compile and run

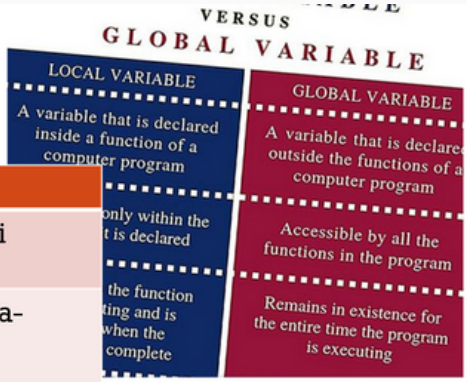
TOPIK 2.0

2.0 Elemen Asas Pengaturcaraan

Pengisytiharan pembolehubah & Operator/ pengendali

Terdapat 2 jenis pembolehubah :

1) Pembolehubah tempatan (*local variable*)
2) Pembolehubah global (*global variable*)



LOCAL VS GLOBAL

Local Variable	Global Variable
diisytiharkan di dalam fungsi	diisytiharkan di luar fungsi dalam program
hanya boleh diakses oleh fungsi itu. Fungsi lain dari program yang sama tidak dapat mengakses pembolehubah tersebut	ia boleh diakses oleh mana-mana pernyataan dalam keseluruhan program.
wujud sehingga fungsi dijalankan. Selepas menyelesaikan pelaksanaan fungsi itu, pembolehubah tempatan dimusnahkan.	wujud sehingga melengkapkan pelaksanaan keseluruhan program.

Operator atau pengendali adalah token yang digunakan untuk menghasilkan hasil pengiraan atau tindakan

Terdapat 6 jenis operator:

1) Operator aritmetik
2) Operator penokokan dan penyusutan
3) Operator hubungan
4) Operator logik
5) Operator umpukan
6) Operator input/output

2.0 Elemen Asas Pengaturcaraan

OPERATOR ARITMETIK, PENOKOKAN & PENYUSUTAN

Operator	Penerangan
-	Operasi tolak
+	Operasi tambah
*	Operasi darab
/	Operasi bahagi
%	Operasi modulus (memulangkan baki dalm bentuk integer)
--	Pengurangan
++	penambahan

Penokokan dan penyusutan

OPERATOR ARITMETIK

Operator(s)	Operation(s)	Order of evaluation (precedence)
()	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
*, / , %	Multiplication, Division , Modulus	Evaluated second. If there are several, they re evaluated left to right
+ or -	Addition or substraction	Evaluated last. If there are several, they are evaluated left to right

2.0 Elemen Asas Pengaturcaraan

OPERATOR HUBUNGAN DAN LOGIK

Operator Hubungan	Penerangan
>	Lebih besar daripada
>=	Lebih besar atau sama dengan
<	Lebih kecil daripada
<=	Lebih kecil atau sama dengan
==	Sama dengan
!=	Tidak sama dengan
Operator logik	
&&	AND
||	OR
!	NOT

OPERATOR HUBUNGAN DAN LOGIK

- && (logical AND)
  - memulangkan TRUE sekiranya kedua-dua keadaan adalah TRUE
- || (logical OR)
  - Memulangkan TRUE sekiranya salah satu daripada keadaan adalah TRUE
- ! (logical NOT, logical negation)
  - Memulangkan TRUE apabila keadaan adalah FALSE dan sebaliknya
  - Merupakan operator unari, iaitu hanya mengambil satu keadaan sahaja

2.0 Elemen Asas Pengaturcaraan

OPERATOR HUBUNGAN (*RELATIONAL*) & PERSAMAAN (*EQUALITY*)

- Berguna untuk membandingkan perhubungan antara dua penyataan.
- Operator hubungan dan persamaan
- Mempunyai keutamaan yang lebih rendah daripada operator aritmetik
- Hasilnya adalah TRUE (1) atau FALSE (0)

<	lebih kecil
<=	lebih kecil atau sama
>	lebih besar
>=	lebih besar atau sama
==	sama
!=	tak sama

OPERATOR HUBUNGAN (RELATIONAL) & PERSAMAAN (EQUALITY)

- contoh:

a = 1; b = 2; c = d = 3;

`return ( c <= a + b )`	`return (1)`
`return ( a > b )`	`return (0)`
`return ( d == a + b )`	`return (1)`

* c <= a + b adalah setara dengan c <= (a + b)

2.0 Elemen Asas Pengaturcaraan

OPERATOR HUBUNGAN (*RELATIONAL*) & PERSAMAAN (*EQUALITY*)

Expression 1	Expression 2	expression1 && expression2
false	false	false
false	true	false
true	false	false
true	true	true

Fig. 2.28 Truth table for the && (logical AND) operator.

Expression 1	Expression 2	Expression1 expression 2
false	false	false
false	true	true
true	false	true
true	true	true

Fig. 2.29 Truth table for the || (logical OR) operator.

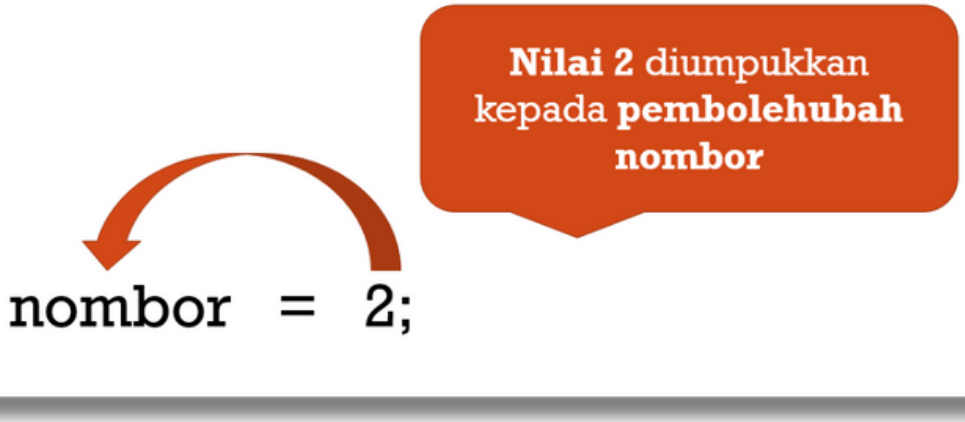
OPERATOR KAWALAN/ UMPUKAN

- Apabila sesuatu pembolehubah diberi nilai yang tertentu di awal aturcara ia dipanggil awalan atau *initialization.*

- Contoh:

```
int nombor = 1;
   float peratus = 0.5;
   char gred = 'A';
int a = x + y;
```

2.0 Elemen Asas Pengaturcaraan



OPERATOR KAWALAN/ UMPUKAN

- Operator assignment/umpukan bagi ungkapan
  c = c + 3;

  boleh diringkaskan kepada **c += 3;**

Contoh:

- d -= 4 (d = d – 4)
- e *= 5 (e = e * 5)
- f /= 3 (f = f / 3)
- g %= 9 (g = g % 9)

2.0 Elemen Asas Pengaturcaraan

OPERATOR INPUT/OUTPUT

- Set jenis dan rutin yang terdapat dalam perpustakaan piawai (*standard library*).
- Perpustakaan piawai C++ bagi I/O ialah **iostream.h**
- Pengisytiharan input/output:

```
cout    //piawai keluaran
cin     //piawai input
```

OPERATOR INPUT/OUTPUT

```
1 // Fig. 1.6: fig01_06.cpp
2 // Addition program
3 #include <iostream>
4
5 int main()
6 {
7    int integer1, integer2, sum;          // declaration
8
9    cout << "Enter first integer\n";  // prompt
10         cin >> integer1;                    // read an integer
11         cout << "Enter second integer\n"; // prompt
12         cin >> integer2;                    // read an integer
13         sum = integer1 + integer2;          // assignment of sum
14         cout << "Sum is " << sum << endl; // print sum
15
16         return 0;   // indicate that program ended successfully
17    }
```

Notice how **cin** is used to get user input.

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

Variables can be output using **cout << variableName**.

endl flushes the buffer and prints a newline.

2.0 Elemen Asas Pengaturcaraan

Penyataan input/ output

```
cout << "Welcome to C++!\n";
```

cout

Merupakan operator penyelitan jujukan (*stream insertion operator*) cout bersama **<<** digunakan untuk mencetak jujukan aksara yang terkandung di antara **"** dan **"** ke atas skrin.

Penyataan output di atas akan mencetak **Welcome to C++**

```
int myVariable;
cin >> myVariable;
```

cin

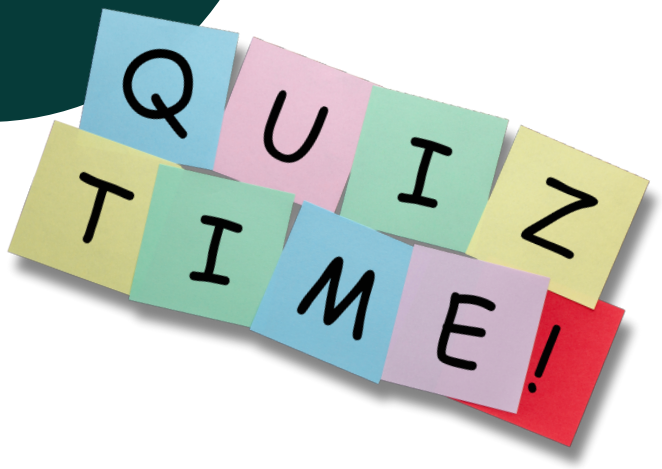
Ia akan menunggu pengguna memasukkan satu nilai input dan kemudian menyimpan nilai tersebut ke dalam pembolehubah di sebelah kanan operator **>>**

Pengguna perlu memasukkan nilai dan menekan kekunci 'Enter' sebelum data dapat dihantar ke komputer.

Penyataan input di atas menunggu pengguna memasukkan nilai, kemudian menyimpan nilai tersebut dalam **myVariable**

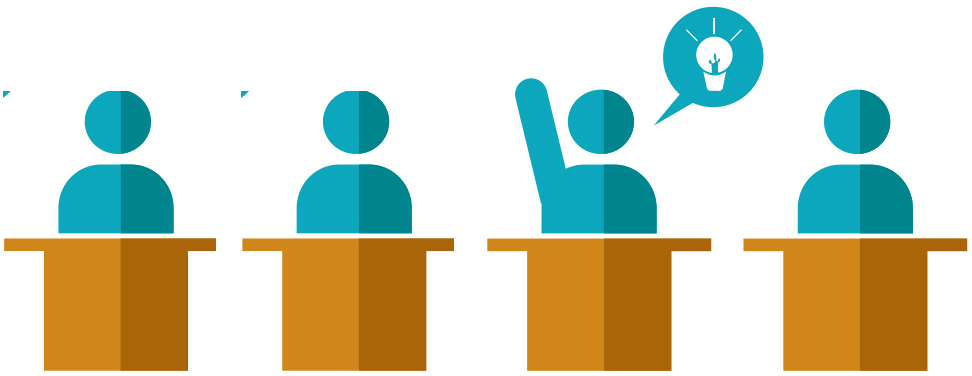
2.0 Elemen Asas Pengaturcaraan

Penilaian Kendiri



KUIZ

CLICK HERE



TOPIK 3.0

3.0 Struktur Kawalan Aturcara

Jenis Struktur Kawalan Aturcara

Terdapat 3 jenis struktur kawalan :

1) Jujukan (*Sequence*)
2) Pilihan (*Selection*)
3) Ulangan (*Repetition*)

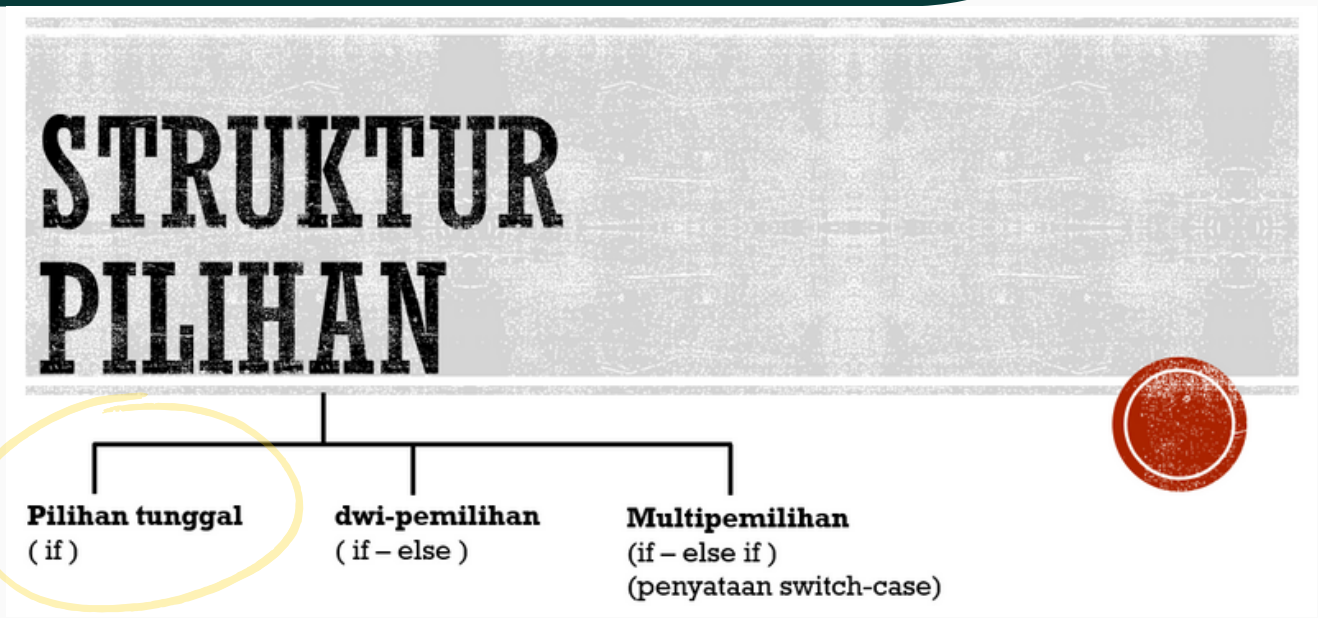
Struktur Kawalan Jujukan

- Aliran secara sehala mengikut urutan linear.
- Paling mudah difahami.
- Melaksanakan arahan baris demi baris mengikut urutan satu aliran sahaja dari atas ke bawah.
-

```
#include<iostream>

using namespace std;

int main(){
    int kuiz, ujian, amali, purata;

    cout<<"MARKAH PENILAIAN (KUIZ, UJIAN & AMALI)";

    cout<<"\n\nMasukkan markah kuiz: ";
    cin>>kuiz;

    cout<<"Masukkan markah ujian: ";
    cin>>ujian;

    cout<<"Masukkan markah amali: ";
    cin>>amali;

    purata=(kuiz+ujian+amali)/3;

    cout<<"\n****************************************\n";
    cout<<"Purata markah bagi penilaian anda ialah : " <<purata;

    return 0;

}
```

3.0 Struktur Kawalan Aturcara

Penyelesaian masalah dengan Struktur Kawalan Pilihan



Struktur Pilihan if

Jika kita inginkan satu set pernyataan dilaksanakan, pernyataan-pernyataan ini perlulah diletakkan di dalam satu blok seperti ini:

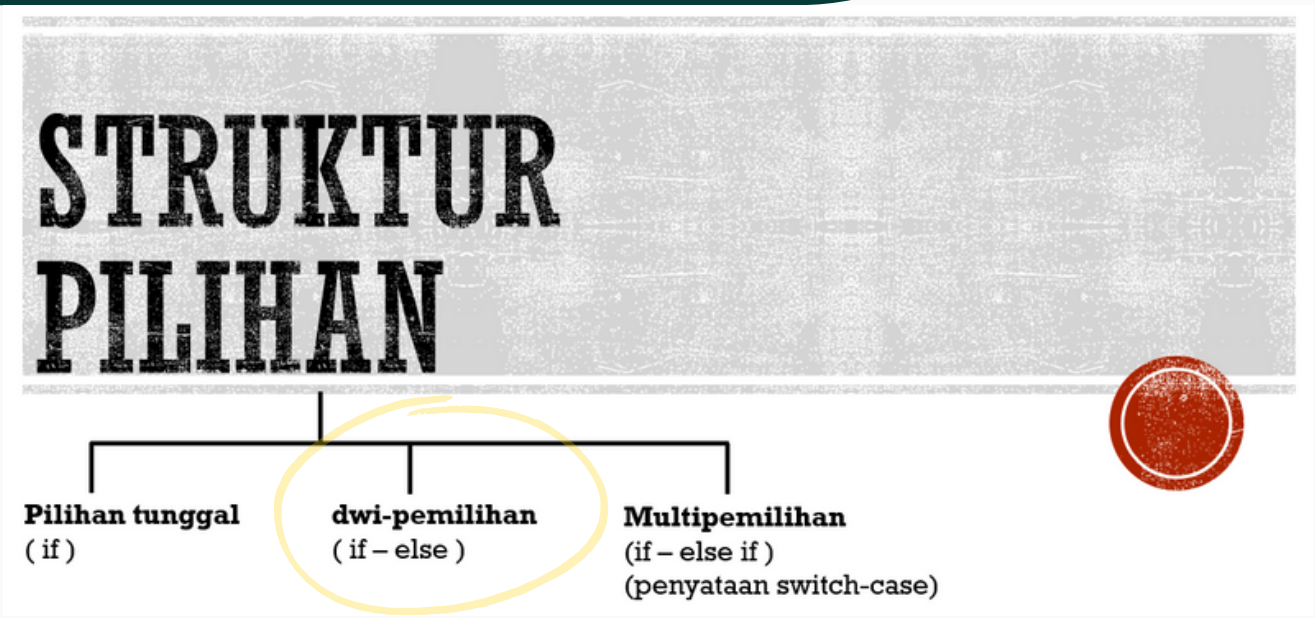
```
if ( syarat ) {
    pernyataan_1;
    pernyataan_2;
        .
        .
pernyataan_n;
}
```

Contoh aturcara

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

int main() {

 int nombor = 1;
 if (nombor == 1)
 cout<<"pembolehubah nombor bernilai 1 "<<endl;
 cout<<"Pernyataan ini di print selepas kenyataan if "<<endl;

return 0;
}
```

```
C:\Users\Pensyarah\Desktop\c++\Latihan.exe
pembolehubah nombor bernilai 1
Pernyataan ini di print selepas kenyataan if

--------------------------------
Process exited after 0.04883 seconds with return value 0
Press any key to continue . . .
```

3.0 Struktur Kawalan Aturcara



Struktur Pilihan if-else

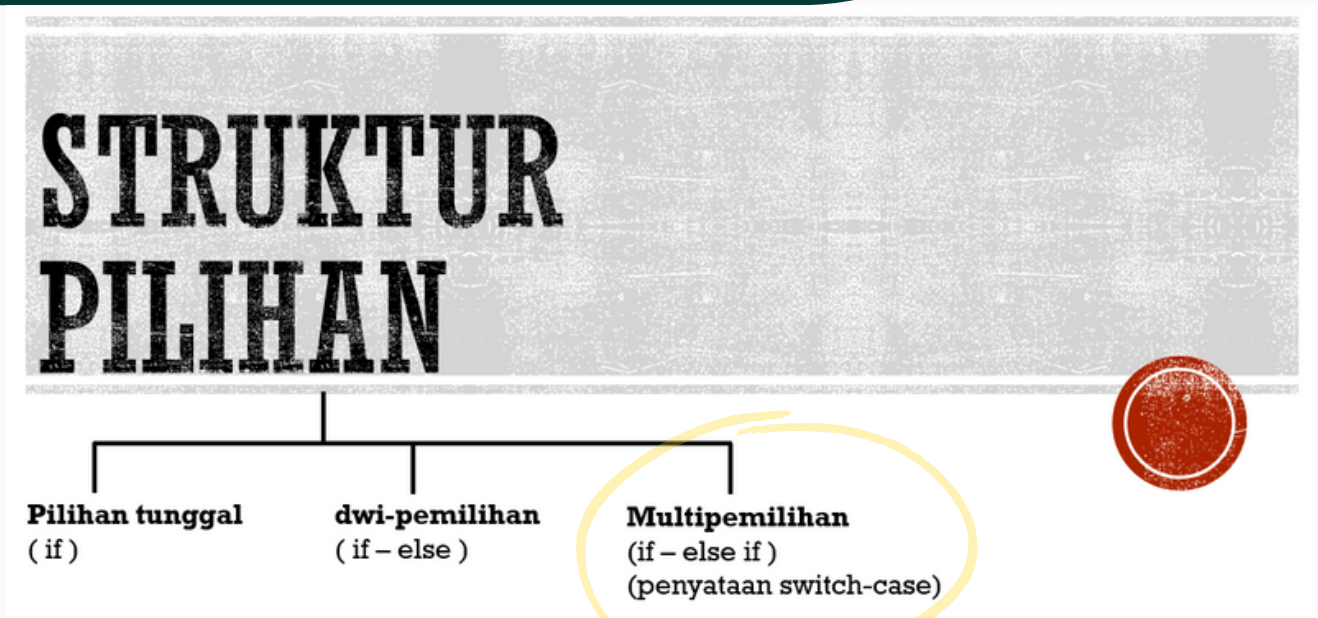
Jika syarat adalah benar, maka pernyataan_1 akan dilaksanakan, dan jika tidak pernyataan_2 akan dilaksanakan.

```
if ( syarat ) {
    pernyataan_1;
else
    pernyataan_2;
}
```

Contoh aturcara

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

int main(){

 char jawapan;

 cout<<"Apakah ibukota malaysia? "<<endl;
 cout<<"a. Jarkarta "<<endl;
 cout<<"b. Kuala Lumpur "<<endl;
 cout<<"Jawapan: ";
 cin>>jawapan;

 if (jawapan == 'b')
    cout<<"Jawapan Anda betul! Tahniah. "<<endl;
 else
    cout<<"Jawapan anda salah, cuba lagi "<<endl;

 return 0;
}
```

3.0 Struktur Kawalan Aturcara



Struktur Pilihan if- else if

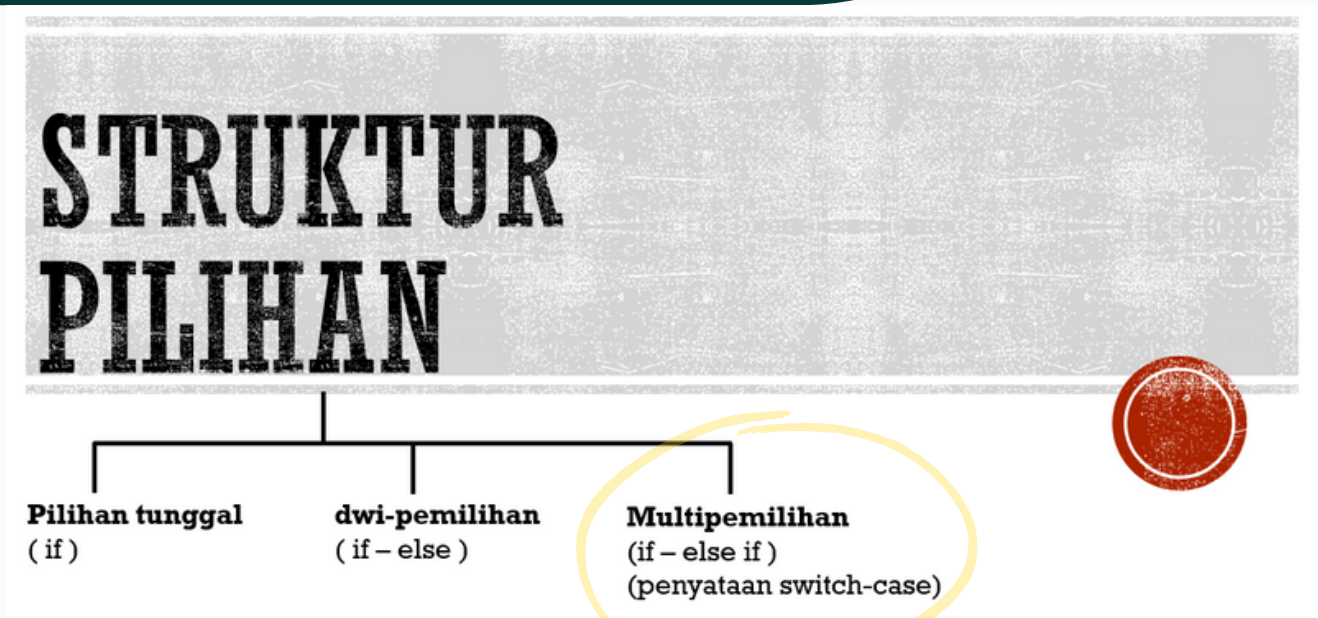
Pernyataan ini membolehkan lebih daripada 2 pilihan seperti berikut:

```
if ( x == 0 )
    cout<<"\nX adalah sifar";
else if ( x > 0 )
    cout<<"\nX adalah positif";
else
    cout<<"\nX adalah negatif";
```

Contoh aturcara

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

int main()
 {
 int num1, num2;

 cout << "Masukkan dua nombor\n"
 cout << "Nombor Pertama: ";
 cin >> num1; // baca no pertama

 cout << "Nombor Kedua: "; // baca no kedua
 cin >> num2;
 cout << "**********************\n";
 if ( num1 == num2 )
 cout << num1 << " adalah sama dengan " << num2 << endl;

 else if ( num1 < num2 )
 cout << num1 << " adalah kurang daripada " << num2 << endl;

 else if ( num1 > num2 )
 cout << num1 << " adalah lebih besar daripada " << num2 << endl;

 cout << "\n**********************\n\n";
 return 0;
}
```

3.0 Struktur Kawalan Aturcara



Struktur Pilihan penyataan switch- case

Digunakan untuk menggantikan pernyataan-pernyataan if- else if. Akan lebih berstruktur dan memudahkan pengaturcaraan

```
switch ( pilihan )
 {
  case  1 :  cout<<“Anda pilih nombor 1”;
                    break;
  case  2 : cout<<“Anda pilih nombor 2”;
                    break;
  case  3 : cout<<“Anda pilih nombor 3”;
                    break;
  default  : cout<<“Anda pilih nombor selain daripada
   1,2 atau 3”;
        }
```

3.0 Struktur Kawalan Aturcara

Penyelesaian masalah dengan Struktur Kawalan Ulangan

STRUKTUR ULANGAN/GELUNG

Terdapat 3 jenis sintak yang digunakan dalam bahasa C++ untuk struktur ulangan/ gelung.

1. **for**
2. **while**
3. **do - while**

Contoh aturcara gelung for

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

int main(){
for (int nombor = 0; nombor < 6; ++nombor){
 cout<<nombor<<endl;
 }
 return 0;
}
```

STRUKTUR GELUNG FOR

- digunakan untuk melakukan gelungan dengan bilangan tertentu
- Bentuk penggunaan pernyataan

```
for ( awalan; syarat ; pengemaskinian)
    pernyataan;
```

```
for ( i = 0; i < 10; i++)
    cout<<i;
```

3.0 Struktur Kawalan Aturcara

STRUKTUR GELUNG WHILE

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

int main()
{
    int x = 12;
    int x2;
 while ( x <= 21 )
 {
 x2 = x * x;
 cout<<"\n" <<x <<" Kuasa Dua ="  <<x2;
 x++;
 }
 return 0;
}
```

- Dalam contoh ini, satu senarai output jadual kuasa dua bagi nilai 12 hingga 21 dipaparkan pada paparan.

STRUKTUR GELUNG DO-WHILE

- Bentuk penggunaan pernyataan.

```
do
    pernyataan;
while ( syarat );
```

```
i = 0;
do
{
    cout<<i;
    i++;
} while ( i < 10 );
```

- Apakah bezanya berbanding gelung while?
  Satu output akan dipaparkan sekurang-kurangnya walaupun syarat adalah palsu

TOPIK 4.0

4.0 Tatasusunan dan Penunjuk

Penggunaan Tatasusunan

Tatasusunan (*array*) dalam C++ merujuk kepada suatu koleksi jujukan nilai yang:

- Mempunyai jenis data yang sama
- Menggunakan nama pembolehubah yang sama; yakni satu pembolehubah
- Mempunyai saiz yang tertentu
- Mempunyai indeks (indeks merupakan lokasi ingatan yang diperuntukan oleh ingatan komputer untuk sesuatu data, indeks bermula dengan nilai 0)

Satu struktur data yang boleh digunakan untuk menyimpan senarai nilai daripada jenis data tertentu

Merupakan teknik yang amat sesuai untuk mencetak atau membaca data yang lebih daripada satu

Sebagai contoh:

> **Markah bagi 10 orang pelajar dapat dicetak dan dibaca dengan mudah dengan hanya menggunakan satu pembolehubah**

Lebih cepat dan memudahkan kerja berbanding dengan cara biasa di mana sesuatu nilai data disimpan menggunakan satu pembolehubah

4.0 Tatasusunan dan Penunjuk

TATASUSUNAN SATU DIMENSI

- Pengisytiharan tatasusunan satu dimensi menggunakan sintaks:

  jenispembolehubah namapembolehubah[saiz] ;

  int markah[10];

- Jenis pembolehubah merupakan jenis data asas (char, int, float, double)
- Nama pembolehubah terdiri dari sebarang nama mengikut peraturan penamaan pengecam/ pembolehubah

TATASUSUNAN SATU DIMENSI

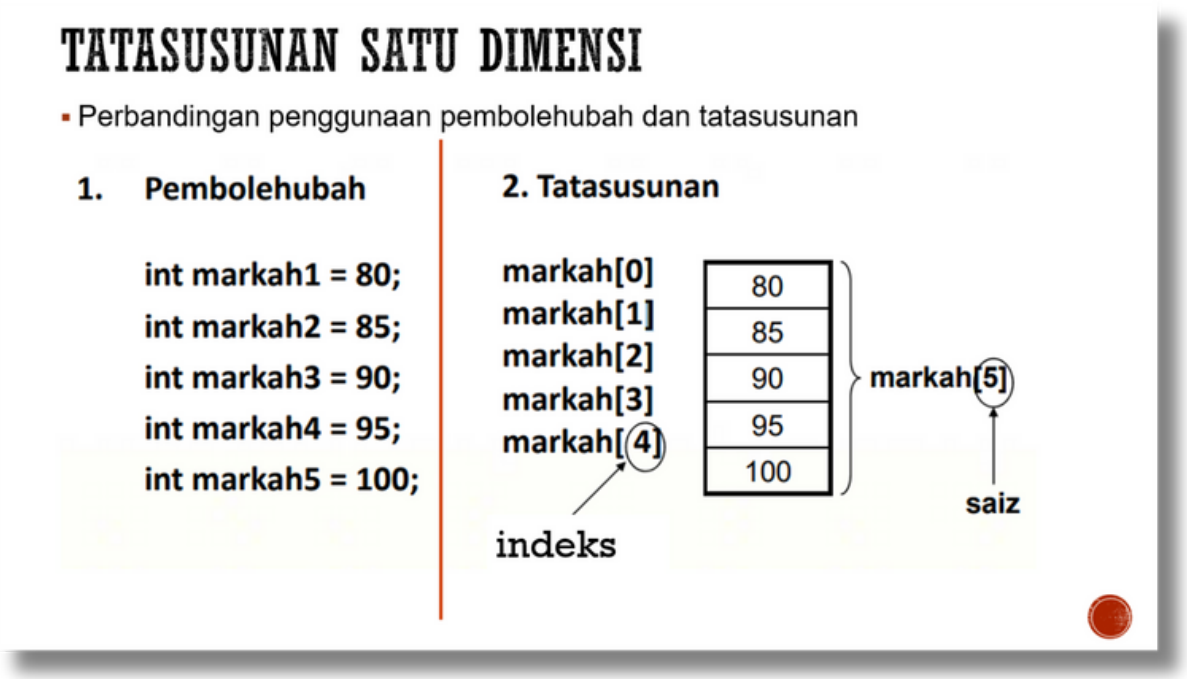
- Saiz merujuk kepada bilangan ruangan ingatan (sel ingatan) atau elemen yang diperuntukkan kepada pembolehubah
- Terdiri daripada pemalar, pembolehubah atau ungkapan bernilai integer
- Contoh:

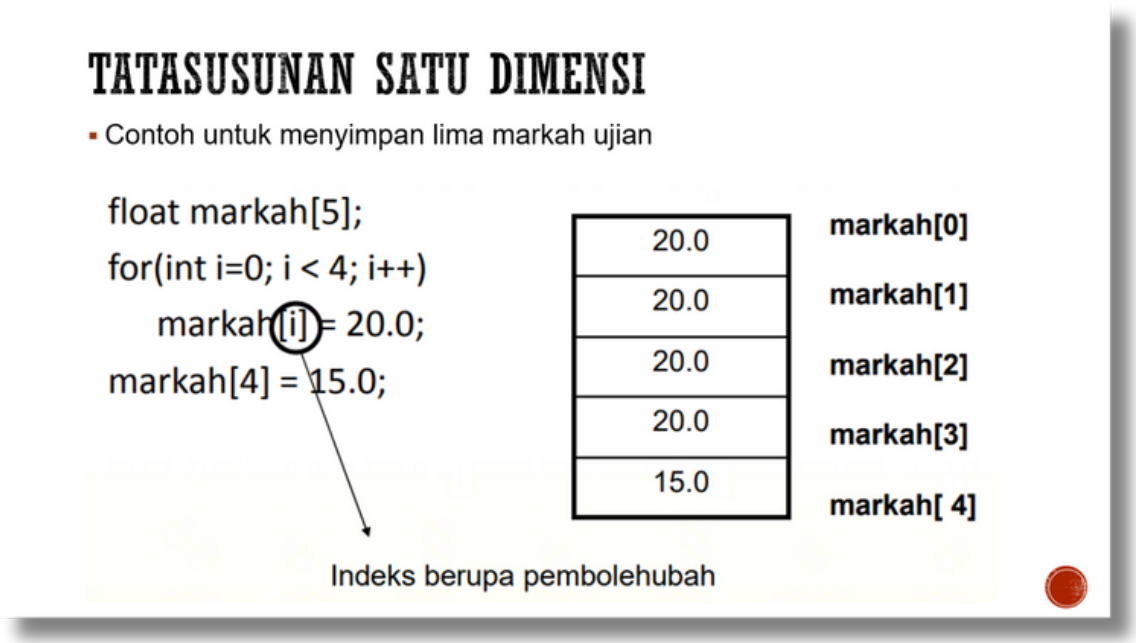
  markah[5];

  gred[SAIZ];

  result[MAKS-1];

4.0 Tatasusunan dan Penunjuk





- Perlu menggunakan indeks
- Indeks merupakan suatu pemalar, pembolehubah atau ungkapan yang bernilai integer yang menyatakan unsur manakah di dalam tatasusunan yang hendak dicapai

4.0 Tatasusunan dan Penunjuk

TATASUSUNAN SATU DIMENSI

- Terdapat dua cara untuk menilai awalkan tatasusunan
  1. Semasa proses pengisytiharan tatasusunan
  2. Semasa di dalam badan aturcara

- Sintaks pengumpukkan nilai awal tatasusunan semasa pengisytiharan adalah:

jenisdata namapembolehubah[saiz] = { senarai data };

int markah[5] = {78, 80, 95, 88, 99};

NILAI AWAL TATASUSUNAN

- Contoh pengumpukkan nilai awal tatasusunan semasa pengisytiharan:

```
int markah[5]={20, 10, 15, 18, 25};
char gred[3]={'a','b','c'};
```

- Saiz tatasusunan dapat diabaikan apabila pengumpukkan nilai berlaku semasa proses pengisytiharan

```
char bangsa[ ]={'m','c','i','L'};
```

4.0 Tatasusunan dan Penunjuk

- Pengumpukkan nilai awal tatasusunan juga boleh berlaku untuk sebahagian data sahaja.
- Contoh:

int umur[5]={18, 19, 20};
char huruf[3]={'a'};

- Bagi unsur yang tiada umpukan nilai, pengkompil akan mengumpukkan

Nilai **0** untuk jenis data **int** dan **float**
nilai **null** untuk **char**

Contoh
int umur[5]={16, 17, 18};

umur[0]	16
umur[1]	17
umur[2]	18
umur[3]	0
umur[4]	0

NILAI AWAL TATASUSUNAN

- Sintaks pengumpukkan nilai awal tatasusunan dalam badan aturcara pula adalah:

  namapembolehubah[indeks]=nilai;

- Pengumpukan nilai berlaku secara satu-persatu untuk semua elemen tatasusunan

```
Markah[0]=20;
Markah[1]=15;
Markah[2]=25;
Markah[3]=10;
Markah[4]=5;
```

4.0 Tatasusunan dan Penunjuk

Pengumpukkan nilai tatasusunan boleh berlaku dalam 3 bentuk:

1.Pemalar

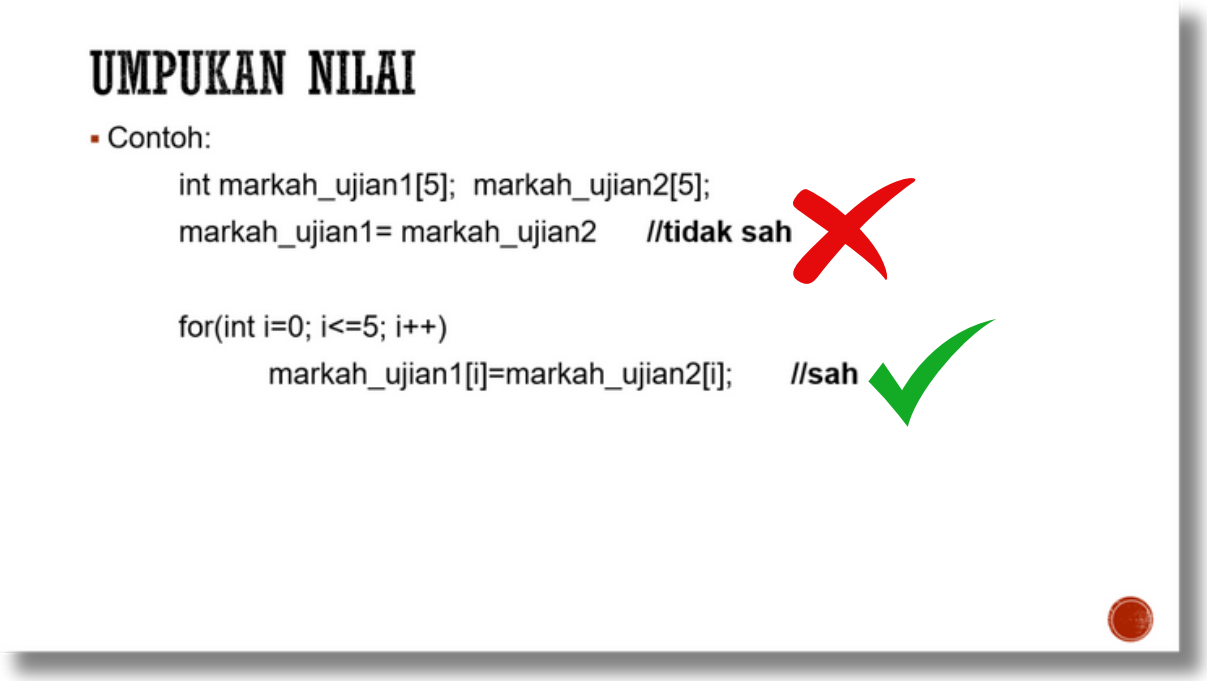
Contoh: **markah[0]=20;**

2.Pembolehubah

Contoh: **markah[1]=skorRendah;**

3.Ungkapan

Contoh: **markah[2]=skorRendah - 1;**

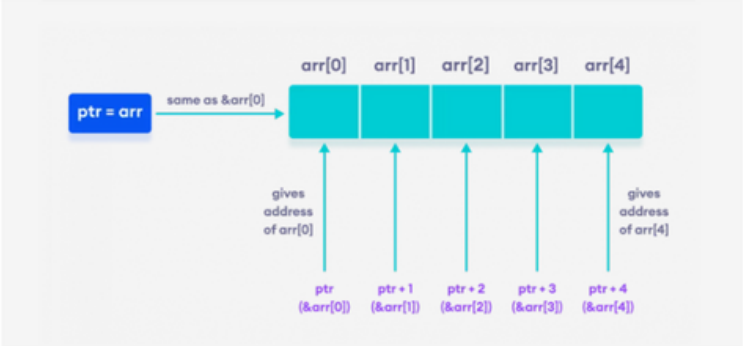


4.0 Tatasusunan dan Penunjuk

Penggunaan Penunjuk

- Penunjuk adalah pembolehubah yang menyimpan alamat pembolehubah lain.
- Penunjuk tidak hanya dapat menyimpan alamat satu pembolehubah, tetapi juga dapat menyimpan alamat sel-sel tatasusunan(*array*).

```
int *ptr ;
int arr[5];
ptr = arr;
```



Di sini, ptr adalah pemboleh ubah penunjuk sementara arr

```
// C++ Program to display address of each element of an array

#include <iostream>
using namespace std;
int main() {

 //declare array arr
 float arr[3];

 // declare pointer variable
 float *ptr;

 cout << "Displaying address using arrays: " << endl;

 // use for loop to print addresses of all array elements
 for (int i = 0; i < 3; ++i) {
   cout << "&arr[" << i << "]= " << &arr[i] << endl;
 }

 ptr = arr;

 cout<<"Displaying address using pointers: "<< endl;

 // use for loop to print addresses of all array elements
 // using pointer notation
 for (int i = 0; i < 3; ++i) {
   cout << "ptr + " << i << " = "<< ptr + i << endl;
 }

 return 0;
}
```

```
C:\Users\Acer\Documents\2022\c++\Projectpointer.exe
Displaying address using arrays:
&arr[0]= 0x6ffdf0
&arr[1]= 0x6ffdf4
&arr[2]= 0x6ffdf8
Displaying address using pointers:
ptr + 0 = 0x6ffdf0
ptr + 1 = 0x6ffdf4
ptr + 2 = 0x6ffdf8
```

4.0 Tatasusunan dan Penunjuk

Perkaitan di antara tatasusunan dan penunjuk

- Nama tatasusunan(*array*) digunakan sebagai penunjuk (*pointer*)

```
// C++ Program to insert and display data entered by using pointer notation.

#include <iostream>

using namespace std;

int main() {

 float arr[5];

 // Insert data using pointer notation
 cout << "Enter 5 numbers: \n";
 for (int i = 0; i <5 ; ++i) {
    cin>> *(arr + i) ;
 }

 // Display data using pointer notation
 cout << "\nDisplaying data: " << endl;

 for (int i = 0; i < 5; ++i) {
  // display value of arr(i)
  cout << *(arr + i) << endl ;
  }

  return 0;
}
```

```
C:\Users\Acer\Documents\2022\c++\Projectpointer.exe
Enter 5 numbers:
2.3
4.6
4.1
3.3
8.2

Displaying data:
2.3
4.6
4.1
3.3
8.2
```

TOPIK 5.0

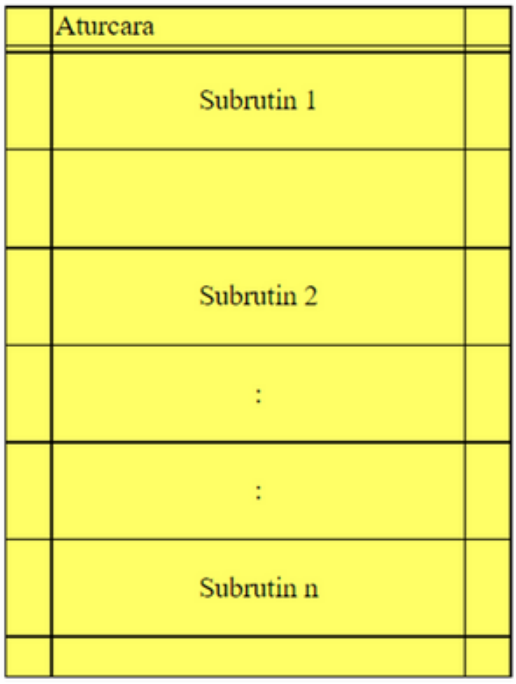
5.0 Fungsi

Penggunaan fungsi

- fungsi adalah blok kod yang dapat digunakan semula dalam program
- dapat menerima satu atau lebih input, melakukan beberapa operasi padanya, dan mungkin atau tidak mengembalikan output ke panggilanfungsi

KELEBIHAN :

- Ia boleh dipanggil dari mana-mana fungsi, beberapa kali



- Dalam C++, semua subrutin dinamakan fungsi (*function*)
- Sebuah aturcara C++ bolehlah dikatakan sebagai satu set fungsi-fungsi yang berinteraksi antara satu dengan lain
- Rutin utama bagi C++ ialah fungsi main(). Pelaksanaan aturcara akan bermula dari fungsi main() ini. Fungsi main() kemudiannya akan memanggil fungsi-fungsi lain bergantung kepada cara aturcara itu ditulis.

5.0 Fungsi

Elemen fungsi, Teknik menghulurkan parameter

Tiga elemen penting fungsi :

1. Takrifan fungsi
2. Panggilan fungsi
3. Prototaip fungsi

Teknik menghulurkan parameter :

1. Tidak memulangkan nilai, tidak menghantar parameter
2. Tidak memulangkan nilai, menghantar parameter
3. Memulangkan nilai, tidak menghantar parameter
4. Memulangkan nilai, menghantar parameter

5.0 Fungsi

2) TIDAK MEMULANGKAN NILAI, MENGHANTAR PARAMETER

```
#include<iostream>

using namespace std;

void kira_luas(int,int);

int main(){

    cout<<"LUAS SEBUAH BILIK\n";
    kira_luas(6,5);

    return 0;
}


void kira_luas(int panjang, int lebar){
    int luas;

    luas=panjang * lebar;
    cout<<"LUAS ="<<luas;
}
```

Prototaip Fungsi

Panggilan Fungsi

Takrifan Fungsi

3) MEMULANGKAN NILAI, TIDAK MENGHANTAR PARAMETER

```
#include<iostream>

using namespace std;

int kira_luas();

int main(){
    int luasBilik;

    cout<<"LUAS SEBUAH BILIK\n";
    luasBilik=kira_luas();
    cout<<"LUAS ="<<luasBilik;

return 0;
}


int kira_luas(){
    int luas;
    int panjang=6;
    int lebar=5;

    luas=panjang * lebar;
    return luas;
}
```

Prototaip Fungsi

Panggilan Fungsi

Takrifan Fungsi

5.0 Fungsi

4) MEMULANGKAN NILAI, MENGHANTAR PARAMETER

```
#include<iostream>

using namespace std;

int kira_luas(int,int);

int main(){
    int luasBilik;

    cout<<"LUAS SEBUAH BILIK\n";
    luasBilik=kira_luas(6, 5);
    cout<<"LUAS ="<<luasBilik;

return 0;
}


int kira_luas(int panjang, int lebar){
    int luas;

    luas=panjang * lebar;
    return luas;
}
```

Prototaip Fungsi

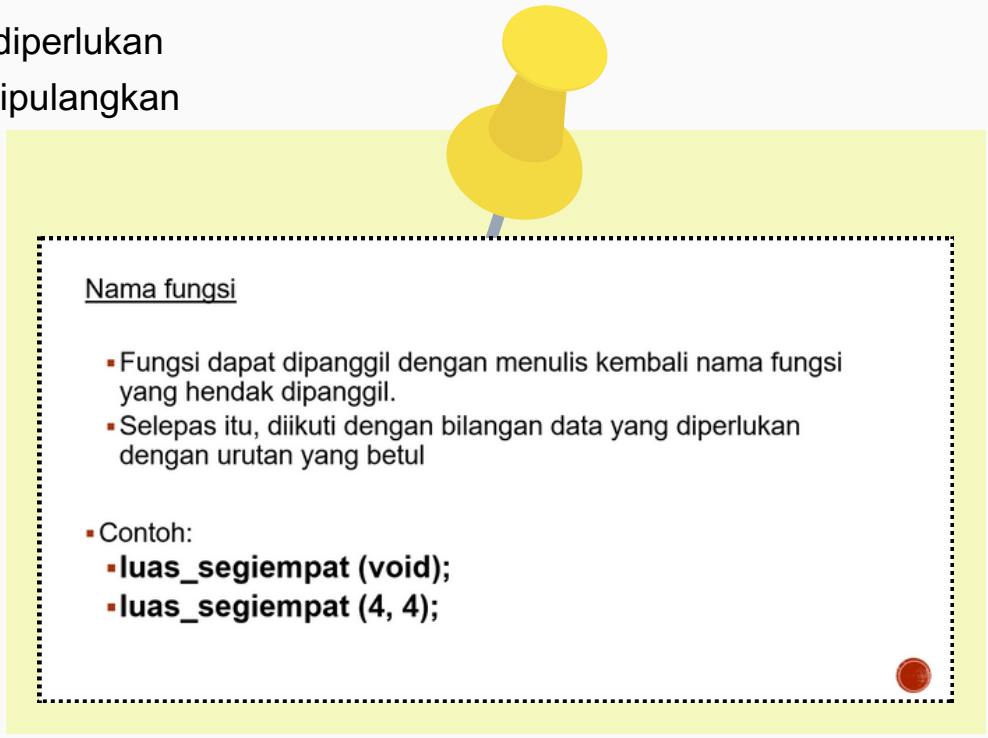
Panggilan Fungsi

Takrifan Fungsi

5.0 Fungsi

Panggilan fungsi

- Untuk melakukan sesuatu tugas, fungsi perlu dipanggil
- Terdapat 3 perkara perlu diketahui sebelum fungsi dapat dipanggil
  - **Nama** fungsi
  - **Parameter** yang diperlukan
  - **Nilai** yang akan dipulangkan



- Data yang dihantar kepada fungsi perlu terdiri daripada jenis yang sama seperti yang telah ditakrifkan pada senarai fungsi
- Bilangan data dan urutan juga perlu sama seperti dalam takrifan fungsi
- Data-data yang dihantar akan digunakan sebagai nilai parameter fungsi yang dipanggil

5.0 Fungsi

CONTOH ATURCARA

- Contoh panggilan fungsi luas_segiempat:

```
luas_segiempat (4, 6)

int luas_segiempat (int panjang, int lebar)
 {
    int luas;
    luas = panjang * lebar;
    return luas;
 }
```

- Contoh cetakan segiempat '*':

```
cetak_segiempat ( 4, 5, '*')

void cetak_segiempat ( int panjang, int lebar, char simbol)
{
    for ( int x = 1; x <= panjang; x++)
     {
          cout<<endl;
          for (int y=1; y<= lebar; y++)
           cout<<simbol;
     }

}
```

5.0 Fungsi

TAKRIFAN FUNGSI

1. Takrifan Fungsi Tanpa Parameter

```
#include <iostream>

using namespace std;

void kira_tambah();

int main(){
    kira_tambah();
}

void kira_tambah(){
    int nom1 =3, nom2=5, jumlah;

    jumlah= nom1+nom2;
    cout<<jumlah;
}
```

TAKRIFAN FUNGSI

2. Takrifan Fungsi Dengan Parameter

```
#include <iostream>

using namespace std;

void kira_tambah(int,int);

int main(){
    kira_tambah(3, 5);
}

void kira_tambah(int nom1, int nom2){
    int jumlah;

    jumlah= nom1+nom2;
    cout<<jumlah;
}
```

Prototaip fungsi

Parameter 3 dan 5 disalin ke parameter nom1 dan nom2 semasa panggilan fingsi

Rujukan

Noraniah Mohd. Yassin, Zalmiyah Zakaria, Dayang Norhayati Abang Jawawi, Norazah Yusof, & Radziah Mohamad (2016). Pengaturcaraan Berstruktur Menggunakan C++. UTM Press. (ISBN:978-983-52-1232-1)

Nor Hasbiah Ubaidullah, Jamilah Hamid, & Saira Banu Omar Khan (2015). Pengaturcaraan Berstruktur C++. Dewan Bahasa dan Pustaka.
(ISBN:978- 9834900991)

Eng L. Zhi (2018). Hands-On GUI Programming with C++ and Qt5. Packt Publishing.
(ISBN:9781788393744)

Stephen R. Davis (2015). Beginning Programming with C++ for Dummies. John Wiley & Sons Inc.
(ISBN:978-1118823873)

Isaac D. Cody (2017). C++: Learn C++ Like A Boss. A Beginners Guide in Coding Programming and Dominating C++. Novice to Expert Guide to Learn and Master C++ Fast (Hacking Freedom and Data Driven). CreateSpace Independent Publishing Platform.
(ISBN:978-1542737647)

ASAS PENGATURCARAAN C++

e ISBN 978-629-99371-0-4

KOLEJ KOMUNITI PAYA BESAR