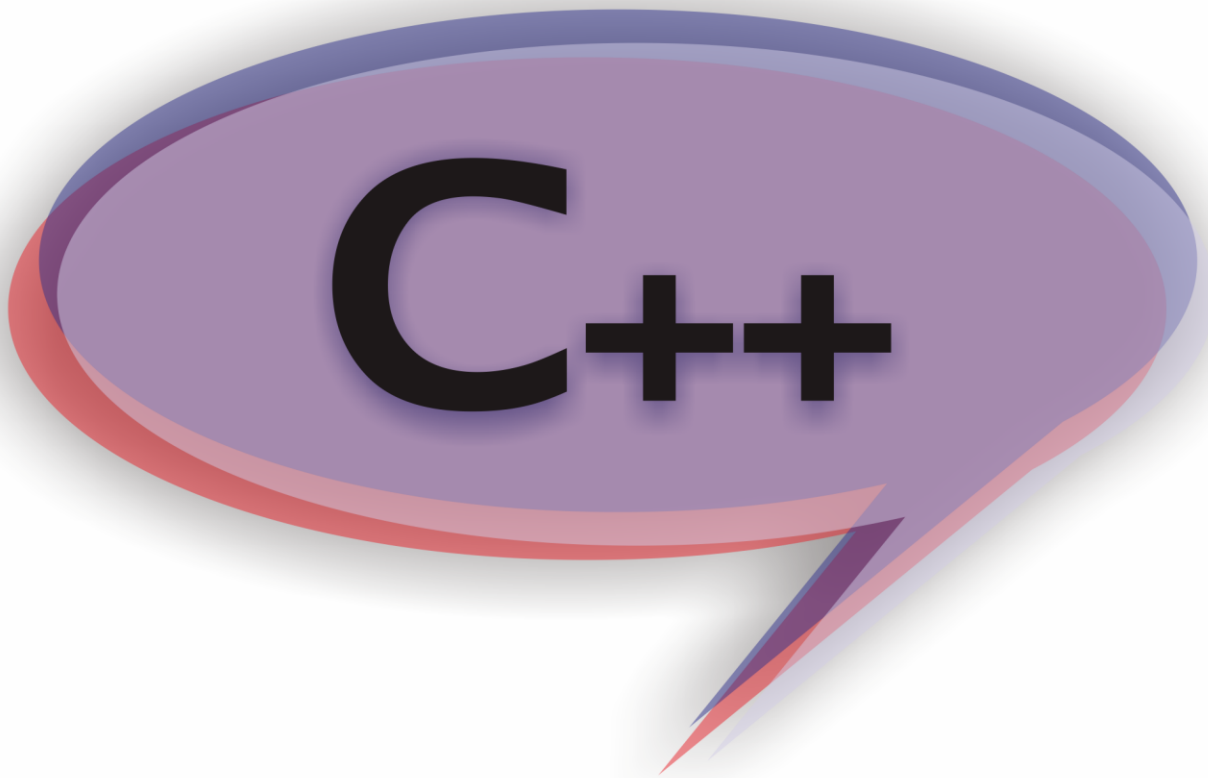




Modul
Algoritma dan Pemrograman Dasar
2015



Software Engineering Laboratory
Where Idea Meets Creativity

Daftar Isi

Daftar Isi.....	i
BAB I.....	1
PENGENALAN C++.....	1
1.1. Include	1
1.2. Fungsi Main()	2
1.3. Komentar	2
1.4. Tanda Semicolon	3
1.5. Mengetahui <i>cout</i>	3
1.6. Variabel.....	3
1.7. Deklarasi Variabel	4
1.8. Deklarasi Variabel	5
1.9. Tipe data.....	6
BAB II.....	9
OPERATOR DAN STATEMENT I/O	9
2.1. Operator	9
2.2. Operator I/O.....	16
BAB III	18
KONDISI	18
3.1. Kondisi.....	18
3.2. Kondisi IF	18
3.3. Pernyataan Switch.....	19
BAB IV	21
PERULANGAN.....	21
4.1. Perulangan	21
4.2. Pernyataan For	21

4.3. Pernyataan While	22
4.4. Pernyataan Do-While.....	23
4.5. Pernyataan Nest For.....	24
BAB V.....	28
FUNGSI DAN PROCEDURE.....	28
5.1. Fungsi.....	28
5.2. Jenis-jenis Fungsi :.....	29
BAB VI.....	33
ARRAY.....	33
7.1. Array	33

BAB I

PENGENALAN C++

C++ adalah bahasa pemrograman komputer yang di buat oleh (Bjarne Stroustrup) merupakan perkembangan dari bahasa C dikembangkan di Bell Labs (Dennis Ritchie) pada awal tahun 1970-an, Bahasa itu diturunkan dari bahasa sebelumnya, yaitu BCL, Pada awalnya, bahasa tersebut dirancang sebagai bahasa pemrograman yang dijalankan pada sistem Unix, Pada perkembangannya, versi ANSI (American National Standart Institute) Bahasa pemrograman C menjadi versi dominan, Meskipun versi tersebut sekarang jarang dipakai dalam pengembangan sistem dan jaringan maupun untuk sistem embedded, Bjarne Stroustrup pada Bel labs pertama kali mengembangkan C++ pada awal 1980-an.

Setiap program C++ mempunyai bentuk umum seperti dibawah ini :

```
#preprocessor directive
void main()
{
    // Batang Tubuh Program Utama
}
```

1.1. Include

Adalah salah satu pengarah *preprocessor directive* yang tersedia pada C++. Preprocessor selalu dijalankan terlebih dahulu pada saat proses kompilasi terjadi. Bentuk umumnya:

```
#include <nama_file>
```

Script tersebut tidak diakhiri dengan tanda semicolon, karena bentuk tersebut bukanlah suatu bentuk pernyataan, tetapi merupakan preprocessor directive. Baris tersebut menginstruksikan kepada kompiler yang menyisipkan file lain dalam hal ini file yang berakhiran .h (file header) yaitu file yang berisi sebagai deklarasi, contohnya :

- **#include <iostream>**

Diperlukan pada program yang melibatkan proses Input (cin) atau Output (cout).

- **#include <conio.h>**

Diperlukan bila melibatkan *clrscr()*, yaitu perintah untuk membersihkan layar.

- **#include <iomanip.h>**

Diperlukan bila melibatkan *setw()* yaitu untuk mengatur lebar dari suatu tampilan data.

- **#include <math.h>**

Diperlukan pada program yang menggunakan *sqrt()* yang bermanfaat untuk operasi matematika kuadrat.

1.2. Fungsi Main()

Fungsi ini menjadi awal dan akhir eksekusi program C++. Main adalah nama judul fungsi. Melihat bentuk seperti itu dapat kita ambil kesimpulan bahwa batang tubuh program utama berada didalam fungsi main(). Pembahasan lebih lanjut mengenai fungsi akan diterangkan kemudian. Yang sekarang ditekankan adalah kita menuliskan program utama kita didalam sebuah fungsi main().

1.3. Komentar

Komentar tidak pernah di-*compile* oleh *compiler*. Dalam C++ terdapat 2 jenis komentar, yaitu :

- **/* Komentar */**

Komentar dapat diletakkan di dalamnya dan dapat mengapit 2 atau lebih baris.

- **// Komentar**

Komentar dapat diletakkan setelah tanda //, namun hanya berguna untuk 1 baris saja.

1.4. Tanda Semicolon

Tanda semicolon “;” digunakan untuk mengakhiri sebuah pernyataan. Setiap pernyataan harus diakhiri dengan sebuah tanda semicolon.

1.5. Mengenal *cout*

Pernyataan *cout* merupakan sebuah objek di dalam C++, yang digunakan untuk mengarahkan data ke dalam standar output (cetak pada layar). Contoh :

```
void main()
{
    cout<<"Halo dunia!";
}
```

Tanda “<<” merupakan sebuah operator yang disebut operator “penyisipan/peletakan”.

1.6. Variabel

Variabel adalah tempat dimana kita dapat mengisi atau mengosongkan nilainya dan memanggil kembali apabila dibutuhkan. Setiap variabel akan mempunyai nama (identifier) dan nilai. contoh sintaks sebagai berikut :

```
V a r i a b e l = e k s p r e s i ;
```

Nama dari suatu variabel dapat ditentukan sendiri oleh program dengan aturan sebagai berikut :

- Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf. Bahasa C++ bersifat *case-sensitive*, yang artinya huruf besar dan kecil dianggap berbeda. Jadi antara **nim**, **NIM**, dan **Nim** dianggap berbeda.
- Tidak boleh mengandung spasi.
- Tidak boleh mengandung simbol-simbol khusus, kecuali garis bawah (underscore). Yang termasuk simbol khusus yang tidak diperbolehkan antara lain : \$, ?, %, #, !, &, *, (,), -, +, =, dsb.
- Panjang bebas, tetapi hanya 32 karakter pertama yang terpakai.

Contoh penamaan variabel yang benar :

Penanaman Yang Benar	Penanaman Yang Salah
namasiswa	nama siswa (salah karena menggunakan spasi)
XY12	12X (salah karena dimulai dengan angka)
harga_total	harga.total (salah karena menggunakan karakter .)
JenisMotor	Jenis Motor (salah karena menggunakan spasi)
alamatrumah	for (salah karena menggunakan kata kunci bahasa pemrograman)

1.7. Deklarasi Variabel

Deklarasi diperlukan bila kita akan menggunakan pengenal (identifier) dalam program. Identifier dapat berupa variabel, konstanta, dan fungsi.

Bentuk umumnya :

```
n a m a _ t i p e   n a m a _ v a r i a b e l ;
```

Contoh :

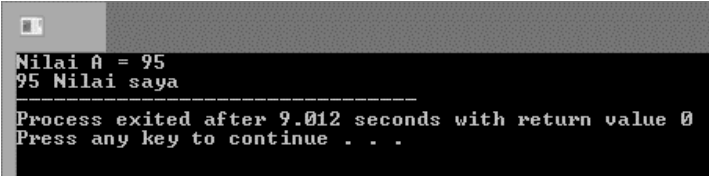
```
int x; //Deklarasi x bertipe integer
char y, huruf, nim[10]; //Deklarasi variabel char
float nilai; //Deklarasi variabel bertipe float
double beta; //Deklarasi variabel bertipe double
int array[5][4]; //Deklarasi variabel bertipe integer
```

Contoh :

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

int main()
{
    int A;
    cout<<"Nilai A = ";cin>>A;
    cout<<A<<" Nilai saya";
}
```

Tampilan :



```
Nilai A = 95
95 Nilai saya
-----
Process exited after 9.012 seconds with return value 0
Press any key to continue . . .
```

1.8. Deklarasi Variabel

Ada dua tipe deklarasi konstanta, yaitu :

1. Menggunakan keyword *const*

Contoh :

```
const float phi = 3.14;
```

Berbeda dengan variabel, konstanta tidak dapat dirubah jika telah diinisialisasi.

2. Menggunakan *#define*

Contoh :

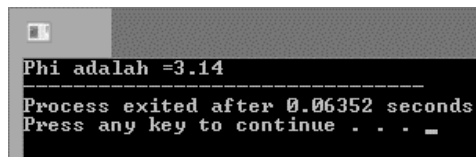
```
#define phi = 3.14;
```

Keuntungan menggunakan *#define* apabila dibandingkan dengan *const* adalah kecepatan kompilasi, karena sebelum kompilasi dilaksanakan, kompiler pertama kali mencari simbol *#define* (oleh sebab itu mengapa # dikatakan preprocessor directive) dan mengganti semua pi dengan nilai 3.14.

Contoh :

```
#include <iostream>
#define phi 3.14
using namespace std;
main()
{
    cout<<"Phi adalah ="<<phi;
```

Tampilan :



```
Phi adalah =3.14
-----
Process exited after 0.06352 seconds
Press any key to continue . . . _
```


1.9. Type data

1.9.1. Tipe Dasar

Adalah tipe data yang dapat langsung dipakai. Daftarnya dapat dilihat pada tabel dibawah ini :

Tipe Dasar	Ukuran Memory (byte)	Jangkuan Nilai	Jumlah digit Presisi
Char	1	-128 hingga +127	-
Int	2	-32768 hingga +32767	-
Long	4	-2.147.438.648 hingga 2.147.438.647	-
Float	4	3,4E-38 hingga 3,4E38	6-7
Double	8	1,7E-308 hingga 1,7E308	15-16
Long Double	10	3.4E-4932 hingga 1.1E4932	19

Perhatikan contoh dibawah ini, tipe data dapat dirubah (*type cast*) dengan cara dibawah ini :

```
float x = 3.345;  
int p = int(x);
```

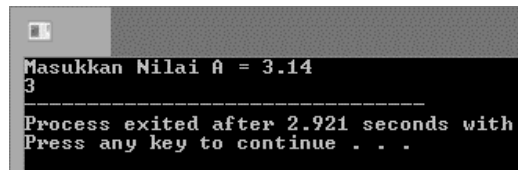
Maka nilai P adalah 3 (terjadi *truncating*).

Tipe data yang berhubungan dengan bilangan bulat adalah char, int, dan long. Sedangkan lainnya berhubungan dengan bilangan pecahan.

Contoh :

```
#include <iostream>  
using namespace std;  
main()  
{  
    int A;  
    cout<<"Masukkan Nilai A = ";cin>>A;  
    cout<<A;  
}
```

Tampilan :



```
Masukkan Nilai A = 3.14
3
-----
Process exited after 2.921 seconds with
Press any key to continue . . .
```

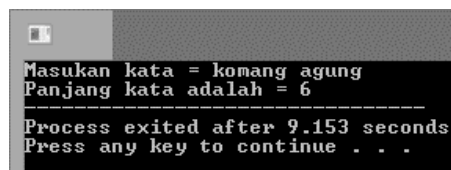
1.9.2. Karakter dan String Literal

String adalah gabungan dari karakter. Perbedaannya dengan tipe data *char* adalah mengenai penyimpanannya. Char menyimpan hanya beberapa karakter saja, sedangkan String dapat menyimpan banyak karakter, kata, dan kalimat sekaligus.

Contoh :

```
#include <iostream>
#include <cstring>
using namespace std;
main()
{
    int panjangteks;
    char kata[100];
    cout<<"Masukan kata = ";
    cin>>kata;
    panjangteks=strlen(kata);
    cout<<"Panjang kata adalah = "<<panjangteks;
}
```

Tampilan :



```
Masukan kata = komang agung
Panjang kata adalah = 6
-----
Process exited after 9.153 seconds
Press any key to continue . . .
```

1.9.3. Keyword dan Identifier

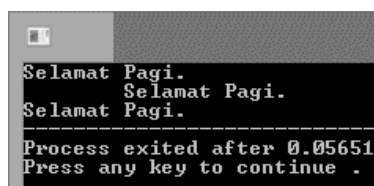
Dalam bahasa pemrograman, suatu program dibuat dari elemen-elemen sintaks individual yang disebut token, yang memuat nama variable, konstanta, keyword, operator, dan tanda baca.

Karakter	Keterangan
\0	Karakter ber-ASCII nol (karakter null)
\a	Karakter bell
\b	Karakter Backspace
\f	Karakter ganti halaman (fromfeed)
\n	Karakter baris baru (newline)
\r	Karakter carriage return (ke awal baris)
\t	Karakter tab horizontal
\v	Karakter tab vertikal
\\	Karakter \
\'	Karakter '
\"	Karakter "
\?	Karakter ?

Contoh :

```
#include <iostream>
using namespace std;
main()
{
    cout<<"Selamat Pagi.\n";
    cout<<"\tSelamat Pagi."<<endl;
    cout<<"Selamat Pagi.\a";
}
```

Tampilan :



```
Selamat Pagi.
Selamat Pagi.
Selamat Pagi.
-----
Process exited after 0.05651
Press any key to continue . .
```

BAB II

OPERATOR DAN STATEMENT I/O

2.1. Operator

Operator merupakan karakter khusus yang berupa simbol atau tanda untuk melakukan suatu operasi atau manipulasi.

2.1.1. Operator Penugasan

Operator Penugasan (*assignment operator*) dalam bahasa C++ berupa tanda sama dengan (“=”).

Contoh :

```
Nilai = 80;  
A = x * y;
```

Variabel “ nilai ” diisi dengan 80 dan variabel “ A ” diisi dengan hasil perkalian x dan y.

2.1.2. Operator Aritmatika

Operator aritmatika merupakan operator yang digunakan untuk melakukan perhitungan pada bilangan.

Tabel operator aritmatika :

Operator	Deskripsi	Contoh
+	Penjumlahan	X + Y
-	Pengurangan	X - Y
*	Perkalian	X * Y
/	Pembagian	X / Y
%	Sisa pembagian Integer (Modulus)	X % Y
~	Negasi (Unary Operator)	~ X

Pada operator penjumlahan, pengurangan, perkalian, dan pembagian dilakukan operasi seperti biasa. Sedangkan, yang di maksud dengan sisa pembagian adalah **sisa dari hasil pembagian** bukan hasil dari pembagian.

Contohnya :

$10 \% 2$

Hasil pembagian = 5

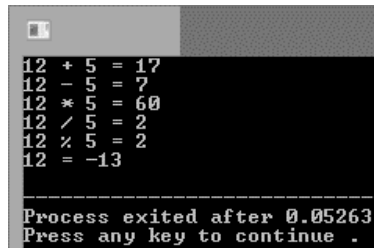
Sisa pembagian = 0

Pada operator negasi hanya membutuhkan satu operand saja.

Contoh :

```
#include <iostream>
using namespace std;
main()
{
    int x = 12, y = 5;
    cout<<x<<" + "<<y<<" = "<<(x+y)<<endl;
    cout<<x<<" - "<<y<<" = "<<(x-y)<<endl;
    cout<<x<<" * "<<y<<" = "<<(x*y)<<endl;
    cout<<x<<" / "<<y<<" = "<<(x/y)<<endl;
    cout<<x<<" % "<<y<<" = "<<(x%y)<<endl;
    cout<<x<<" = "<<(~x)<<endl;
}
```

Tampilan :



```
12 + 5 = 17
12 - 5 = 7
12 * 5 = 60
12 / 5 = 2
12 % 5 = 2
12 = -13
-----
Process exited after 0.05263
Press any key to continue . .
```

Penjelasan :

Karena tipe datanya integer, maka $12/5 = 2$. Alasannya karena tipe data integer tidak dapat menampung nilai pecahan. Pada operasi negasi nilai 12 merupakan nilai positif maka operasinya $x + 1 = \sim x$, sehingga $12 + 1 = -13$.

2.1.3. Operator Hubungan / Relasi

Operator hubungan / relasi digunakan untuk membandingkan hubungan antara dua buah operand (sebuah nilai atau variabel). Pada operator relasi menghasilkan kondisi **benar** atau **salah**.

Operator	Deskripsi	Contoh	
==	Sama dengan (bukan assignment)	X==Y	X sama dengan Y
!=	Tidak sama dengan	X!=Y	X tidak sama dengan Y
>	Lebih besar	X > Y	X lebih besar dari Y
<	Lebih Kecil	X < Y	X lebih kecil dari Y
>=	Lebih besar atau sama dengan	X>=Y	X lebih besar atau sama dengan Y
<=	Lebih kecil atau sama dengan	X<=Y	X lebih kecil atau sama dengan Y

Contoh :

```
#include <iostream>
using namespace std;
main ()
{
    int x = 8, y = 14;
    if (x==y)
    {
        cout<<x<<" sama dengan "<<y<<endl;
    }
    else if (x!=y);
    {
        cout<<x<<"      tidak      sama      dengan
"<<y<<endl;
    }
    else if (x>y)
    {
```

```

        cout<<x<<"    lebih    besar    dari
"<<y<<endl;
    }
    else if (x<y)
    {
        cout<<x<<"    lebih    kecil    dari
"<<y<<endl;
    }
    else if (x>=y)
    {
        cout<<x<<"    lebih    besar    atau    sama
dengan "<<y<<endl;
    }
    else if (x<=y)
    {
        cout<<x<<"    lebih    kecil    atau    sama
dengan "<<y<<endl;
    }
}

```

Tampilan :



2.1.4. Operator Naik dan Turun (*Increment dan Decrement*)

Operator *increment* merupakan operator yang dapat menambahkan (menaikkan) suatu nilai. Operator increment ini ditandai dengan tanda “ ++ ”. Sedangkan, operator decrement merupakan operator yang mengurangi (menurunkan) suatu nilai. Operator decrement ini ditandai dengan tanda “ -- ”.

Contoh :

```

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

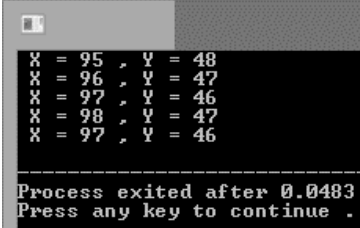
```

```

int x = 95 , y = 48;
cout<<" X = "<<x<<" , Y = "<<y<<endl;
++x, --y;
cout<<" X = "<<x<<" , Y = "<<y<<endl;
x++, y--;
cout<<" X = "<<x<<" , Y = "<<y<<endl;
x++, y++;
cout<<" X = "<<x<<" , Y = "<<y<<endl;
x--, y--;
cout<<" X = "<<x<<" , Y = "<<y<<endl;
}

```

Tampilan :



```

X = 95 , Y = 48
X = 96 , Y = 47
X = 97 , Y = 46
X = 98 , Y = 47
X = 97 , Y = 46
-----
Process exited after 0.0483
Press any key to continue .

```

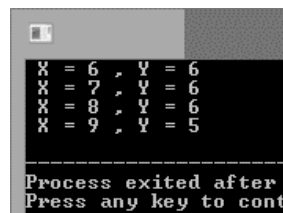
Penjelasan :

Terlihat bahwa operator pre-increment (penambahan sebelum ekspresi atau proses lain dijalankan) dan post-increment (penambahan sesudah ekspresi atau proses lain dijalankan) memiliki akibat yang sama, yaitu menambahkan nilai satu pada X dan memasukkan nilai tersebut kembali ke X ($x = x + 1$). Hal yang sama juga terjadi pada operator pre-decrement dan post-decrement yang memberikan akibat yang sama, yaitu mengurangi nilai satu pada y ($y = y - 1$).

Contoh :

```
#include <iostream>
using namespace std;
main ()
{
    int x = 5, y;
    y = ++x;
    cout<<" X = "<<x<<" , Y = "<<y<<endl;
    y = x++;
    cout<<" X = "<<x++<<" , Y = "<<y<<endl;
    cout<<" X = "<<x<<" , Y = "<<y<<endl;
    cout<<" X = "<<++x<<" , Y = "<<--y<<endl;
}
```

Tampilan :



Penjelasan :

Dalam penugasan yang pertama, x adalah pre-increment, menaikkan nilainya menjadi 6, yang selanjutnya dimasukkan ke y. Dalam penugasan kedua, x adalah post-increment, sehingga 6 dimasukkan ke y kemudian x dinaikkan itu sebabnya nilai x = 7, y = 6. Dalam penugasan ketiga, x pre-increment, sehingga nilai x dinaikkan menjadi = 8, sedangkan nilai y tetap. Dalam penugasan keempat x adalah pre-increment sehingga nilai x dinaikkan menjadi 9, sedangkan y adalah pre-decrement sehingga nilai y = 5.

2.1.5. Operator Bitwise (Logika)


Operator logika merupakan operator yang berhubungan dengan manipulasi bit. Operator ini hanya bisa digunakan pada operand bertipe data int atau char.

Operator	Deskripsi	Contoh
<<	Geser n bit ke kiri (shift left)	X << Y
>>	Geser n bit ke kanan (shift right)	X >> Y
&&	AND	X && Y
	OR	X Y
^	XOR	X ^ Y
~	NOT	~ X

Contoh :

```
#include <iostream>
using namespace std;
main ()
{
    int x = 10, y = 2;
    cout<<(x<<y)<<endl;
    cout<<(x>>y)<<endl;
    cout<<(x&y)<<endl;
    cout<<(x|y)<<endl;
    cout<<(x^y)<<endl;
    cout<<(~x)<<endl;
}
```

Tampilan :



```
40
2
2
10
8
-11
-----
Process exited after 0.05397
Press any key to continue . .
```

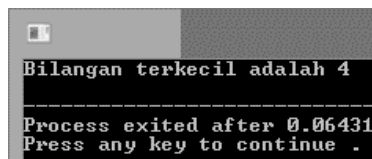
2.1.6. Operator Kondisi

Operator kondisi digunakan untuk memperoleh nilai dari dua kemungkinan *ungkapan1* ? *ungkapan2* : *ungkapan3*. Bila nilai *ungkapan1* benar, maka nilainya sama dengan *ungkapan2*, bila tidak maka nilainya sama dengan *ungkapan3*.

Contoh :

```
#include <iostream>
using namespace std;
main()
{
    int m =4, n=14;
    int min=m<n?m:n;
    cout<<"Bilangan terkecil adalah "<<min<<endl;
}
```

Tampilan :



Penjelasan :

Cara menggunakan operator kondisi adalah menentukan kondisi dengan bantuan operator hubungan. Pada contoh diatas adalah pada "m<n". Kemudian tambahkan simbol "?". Kemudian aksi yang akan muncul pada m:n. Sehingga membentuk :

```
min=operand1<operand2?jika_benar:jikasalah;
```

2.2. Operator I/O

Pada C++ terdapat 2 jenis I/O dasar, yaitu :

1. Statemen Input adalah Statemen / fungsi yang digunakan untuk membaca data dari inputing device (keyboard/mouse), contoh : cout (character out).

2. Statemen Output adalah Statemen yang digunakan untuk menuliskan data ke layar monitor, contoh : cin (character in).

BAB III

KONDISI

3.1. Kondisi

Statement kondisi biasa di gunakan untuk pengambilan sebuah perintah atau keputusan kondisional (bersyarat) di mana ada 2 atau lebih kondisi yang harus kita pilih (true/false), ketika syarat dari kondisi tersebut terpenuhi, maka kondisi tersebut bernilai benar atau true, jika syarat kondisi tersebut belum terpenuhi maka kondisi tersebut bernilai salah atau false.

3.2. Kondisi IF

Sebuah pernyataan yang dapat dipakai untuk mengambil keputusan berdasarkan suatu kondisi. Bentuk pernyataan ini ada dua macam :

- IF
- ELSE

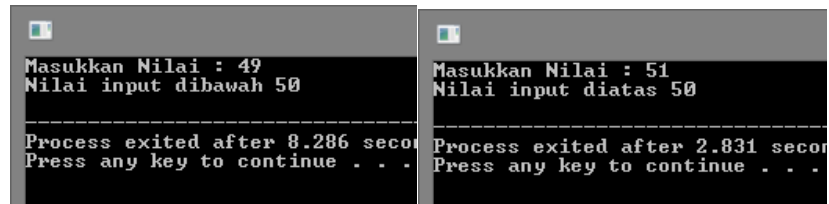
Bentuk umum dari kondisi IF bisa dilihat pada script dibawah ini :

```
If (kondisi)
    Statement;
```

Pernyataan dilaksanakan jika dan hanya jika kondisi yang diinginkan terpenuhi, jika tidak program tidak memberikan hasil apa-apa.

Contoh :

```
#include <iostream>
using namespace std;
main()
{
    int nilai;
    cout<<"Masukkan Nilai : ";
    cin>>nilai;
    if (nilai > 50)
    {
        cout<<"Nilai input diatas 50"<<endl;
    }
    else
    {
        cout<<"Nilai input dibawah 50"<<endl;
    }
    return 0;
}
```



Pernyataan dilaksanakan jika dan hanya jika kondisi yang diinginkan terpenuhi, jika tidak program tidak memberikan hasil apa-apa.

Pernyataan1 dilaksanakan jika dan hanya jika kondisi yang diinginkan terpenuhi, jika tidak, lakukan pernyataan2. Jika Anda tidak mempergunakan pernyataan *else* program tidak akan error, namun jika anda mempergunakan pernyataan *else* tanpa didahului pernyataan *if*, maka program akan error. Jika pernyataan1 atau pernyataan2 hanya terdiri dari satu baris, maka tanda { } tidak diperlukan, namun jika lebih maka diperlukan.

3.3. Pernyataan Switch

Pernyataan **switch** adalah pernyataan yang digunakan untuk menjalankan salah satu pernyataan dari beberapa kemungkinan pernyataan, berdasarkan nilai dari sebuah ungkapan dan nilai penyeleksian. Pernyataan **if...else if** jamak dapat dibangun dengan pernyataan **switch**.

Bentuk umum dari Switch :

```
switch (ekspresi)
{
    case konstanta1:
        pernyataan1;
        break;
    case konstanta2:
        pernyataan2;
        break;
    case konstanta3:
        pernyataan3;
        break;
    default:
        pernyataanlain;
}
```

Hal-hal yang perlu diperhatikan :

1. Dibelakang keyword case harus diikuti oleh sebuah konstanta, tidak boleh diikuti oleh ekspresi ataupun variable.
2. Konstanta yang digunakan bertipe int atau char.
3. Jika bentuknya seperti diatas maka apabila *ekspresi* sesuai dengan konstanta2 maka pernyataan2, pernyataan3 sampai dengan pernyataan lain dieksekusi. Untuk mencegah hal tersebut, gunakan keyword **break**. Jika keyword **break** digunakan maka setelah pernyataan2 dieksekusi program langsung keluar dari pernyataan **switch**. Selain digunakan dalam **switch**, keyword *break* banyak digunakan untuk keluar dari pernyataan yang berulang (looping).
4. Pernyataan lain dieksekusi jika konstanta1 sampai konstantaN tidak ada yang memenuhi *ekspresi*.

Contoh :

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

main()
{
    int nilai;
    cout<<"Masukkan nilai : ";
    cin>>nilai;
    switch (nilai)
    {
        case 1:
            cout<<"Nilai 1"<<endl;
            break;
        case 2:
            cout<<"Nilai 2"<<endl;
            break;
        case 3:
            cout<<"Nilai 3"<<endl;
            break;
        default:
            cout<<"Salah,          nilai          diluar
jangakauan."<<endl;
    }
}
```

BAB IV

PERULANGAN

4.1. Perulangan

Perulangan pada C++ merupakan sebuah / sekelompok instruksi yang diulang untuk jumlah pengulangan tertentu. Baik yang terdefinisi sebelumnya ataupun tidak. Struktur pengulangan terdiri atas dua bagian :

1. Kondisi pengulangan yaitu ekspresi boolean yang harus dipenuhi untuk melaksanakan pengulangan.
2. Isi atau badan pengulangan yaitu satu atau lebih pernyataan (aksi) yang akan diulang.

Perintah atau notasi dalam struktur pengulangan adalah pernyataan **For**, pernyataan **While**, pernyataan **Do-While**, pernyataan **Nested For**.

4.2. Pernyataan For

Pernyataan for digunakan untuk menghasilkan pengulangan(looping) beberapa kali tanpa penggunaan kondisi apapun. Pada umumnya looping yang dilakukan oleh for telah diketahui batas awal, syarat looping dan perubahannya.

Pada umumnya looping yang dilakukan oleh for telah diketahui batas awal, syarat looping dan perubahannya. Selama *kondisi* terpenuhi, maka pernyataan akan terus dieksekusi.

Bentuk umum:

```
for (inisialisasi ; kondisi ; perubahan)
{
    statement;
}
```

Contoh :

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

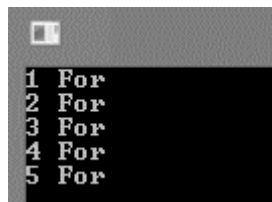


```

{
    int i;
    for (i=1;i<=5;i++)
    {
        cout<<i<<" For"<<endl;
    }
    getch();
}

```

Hasil Tampilan :



Analisa Program :

Pada program diatas, terdapat variabel i bertipe data integer. Kemudian variabel i akan di ulang dengan pernyataan for sesuai dengan kondisinya, yaitu i = 1 sampai i <= 5, i++ diikuti dengan output “For” yang akan di ulang sampai kondisi tersebut tidak terpenuhi lagi.

4.3. Pernyataan While

Pernyataan while merupakan salah satu pernyataan yang berguna untuk memproses suatu pernyataan atau beberapa pernyataan beberapa kali. Pernyataan while memungkinkan statemen-statemen yang ada didalamnya tidak dilakukan sama sekali.

Bentuk umum :

```

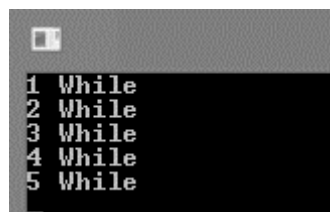
Inisialisasi;
while (kondisi)
{
    pernyataan;
    perubahan;
}

```

Contoh:

```
#include <iostream>
using namespace std;
main()
{
    int i=1;
    while (i<=5)
    {
        cout<<i<<" While"<<endl;
        i++;
    }
}
```

Hasil Tampilan :



```
1 While
2 While
3 While
4 While
5 While
```

Analisa Program :

Pada program diatas dimulai dengan variabel $i = 1$, ketika $i \leq 5$ maka akan menjalankan perintah output i diikuti kata “While” kemudian $i++$ sampai semua kondisi terpenuhi.

4.4. Pernyataan Do-While

Pernyataan do-while mirip seperti pernyataan while, hanya saja pada do-while pernyataan yang terdapat didalamnya minimal akan sekali dieksekusi.

Bentuk umum :

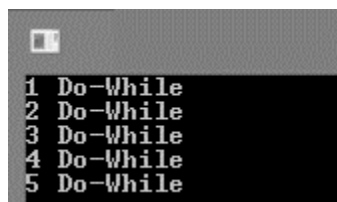
```
Inisialisasi;
do
{
    pernyataan;
}
```

```
while (kondisi);
```

Contoh :

```
#include <iostream>
using namespace std;
main()
{
    int i=1;
    do
    {
        cout<<i<<" Do-While"<<endl;
        i++;
    }
    while (i<=5);
}
```

Hasil Tampilan :



```
1 Do-While
2 Do-While
3 Do-While
4 Do-While
5 Do-While
```

Analisa Program :

Pada program diatas dimulai dengan variabel $i = 1$, kemudian program akan mengerjakan perintah output i diikuti kata Do-While, ketika $i \leq 5$ maka akan perulangan tersebut sampai kondisi tidak terpenuhi lagi.

4.5. Pernyataan Nested For

Pernyataan nested for adalah suatu perulangan for di dalam perulangan for yang lain. Di dalam penggunaan nested for, perulangan yang di dalam terlebih dahulu dihitung hingga selesai, kemudian perulangan yang di luar diselesaikan.

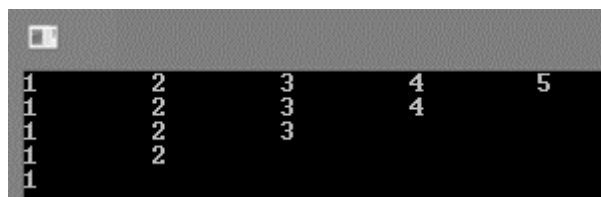
Bentuk umum :

```
for (inisialisasi ; kondisi ; perubahan)
{
    for (inisialisasi ; kondisi ; perubahan)
    {
        statement;
    }
}
```

Contoh :

```
#include<iostream>
using namespace std;
main()
{
    int x, y;
    for (x=5;x>=1;x--)
    {
        for (y=1;y<=x;y++)
        {
            cout<<y<<"\t";
        }
        cout<<endl;
    }
}
```

Hasil Tampilan :



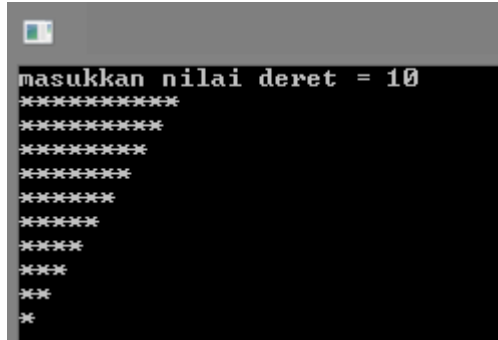
```
1      2      3      4      5
1      2      3      4
1      2      3
1      2
1
```

Analisa Program :

Pada program diatas terdapat variabel x dan y bertipe integer. Pada for pertama (`for (x=5;x>=1;x--)`) digunakan untuk mengerjakan output vertikal, sedangkan for kedua (`for (y=1;y<=x;y++)`) digunakan untuk mengerjakan output horizontal. Dimana pada saat x=5 maka akan mengerjakan output mulai dari y=1 sampai y=x (dimana x=5). Kemudian pada saat x=4 maka akan mengerjakan output mulai dari y=1 sampai y=x (dimana x=4). Kemudian pada saat x=3 maka akan mengerjakan output mulai dari y=1 sampai y=x (dimana x=3). Kemudian pada saat x=2 maka akan mengerjakan output mulai dari y=1 sampai y=x (dimana x=2). Kemudian pada saat x=1 maka akan mengerjakan output mulai dari y=1 sampai y=x (dimana x=1). Ketika x=0 maka program akan berhenti karena kondisi x tidak memenuhi lagi sehingga program selesai dijalankan.

Latihan:

1. Buatlah program untuk mencetak deret 10 8 6 4 2 (menggunakan While dan Do-While) !
2. Buatlah program untuk hasil tampilan dibawah ini !



```
masukkan nilai deret = 10
*****
*****
*****
*****
*****
*****
*****
*****
****
****
***
***
**
**
*
```

BAB V

FUNGSI DAN PROCEDURE

5.1. Fungsi

Function/fungsi adalah satu blok kode yang melakukan tugas tertentu atau satu blok instruksi yang di eksekusi ketika dipanggil dari bagian lain dalam suatu program.

Keuntungan modularisasi program :

1. Menghindari penulisan teks program yang sama secara berulang kali.
2. Kemudahan menulis dan menemukan kesalahan (debug) program.

Bentuk umum deklarasi fungsi:

```
TipeNilaiBalik fungsi (tipeparameter, ... ) ;
```

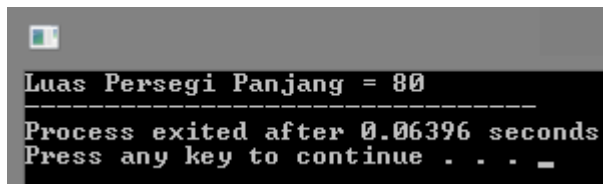
Keterangan :

- **tipeNilaiBalik** = Tipe nilai yang dikembalikan dengan statemen 'return'.
Tipe default nya : 'int'. Untuk menyatakan fungsi yang tidak mengembalikan nilai balik, dideklarasikan sebagai : 'void'.
- **fungsi** = nama fungsi tersebut
- **tipeparameter** = Tipe parameter, bila parameter lebih dari satu (1), masing-masing dipisahkan dengan tanda koma (,) untuk menyatakan fungsi tanpa parameter dispesifikasikan : 'void'. Bila tipe parameter tidak dispesifikasikan, defaultnya : 'void' Fungsi harus dideklarasikan terlebih dahulu sebelum didefinisikan. Maksudnya adalah memberitahu compiler jumlah dan tipe parameter yang diterima dan nilai balik fungsi (bila ada) agar compiler dapat memeriksa ketepatannya. **Definisi fungsi** itu sendiri adalah menspesifikasikan tugas fungsi tersebut.

Contoh:

```
#include <iostream>
#include <conio.h>
using namespace std;
int Luas (int a, int b )
{
    int L;
    L = a * b;
}
int main ( )
{
    int x ;
    x = Luas (8,10) ;
    cout << "Luas Persegi Panjang = "<< x ;
    return 0 ;
}
```

Hasil Tampilan :



```
Luas Persegi Panjang = 80
-----
Process exited after 0.06396 seconds
Press any key to continue . . . _
```

5.2. Jenis-jenis Fungsi :

1. Fungsi dengan Nilai Balik

Bentuk umumnya:

```
tipe_nilai_balik nama_fungsi(tipe_parameterA,
tipe_parameterB,...)
{
    pernyataan_1;
    ...
    pernyataan_n;
    return nilai_balik;
}
tipe_nilai_balik nama_fungsi(tipe_parameterA,
tipe_parameterB, ...)
```

Dari bentuk umum tersebut, ada 3 poin penting yang perlu dijelaskan, yaitu:

a. tipe_nilai_balik

Menentukan tipe nilai yang diberikan oleh fungsi ketika fungsi dipanggil. Nilai balik ditentukan melalui pernyataan *return*.

b. nama_fungsi

Digunakan untuk penginisialan fungsi yang akan dibuat.

c. parameter

Digunakan untuk melewati nilai ke fungsi. Antar parameter dipisahkan oleh tanda koma (.). Jika tak ada parameter, judul fungsi berupa:

```
tipe_nilai_balik nama_fungsi()
```

Contoh:

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

Manfaat yang bisa kita gunakan dalam fungsi dengan nilai balik adalah kita dapat melakukan *cout* terhadap hasil kerja dari fungsi tersebut dengan memanfaatkan nilai balik tersebut. Contohnya adalah pada program dibawah ini:

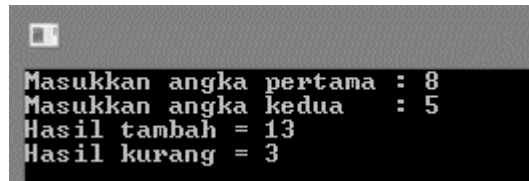
```
#include <iostream>
#include <conio.h>
using namespace std;

int tambah(int a1, int a2); //inisialisasi fungsi tambah
int kurang(int b1, int b2); //inisialisasi fungsi kurang
main()
{
    int a,b,t,k;
    cout<<"Masukkan angka pertama : ";cin>>a;
    cout<<"Masukkan angka kedua   : ";cin>>b;
    t=tambah(a,b); //memasukkan angka ke fungsi t
    k=kurang(a,b); //memasukkan angka ke fungsi k
    cout<<"Hasil tambah = "<<t<<"\n"; //akan muncul
    hasilnya
    cout<<"Hasil kurang = "<<k<<"\n"; //dari nilai yang
    dibalikkan
    getch();
}

int tambah(int a1, int a2) //angka yang dimasukkan
diproses
{
    int tmbh;
    tmbh=a1+a2;
    return tmbh; //hasil rumus akan dibalikkan ke fungsi
}

int kurang(int b1, int b2)
{
    int krng;
    krng=b1-b2;
    return krng;
}
```

Hasil Tampilan :



```
Masukkan angka pertama : 8
Masukkan angka kedua   : 5
Hasil tambah = 13
Hasil kurang = 3
```

Jika anda menulis program tersebut, hasilnya akan sama dengan program sebelumnya. Namun yang membedakan adalah cara penulisan program. Di program ini, tiap rumus memiliki tubuh/bagian tersendiri dan terpisah. Sehingga untuk merubahnya sangat mudah, jika kita memiliki rumus penghitungan yang banyak.

Perlu diingat, bahwa program C++ membaca program selalu dimulai dari main. Inisialisasi fungsi di awal diperlukan, karena sebagai pemberitahuan bahwa ada fungsi diluar dari main. Sehingga, saat eksekusi main, fungsi yang tertera di main akan dieksekusi, walaupun fungsi tersebut diluar dari main.

2. Fungsi tanpa Nilai Balik

Fungsi dengan tipe void berarti tidak memiliki nilai balik.

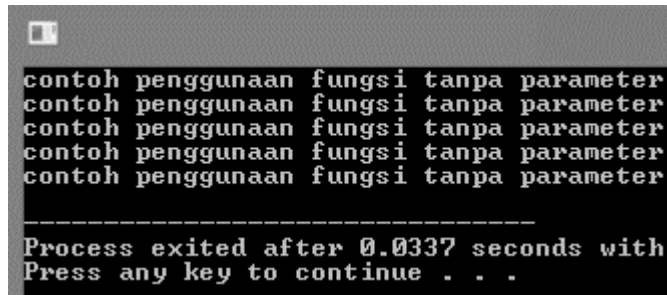
Contoh:

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

void perulangan()
{
    for (int x=0;x<5;x++)
    {
        cout<<"contoh penggunaan fungsi tanpa parameter \n";
    }
}

int main()
{
    perulangan();
}
```

Hasil Tampilan :



```
contoh penggunaan fungsi tanpa parameter
contoh penggunaan fungsi tanpa parameter
contoh penggunaan fungsi tanpa parameter
contoh penggunaan fungsi tanpa parameter
contoh penggunaan fungsi tanpa parameter
-----
Process exited after 0.0337 seconds with
Press any key to continue . . .
```

Pada contoh diatas, kita memiliki sebuah fungsi void pada *perulangan*. Disana, tidak ada statement *return*. Contoh lainnya supaya dapat lebih mengerti, dapat disimak pada Program A ini:

```
#include <iostream>
using namespace std;
void garis(int n); //void dengan paramater
main()
{
    garis(25);
}
void garis(int n)
{
    int a;
    for (a=1; a<=n; a++) //batas ditentukan dari parameter
        cout<<'*';
    cout<<'\n';
}
```

Hasil Tampilan :



Kedua program tersebut akan menghasilkan hasil yang sama. Perbedaan dari kedua program tersebut adalah program A memiliki parameter pada fungsi tanpa nilai baliknya. Parameter tersebut digunakan sebagai pembantu dari looping for. Sedangkan pada Program B, program tersebut tidak memiliki parameter (void). Sehingga, untuk membantu program looping for, user harus memasukkannya ke program.

BAB VI

ARRAY

7.1. Array

Array adalah sebuah variabel yang menyimpan sekumpulan data yang memiliki tipe sama .setiap data tersebut menempati lokasi atau alamat memory yang berbeda-beda dan selanjutnya di sebut dengan elemen array. Elemen array tersebut kemudian dapat kita akses melalui indeks yang terdapat di dalamnya namun penting sekali untk di perhatikan bahwa dalam C++, Indeks array selalu di mualai dari 0, bukan 1.

Untuk mendeklarasian sebuah array kita harus menggunakan tanda [..](bracket) Bentuk umum dari variabel array dapat ditulis seperti dibawah ini :

```
Tipe_data nama_variabel[indeks]; //ketentuan  
int a[5]; //contoh
```

Pada contoh diatas, data yang dapat ditampung yaitu 5 data. Indeks array selalu dimulai dari angka 0. Sehingga pada kasus diatas, data yang tersimpan dimulai dari indeks ke 0 hingga 4.

7.1.1. Array 1 Dimensi

Contoh sebelumnya yang kita praktekkan sebelumnya yaitu merupakan array 1 dimensi. Cara diatas merupakan cara untuk melakukan inisialisasi. Untuk mengisikan data ke dalam element-element array kita dapat melakukannya langsung untuk setiap element , contohnya :

```
A[0]=1  
A[1]=2  
A[2]=3  
A[3]=4  
.....  
Dst
```

Namun cara di atas tidak efisien karena jika memerlukan data yang banyak harus menuliskan script yang banyak juga. Cara yang lebih umum dan banyak di gunakan untuk mengisikan nilai ke dalam element array adalah dengan menggunakan perulangan (looping).

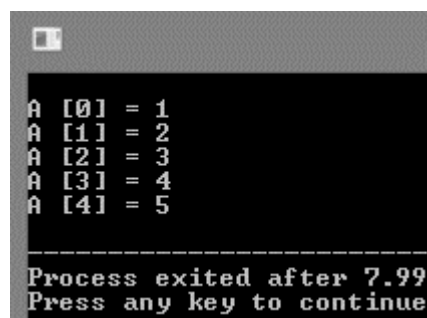
Contoh jika ingin mengisi 5 elemen array ,maka kita dapat menulis sintaks seperti berikut :

```
for(int j=0; j<5; j++){  
    cout<<"A["<<j<<" ] = ";cin>>a[j];  
}
```

Contoh :

```
#include <iostream>  
using namespace std;  
  
int main()  
{  
    int a[5];  
    int j;  
    // mengisi nilai ke dalam element array  
    cout<<endl;  
    for(j=0;j<5;j++)  
    {  
        cout<<"A ["<<j<<" ] = ";cin>>a[j];  
    }  
}
```

Hasil Tampilan:

A screenshot of a terminal window showing the output of a C++ program. The output displays the initialization of an array 'A' with 5 elements. Each element is assigned a value from 1 to 5. The output is as follows:
A [0] = 1
A [1] = 2
A [2] = 3
A [3] = 4
A [4] = 5

Process exited after 7.99
Press any key to continue

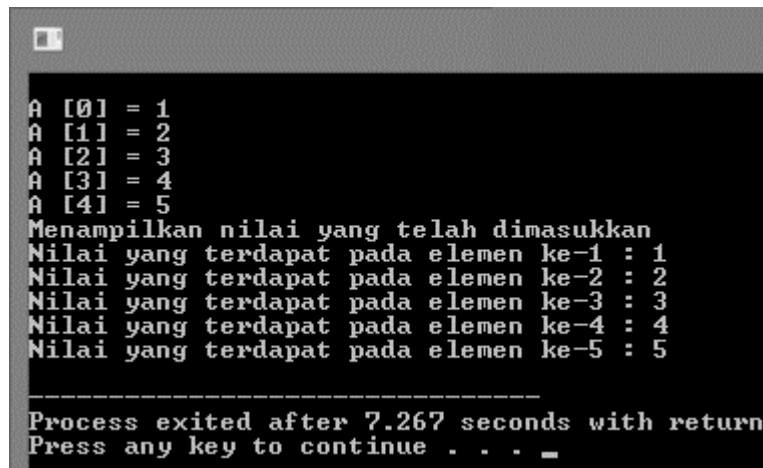
Pada program diatas terdapat deklarasi array yang bertipe data integer dengan jumlah array sebanyak 5. Untuk melakukan input data disertakan perulangan yang dimulai dari 0 sampai <5 yang berarti dari 0 sampai 4.

Setelah memahami cara mengisi nilai ke dalam elemen array, sekarang kita memahami cara menampilkan nilai-nilai yang telah di inputkan ke dalam elemn-elemen tsb. Konsepnya sama, kita

akan menggunakan perulangan untuk menampilkannya. Tambahkan syntag berikut pada program diatas :

```
cout<<"Menampilkan nilai yang telah dimasukkan"<<endl;
for (int j=0; j<5; j++) {
    cout<<"Nilai yang terdapat pada elemen ke-";
    cout<<j+1<<" : "<<a[j]<<endl;
}
```

Hasil tampilan :

A screenshot of a terminal window showing the output of a C++ program. The output displays the initialization of an array 'A' with 5 elements, followed by a loop that prints each element's value. The output is as follows:

```
A [0] = 1
A [1] = 2
A [2] = 3
A [3] = 4
A [4] = 5
Menampilkan nilai yang telah dimasukkan
Nilai yang terdapat pada elemen ke-1 : 1
Nilai yang terdapat pada elemen ke-2 : 2
Nilai yang terdapat pada elemen ke-3 : 3
Nilai yang terdapat pada elemen ke-4 : 4
Nilai yang terdapat pada elemen ke-5 : 5
-----
Process exited after 7.267 seconds with return
Press any key to continue . . . _
```

7.1.2. Array 2 Dimensi

Hampir sama dengan array 1 dimensi, yang membuat array 2 dimensi berbeda adalah daya tampung datanya. Jika di array 1 dimensi, kita perlu inisialisasi 1 angka untuk menunjukkan berapa data yang bisa dimasukkan. Di array 2 dimensi, kita bisa menginisialisasikan 2 angka. Array seperti ini sering di gunakan untuk pemerosesan matrik, Bentuk umum dari array 2 dimensi adalah sebagai berikut :

```
tipedata nama_array[jmlh baris][jmlah kolom];
float bil[2][5];
```

Ilustrasi table array sebagai berikut :

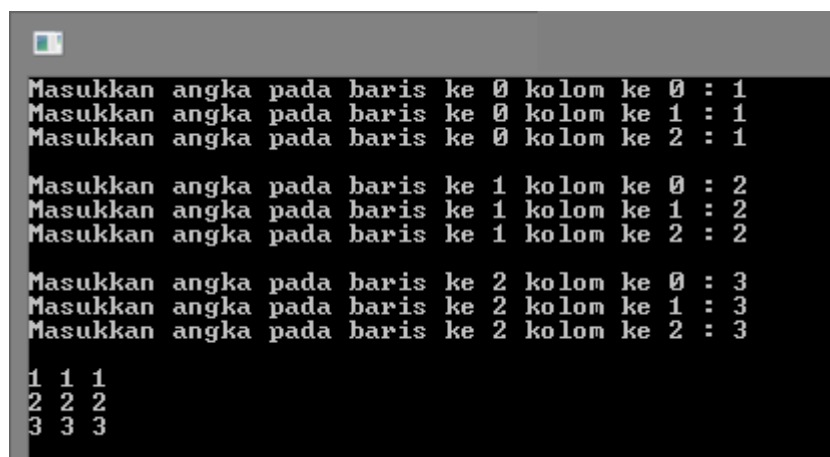
[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]

Contoh diatas menunjukkan bahwa kita dapat mengisi data dari array 00, 01, 02, 03, 04, 10, 11 dan seterusnya. Sehingga, data yang diisi totalnya menjadi $2 \times 5 = 10$ data.

Contoh :

```
#include <iostream>
using namespace std;
main()
{
    int matrix[3][3]; //inisialisasi awal
    int i,j;
    for(i=0;i<=2;i++)
    {
        for(j=0;j<=2;j++)
        {
            cout<<"Masukkan angka pada baris ke
"<<i<<" kolom ke "<<j<<" : ";
            cin>>matrix[i][j];
        }
        cout<<endl;
    }
    for(i=0;i<=2;i++)
    {
        for(j=0;j<=2;j++)
        {
            cout<<matrix[i][j]<<" ";
        }
        cout<<endl;
    }
}
```

Hasil Tampilan :



```
Masukkan angka pada baris ke 0 kolom ke 0 : 1
Masukkan angka pada baris ke 0 kolom ke 1 : 1
Masukkan angka pada baris ke 0 kolom ke 2 : 1

Masukkan angka pada baris ke 1 kolom ke 0 : 2
Masukkan angka pada baris ke 1 kolom ke 1 : 2
Masukkan angka pada baris ke 1 kolom ke 2 : 2

Masukkan angka pada baris ke 2 kolom ke 0 : 3
Masukkan angka pada baris ke 2 kolom ke 1 : 3
Masukkan angka pada baris ke 2 kolom ke 2 : 3

1 1 1
2 2 2
3 3 3
```

Contoh :

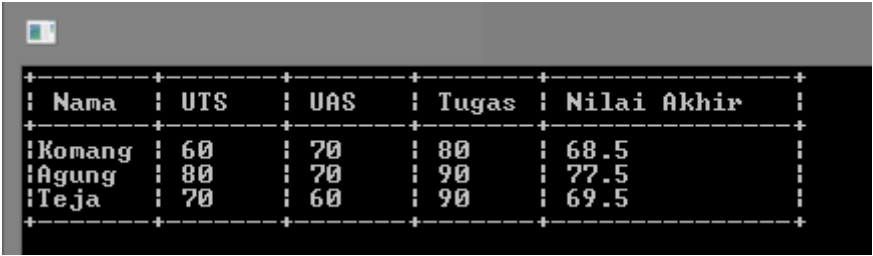
Menghitung nilai UTS, UAS, Tugas dan Nilai Akhir menggunakan array 1 dimensi dan array 2 dimensi.

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

main()
{
    string nama[3] = {"Komang", "Agung", "Teja"};
    float      nilai[3][3] =
    {{60, 70, 80}, {80, 70, 90}, {70, 60, 90}};
    float nilai_akhir=0;

    cout<<"-----+-----+-----+-----+-----+
-----+"<<endl;
    cout<<"| Nama\t| UTS\t| UAS\t| Tugas\t| Nilai
Akhir\t|"<<endl;
    cout<<"-----+-----+-----+-----+-----+
-----+"<<endl;
    for(int i=0; i<3; i++)
    {
        cout<<"| "<<nama[i]<<"\t| ";
        for(int j=0; j<3; j++)
        {
            cout<<nilai[i][j]<<"\t| ";
        }
        nilai_akhir = (0.35 * nilai[i][0])+(0.45 *
nilai[i][1])+(0.2 * nilai[i][2]);
        cout<<nilai_akhir<<"\t\t| ";
        cout<<endl;
    }
    cout<<"-----+-----+-----+-----+-----+
-----+"<<endl;
}
```

Hasil Tampilan :



Nama	UTS	UAS	Tugas	Nilai Akhir
Komang	60	70	80	68.5
Agung	80	70	90	77.5
Teja	70	60	90	69.5