UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

AlgoritmadanPemrograman

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	01	MK87001	Tim Dosen

Abstract

Moduliniberisitentangdefenisidasardari program, algoritma, pemrograman, notasialgoritma, besertabeberapacontohsoalterkaitmateri

Kompetensi

Diharapkanmahasiswadapat:
- Mengetahuidefenisidari program, algoritma, pemrograman.
- Menyelesaikanmasalah-masalahdasardenganalgoritma yang benar

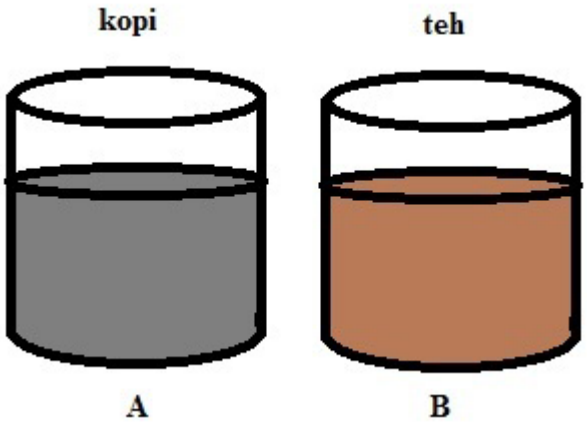
Pendahuluan

Defenisi

- Pemrogramanmerupakansuatukegiatanmerancangataumenulissebuahprogram.
- Program adalahSuatualgoritmayang ditulisdalambahasakomputer, biasanyaterdiridarikumpulaninstruksi-instruksi.
- Bahasakomputer yang digunakanuntukmenulis program disebutdenganbahasapemrograman.
- Algoritmamerupakanalurpikiranatauurutanlangkah-langkahdalammenyelesaikansebuahmasalah.
- Masalahataupersoalanmerupakansesuatu yang bisaberupapertanyaanatautugas yang dicarijawabannya.

ContohMasalah 1.1

Tedapat 2 buahgelas, yaituGelas A yang berisi Kopi, danGelas B yang berisiTeh.Dimanaditugaskanuntukmenukarisidarikeduagelastersebut, sehinggaTehberpindahkeGelas A dan kopi keGelas B.

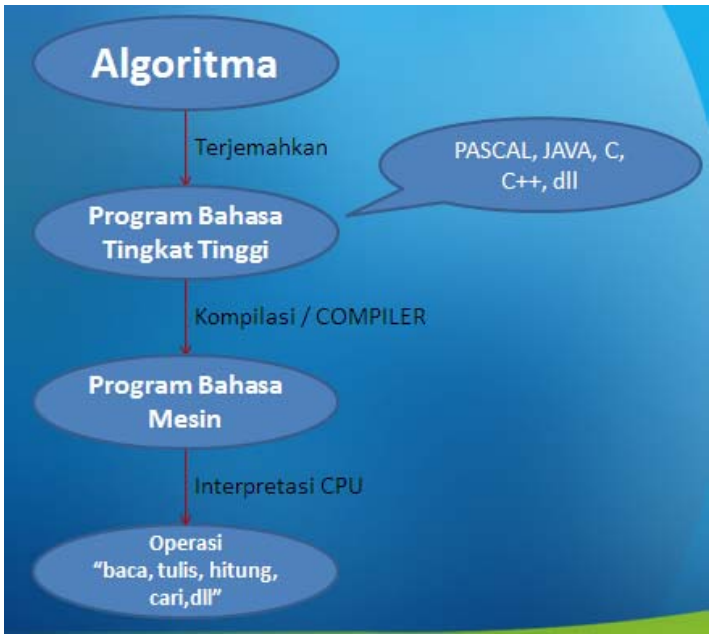


Gambar 1.1.2 Gelas A dan B

Solusinyaadalah ?

TahapanPelaksanaan Program

Ada beberapatahapan yang harusdilaluiolehsebuah program, sampaiakhirnyadieksekusioleh CPU, yaitusepertipadaGambar 1.2.



Gambar 1.2 TahapanPelaksanaan Program

Keterangan

1. **Tahap 1**

   Sebuah program berisiurutanlangkah-langkahdalammenyelesaikansebuahmasalah, ataudisebutjugasebagaisebuahalgoritma.Dimanaalgoritmaininantinyadisajikandalamberbagaibentukataudisebutsebagainotasialgoritmik, antara lain adalahsebagaiberikut:

   - KalimatDeskriptif

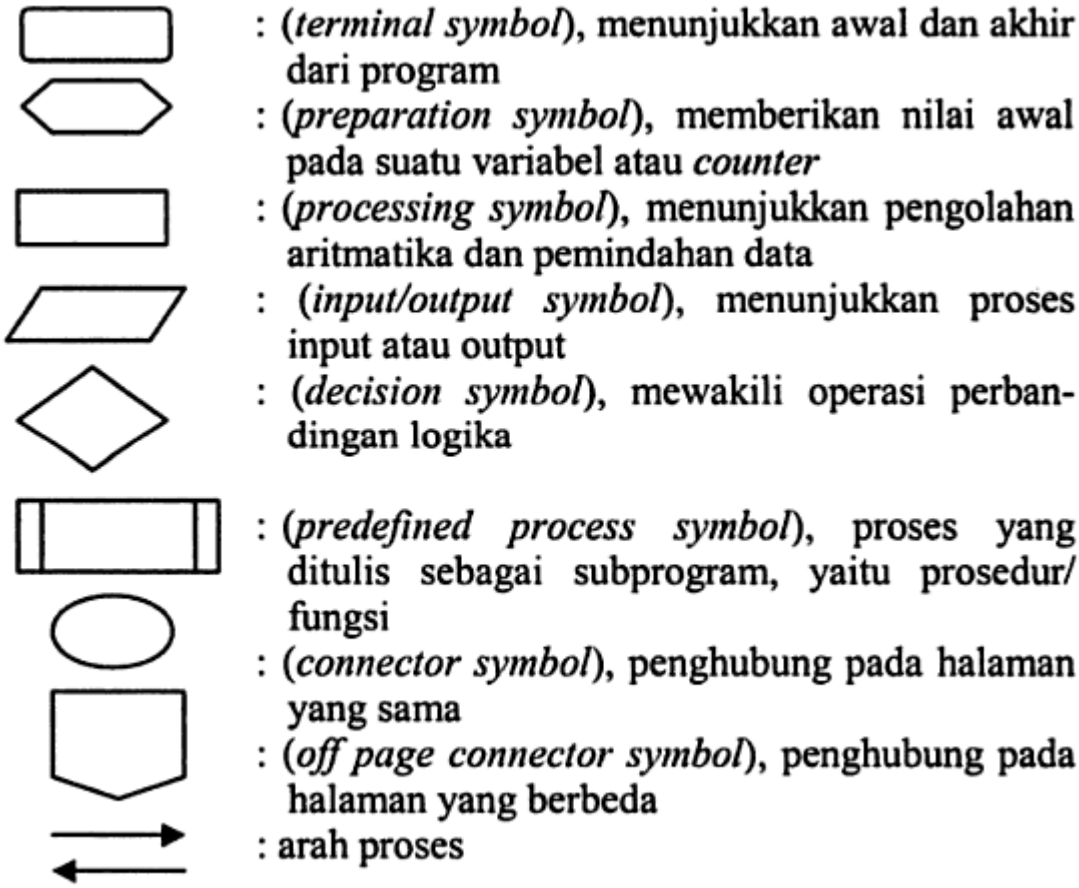
     Notasiinimenyatakanlangkah-langkahalgoritmadalambentukkalimatdeskriptif.PadaContohMasalah 1.1, dapatkitaselesaikandenganurutanalgoritmasebagaiberikut.

     - Siapkan 1 Gelaskosong C
     - Tuangkanisigelas A (Kopi) kegelas C
     - Tuangkanisigelas B (Teh) kegelas A
     - Tuangkanisigelas C (Kopi) kegelas B

- Flowchart

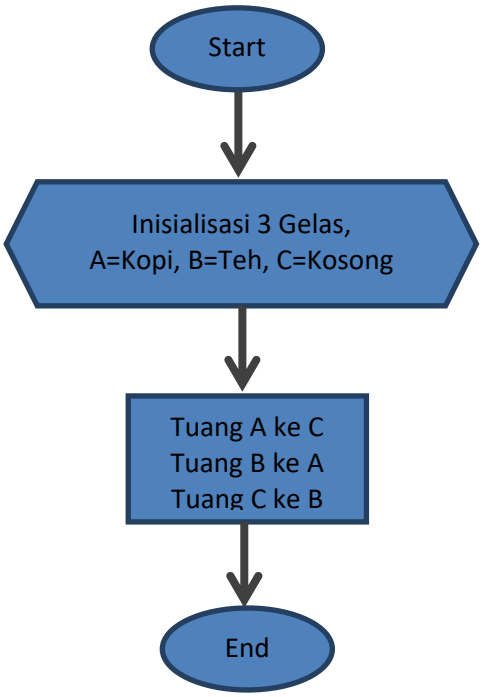
  Merupakansebuahbaganalir, yang menggambarkanaliraninstruksidarisebuahalgoritmadalambentukgeometri. Flowchart cocokdigunakanuntukmenggambarkanalgoritmapadamasalahkecil, sedangkanuntukmasalahbesar, kurangcocok, karenaakanmemerlukanberlembar-lembarkertasuntukmenyajikanseluruhalgoritmanya.

  Notasi-notasipada flowchart adalahsebagaiberikut.



Gambar 1.3 Notasi Flow chart

Flowchart dariContohMasalah1.1, dapatdilihatpadaGambar 1.4



Gambar 1.4 Flowchart ContohMasalah 1.1

- Pseudo Code

  Pseudo artinyaadalahpura-puramenyerupai, sehingga pseudo code adalahsebuahkode/tanda/ceritera yang menyerupaipenjelasancarauntukmenyelesaikanmasalah, ditulisdalambahasayang mendekatibahasapemrogramantingkattinggiataumerupakancampuranbahasaman usiadengansebuahbahasapemrograman.

  Pseudo code dariContohMasalah 1.1, adalahsebagaiberikut.

```
Deklarasi :
Gelas_A, Gelas_B, Gelas_C : string

Algoritma:
Gelas_A=”Kopi”
Gelas_B=”Teh"
Gelas_C=”Kosong”
Gelas_C = Gelas_A
Gelas_A=Gelas_B
Gelas_B=Gelas_C
```

2. **Tahap 2**

   Setelahalgoritmaditulis, makatahapanberikutnyaadalah programmer akanmenterjemahkanalgoritmatadi, kedalamsebuahBahasaPemrogramantinggi.

   Jenisbahasapemrogramansangatbanyakdanberagam, berdasarkantingkatkanbahasanya, bahasapemrogramandapatdibagimenjadi:

   - Bahasatingkatrendah

     Bahasainidirancang, agar setiapinstruksinyalangsungdikerjakanoleh computer, tanpaharusmelalui translator.Contohnyaadalahbahasamesin..Bahasamesinmerupakansekumpulank odebiner(0 dan 1).
   - Bahasatingkattinggi

     Bahasatingkattinggi, membuatsebuah program menjadilebihmudah, karenabahasanyamendekatibahasamanusia (bahasainggris), sehinggalebihmudahdipahami.Tetapi program yang ditulisdenganbahasatingkattinggi, tidakdapatlangsungdilaksanakanoleh computer, harusditerjemahkanterlebihdahuluolehsebuah translator bahasapemrograman yang disebut compiler, kedalambahasamesin.

     Contoh : Pascal, Java, C, C++, Visual Basic, Fortran, dll.

3. **Tahap 3**

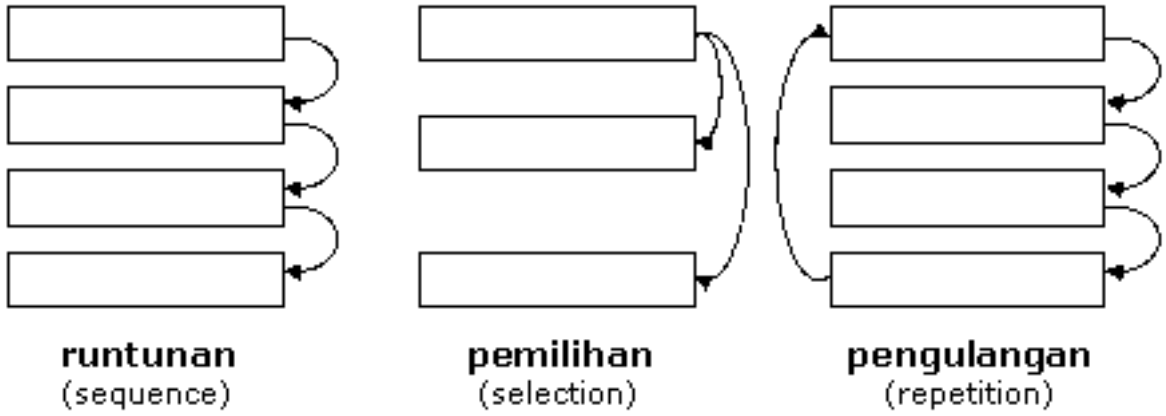
   Setelah program selesaidibuat, program tersebutakan di kompilasi. Padatahapaninisetiapinstruksi yang ditulisakandicekstruturdanpenulisannya, apakahsudahsesuaidenganaturandaribahasapemrogramman yang digunakanatautidak. Jikasudahbenar, makainstruksidalambahasatingkattinggi, akanditerjemahkandalambahasaMesin

4. **Tahap 4**

   Setelah program menjadibahasamesin, makainstruksinyadapatdipahamioleh CPU, dandieksekusi, sehingganantinyadapatmenjadioperasibaca, tulis, hitungperbandingan, dan lain-lain.

KonstruksiDasarAlgoritma

Algoritmamerupakanurutaninstruksi-instruksidalammenyelesaikansebuahmasalah.Urutaninstruksi-instruksiinidapatberupasebuahruntunanaksi, pemilihanaksiataupengulanganaksi.Sehinggasebuahalgoritmadapatdibangundari Runtunan (sequence), pemilihan (selection) danpengulangan (looping), yang dapatdilihatpadaGambar 1.5.



Gambar 1.5.KonstruksidasarAlgoritma

1. Runtunan (Sequence)

Sebuahruntunanterdiridarisatuataulebihpernyataan/instruksi yang dikerjakansecaraberurutan, dimana1 instruksidilaksanakansetelahinstruksiberikutnya.Jikaurutandirubah, makahasilbisaberubah.

Contoh.

PadaContohMasalah 1.1, urutanalgoritmanyaadalahdenganruntunaninstruksi 1,2,3,4, dieksekusisecaraber-urutan

1. Siapkan 1 Gelaskosong C
2. Tuangkanisigelas A (Kopi) kegelas C
3. Tuangkanisigelas B (Teh) kegelas A
4. Tuangkanisigelas C (Kopi) kegelas B

Jikaruntunannyakitagantimenjadi 1,2,4,3, makahasildarimasalahinitidaktercapai, karenakitatidakmenukarisikeduaGelas A dan B, tetapimencampurkanisikeduagelastersebut.

2. Pemilihan (Selection)

Dalampemilihan, sebuahInstruksi/kumpulandikerjakanjikakondisitertentudipenuhi.Biasanyadinyatakandengani nstruksiIF-THEN (Jika – Maka).

Contoh:Kendaraan di traffic light

- JikaLampu Traffic Light berwarnaMerah MakakendaraanberhentiJikaLampu Traffic Light berwarnaKuning MakakendaraanJalanhati-hati
- JikaLampu Traffic Light berwarnaHijau MakakendaraanJalan

3. Pengulangan (Looping)

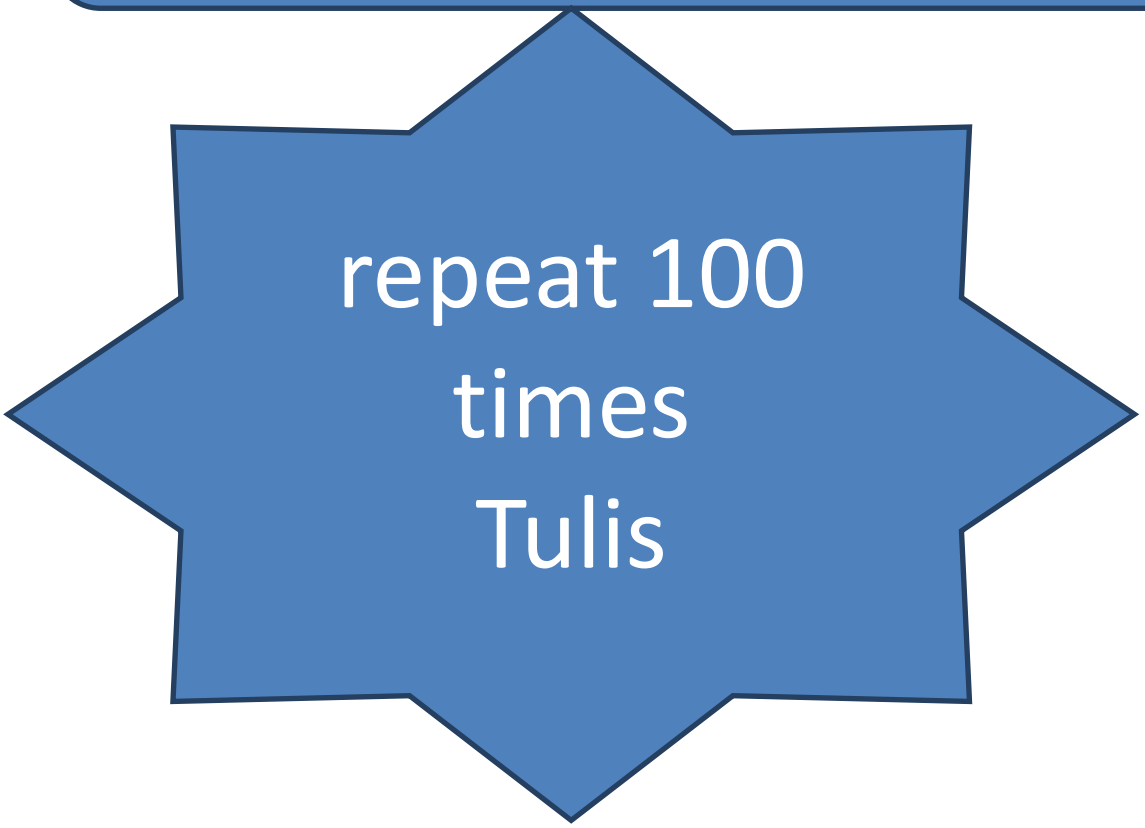
Dalampengulangan, makasebuahinstruksiataukumpulaninstruksiakandieksekusisecaraberulang, dalamhitungan yang terbatas (finite). Kelebihankomputeradalahmampumelakukanpekerjaan yang samaberulang kali tanpakenallelah.

Contoh :

Menulisperjanjian 100 kali “Sayaberjanjitidakakanmalaslagi”

- BagaimanaAlgoritmanya ?

1. Menulispadabarispertamakalimat “Sayaberjanjitidakakanmalaslagi”
2. Dilulangpadabariskedua, menuliskalimat “Sayaberjanjitidakakanmalaslagi”
3. Diulangpadabarisketiga – seratus, menuliskalimat “Sayaberjanjitidakakanmalaslagi”



DaftarPustaka

1. Kristianto. Andri, AlgoritmadanPemrogramandengan C++ Edisi 3, Yokyakarta, GrahaIlmu, 2013
2. Munir. Rinaldi, AlgoritmadanPemrogramanDalamBahasa Pascal dan C, Bandung Informatika, 2007
3. Sjukani. MohAlgoritma (AlgoritmadanStruktur Data1) dengan C, C++, dan Java, Jakarta, MitraKencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial

UNIVERSITAS
MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

PengenalanBahasa C++

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	02	MK87001	Tim Dosen

Abstract

Moduliniberisitentangstruktur program C++, VariabelKonstanta, danTipe data, instruksi output pada C++

Kompetensi

Diharapkanmahasiswadapat:
- Mengetahuistrukturprogram C++
- Mendefenisikannamavariabeldankonstantan, sertatipe data
- Membuatsebuah program sederhanadanmencetakhasilnyakelayar.

Program danBahasaPemrograman C++

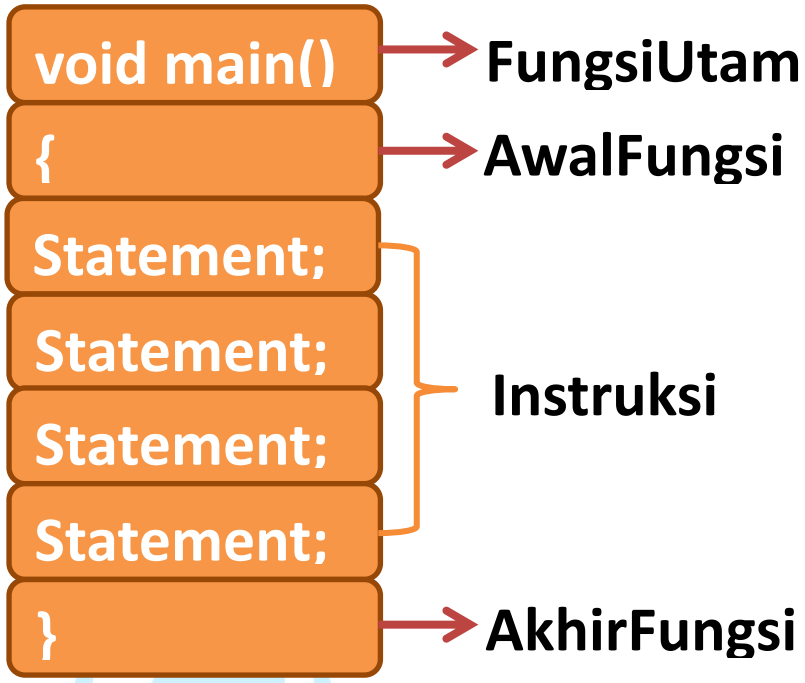
Defenisi

Program adalahSuatualgoritmayang ditulisdalambahasakomputer, biasanyaterdiridari 1 ataukumpulaninstruksi-instruksi.Dimanainstruksi-instruksitersebutharusbersifatlogis (instruksi-instruksitersebutdapatdikerjakandenganbenar).

Sebuahprogram yangditulisdalamsebuahbahasa (bahasapemrogram). Dimanabahasapemrograman yang akandipakaidalammatakuliahiniadalahsebuahbahasapemrogramantingkattinggiyaitu C++.

StrukturBahasaPemrograman C++

Instruksi-instruksi yang ditulisdalambahasapemrograman C++, ditulisdalamsebuahkelompok yang disebutdengan*function* ataufungsi.Dimanasebuahfungsi, bisaterdiridari 1 ataubanyakinstruksi.Sebuah program dalambahasapemrograman C++ harusmemiliki 1 fungsi minimal yaitufungsiutama (*main function*).

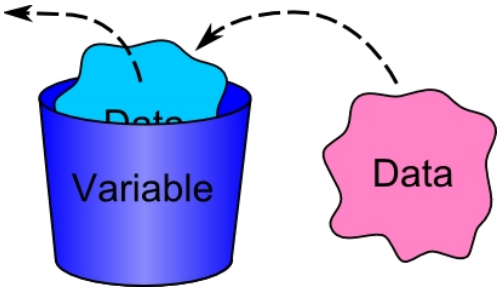


Gambar 2.1 StrukturBahasaPemrograman C++

Variabel, KonstantadanTipe Data

Variabel

Defenisidarisebuahvariabeladalahsebuahlokasidimemorykomputer, dimanakitadapatmenyimpandanmengambilsebuahnilai.



Gambar 2.2.Ilustrasivariable

Agar sebuahvariabeldikenalidenganmudah, makasepertihalnyabendaataumanusia, yang untukindetitasnyamenggunakannama, makasebuahvariabeljugaharusmempunyainama. Untukmendefenisikannamasebuahvariabel, makapersyaratan yang harusdipenuhiadalahsebagaiberikut:

- **Unik**
  Defenisinamauntuksebuahvariabel, harusbersifatunik. Dimanaantara 1 variabeldenganvariabellainnyatidakdiizinkanuntukmemilikinama yang sama.

- **Tidakbolehsamadengan keyword**
  Keyword adalah kata kunci yang digunakandalampemrogramanuntukfungsi-fungsitertentu.Dalampemrogramanc++ keyword dikenalioleh compiler dengancaramendeklarasikan file headernya, tetapiadajuga keyword yang tidakperluuntukmendeklarasikan file header. Keyword merupakanpernyataansingkatdanmempunyaifungsitertentusehinggasangatfleksibeldanmudahdalampenggunaandalampemrograman.

Berikutmerupakan keyword dasardalamc++.

- C++ mempunyai 32 buah kata kuncikelompokpertama yang merupakanturunandaribahasa C, di antaranya:

auto	const	double	float	int	short	struct	unsigned
break	continue	else	for	long	signed	switch	void
case	default	enum	goto	register	sizeof	typedef	volatile
char	do	extern	if	return	static	union	while

- Kata kuncikelompokkeduaberjumlah 30. Kata-kata iniadalahbarudanhanyaada di bahasa C++.

asm	dynamic_cast	namespace	reinterpret_cast	try
bool	explicit	new	static_cast	typeid
catch	false	operator	template	typename
class	friend	private	this	using
const_cast	inline	public	throw	virtual
delete	mutable	protected	true	wchar_t

- **Kombinasihurufkecil/besar, angka, underscore**

  Namasebuahvariabelhanyabolehmenggunakankombinasidarihurufkecil (a,b,…,z), atauhurufbesar (A,B,…,Z), atauangka (0,1,…,9), atau underscore (_).

- **Karakter 1 ➔huruf/underscore**

  Untukkarakterpertamadarisebuahvariabel, harusdimulaidenganhuruf (besar/kecil) dan underscore (_). Tidakdiizinkanuntukmenggunakanangka/simbolatauselainhurufdan underscore, padakarakterpertamadarinamasebuahvariabel.

- **Tidakmenggunakanspasi**

  Dalampenamaanvaribeljugatidakdizinkanmenggunakanspasi.

- **Case sensitive**

  Bahasapemrograman C++ bersifat case sensisitive, dimanahurufbesardanhurufkecildianggapberbedaolehcompilernya.

Berikutadalahcontoh-contohnama – namavariabel yang diizinkanatautidakdiizinkan.

CONTOH BENAR
1. Nilai
2. NILAI
3. nilai
4. _nilai
5. Nilai_Akhir
6. NilaiAkhir

CONTOH SALAH
1. NilaiAkhir
2. 1Nilai
3. Public
4. Nilai-Akhir

KeteranganContohBenar

1. Contoh 1,2dan 3 walaupunkatanyasama, tetapidianggapsebagai 3 variabelberbeda, karena Case Sensitive.

KeteranganContoh Salah

1. NilaiAkhir, salahkarenamenggunakanspasi
2. 1Nilai, salahkarenadimulaidenganangka
3. Public, salahkarenamenggunakan kata-kata pada keyword
4. Nilai-akhir, salahkarenamenggunakantanda minus

Konstanta

Konstantamerupakansebuahlokasimemoridimanasebuahnilaidisimpan, sepertihalnyasebuahvariabel, tetapimemilikiperbedaan, dimanavariabelnilainyabisaberubah, bergantungkepadainstruksi yang diberikan, sedangkankonstantanilainyaadalahtetap.Konstantaharusdidefenisikanpadaawal program.

Salah satucarauntukmendefenisikankonstantaadalahdengancara literal. Literal adalahmemberikansebuahnilaisecaralangsungdalamsebuah program.

Contohkonstanta:

Phi=3.14

Tipe Data

Setiap data yang digunakan, baik yang disimpandalambentukvariabel, maupun yang disimpanbukandalamsebuahvariabel, mempunyaitipe-tipetertentu, sepertiberjenis 1 karaktersaja, kumpulankarakterdannumerik.

Pada C++ adabeberapatipe data dasar yang digunakan.SepertiTabel 2.1 berikut.

Tipe data	Penulisan	Jumlah Byte	Jangkauannilai
Karakter	char atau signed char	1	-128 s.d. 127
	unsigned char	1	0 s.d 255
Integer (bilanganbulat)	Intatau signed intatau signed	2	-32768 - 32767
	unsigned intatau unsigned	2	
	Signed Long int atau atau signed long, long int atau long	2	2147483648 s/d 2147483647 (2.1 Milyar)
	Unsigned long int atau unsigned long	2	0 – 4294967295 (4.2 Milyar
Bilangan pecahan	Float	4	3.4E-38 – 3.4E38
	Double	8	1.7E-308-1.7E308 -1.7E-308- -1.7E308
	Long double	10	3.4E-4932-1.14E4932 -1.1E4932—3.4E4932

Contoh

1. Soal

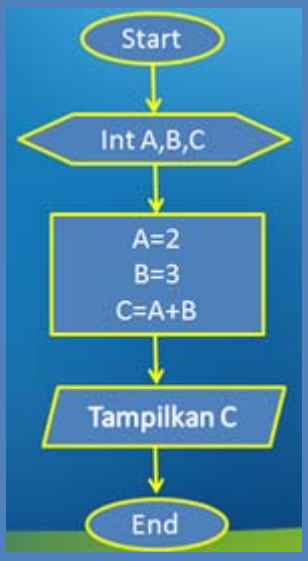
Buatlah pseudo code, flowchart dan program untukmenjumlahkan 2 bilanganyaitubilangan 2 dan 3

Jawab.

Pseudo code

1. **Siapkan 3 variabel, bertipe data int (integer) ➔ A, B, C**
2. **Isi A = 2**
3. **Isi B = 3**
4. **Jumlahkan A+B, simpanke C**
5. **Tampilkannilai C**

Flowchart



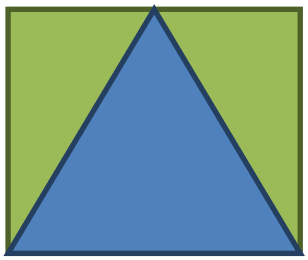
Program

```
#include <iostream.h>
void main()
{
intA,B,C;
A=2;
B=3;
C=A+B;
cout<<"HasilJumlah 2+3="<<C;
}
```

2. Soal

Buat pseudo code ,flowchart dan program untukkasusdibawahini:

Diketahuisebuahbangundatardenganbentuksepertiberikut:



DenganPanjangsisi = 8

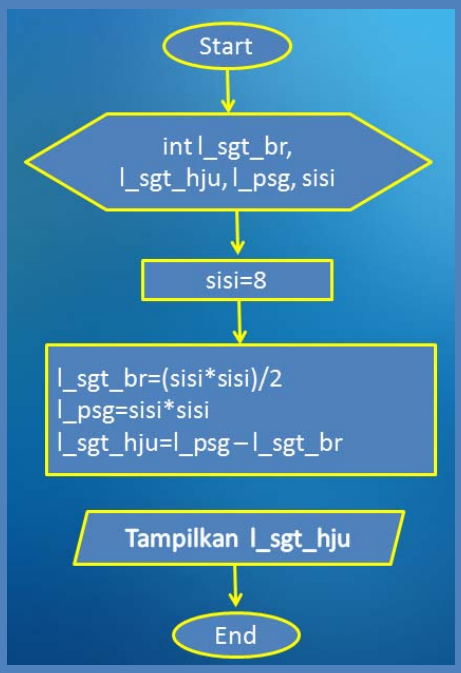
Hitunglahluasdaerahberwanahijau

Jawab

Pseudo code

1. **Siapkan 3 variabel, bertipe data int (integer) ➔sisi, l_sgt_br, l_psg, l_sgt_hju**
2. **sisi=8**
3. **l_sgt_br=(sisi*sisi)/2**
4. **l_psg=sisi*sisi**
5. **l_sgt_hju=l_psg – l_sgt_br**
6. **cetakl_sgt_hju**

FLOW CHART



Program

```
#include <iostream.h>
void main()
{
Intsisi,l_sgt_br,l_psg, l_sgt_hju;
sisi=8;
l_sgt_br=(sisi*sisi)/2;
l_psg=sisi*sisi;
l_sgt_hju=l_psg – l_sgt_br;
cout<<"luassegitigahijau
:"<<l_sgt_hju;
}
```

DaftarPustaka

1. Kristianto. Andri, AlgoritmadanPemrogramandengan C++ Edisi 3, Yokyakarta, GrahaIlmu, 2013
2. Munir. Rinaldi, AlgoritmadanPemrogramanDalamBahasa Pascal dan C, Bandung Informatika, 2007

3. Sjukani. MohAlgoritma (AlgoritmadanStruktur Data1) dengan C, C++, dan Java, Jakarta, MitraKencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

Input/Output Dasar

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	03	MK87001	Tim Dosen

Abstract

Modul ini berisi tentang bagaiaman pembacaan input pada C++ dengan menggunakan standard input keyboard, dan juga menampilkan output kelayar monitor

Kompetensi

Diharapkanmahasiswadapat:

- Mengetahui cara pembacaan input dari keyboard untuk berbagai tipe data
- Mengetahui cara menampilkan output kelayar monitor
- Membuat program yang mengimplementasikan input/output dasar

Pendahuluan

Defenisi

Pada modul-modul sebelumnya, contoh program yang berinteraksi dengan user belum terlalu maksimal, hanya mencetak output. Dengan menggunakan input dan output library, maka program dapat berinteraksi dengan user, tidak hanya sekedar mencetak output kelayar, tetapi juga dapat membaca input dari keyboard.

C++ menggunakan sebuah streams untuk melaksanakan operasi input dan output. Sebuah stream merupakan sebuah objek dimana sebuah program dapat memasukkan atau mengambil karakter ke.dari objek tersebut. Standard C++ library termasuk file header iostream, dimana objek-objek standar input dan output stream dideklarasikan.

Standard Output → cout

Standard output dari sebuah program adalah layar monitor, dan C++ stream object mendefenisikan operasi tersebut dengan cout

Sintaks cout

cout<<var

Contoh penggunaan cout

```
1 cout <<"Dasar Pemrograman";
  // mencetak tulisan Dasar Pemrograman pada layar
  // pada contoh diatas, kata Dasar Pemrograman menggunakan tanda kutip dua, karena
  merupakan karakter atau string.
2 cout << 120;
  // mencetak angka 120 pada layar
  // Tidak menggunakan tanda kutip dua, karena bertipe numerik
3 cout << x;
  // mencetak isi dari sebuah variabel x pada layar
```

```
// Tidak menggunakan tanda kutip dua, karena merupakan sebuah variabel
```

Operator**lebih kecil dari** (<<) bisa digunakan lebih dari satu kali dalam sebuah statement tunggal. Operator ini digunakan lebih dari satu kali, apabila kita ingin mencetak sebuah kombinasi dari variabel dan konstanta atau lebih dari satu varabel

Contoh

```
cout <<"Hallo, "<<"Saya "<<"Instruksi C++";
cout <<"Hallo saya berumur "<< age <<" tahun lah alamat saya "<<alamat;
```

NOTE:

cout tidak menambahkan line break.

Contoh

```
1 cout <<"Kalimat  A.";
2 cout <<"Yang ini Kalimat B.";
```

Outputnya adalah:
Kalimat A. Yang ini kalimat B

untuk menambahkan sebuah line break pada output, maka harus secara eksplisit ditambahkan karakter new-line pada cout.

Pada C++ sebuah new-line character dinyatakan dengan →

- \n(backslash, n)
- endl manipulator

Contoh

```
cout <<"Kalimat  A \n";
cout <<"Yang ini Kalimat B.";
```

Outputnya adalah:
Kalimat A.
Yang ini kalimat B

```
cout <<"Kalimat  A."<<endl;
cout <<"Yang ini Kalimat B.";
```

Outputnya adalah:
Kalimat A.
Yang ini kalimat B

Standard Input (cin).

Standard input device adalah keyboard. Untuk menangani standard input dalam C++ adalah dengan menggunakan cin stream.

Sintak →cin >> var

Contoh

```
#include <iostream.h>
void main()
{
int N;
cin >> N;
}
```

Inputan Karakter

Tipe data karakter merupakan jenis data yang hanya diwakili oleh 1 karakter saja (bisa huruf atau angka). Berapapun jumlah karakter yang kita inputkan, yang akan disimpan kedalam variabel nya di memori hanyalah 1 karakter saja.

Untuk membaca data dengan tipe data karakter, ada 3 cara yang dilakukan oleh C++ yaitu sebagai berikut:

1. Menggunakan sintak cin>>var;

Contoh

```
#include <iostream.h>
void main()
{
char C;
cin >> C;
cout << C;
}
```

Output

```
8
8
---
890
8
```

2. Menggunakan sintak var=getch();

   Untuk menggunakan instruksi getch(), maka kita harus menambahkan conio.h sebagai library yang mendeklarasikan instruksi ini.

   Dengan menggunakan getch(), maka inputan yang kita masukkan, tidak akan terlihat kelayar monitor. Tombol keyboard yang kita tekan juga berfungsi sebagai tombol enter

CONTOH

```
#include <iostream.h>
#include <conio.h>
void main()
{
char C;
C=getch();
cout << C;
}
```

Output

```
8
```

3. Menggunakan sintak var=gethce();

   Untuk menggunakan instruksi getche(), maka kita juga harus menambahkan conio.h sebagai library yang mendeklarasikan instruksi ini. Perbedaannya dengan getch() adalah, pada getche(), inputan yang dimasukkan terlihat pada layar monitor.

   Dengan menggunakan getche(), tombol yang diinput juga berfungsi sebagai enter, dan output akan ditampilkan pada baris yang sama dengan input.

CONTOH

```
#include <iostream.h>
#include <conio.h>
void main()
{
char C;
C=getche();
cout << C;
}
```

Output

```
88
```

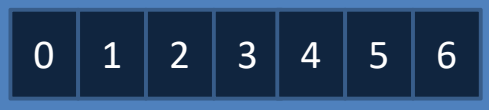
Inputan string

String merupakan kumpula dari karakter. Untuk mendeklarasikan sebuah variabel yang bertipe data string, sama halnya mendeklarasikan variabel dengan tipe data karakter, hanya untuk string kita perlu menambahkan jumlah karakter yang akan dipesan dimemori, dan ini dikenal dengan nama array.

CONTOH DEKLARASI VARIABEL

char C[7]; -→ array

KONDISI DIMEMORI



Ada 2 cara dalam membaca inputan dengan jenis data string yaitu sebagai berikut.

1. Menggunakan sintak cin>>var

CONTOH

```
#include <iostream.h>
void main()
{
char C[7];
cin >> C;
cout << C;
}
```

OUTPUT

1. ABCDE
   ABCDE
2. ABCDEFGH
   ABCDEFG
3. ABC DE
   ABC

Dengan menggunakan cara ini, jika ada kumpulan karakter yang dimasukkan dipisahkan dengan spasi, maka yang dibaca adalah kumpulan karakter yang pertama sebelum spasi, seperti pada output contoh 3

2. Menggunakan sintak cin.getline(var,(sizeof(var)));
   Dengan menggunakan perintah ini, masalah kumpulan karakter yang terpisah dengan spasi, dapat diatasi.

CONTOH

```
#include <iostream.h>
void main()
{
char C[7];
cin.getline (C,7);
cout << C;
}
```

Atau penulisannya juga bisa seperti berikut:

```
#include <iostream.h>
void main()
{
char C[7];
cin.getline (C, (sizeof(C)));
cout << C;
}
```

OUTPUT

1. ABC DE
   ABC DE

Inputan Numerik

Untuk inputan numerik digunakan untuk data-data yang akan diproses secara matematika.

1. Tipe data integer

CONTOH

```
#include <iostream.h>
void main()
{
int C;
cin >> C;
cout << C;
}
```

2. Tipe data long integer

CONTOH

```
#include <iostream.h>
void main()
{
signed long int C;
cin >> C;
cout << C;
}
```

3. Tipe data float

CONTOH

```
#include <iostream.h>
void main()
{
float C;
cin >> C;
cout << C;
}
```

Contoh-contoh

1. Buatlah program utuk membaca inputan & menampilkan data pribadi

Input

NIM

Nama

Tempat Lahir

Tanggal Lahir

Output

DATA PRIBADI

NIM :

NAMA :

Tempat/Tanggal Lahir:

PSUDO CODE

1. Siapkan 4 variabel (Nim, nama, tempat dan tanggal lahir) tipe data string
2. Input Nim, nama, tempat dan tanggal lahir
3. Cetak Nim, nama, tempat dan tanggal lahir

FLOW CHART

PROGRAM

```
#include <iostream.h>
void main()
{
char nim[9], nama[10], tpt_lhr[10],
tgl_lhr[8];
cout<<"Inputkan NIM anda          :";
cin.getline(nim,9);

cout<<"Inputkan Nama anda         :";
cin.getline(nama,10);

cout<<"Inputkan Tempat Lahir anda  :";
cin >> tpt_lhr;

cout<<"Inputkan Tanggal Lahir anda :";
cin >> tgl_lhr;

cout<<endl<<endl;
cout << "          DATA PRIBADI"<<endl;
cout << "          _____"<<endl;
cout << "NIM               : "<<nim<<endl;
cout << "Nama              :
"<<nama<<endl;
cout << "Tempat/Tanggal Lahir:
"<<tpt_lhr<<"/"<<tgl_lhr<<endl;
}
```

Latihan & Tugas Rumah

1. LATIHAN

   Buatlah pseudocode, flowchart dan program untuk mencari Luas dan keliling bangun datar dibawah ini

   a. Persegi Panjang
   b. Segitiga
   c. Bujur Sangkar
   d. Lingkaran

2. TUGAS RUMAH

   Buat program untuk mencari KELILING, LUAS dan VOLUME untuk bangun ruang dibawah ini.

   1. KUBUS
   2. BALOK
   3. KERUCUT
   4. BOLA
   5. TABUNG

   Pada setiap program, ditambahkan

   • inputan NIM, NAMA, Tempat dan Tanggal Lahir

   Output

   KUBUS

   Oleh

   NIM :
   Nama :
   Tempat/Tgl Lahir :

   Panjang Sisi :
   Keliling Kubus :
   Luas Kubus :
   Volume Kubus:

DaftarPustaka

1. Kristianto. Andri, Algoritma dan Pemrograman dengan C++ Edisi 3, Yokyakarta, Graha Ilmu, 2013
2. Munir. Rinaldi, Algoritma dan Pemrograman Dalam Bahasa Pascal dan C, Bandung Informatika, 2007
3. Sjukani. Moh Algoritma (Algoritma dan Struktur Data1) dengan C, C++, dan Java, Jakarta, Mitra Kencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial, diakses tanggal 10 September 2013

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

PERCABANGAN

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	04	MK87001	Tim Dosen

Abstract

Modul ini berisi tentang bagaimana membuat algoritma pemilihan atau percabangan

Kompetensi

Diharapkan mahasiswa dapat:

- Membuat algoritma dengan flowchart untuk percabangan
- Mengetahui sintak percabangan dengan c++
- Membuat program yang mengimplementasikan percabangan

Pendahuluan

Defenisi

Kondisi percabangan merupakan sebuah statement yang menyebabkan program untuk merubah bagian eksekusi berdasarkan nilai dari sebuah ekspresi.

Statement yang paling umum digunakan adalah **if**

Statement if

Jika meng-eksekusi sebuah statement **if,** maka akan menyebabkan eksekusi sebuah statement atau block statement , hanya jika kondisinya terpenuhi.

Sintak

```
if (Kondisi)
statement;
```

Kondisi adalah suatu pernyataan atau ekspresi logika yang akan dievaluasi. Kondisi ini memiliki 2 nilai yaitu benar (True) atau salah (False). Jika kondisi adalah benar, maka statement akan dieksekusi. Jika kondisi adalah salah, maka statement tidak dieksekusi, dan program akan melanjutkan eksekusi instruksi berikutnya.

Operator Relational

Untuk melakukan evaluasi sebuah perbandingan antara 2 pernyataan, maka digunakan operator relasional. Hasil dari operasi relational adalah bernilai Boolean yaitu true atau false.

Simbol Operator	Maksud
==	Sama dengan (equal to)
>	Lebih besar dari (greather than)
<	Lebih kecil dari (Less than)
>=	Lebih besar atau sama dengan (greather than or equal to)
<=	Lebih kecil atau sama dengan (Less than or equal to)
!=	Tidak sama dengan (Not equal to)

Contoh penulisan operator relational

Contoh	Keterangan
If(A<=B)	Benar, tanpa ada spasi
If (A<=B)	Benar
If(A<=B)	Benar
If(A <= B)	Benar
If(A< =B)	Salah, ada spasi pada operator

Bentuk Umum Statement if

Statement if ada 2 macam yaitu

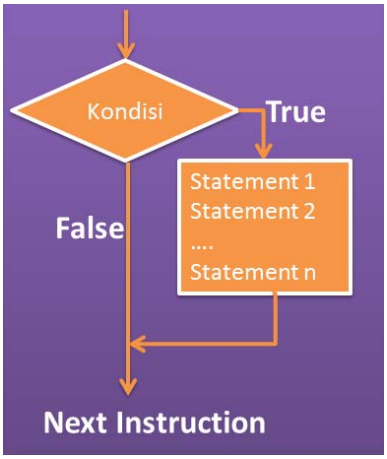
- IF-THEN (Jika - Maka)
- IF-THEN-ELSE (Jika – Maka – Kalau Tidak)

IF - THEN

Sintaks

```
if (kondisi)
{
    Statement True
}
Next Instruction
```

Flowchart



Statement true

Terdiri dari satu atau lebih statement atau instruksi.

Jika terdiri dari 1 statement aja, maka disebut statement tunggal.

Jika terdiri dari 2 atau lebih statement, maka disebut dengan blok statement yang harus diapit oleh tanda kurung kerawal buka “{“ dan tutup “ }”

Cara kerja

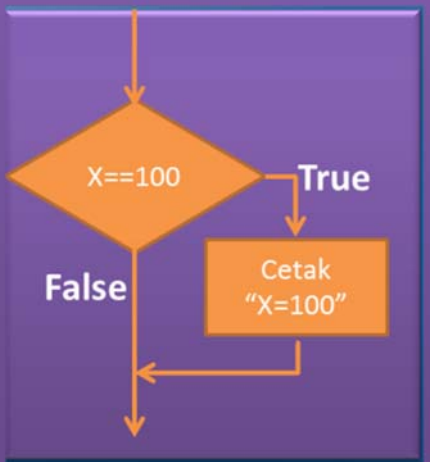
1. Memeriksa nilai kondisi
2. Jika kondisi :
   - Bernilai TRUE
     - Maka eksekusi statement-statement yang berada dalam **blok statement-true**.
     - Setelah selesai mengerjakan semua statement dalam **blok statement true**, langsung meloncat ke **Next Instruction.**
   - Bernilai FALSE
     - Maka **blok statement-true**,tidak akan dieksekusi.
     - Langsung mengerjakan **Next Instruction**

Contoh

Statement tunggal

```
if (x == 100)
cout << "x is 100";
```

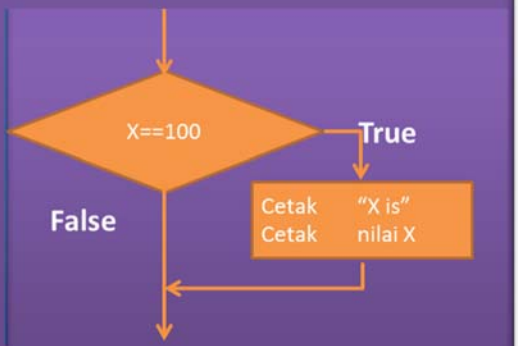
Flow Chart



Blok Statement

```
if (x == 100)
{
cout << "x is ";
cout << x;
}
```

Flow Chart

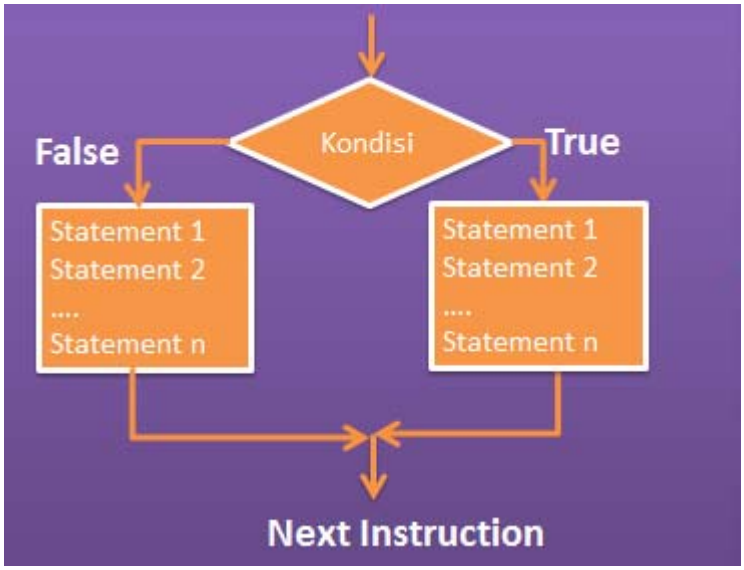


IF – THEN-ELSE

Sintaks

```
if (kondisi)
{
    Statement True
}
else
{
    Statement False
}
Next Instruction
```

Flowchart



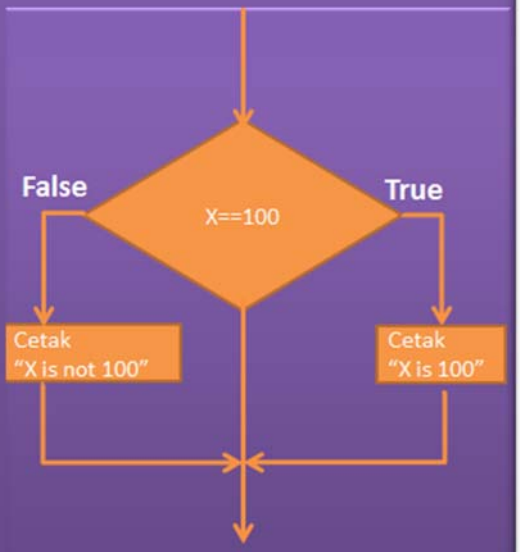
Cara kerja

1. Memeriksa nilai kondisi
2. Jika kondisi :
   - Bernilai TRUE
     - Maka eksekusi statement-statement yang berada dalam **blok statement-true**.
     - Setelah selesai mengerjakan semua statement dalam **blok statement true**, langsung meloncat ke **Next Instruction.**
   - Bernilai FALSE
     - Maka eksekusi statement-statement yang berada dalam **blok statement-false**.
     - Setelah selesai mengerjakan semua statement dalam **blok statement false**, langsung meloncat ke **Next Instruction.**

Contoh

```
if (x == 100)
cout << "x is 100";
else
cout << "x is not 100";
```

Flow Chart
=========

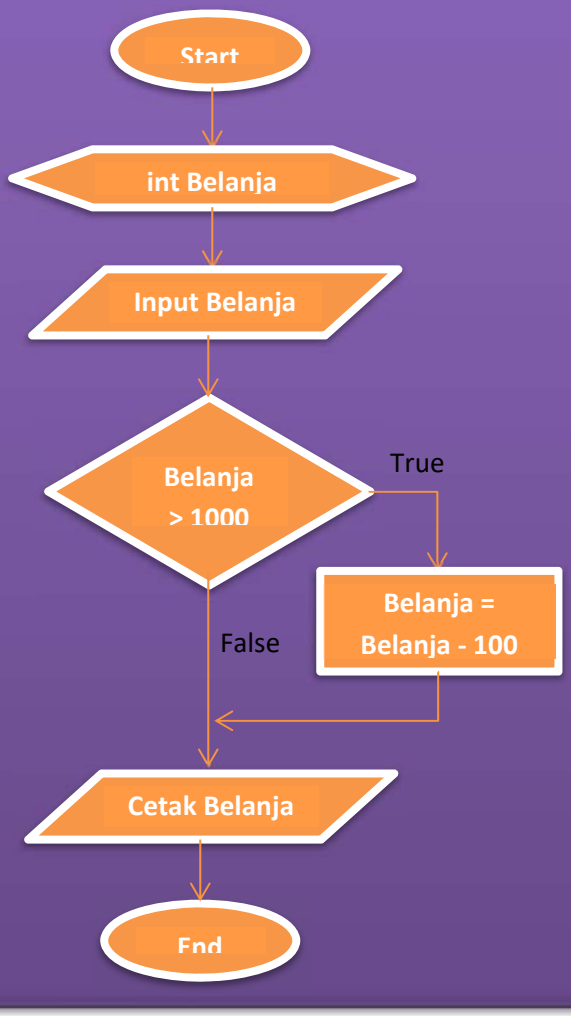


Contoh-contoh Soal

Soal 1

Sebuah toko memberikan diskon sebesar Rp. 100, jika nilai belanjaan lebih besar dari Rp. 1000.

Flow Chart



Program

```
#include <iostream.h>
void main()
{
int belanja;
cin>>belanja;
if (belanja > 1000)
        belanja=belanja-100;
cout<<belanja;
}
```

Contoh output

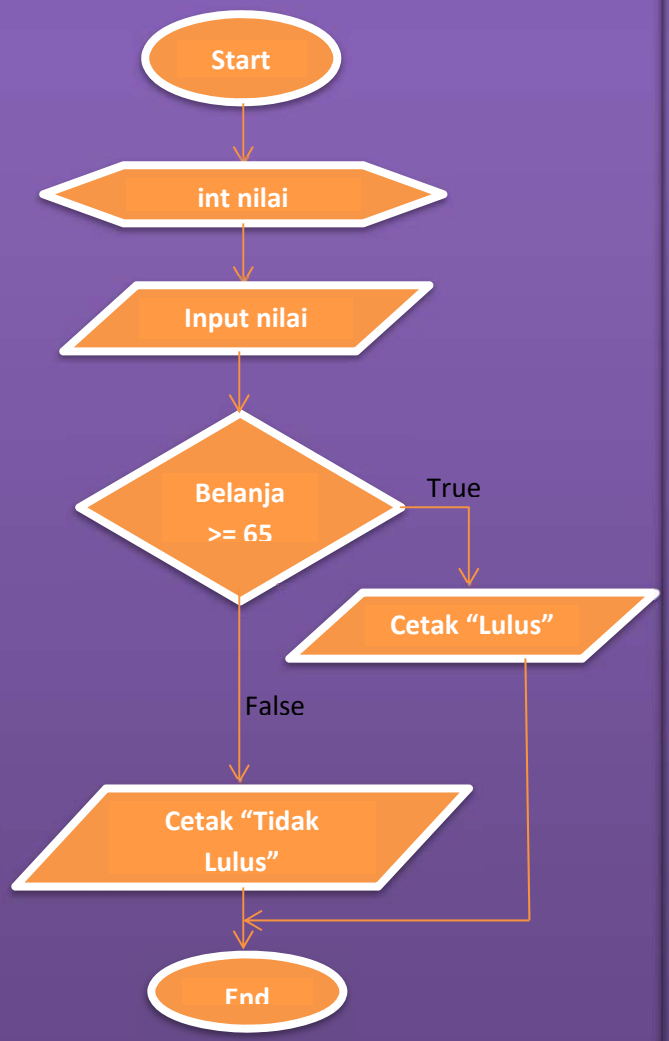
```
---------------------------------------
Input : 2000
Hasil : 1900
Input : 1000
Hasil : 1000
Input : 900
Hasil : 900
```

Soal 2

Menentukan lulus atau tidak lulus

- Jika nilai >=65 maka dinyatakan lulus
- Jika nilai <65 maka tidak lulus

Flowchart



Program

```
#include <iostream.h>
void main()
{
int nilai;
cin>>nilai;
if (nilai >=65)
        cout<<"Keterangan Lulus";
else
        cout<<"Keterangan Tidak Lulus";
}
```

DaftarPustaka

1. Kristianto. Andri, Algoritma dan Pemrograman dengan C++ Edisi 3, Yokyakarta, Graha Ilmu, 2013
2. Munir. Rinaldi, Algoritma dan Pemrograman Dalam Bahasa Pascal dan C, Bandung Informatika, 2007
3. Sjukani. Moh Algoritma (Algoritma dan Struktur Data1) dengan C, C++, dan Java, Jakarta, Mitra Kencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial, diakses tanggal 10 September 2013

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

PERCABANGAN LANJUT MULTI CONDITIONS

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	05	MK87001	Tim Dosen

Abstract

Modul ini berisi tentang bagaimana membuat algoritma pemilihan atau percabangan

Kompetensi

Diharapkan mahasiswa dapat:

- Membuat algoritma dengan flowchart untuk percabangan
- Mengetahui sintak percabangan dengan c++
- Membuat program yang mengimplementasikan percabangan

Multi Conditions

Operator Logika

Satu kondisi terkadang tidak bisa memenuhi sebuah syarat, sehingga diperlukan dua atau lebih kondisi. Untuk menggabung kondisi-kondisi tersebut digunakan operator yang disebut operator logika.

Ada tiga macam operator logika yaitu:

- NOT ditulis !
- AND ditulis &&
- OR ditulis ||

Operator NOT

Operator NOT, bukan untuk menggabungkan dua buah kondisi, tapi bekerja sebagai pembalik nilai logika TRUE menjadi FALSE, FALSE menjadi TRUE sehingga sering disebut Unary Operator.

Contoh:

Misal nilai A=5 dan B=2

Kondisi	Nilai
A==B	False
A>B	True
A<B	False
A>=B	True
A<=B	False
A!=B	True

NOT Kondisi	Nilai
!(A==B)	True
!(A>B)	False
!(A<B)	True
!(A>=B)	False
!(A<=B)	True
!(A!=B)	False

Sintaks

if NOT kondisi-1

if (!C1)

Contoh

if (! (A>B))

=

if (A<=B)

Penulisan	Sama maksudnya dengan
!(A==B)	A != B
!(A>B)	A <= B
!(A<B)	A >= B
!(A>=B)	A < B
!(A<=B)	A > B
!(A!=B)	A == B

Operator AND

Operator AND menggabungkan dua buah kondisi menjadi satu nilai. Operasi ini akan bernilai TRUE jika kedua kondisi yang digabungkan bernilai TRUE, atau kedua syarat terpenuhi.

Sintaks

```
if (kondisi1 && kondisi2)
atau
if ((kondisi1) && (kondisi2))
```

Contoh

```
1. if (kode_Jekel == 1 && umur <=25)
2. if (nilai>=60 && nilai < 70)
```

Tabel berikut merupakan sebuah table kebenaran yang akan memperlihatkan evaluasi dari gabungan 2 buah nilai A dan B, dengan menggunakan operator AND

A	B	A AND B
True	True	True
True	False	False
False	True	False
False	False	False

Contoh

Bila nilai

A=5

B=2

C=6

D=4

Maka

if (A>B && C>D)	Bernilai TRUE A>B → 5>2 adalah TRUE C>D → 6>4 adalah TRUE
if (A>B && B>D)	Bernilai FALSE A>B → 5>2 adalah TRUE B>D → 2>4 adalah FALSE
if (A>C && B>D)	Bernilai FALSE A>C → 5>6 adalah FALSE B>D → 2>4 adalah FALSE
if (A>B && !(B>D))	Bernilai FALSE A>B → 5>2 adalah TRUE !(B>D) → !(2>4) adalah TRUE

Operator OR

Operator OR menggabungkan dua buah kondisi menjadi satu nilai. Dimana akan bernilai TRUE jika salah satu atau kedua kondisi bernilai TRUE. Dan akan bernilai FALSE jika kedua kondisinya adalah FALSE.

Atau dengan kata lain, cukup satu syarat saja yang terpenuhi.

Sintaks

if (kondisi1 || kondisi2)

atau

if ((kondisi1) || (kondisi2))

Contoh

----

1. if (kode_Jekel == 1 || umur <=25)

2. if (nilai1>=60 || nilai2 < 70)

Tabel berikut merupakan table kebenaran dari operator OR yang akan memperlihatkan evaluasi dari gabungan 2 buah nilai A atau B.

A	B	A OR B
True	True	True
True	False	True
False	True	True
False	False	False

Contoh

Bila nilai

A=5

B=2

C=6

D=4

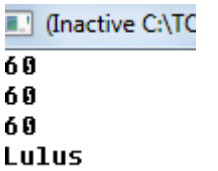
Maka

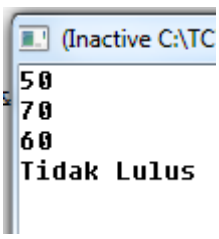
if (A>B || C>D)	Bernilai TRUE A>B → 5>2 adalah TRUE C>D → 6>4 adalah TRUE
if (A>B || B>D)	Bernilai TRUE A>B → 5>2 adalah TRUE B>D → 2>4 adalah FALSE
if (A>C || B>D)	Bernilai FALSE A>C → 5>6 adalah FALSE B>D → 2>4 adalah FALSE
if (A>B II !(B>D))	Bernilai TRUE A>B → 5>2 adalah TRUE !(B>D) → !(2>4) adalah TRUE

Contoh Soal

Soal 1

Buatlah flowchart dan program untuk menginput nilai ujian masuk seorang mahasiswa. Dimana ada 3 buah materi ujian. Mahasiswa tersebut akan **lulus** jika ketiga materi yang diuji mendapat nilai masing-masing >=60, dan jika ke salah satu ada yang kurang dari 60 maka cetak keterangan **tidak lulus.**





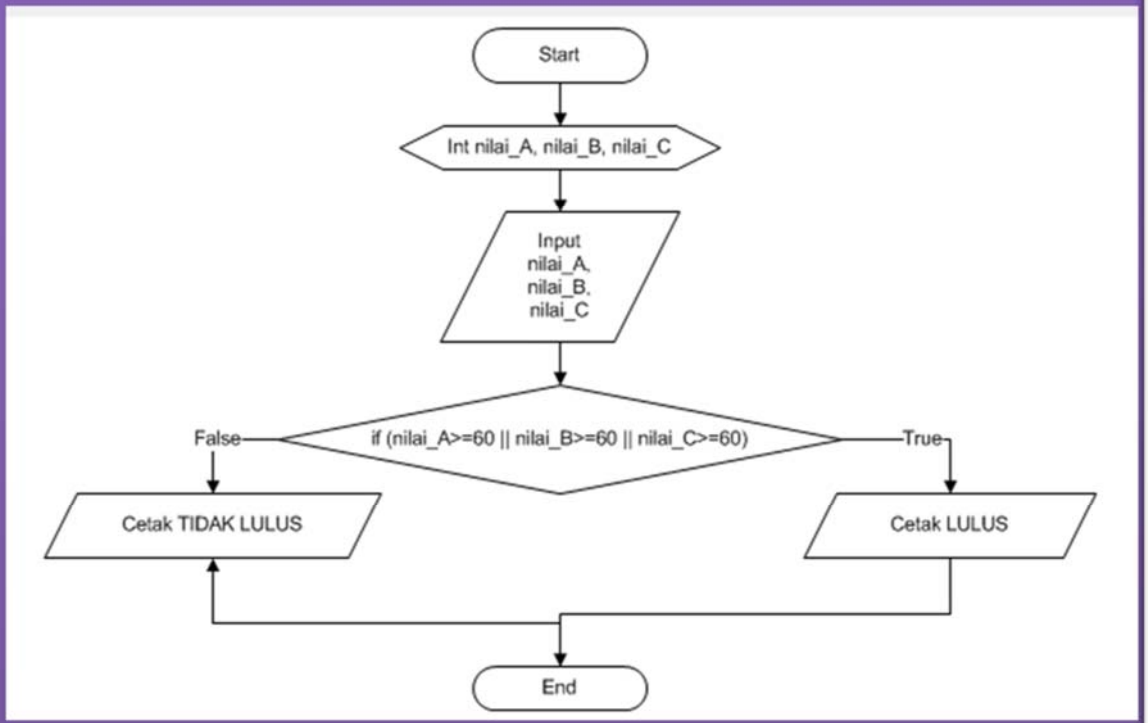
Sintaks **Flowchart**

Soal 2

Buatlah flowchart dan program untuk menginput nilai ujian masuk seorang mahasiswa. Dimana ada 3 buah materi ujian. Mahasiswa tersebut akan **lulus** jika salah satu materi yang diuji mendapat nilai >=60, dan jika ke ketiga materi yang diuji mendapat nilai kurang dari 60 maka cetak keterangan **tidak lulus.**

Flow Chart

==========



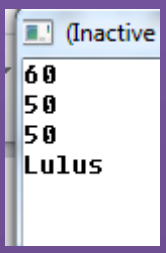
Program

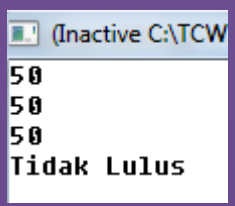
========

```
#include<iostream.h>
void main()
{
int A,B,C;
cin>>A;
cin>>B;
cin>>C;
if (A>=60 || B>=60 || C>=60)
cout<<"Lulus";
else
cout<<"Tidak Lulus";
}
```

Hasil

========





Soal-soal Latihan

Soal 1.

Buatlah algoritma dengan menggunakan flowchart dan juga program untuk inputan nilai satu mata kuliah, dengan ketentuan sebagai berikut:

- Nilai yang diinput adalah antara 0 – 100
- Proses
    - Jika nilai berada antara 80 – 100, maka cetak nilai A
    - Jika nilai berada antara 70 – 79, maka cetak nilai B+
    - Jika nilai berada antara 65 – 69, maka cetak nilai B
    - Jika nilai berada antara 60 – 64, maka cetak nilai C+
    - Jika nilai berada antara 55 – 59, maka cetak nilai C
    - Jika nilai berada antara 40 – 54, maka cetak nilai D
    - Jika nilai berada antara 0 – 40, maka cetak nilai E

Soal 2.

Buatlah algoritma dengan menggunakan flowchart dan juga program untuk kasus dibawah ini.

Inputan

- Jumlah barang
- Harga satuan

Proses

- Harga Barang = jumlah barang * harga satuan
- Harga yang harus dibayar = harga barang – potongan
- Potongan dihitung berdasarkan
    - Bila harga barang > 1 juta rupiah, maka mendapatkan potongan sebesar 10% dari harga barang
    - Bila harga barang 501.000,- – 1.000.000,- maka mendapat potongan sebesar 5% dari harga barang
    - Bila harga barang kurang dari 501.000,- maka tidak mendapat potongan

Output

- Jumlah Barang
- Harga Satuan
- Harga Barang
- Potongan
- Harga yang harus dibayar

DaftarPustaka

1. Kristianto. Andri, Algoritma dan Pemrograman dengan C++ Edisi 3, Yokyakarta, Graha Ilmu, 2013
2. Munir. Rinaldi, Algoritma dan Pemrograman Dalam Bahasa Pascal dan C, Bandung Informatika, 2007
3. Sjukani. Moh Algoritma (Algoritma dan Struktur Data1) dengan C, C++, dan Java, Jakarta, Mitra Kencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial, diakses tanggal 10 September 2013

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

PERCABANGAN LANJUT SWITCH

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	06	MK87001	Tim Dosen

Abstract

Modul ini berisi tentang bagaimana membuat algoritma pemilihan atau percabangan

Kompetensi

Diharapkan mahasiswa dapat:

- Membuat algoritma dengan flowchart untuk percabangan
- Mengetahui sintak percabangan dengan c++
- Membuat program yang mengimplementasikan percabangan

Switch - case

Defenisi

Pernyataan switch memiliki sintaks yang khas. Tujuannya adalah untuk memeriksa beberapa nilai konstan dari sebuah pernyataan.

Sintaks

```
switch (kondisi)
{
 case konstan1:
        blok instruksi 1;
        break;
 case konstan2:
        blok instruksi 2;
         break;
 .
 .
 .
 default:
        default blok instruksi
}
```

Cara Kerja

1. Switch melakukan evaluasi kondisi dan memeriksa jika kondisi tersebut sama dengan **konstan1**. Jika sama, maka program akan menjalankan **blok instruksi 1**, sampai menemukan statement **break**.

   Pada saat menemukan statement break, maka program akan melompat ke akhir dari switch.
2. Jika kondisi tidak sama dengan **konstan1**, maka akan diperiksa **konstan2**. Jika sama, maka program akan menjalankan **blok instruksi 2**, sampai menemukan statement **break**.

   Pada saat menemukan statement break, maka program akan melompat ke akhir dari switch.
3. Terakhir, jika nilai dari kondisi tidak cocok dengan konstan yang ada, maka program akan menjalankan **blok instruksi default,** jika ada (sifat optional)

Default dan break

default

Jika semua kondisi yang ada di Case tidak ada bernilai TRUE, maka yang akan dikerjakan adalah blok instruksi yang ada pada default.

break

Setelah mengerjakan blok instruksi yang ditunjuk oleh case yang bernilai TRUE, maka dengan instruksi break, program akan langsung meloncat ke instruksi berikutnya.

Perbandingan Switch dengan if

switch example	if-else equivalent

`switch (x) {`
`  case 1:`
`  cout << "x is 1";`
`  break;`
`  case 2:`
`  cout << "x is 2";`
`  break;`
`  default:`
`  cout << "value of x unknown";`
`  }`
	`if (x == 1) {`
	`  cout << "x is 1";`
	`  }`
	`else if (x == 2) {`
	`  cout << "x is 2";`
	`  }`
	`else {`
	`  cout << "value of x unknown";`
	`  }`

Contoh 1 switch - case

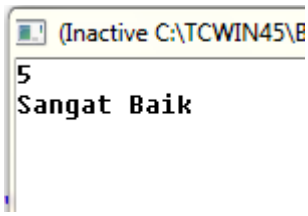
Buatlah flowchart dan program menggunakan switch case, untuk menginput sebuah nilai N, dan kemudian mencetak peringkat nilai sesuai dengan nilai N yang diinputkan.

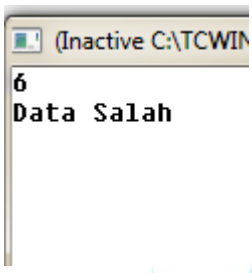
N	Peringkat
1	Kurang Sekali
2	Kurang
3	Cukup
4	Baik
5	Baik Sekali

Sintaks **Flowchart**

```
Program
======

#include<iostream.h>
void main()
{
int N;
cin>>N;
switch(N)
{
case 1:
        cout<<"Kurang Sekali";
        break;
case 2:
        cout<<"Kurang";
        break;
case 3:
        cout<<"Cukup";
        break;
case 4:
        cout<<"Baik";
        break;
case 5:
        cout<<"Sangat Baik";
        break;
default:
                cout<<"Data Salah";
}
}
```

Output



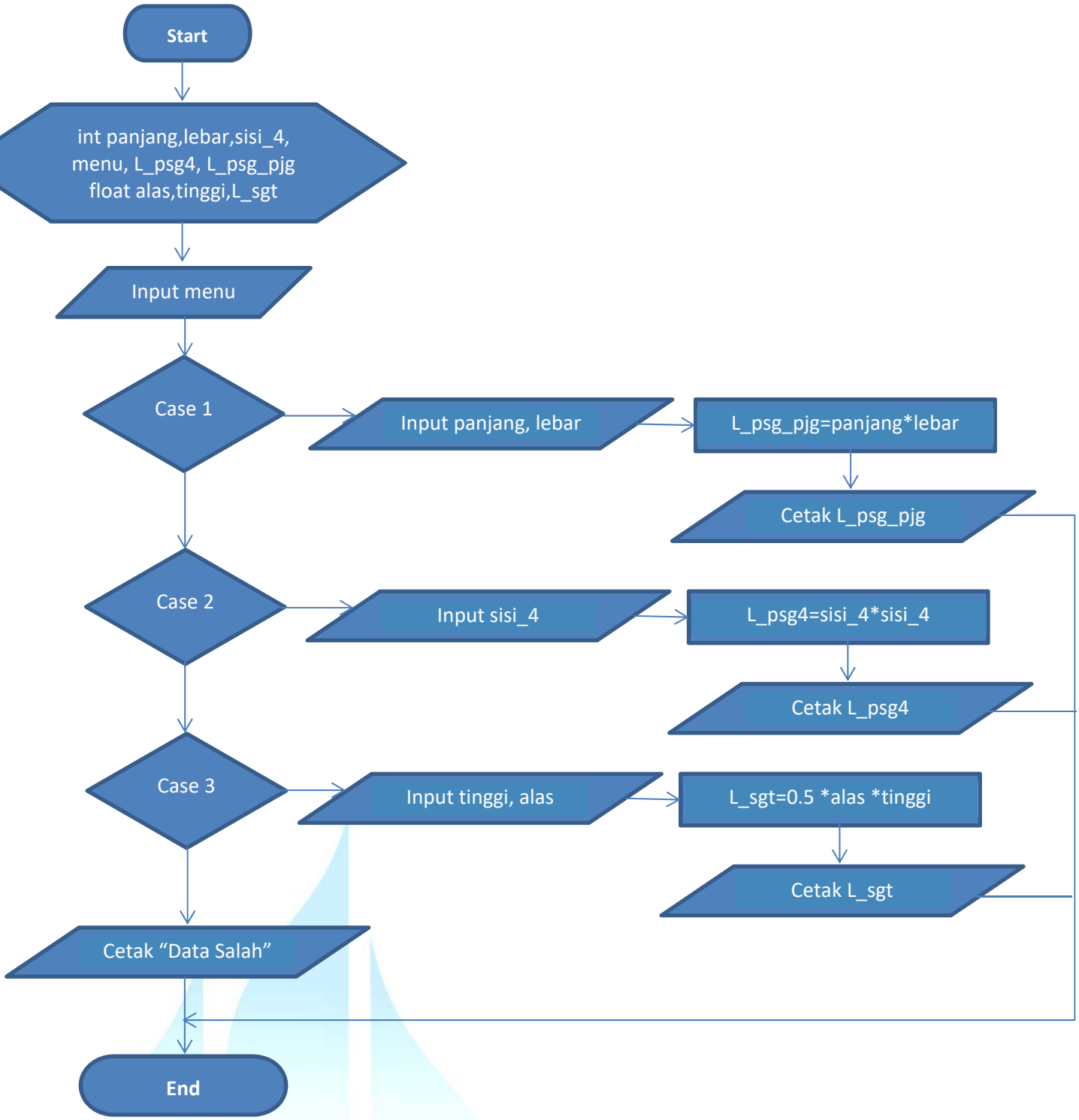


Contoh 2 switch - case

Buatlah flowchart dan program dengan menu-menu sebagai berikut:

1. Mencari Luas segitiga
2. Mencari Luas Persegi Panjang
3. Mencari Luas Persegi Empat

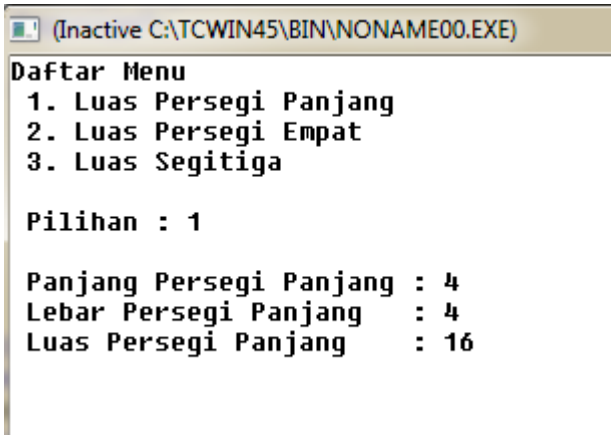
Flowchart

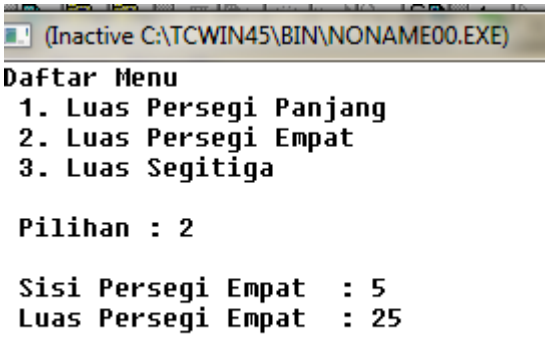


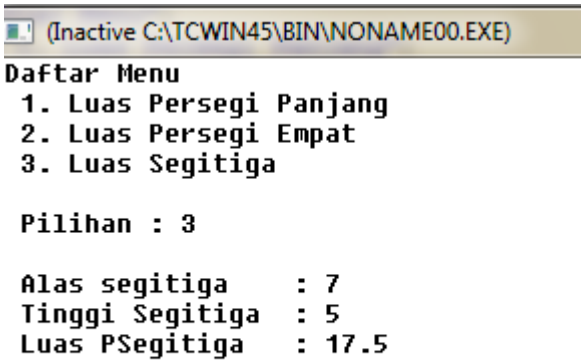
Program

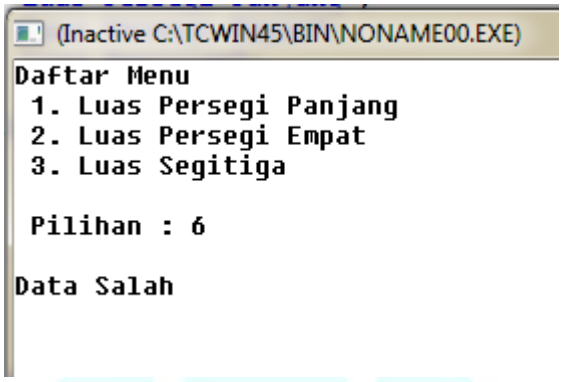
```
#include<iostream.h>
void main()
{
int panjang,lebar,sisi,L_psg4,L_psg_pjg,menu;
float alas,tinggi,L_sgt;

cout<<"Daftar Menu";
cout<<"\n 1. Luas Persegi Panjang";
cout<<"\n 2. Luas Persegi Empat";
cout<<"\n 3. Luas Segitiga";
cout<<endl;
cout<<"\n Pilihan : ";
cin>>menu;
cout<<endl;

switch(menu)
{
case 1:
        {
                cout<<" Panjang Persegi Panjang : ";   cin>>panjang;
                cout<<" Lebar Persegi Panjang   : ";   cin>>lebar;
                L_psg_pjg=panjang*lebar;
                cout<<" Luas Persegi Panjang    : "<<L_psg_pjg;
        }
        break;
case 2:
        {
                cout<<" Sisi Persegi Empat  : ";   cin>>sisi;
                L_psg4=sisi*sisi;
                cout<<" Luas Persegi Empat  : "<<L_psg4;
        }
        break;
case 3:
        {
                cout<<" Alas segitiga    : ";   cin>>alas;
                cout<<" Tinggi Segitiga  : ";   cin>>tinggi;
                L_sgt=0.5*alas*tinggi;
                cout<<" Luas PSegitiga   : "<<L_sgt;
        }
        break;
default:
                cout<<"Data Salah";
}
}
```

Output


(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
Daftar Menu
1. Luas Persegi Panjang
2. Luas Persegi Empat
3. Luas Segitiga
Pilihan : 1
Panjang Persegi Panjang : 4
Lebar Persegi Panjang : 4
Luas Persegi Panjang : 16


(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
Daftar Menu
1. Luas Persegi Panjang
2. Luas Persegi Empat
3. Luas Segitiga
Pilihan : 2
Sisi Persegi Empat : 5
Luas Persegi Empat : 25


(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
Daftar Menu
1. Luas Persegi Panjang
2. Luas Persegi Empat
3. Luas Segitiga
Pilihan : 3
Alas segitiga : 7
Tinggi Segitiga : 5
Luas PSegitiga : 17.5


(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
Daftar Menu
1. Luas Persegi Panjang
2. Luas Persegi Empat
3. Luas Segitiga
Pilihan : 6
Data Salah

DaftarPustaka

1. Kristianto. Andri, Algoritma dan Pemrograman dengan C++ Edisi 3, Yokyakarta, Graha Ilmu, 2013
2. Munir. Rinaldi, Algoritma dan Pemrograman Dalam Bahasa Pascal dan C, Bandung Informatika, 2007
3. Sjukani. Moh Algoritma (Algoritma dan Struktur Data1) dengan C, C++, dan Java, Jakarta, Mitra Kencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial, diakses tanggal 10 September 2013

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

String

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	07	MK87001	Tim Dosen

Abstract

Modul ini berisi tentang manipulasi string

Kompetensi

Diharapkan mahasiswa dapat:
- Memahami manipulasi string

Introduction to strings

Defenisi

Variables that can store non-numerical values that are longer than one single character are known as strings.

The C++ language library provides support for strings through the standard string class. This is not a fundamental type, but it behaves in a similar way as fundamental types do in its most basic usage.

A first difference with fundamental data types is that in order to declare and use objects (variables) of this type we need to include an additional header file in our source code: <string> and have access to the std namespace (which we already had in all our previous programs thanks to the using namespace statement).

A string is an array of characters whose last character is \0. A typical string, such as Pacifique, is graphically represented as follows:

P	a	c	i	f	i	q	u	e	\0

The last character \0 is called the null-terminating character. For this reason, a string is said to be null-terminated. The string library ships with a lot of functions used to perform almost any type of operation on almost any kind of string. Used under different circumstances, the string functions also have different syntaxes.

The strings that you can use in your program may be defined in various libraries depending on your compiler but most of the time, they are available once you include the string library that is defined in the std namespace. For example, the functions we are about to review are part of the C language. The strings that are part of the (C++) Standard Template Library (STL) are defined in the string class of the std namespace. Based on this, most compilers make all these functions accessible once you include the string library and the std namespace in your program.

Example

```
// my first string
#include <iostream>
#include <string>
using namespace std;

int main ()
{
  string mystring = "This is a string";
  cout << mystring;
  return 0;
}
```

```
This is a string
```

As you may see in the previous example, strings can be initialized with any valid string literal just like numerical type variables can be initialized to any valid numerical literal. Both initialization formats are valid with strings:

string mystring = "This is a string";

string mystring ("This is a string");

Strings can also perform all the other basic operations that fundamental data types can, like being declared without an initial value and being assigned values during execution:

```
// my first string
#include <iostream>
#include <string>
using namespace std;

int main ()
{
  string mystring;
  mystring = "This is the initial string content";
  cout << mystring << endl;
  mystring = "This is a different string content";
  cout << mystring << endl;
  return 0;
}
```

```
This is the initial string content
This is a different string content
```

Class String

Strings are objects that represent sequences of characters. The standard string class provides support for such objects with an interface similar to that of standard containers, but adding features specifically designed to operate with strings of characters.

The string class is an instantiation of the basic_string class template that uses char as the character type, with its default char_traits and allocator types (see basic_string for more info on the template).

Member types

member type	definition
value_type	char
traits_type	char_traits<char>
allocator_type	allocator<char>
reference	char&
const_reference	const char&
pointer	char*
const_pointer	const char*
iterator	a random access iterator to char (convertible to const_iterator)
const_iterator	a random access iterator to const char
reverse_iterator	reverse_iterator<iterator>
const_reverse_iterator	reverse_iterator<const_iterator>
difference_type	ptrdiff_t
size_type	size_t

Member functions

(constructor)	Construct string object (public member function)
(destructor)	String destructor (public member function)
operator=	String assignment (public member function)

Iterators:

begin	Return iterator to beginning (public member function)
end	Return iterator to end (public member function)
rbegin	Return reverse iterator to reverse beginning (public member function)
rend	Return reverse iterator to reverse end (public member function)
cbegin C++11	Return const_iterator to beginning (public member function)
cend C++11	Return const_iterator to end (public member function)
crbegin C++11	Return const_reverse_iterator to reverse beginning (public member function)
crend C++11	Return const_reverse_iterator to reverse end (public member function)

Capacity:

size	Return length of string (public member function)
length	Return length of string (public member function)
max_size	Return maximum size of string (public member function)
resize	Resize string (public member function)
capacity	Return size of allocated storage (public member function)
reserve	Request a change in capacity (public member function)
clear	Clear string (public member function)
empty	Test if string is empty (public member function)
shrink_to_fit C++11	Shrink to fit (public member function)

Element access:

operator[]	Get character of string (public member function)
at	Get character in string (public member function)
back C++11	Access last character (public member function)
front C++11	Access first character (public member function)

Modifiers:

operator+=	Append to string (public member function)
append	Append to string (public member function)
push_back	Append character to string (public member function)
assign	Assign content to string (public member function)
insert	Insert into string (public member function)
erase	Erase characters from string (public member function)
replace	Replace portion of string (public member function)
swap	Swap string values (public member function)
pop_back C++11	Delete last character (public member function)

String operations:

c_str	Get C string equivalent (public member function)
data	Get string data (public member function)
get_allocator	Get allocator (public member function)
copy	Copy sequence of characters from string (public member function)
find	Find content in string (public member function)
rfind	Find last occurrence of content in string (public member function)
find_first_of	Find character in string (public member function)
find_last_of	Find character in string from the end (public member function)
find_first_not_of	Find absence of character in string (public member function)
find_last_not_of	Find non-matching character in string from the end (public member function)
substr	Generate substring (public member function)
compare	Compare strings (public member function)

fx Member constants

npos	Maximum value for size_t (public static member constant)

fx Non-member function overloads

operator+	Concatenate strings (function)
relational operators	Relational operators for string (function)
swap	Exchanges the values of two strings (function)
operator>>	Extract string from stream (function)
operator<<	Insert string into stream (function)
getline	Get line from stream into string (function)

String Manipulation Functions

The C++ and its parent the C languages do not have a string data type. In C and C++, strings are created from arrays. Therefore, the C++ language relies on operations performed on the arrays of characters or pointers to char. The functions used for this purpose are numerous and you should know what they are used for.

As a reminder, thanks to the features of arrays of characters and the friendship between arrays of characters and strings, there are two main ways you can declare and initialize a string:
char *Thing = “Telephone”;

char Major[] = “Computer Sciences”;

The Length of a String

In many operations, you will want to know how many characters a string consists of. To find the number of characters of a string, use the strlen() function. Its syntax is:

```
int strlen(const char* Value);
```

The strlen() function takes one argument, which is the string you are considering. The function returns the number of characters of the string. Here is an example:

```
#include <iostream>
using namespace std;
int main()
{
	char *School = "Manchester United";
	int Length = strlen(School);
	cout << "The length of \"" << School
	    << "\" is " << Length << " characters\n\n";
	return 0;
}
```

This would produce:

The length of "Manchester United" is 17 characters

The strcat() Function

If you have two strings, to append one to another, use the strcat() function. Its syntax is:

```
char *strcat(char *Destination, const char *Source);
```

The strcat() function takes two arguments. The second argument, called the source string, is the string you want to add to the first string; this first string is referred to as the destination. Although the

function takes two arguments. It really ends up changing the destination string by appending the second string at the end of the first string. This could be used to add two strings. Here is an example:

```
#include <iostream>
using namespace std;
int main()
{
   char *Make = "Ford ";
   char *Model = "Explorer";
   cout << "Originally, Make = " << Make;
   strcat(Make, Model);
   cout << "\n\nAfter concatenating, Make = " << Make << endl;
   return 0;
}
```

This would produce:

```
Originally, Make = Ford


After concatenating, Make = Ford Explorer
```

The strncat() Function

Like the strcat() function, the strncat() function is used to append one string to another. The difference is that, while the strcat() considers all characters of the source string, the strncat() function allows you to specify the number of characters from the source string that you want to append to the destination string. This means that, if the source string has 12 characters, you can decide to append only a set number of its characters. The syntax is:

```
char* strncat(char* Destination, const char* Source, int Number);
```

Besides the same arguments as the strcat() function, the Number argument sets the number of characters considered from Source. To perform the concatenation, the compiler would count characters from left to right on the source string. Here is an example:

```
#include <iostream>
using namespace std;
int main()
{
   char *Make = "Ford ";
   char *Model = "Explorer";
```

```
   cout << "Originally, Make = " << Make;
   strncat(Make, Model, 3);
   cout << "\n\nAfter concatenating, Make = " << Make;
   return 0;  }
```

This would produce:

```
Originally, Make = Ford
After concatenating, Make = Ford Exp
```

Copying One String Into Another

The **strcpy()** function is used to copy one string into another string. In English, it is used to replace one string with another. The syntax of the **strcpy()** function is:

```
char* strcpy(char* Destination, const char* Source);
```

This function takes two arguments. The first argument is the string that you are trying to replace. The second argument is the new string that you want to replace. There are two main scenarios suitable for the **strcpy()** function: To replace an existing string or to initialize a string.

The following would initialize a string:

```
char CarName[20];
strcpy(CarName, "Toyota Camry");
cout << "Car Name: " << CarName;
```

If you have two strings and copy one into another, both strings would hold the same value:

```
#include <iostream>
using namespace std;
int main()
{
   char carName1[] = "Ford Escort";
   char carName2[] = "Toyota 4-Runner";
   cout << "The String Copy Operation";
   cout << "\nFirst Car: " << carName1;
   cout << "\nSecond Car: " << carName2;
   strcpy(carName2, carName1);
   cout << "\n\nAfter using strcpy()...";
   cout << "\nFirst Car: " << carName1;
   cout << "\nSecond Car: " << carName2 << endl;
   return 0;
}
```

This would produce:

The String Copy Operation

First Car: Ford Escort

Second Car: Toyota 4-Runner

After using strcpy()...

First Car: Ford Escort

Second Car: Ford Escort

The strncpy() Function

The **strncpy()** function works like the **strcpy()** function. As a difference, the **strncpy()**function allows you to specify the number of characters that the compiler would copy from the source string. Here is the syntax:

```
char* strncpy(char* Destination, const char* Source, int Number);
```

The Number argument specifies the number of characters that will be copied from the Source string. Here is an example:

```
#include <iostream>
using namespace std;
int main()
{
   char CarName1[] = "Ford Escort";
   char CarName2[] = "Toyota 4-Runner";
   cout << "The String Copy Operation";
   cout << "\nFirst Car: " << CarName1;
   cout << "\nSecond Car: " << CarName2;
   strncpy(CarName2, CarName1, 8);
   cout << "\n\nAfter using strncpy() for 8 characters";
   cout << "\nFirst Car: " << CarName1;
   cout << "\nSecond Car: " << CarName2 << endl;
   return 0;
}
```

This would produce:

The String Copy Operation

First Car: Ford Escort

Second Car: Toyota 4-Runner

After using strncpy() for 8 characters

First Car: Ford Escort

Second Car: Ford Esc-Runner

The strdup() Function

The strdup() function is used to make a copy of create a duplicate of that string. Its syntax is:

```
char* strdup(const char *S);
```

This function takes as an argument the string you want to duplication and returns the duplicated string.

```
#include <iostream>
using namespace std;
int main()
{
   char *FirstName1 = "Charles";
   char *FirstName2 = "Andy";
   char *LastName1 = "Stanley";
   char *LastName2;
   LastName2 = strdup(LastName1);
   cout << "Father: " << FirstName1 << ' ' << LastName1 << endl;
   cout << "Son: " << FirstName2 << ' ' << LastName2;
   return 0;
}
```

This would produce:

Father: Charles Stanley

Son: Andy Stanley

Comparing Strings

Various routines are available for strings comparison. The C-String library is equipped with functions that perform the comparisons by characters. Alternatively, some compilers like Borland C++ Buider ships with additional multiple functions to perform these comparisons on null-terminated strings.

The strcmp() Function

The strcmp() function compares two strings and returns an integer as a result of its comparison. Its syntax is:

```
int strcmp(const char* S1, const char* S2);
```

This function takes two strings, S1 and S2 and compares them. It returns

- A negative value if S1 is less than S2
- 0 if S1 and S2 are equal
- A positive value if S1 is greater than S2

```
#include <iostream>
using namespace std;
int main()
{
        char *FirstName1 = "Andy";
        char *FirstName2 = "Charles";
        char *LastName1 = "Stanley";
        char *LastName2 = "Stanley";
        int Value1 = strcmp(FirstName1, FirstName2);
        int Value2 = strcmp(FirstName2, FirstName1);
        int Value3 = strcmp(LastName1, LastName2);
        cout << "The result of comparing " << FirstName1
                << " and " << FirstName2 << " is\t" << Value1 << endl;
        cout << "The result of comparing " << FirstName2
                << " and " << FirstName1 << " is\t" << Value2 << endl;
        cout << "The result of comparing " << LastName1
                << " and " << LastName2 << " is\t" << Value3;
        return 0;
}
```

This would produce:

The result of comparing Andy and Charles is -2

The result of comparing Charles and Andy is 2

The result of comparing Stanley and Stanley is 0

The strncmp() Function

The **strncmp()** function compares two strings using a specified number of characters and returns an integer as a result of its findings. Its syntax is:

```
int strncmp(const char* S1, const char* S2, int Number);
```

This function takes three arguments. The first two arguments are the strings that need to be compared. The 3rd argument specifies the number of characters considered for the comparison. It returns

- A negative value if S1 is less than S2
- 0 if S1 and S2 are equal
- A positive value if S1 is greater than S2

The stricmp() Function

The **stricmp()** function compares two strings without regard to their case. In other words, this function does not take into consideration if there is a mix of uppercase and lowercase characters in the strings. The syntax of the function is:

```
int stricmp(const char* S1, const char* S2);
```

This function takes two strings, S1 and S2 and compares them. It returns

- A negative value if S1 is less than S2
- 0 if S1 and S2 are equal
- A positive value if S1 is greater than S2

The strnicmp() Function

The **strnicmp()** function compares two strings without regard to their case but considers only a specified number of characters. The syntax of the function is:

```
int strnicmp(const char* S1, const char* S2, int n);
```

This function takes two strings, S1 and S2 and compares them. It returns

- A negative value if S1 is less than S2
- 0 if S1 and S2 are equal
- A positive value if S1 is greater than S2

Working With Individual Characters

The strchr() Function

The **strchr()** function looks for the first occurrence of a certain character in a string. Its syntax is:

```
char* strchr(const char* S, char c);
```

This function takes two arguments. The second argument specifies what character to look for in the first argument which is a string. If the character c appears in the string S, the function would return a new string whose value starts at the first occurrence of c in S. If the character c does not appear in the string S, then the function would return NULL.

The strrchr() Function

The **strrchr()** function examines a string starting at the end (right side) of the string and looks for the first occurrence of a certain character. Its syntax is:

```
char* strrchr(const char* S, char c);
```

The first argument is the string that needs to be examined. The function will scan the string S from right to left. Once it finds the first appearance of the character c in the string, it would return a new string whose value starts at that first occurrence. If the character c does not appear in the string S,

then the function would return NULL.

Working With Sub-Strings

The strstr() Function

The **strstr()** function looks for the first occurrence of a sub-string in another string and returns a new string as the remaining string. Its syntax is:

```
char* strstr(const char* Main, const char *Sub);
```

The first argument of the function is the main string that would be examined. The function would look for the second argument, the Sub string appearance in the main string. If the Sub string is part of the Main string, then the function would return a string whose value starts at the first appearance of Sub and make it a new string. If Sub is not part of the Main string, the function would return a NULL value.

Working With Character Cases

The strlwr() Function

The **strlwr()** function is used to convert a string to lowercase. Its syntax is:

```
char *strlwr(const char *S);
```

This function takes, as argument, the string that needs to be converted. During conversion, if a Latin character were in uppercase, it would be converted to lowercase. Otherwise, it would stay “as if”. This means any symbol that is not a readable character would not be converted.

```
#include <iostream>
using namespace std;
int main()
{
   char CustomerAddress[] = "4812 LOCKWOOD Drive #F04";
   cout << "Customer Address: " << CustomerAddress << endl;
   char *ShippingAddress = strlwr(CustomerAddress);
```

```
    cout << "Shipping Address: " << ShippingAddress << endl;
    return 0;
}
```

This would produce:

Customer Address: 4812 LOCKWOOD Drive #F04

Shipping Address: 4812 lockwood drive #f04

The strupr() Function

The strupr() function is used to convert a string to uppercase. Its syntax is:

```
char *strupr(const char *S);
```

Each lowercase character in the function's argument, S, would be converted to uppercase. Any character or symbol that is not in lowercase would not be changed.

```
#include <iostream>
using namespace std;
int main()
{
    char Drink[] = "100% Apple Juice";
    char *SayItLoud;
    cout << "What is that drink? " << Drink << endl;
    SayItLoud = strupr(Drink);
    cout << "Say it loud: " << SayItLoud << endl;
    return 0;
}
```

This would produce:

What is that drink? 100% Apple Juice

Say it loud: 100% APPLE JUICE

Formatting Strings

The sprintf() Function

The **sprintf()** function is used to format data and specify how it should display. Its syntax is:

```
int sprintf(char* Buffer, const char* S, Arguments…);
```

This function takes at least two arguments and could take more. The first argument is a null-terminated string that could display one or a few words and the formula to use when displaying the second or more argument. To display a value as part of the Buffer, type a double-quote followed by the string if any, followed by the % sign, follow by one of the following characters:

Character	Used to Display	Character	Used to Display
s	A string	f	Floating-point value
c	A character	d	An integer

DaftarPustaka

1. Kristianto. Andri, Algoritma dan Pemrograman dengan C++ Edisi 3, Yokyakarta, Graha Ilmu, 2013
2. Munir. Rinaldi, Algoritma dan Pemrograman Dalam Bahasa Pascal dan C, Bandung Informatika, 2007
3. Sjukani. Moh Algoritma (Algoritma dan Struktur Data1) dengan C, C++, dan Java, Jakarta, Mitra Kencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial, diakses tanggal 10 September 2013

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

Looping For, While

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	08	MK87001	Tim Dosen

Abstract

Modul ini berisi tentang perulangan menggunakan For dan While

Kompetensi

Diharapkan mahasiswa dapat:

- Memahami proses perulangan dengan menggunakan For dan While
- Membuat program perulangan menggunakan For dan While

Pendahuluan

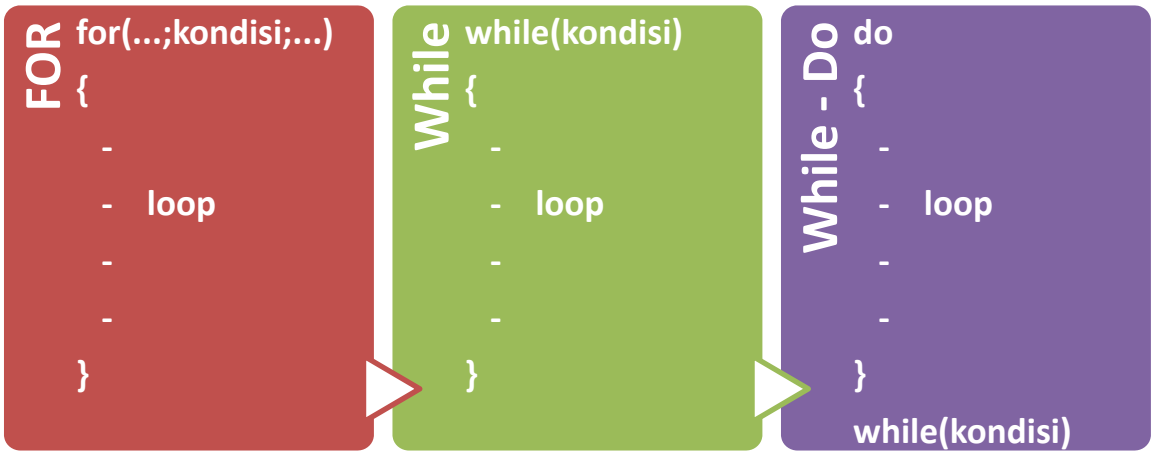
Definisi

Loop atau perulangan memiliki tujuan untuk mengulang sebuah statement beberapa kali tergantung kepada jumlah yang diinginkan atau tergantung kondisi yang disyaratkan.

Ada beberapa jenis perulangan yang dapat digunakan dalam bahasa C++ yaitu:

1. For
2. While
3. Do-while

Syntaks

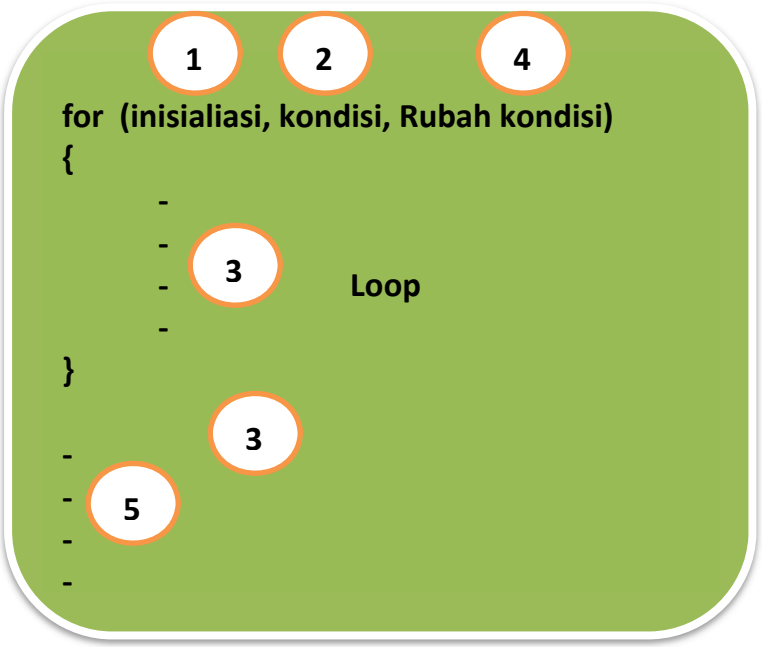


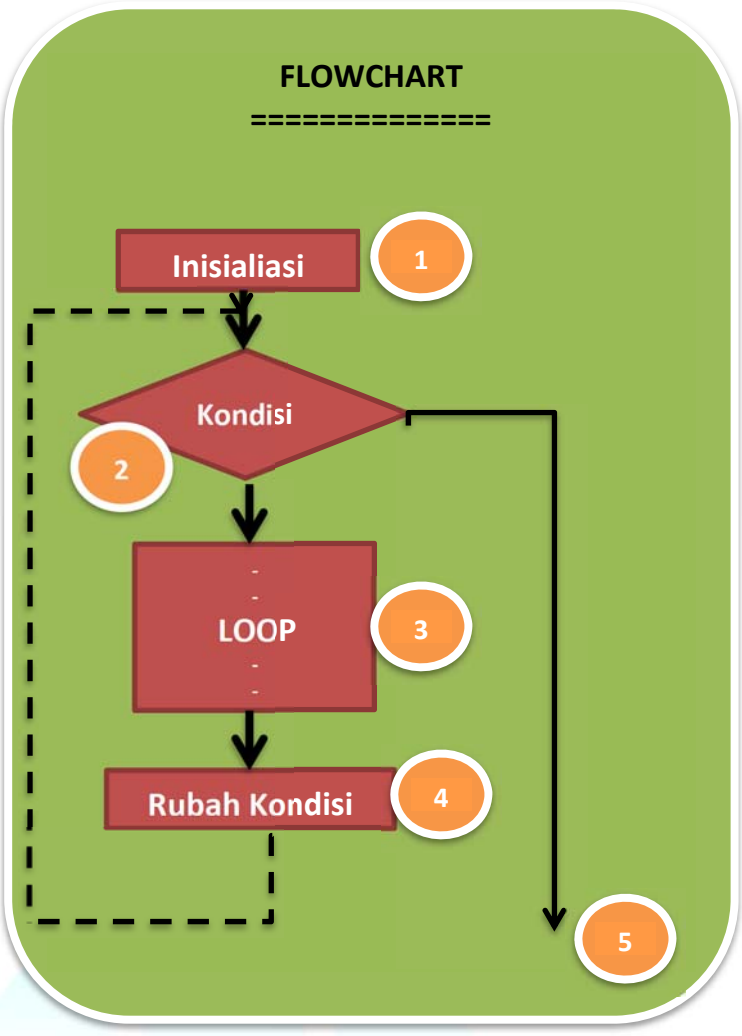
Catt:

Kondisi → sebuah ungkapan atau pernyataan yang mengandung nilai Benar atau Salah

FOR

Format nya adalah :





Keterangan

1. Mengeksekusi **Inisialisasi**

   Berisi nilai awal sebuah variabel yang disetting untuk sebuah variabel conter. Inisialisasi hanya dieksekusi sekali.

2. Memeriksa **Kondisi**

   Jika bernilai **Benar**

   - Laksanakan instruksi yang ada pada **loop (point 3)**.
   - Eksekusi **point 4**, yang berfungsi untuk merubah nilai dan mempengaruhi kondisi.
   - Kembali ke **Point 2**, memeriksa kondisi, dan seterusnya

   Jika bernilai **Salah,** loncat ke **point 5** yaitu instruksi berikutnya diluar looping atau proses looping berakhir.

Contoh

Mencetak tulisan “Universitas Mercu Buana” sebanyak 5 kali.

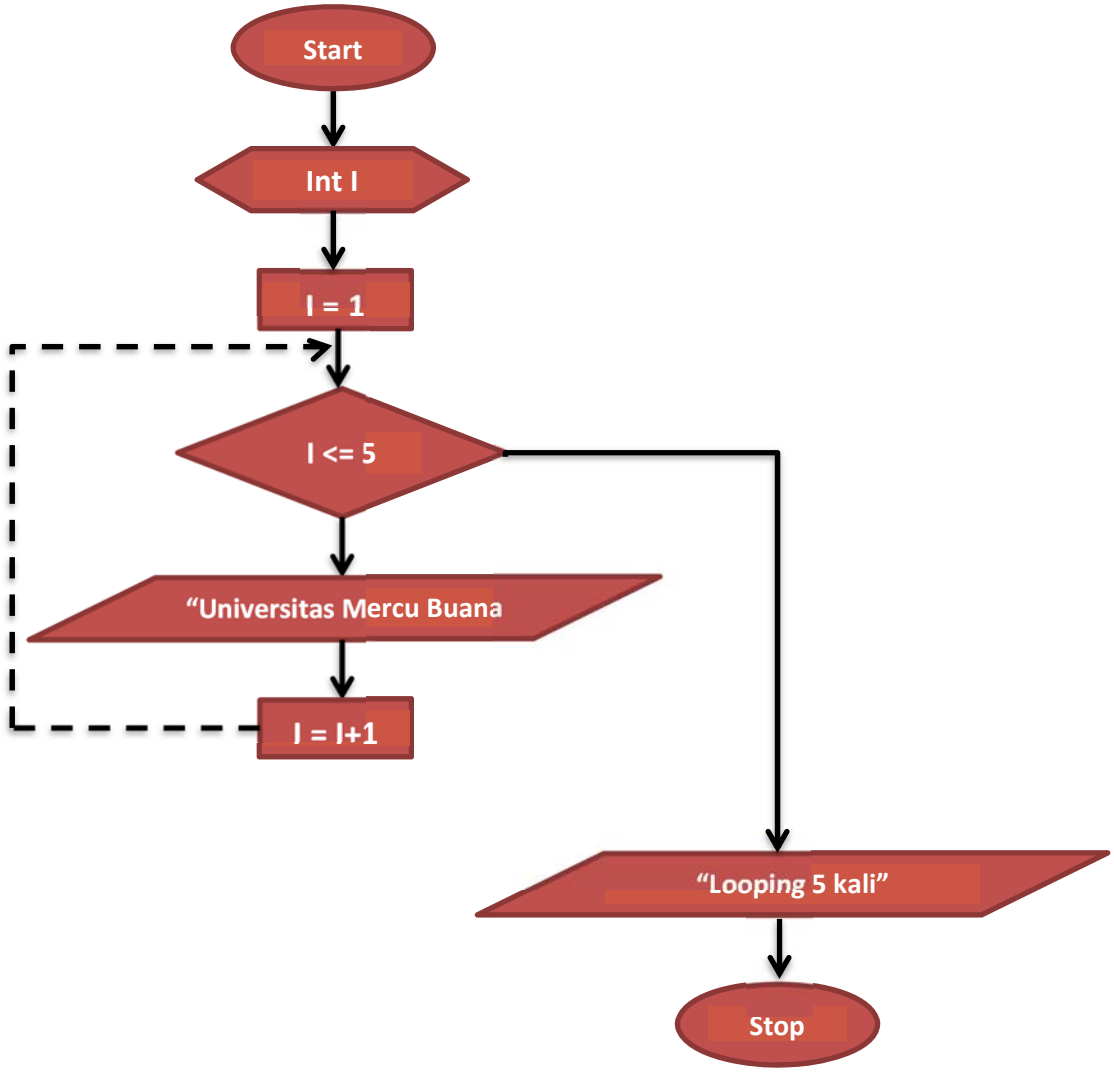
Solusi

1. Variabel → dibutuhkan sebuah variabel yang berfungsi untuk menghitung banyak cetakan, dalam contoh ini misalnya adalah **I** dengan tipe data **integer.**
2. **Kondisi** nya adalah:

   Jika nilai **I** diset 1 maka kondisinya adalah **I<=5**

   Jika nilai **I** diset 0 maka kondisinya adalah **I<5**

FLOWCHART



Perkembangan nilai I

Nilai I	Kondisi I<=5	Tercetak	I=I+1
1	True	Universitas Mercu Buana	2
2	True	Universitas Mercu Buana	3
3	True	Universitas Mercu Buana	4
4	True	Universitas Mercu Buana	5
5	False	Universitas Mercu Buana	6
6		Looping 5 Kali	Keluar Loop

Program

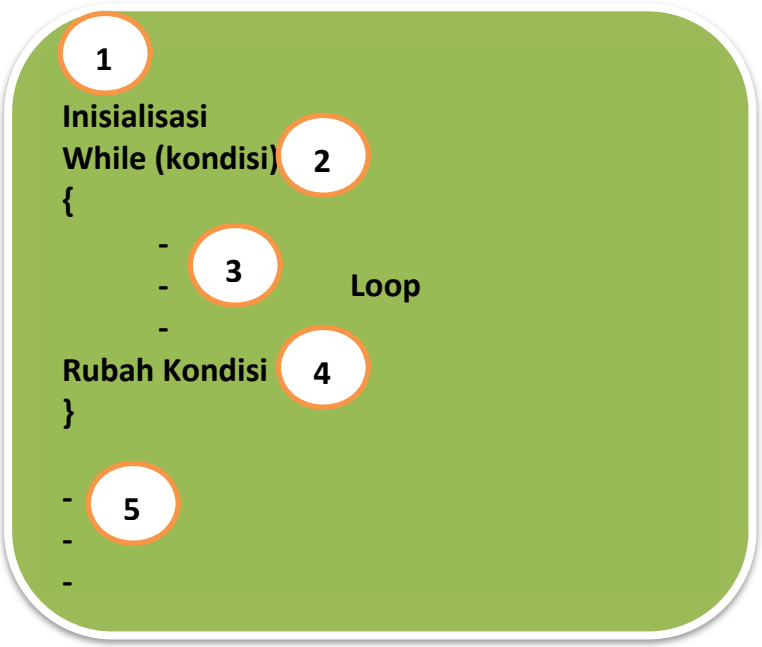
```
#include <iostream.h>
void main()
{
   int I;
       1   2   4
   for (I=1; I<=5; I++)
   {
    cout << “Universitas Mercu Buana \n ”;    3
    }
  cout << "Looping 5 Kali";  5
}
```

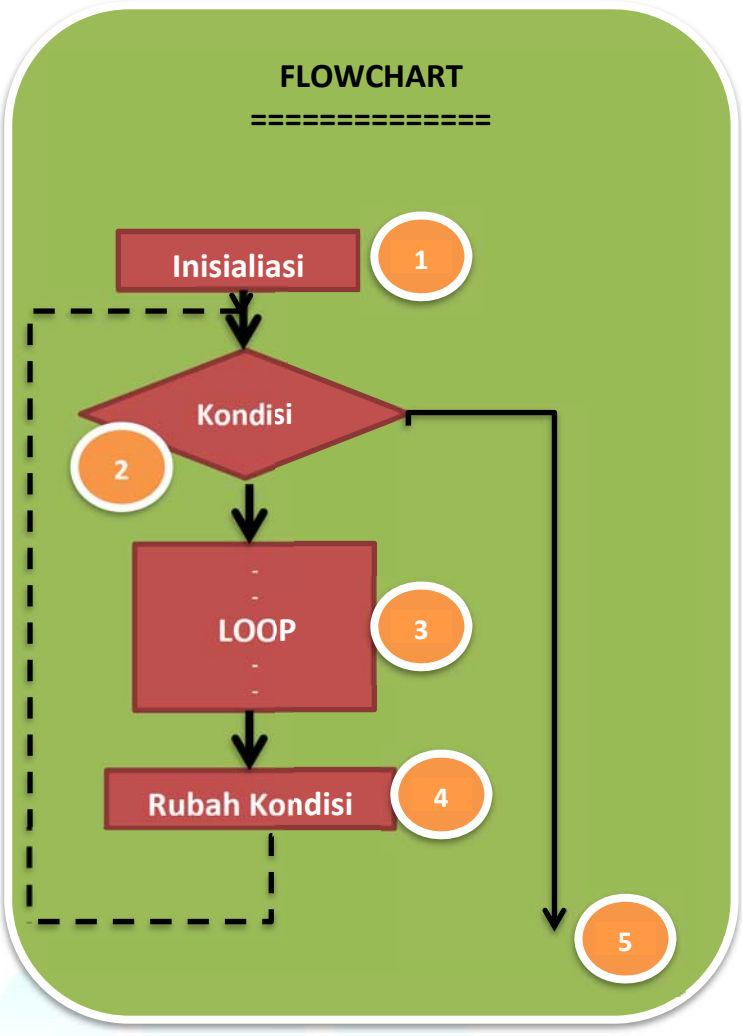
OUTPUT

```
Universitas Mercu Buana
Universitas Mercu Buana
Universitas Mercu Buana
Universitas Mercu Buana
Universitas Mercu Buana
Looping 5 Kali
```

WHILE

Format nya adalah :





Keterangan

1. Mengeksekusi **Inisialisasi**

   Berisi nilai awal sebuah variabel yang disetting untuk sebuah variabel conter. Inisialisasi hanya dieksekusi sekali.

3. Memeriksa **Kondisi**

   Jika bernilai **Benar**

   - Laksanakan instruksi yang ada pada **loop (point 3)**.
   - Eksekusi **point 4**, yang berfungsi untuk merubah nilai dan mempengaruhi kondisi.
   - Kembali ke **Point 2**, memeriksa kondisi, dan seterusnya

   Jika bernilai **Salah,** loncat ke **point 5** yaitu instruksi berikutnya diluar looping atau proses looping berakhir.

Contoh

Mencetak tulisan “Universitas Mercu Buana” sebanyak 5 kali.

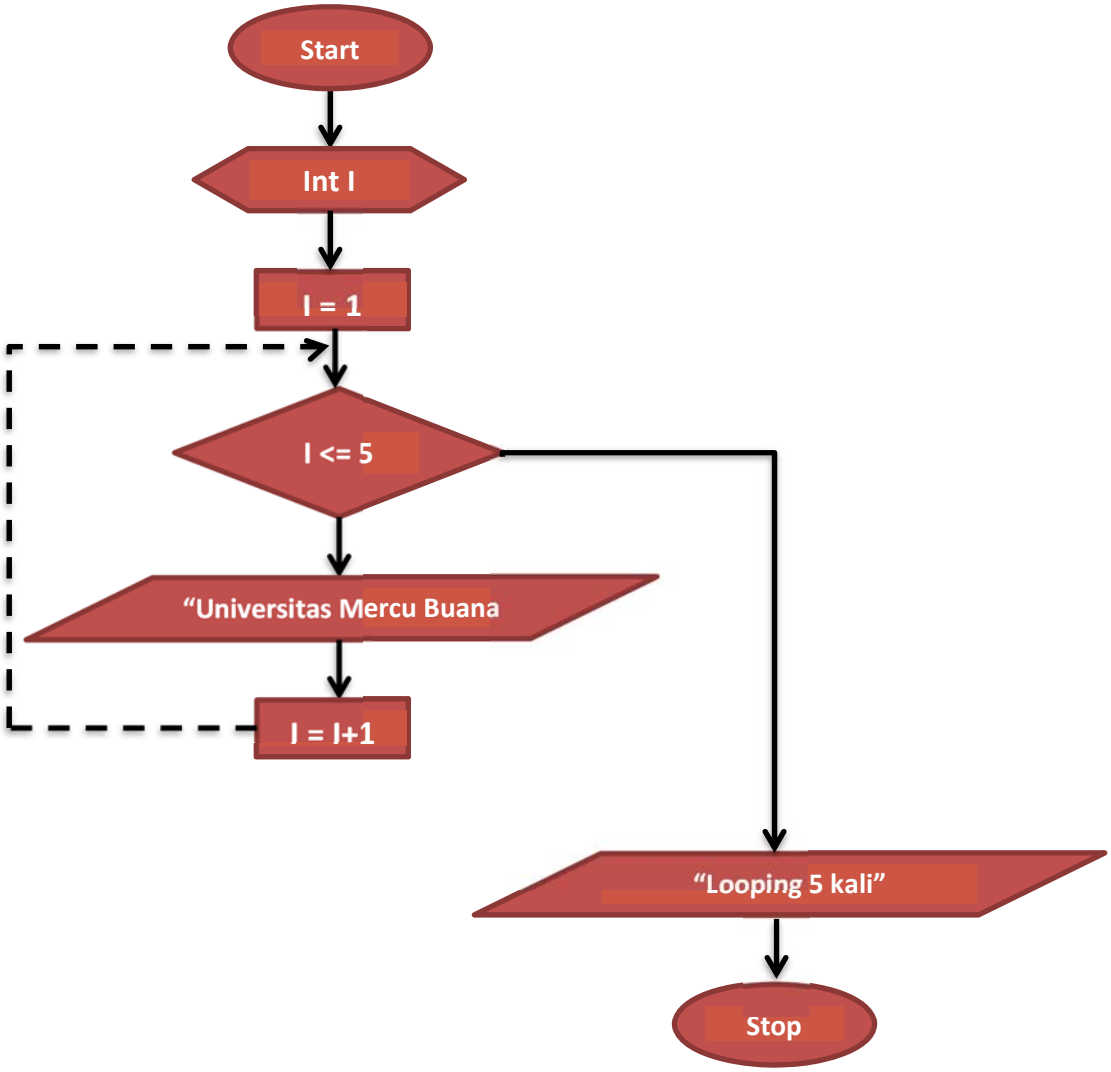
Solusi

2. Variabel → dibutuhkan sebuah variabel yang berfungsi untuk menghitung banyak cetakan, dalam contoh ini misalnya adalah **I** dengan tipe data **integer.**
3. **Kondisi** nya adalah:

   Jika nilai **I** diset 1 maka kondisinya adalah **I<=5**

   Jika nilai **I** diset 0 maka kondisinya adalah **I<5**

FLOWCHART



Perkembangan nilai I

Nilai I	Kondisi I<=5	Tercetak	I=I+1
1	True	Universitas Mercu Buana	2
2	True	Universitas Mercu Buana	3
3	True	Universitas Mercu Buana	4
4	True	Universitas Mercu Buana	5
5	False	Universitas Mercu Buana	6
6		Looping 5 Kali	Keluar Loop

Program

```
#include <iostream.h>
void main()
{
   int I;
    I=1;   (1)
   while (I<=5)  (2)
   {
    cout << “Universitas Mercu Buana \n”;  (3)
    I++;  (4)
    }
  cout << "Looping 5 Kali"; (5)
}
```

OUTPUT

```
Universitas Mercu Buana
Universitas Mercu Buana
Universitas Mercu Buana
Universitas Mercu Buana
Universitas Mercu Buana
Looping 5 Kali
```

DaftarPustaka

1. Kristianto. Andri, Algoritma dan Pemrograman dengan C++ Edisi 3, Yokyakarta, Graha Ilmu, 2013
2. Munir. Rinaldi, Algoritma dan Pemrograman Dalam Bahasa Pascal dan C, Bandung Informatika, 2007
3. Sjukani. Moh Algoritma (Algoritma dan Struktur Data1) dengan C, C++, dan Java, Jakarta, Mitra Kencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial, diakses tanggal 10 September 2013

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

Loop – While do

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	09	MK87001	Tim Dosen

Abstract

Modul ini berisi tentang perulangan menggunakan While-Do

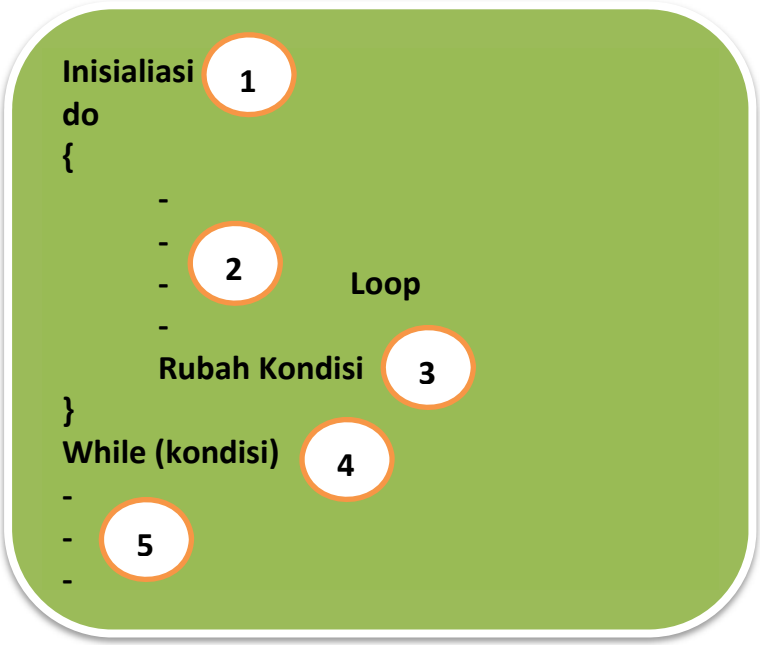
Kompetensi

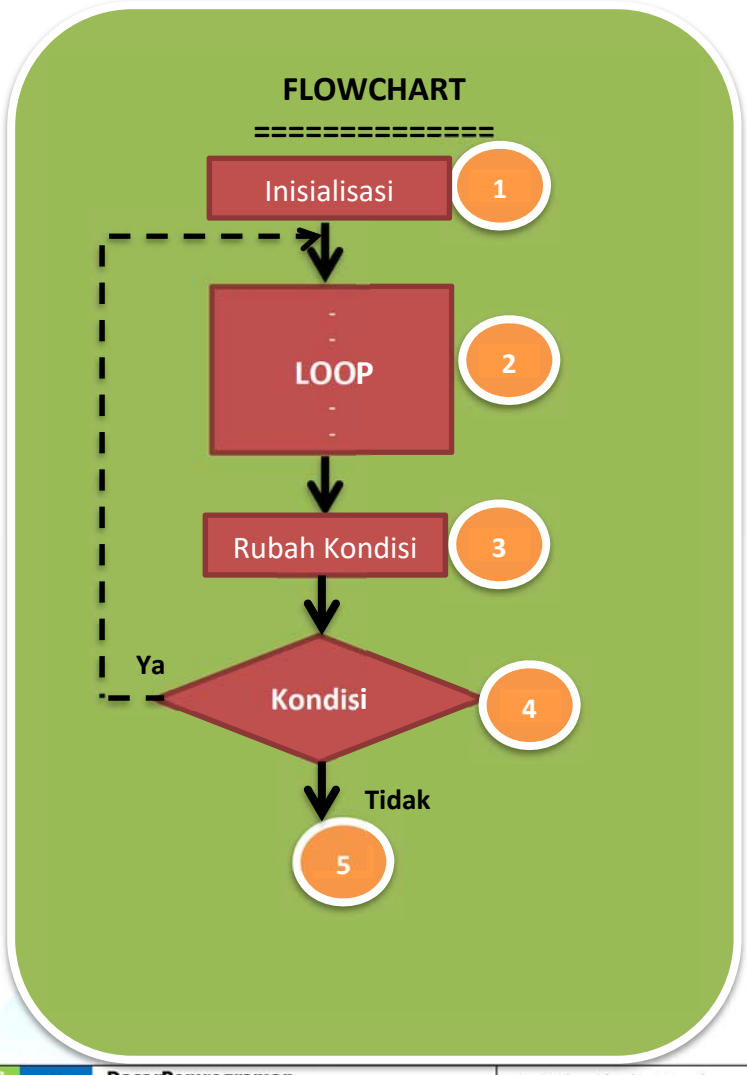
Diharapkan mahasiswa dapat:

- Memahami proses perulangan dengan menggunakan While Do
- Membuat program perulangan menggunakan While Do

While – Do

Format nya adalah :





Keterangan

1. Mengeksekusi Inisialisasi **(Point 1)**
2. Mengeksekusi **Point 2,** instruksi perulangan.
3. Mengeksekusi **Point 3 Rubah Kondisi**
4. Memeriksa **Kondisi (Point 4)**

   Jika bernilai **Benar**

   - Laksanakan instruksi yang ada pada **loop (point 2)**.
   - Kembali ke **Point 3**, memeriksa kondisi, dan seterusnya

   Jika bernilai **Salah,** loncat ke **point 5** yaitu instruksi berikutnya diluar looping atau proses looping berakhir.

Contoh

Mencetak tulisan “Universitas Mercu Buana” sebanyak 5 kali.

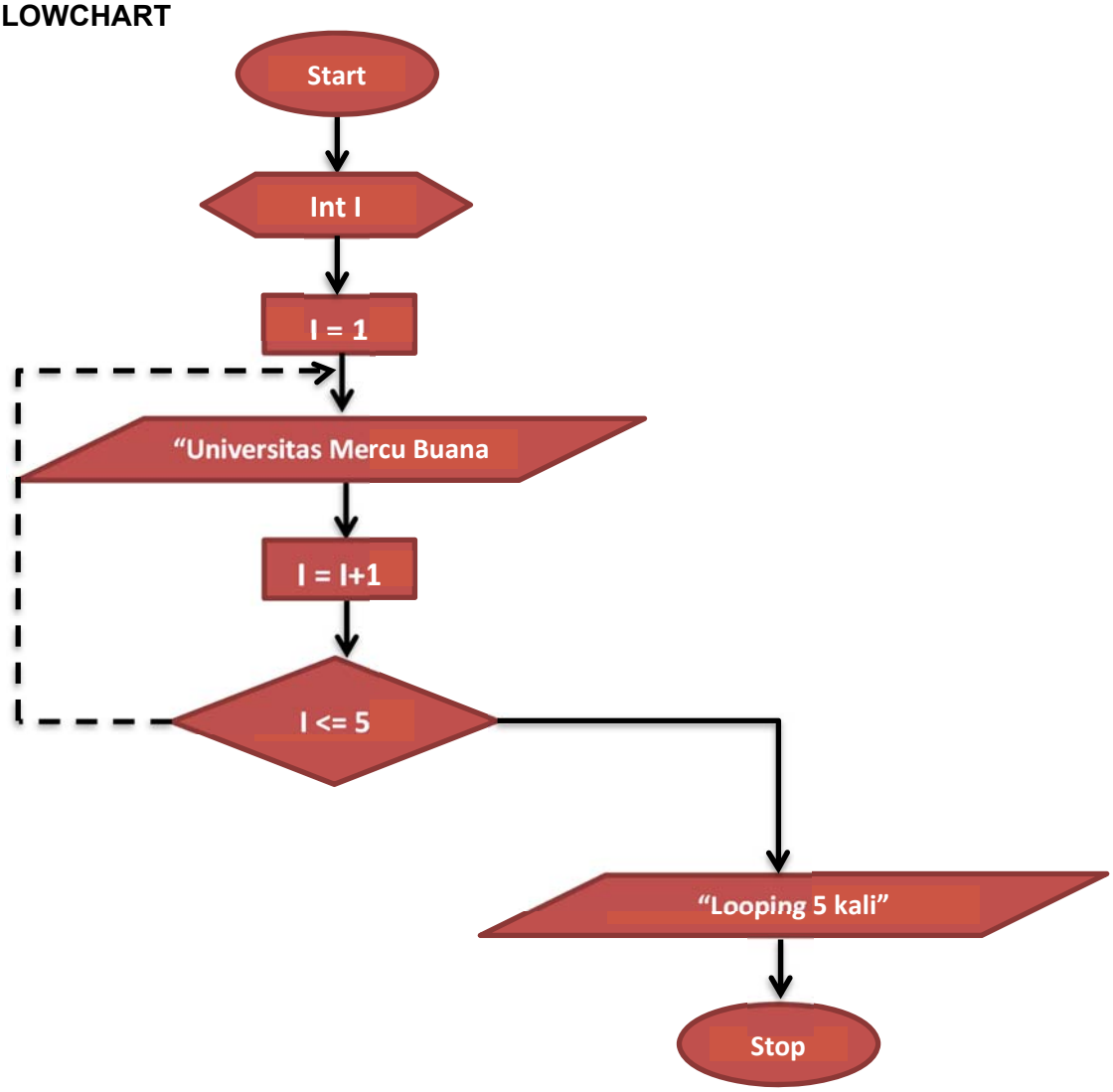
Solusi

1. Variabel → dibutuhkan sebuah variabel yang berfungsi untuk menghitung banyak cetakan, dalam contoh ini misalnya adalah **I** dengan tipe data **integer.**
2. **Kondisi** nya adalah:

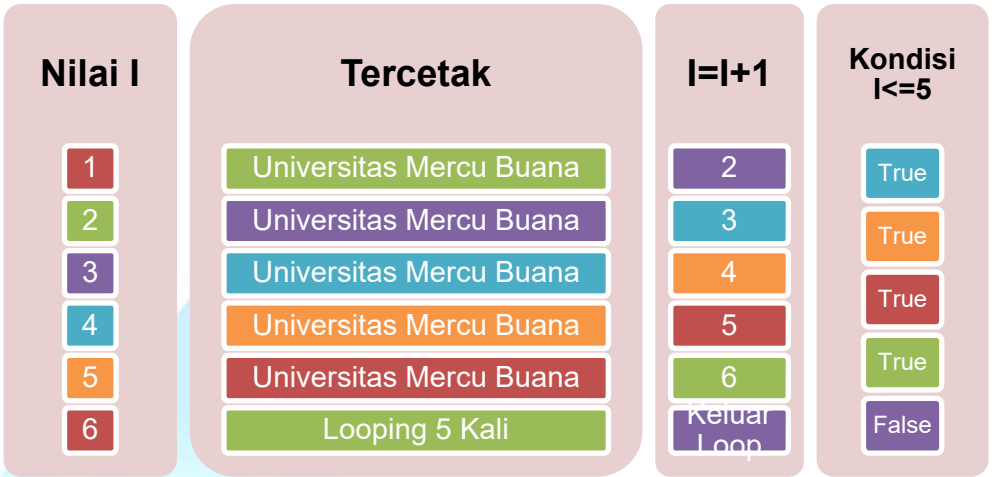
   Jika nilai **I** diset 1 maka kondisinya adalah **I<=5**

   Jika nilai **I** diset 0 maka kondisinya adalah **I<5**

FLOWCHART



Perkembangan nilai I



Program

```
#include<iostream.h>

void main()

{

 int I;

 I=1;

 do

 {

        cout<<"Universitas Mercu Buana \n" ;

        I++;

 }

 while (I<=5);

 cout<<"Looping 5 kali \n" ;

}
```

OUTPUT

```
Universitas Mercu Buana
Universitas Mercu Buana
Universitas Mercu Buana
Universitas Mercu Buana
Universitas Mercu Buana
Looping 5 Kali
```

Contoh-Contoh Soal

Soal 1

Menampilkan Angka 1 – 5

Program dengan For

```
#include<iostream.h>
void main()
{
 int I;
 for (I=1;I<=5;I++)
 {
 cout<<I<<"\n";
 }
}
```

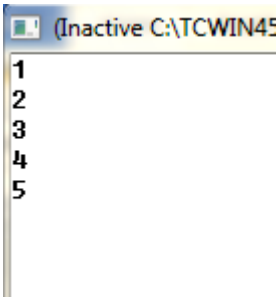
Program dengan While

```
#include<iostream.h>
void main()
{
 int I;
 I=1;
 while (I<=5)
 {
 cout<<I<<"\n";
 I++ ;
 }
}
```

Program dengan Do- While

```
#include<iostream.h>
void main()
{
 int I;
 I=0;
 do
 {
 I++;
 cout<<I<<"\n";
 }
 while (I < 5);
}
```

Layout



Soal 2

Menampilkan deretan angka sampai nilai maksimal yang diinputkan

Program dengan For

```
#include<iostream.h>
void main()
{
 int I,N;
 cout<<"Inputkan Nilai Maksimum :";
 cin>>N;
 for (I=1; I<=N;I++)
 {
 cout<<I<<"\n";
 }
 }
```

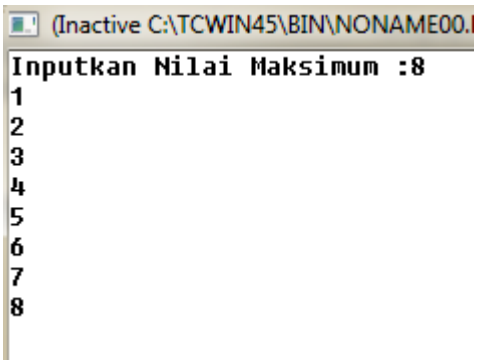
Program dengan While

```
#include<iostream.h>
void main()
{
 int I,N;
 cout<<"Inputkan Nilai Maksimum :";
 cin>>N;
 I=1;
 while (I<=N)
 {
 cout<<I<<"\n";
 I++ ;
 }
}
```

Program dengan do-While

```
#include<iostream.h>
void main()
{
 int I,N;
 I=0;
 cout<<"Inputkan Nilai Maksimum :";
 cin>>N;
 do
 {
 I++;
 cout<<I<<"\n";
 }
 while (I < N);
}
```

Layout



DaftarPustaka

1. Kristianto. Andri, Algoritma dan Pemrograman dengan C++ Edisi 3, Yokyakarta, Graha Ilmu, 2013
2. Munir. Rinaldi, Algoritma dan Pemrograman Dalam Bahasa Pascal dan C, Bandung Informatika, 2007
3. Sjukani. Moh Algoritma (Algoritma dan Struktur Data1) dengan C, C++, dan Java, Jakarta, Mitra Kencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial, diakses tanggal 10 September 2013

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

Loop – Break dan Continue

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	10	MK87001	Tim Dosen

Abstract

Modul ini berisi tentang perulangan dalam menggunakan Break & Continue

Kompetensi

Diharapkan mahasiswa dapat:

- Memahami proses perulangan dengan menggunakan Break dan Continue
- Membuat program perulangan menggunakan Brea & Continue

Jump statements

Break statement

Fungsi **break** adalah untuk meninggalkan sebuah loop ketika kondisi untuk akhiri perulangan tidak terpenuhi. Break dapat digunakan untuk mengakhiri sebuah infinite loop atau mengakhiri dengan tidak normal.

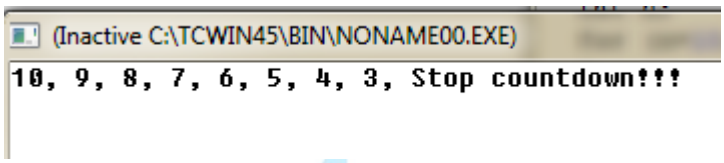
Contoh 1

Program untuk melakukan perhitungan mundur . Jika nilai hitungan sudah sampai dengan 3, maka proses dihentikan.

Program

```
#include <iostream.h>

void main ()
{
  int n;
  for (n=10; n>0; n--)
  {
        cout << n << ", ";
        if (n==3)
        {
              cout << "Stop countdown!!!";
              break;
        }
  }
}
```

Layout



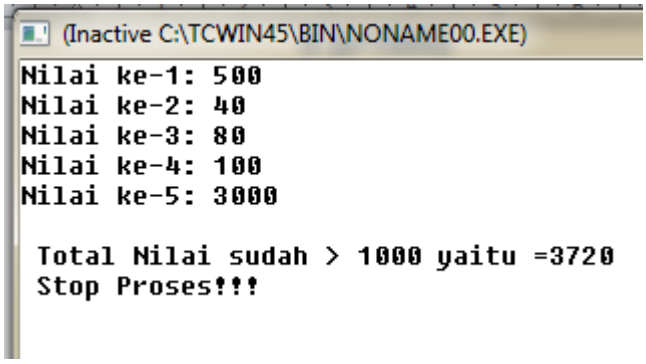
Contoh 2

Program untuk menghitung total nilai yang diinputkan, dimana nilai yang diinputkan adalah maksimum 10 nilai. Jika total nilai yang diinput >1000, maka proses dihentikan.

Program

```
#include <iostream.h>

void main ()
{
 int i,n,t;
 t=0;
 for (i=1;i<=10;i++)
 {
        cout<<"Nilai ke-"<<i<<": ";
        cin>>n;
        t=t+n;
        if (t>1000)
        {
              cout<<"\n Total Nilai sudah > 1000 yaitu ="<<t;
              cout <<"\n Stop Proses!!!";
              break;
        }
 }
}
```

Layout



Continue statement

Continue menyebabkan program di-skip pada bagian tertentu.

Contoh

Program untuk men-skip angka 5 pada sebuah perhitungan mundur:

```
1   // continue loop example
2   #include <iostream>
3   using namespace std;
4
5   int main ()
6   {
7     for (int n=10; n>0; n--) {
8       if (n==5) continue;
9       cout << n << ", ";
10    }
11    cout << "FIRE!\n";
12    return 0;
13  }
```

```
10, 9, 8, 7, 6, 4, 3, 2, 1, FIRE!
```

The goto statement

goto allows to make an absolute jump to another point in the program. You should use this feature with caution since its execution causes an unconditional jump ignoring any type of nesting limitations.

The destination point is identified by a label, which is then used as an argument for the goto statement. A label is made of a valid identifier followed by a colon (:).

Generally speaking, this instruction has no concrete use in structured or object oriented programming aside from those that low-level programming fans may find for it. For example, here is our countdown loop using goto:

```
1  // goto loop example
2
3  #include <iostream>
4  using namespace std;
5
6  int main ()
7  {
8    int n=10;
9    loop:
10   cout << n << ", ";
11   n--;
12   if (n>0) goto loop;
13   cout << "FIRE!\n";
14   return 0;
15 }
```

```
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, FIRE!
```

The exit function

exit is a function defined in the cstdlib library.

The purpose of exit is to terminate the current program with a specific exit code. Its prototype is:

```
void exit (int exitcode);
```

The exitcode is used by some operating systems and may be used by calling programs. By convention, an exit code of 0 means that the program finished normally and any other value means that some error or unexpected results happened.

The selective structure: switch.

The syntax of the switch statement is a bit peculiar. Its objective is to check several possible constant values for an expression. Something similar to what we did at the beginning of this section with the concatenation of severalif and else if instructions. Its form is the following:

```
switch (expression)
{
  case constant1:
     group of statements 1;
     break;
  case constant2:
     group of statements 2;
     break;
  .
  .
  .
  default:
     default group of statements
}
```

It works in the following way: switch evaluates expression and checks if it is equivalent to constant1, if it is, it executes group of statements 1 until it finds the break statement. When it finds this break statement the program jumps to the end of the switch selective structure.

If expression was not equal to constant1 it will be checked against constant2. If it is equal to this, it will executegroup of statements 2 until a break keyword is found, and then will jump to the end of the switch selective structure.

Finally, if the value of expression did not match any of the previously specified constants (you can include as manycase labels as values you want to check), the program will execute the statements included after the default:label, if it exists (since it is optional).

Both of the following code fragments have the same behavior:

switch example	**if-else equivalent**

switch (x) {
  case 1:
  cout << "x is 1";
  break;
  case 2:
  cout << "x is 2";
  break;
  default:
  cout << "value of x unknown";
  }
	if (x == 1) {
	  cout << "x is 1";
	  }
	else if (x == 2) {
	  cout << "x is 2";
	  }
	else {
	  cout << "value of x unknown";
	  }

The switch statement is a bit peculiar within the C++ language because it uses labels instead of blocks. This forces us to put break statements after the group of statements that we want to be executed for a specific condition. Otherwise the remainder statements - including those corresponding to other labels- will also be executed until the end of the switch selective block or a break statement is reached.

For example, if we did not include a break statement after the first group for case one, the program will not automatically jump to the end of the switch selective block and it would continue executing the rest of statements until it reaches either a break instruction or the end of the switch selective block. This makes it unnecessary to include braces { } surrounding the statements for each of the cases, and it can also be useful to execute the same block of instructions for different possible values for the expression being evaluated. For example:

```
switch (x) {
  case 1:
  case 2:
  case 3:
    cout << "x is 1, 2 or 3";
    break;
  default:
    cout << "x is not 1, 2 nor 3";
  }
```

Notice that switch can only be used to compare an expression against constants. Therefore we cannot put variables as labels (for example case n: where n is a variable) or ranges (case (1..3):) because they are not valid C++ constants.

If you need to check ranges or values that are not constants, use a concatenation of if and else if statements.

DaftarPustaka

1. Kristianto. Andri, Algoritma dan Pemrograman dengan C++ Edisi 3, Yokyakarta, Graha Ilmu, 2013
2. Munir. Rinaldi, Algoritma dan Pemrograman Dalam Bahasa Pascal dan C, Bandung Informatika, 2007
3. Sjukani. Moh Algoritma (Algoritma dan Struktur Data1) dengan C, C++, dan Java, Jakarta, Mitra Kencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial, diakses tanggal 10 September 2013

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

Nested Loop

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	11	MK87001	Tim Dosen

Abstract

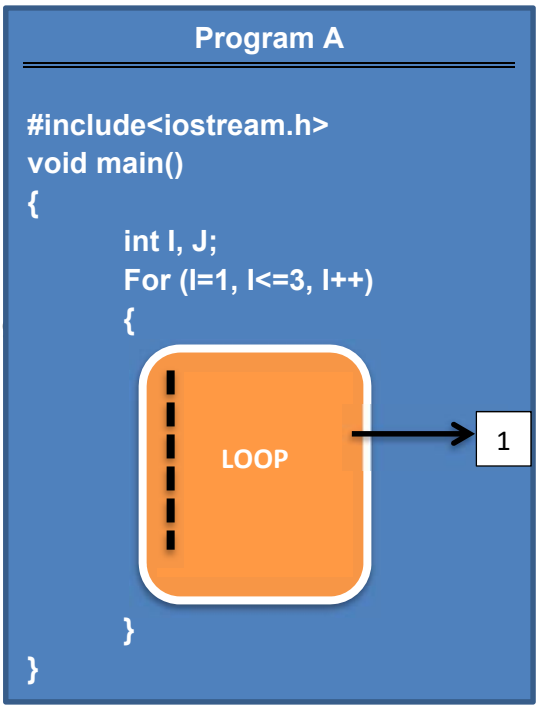
Modul ini berisi tentang perulangan bersarang

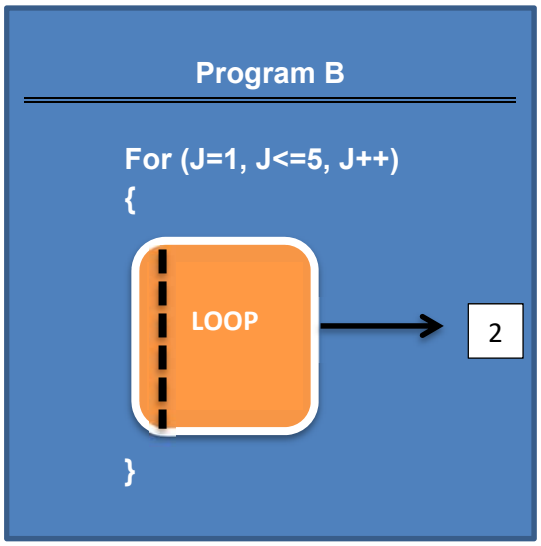
Kompetensi

Diharapkan mahasiswa dapat:

- Memahami proses perulangan bersarang
- Membuat program perulangan bersarang

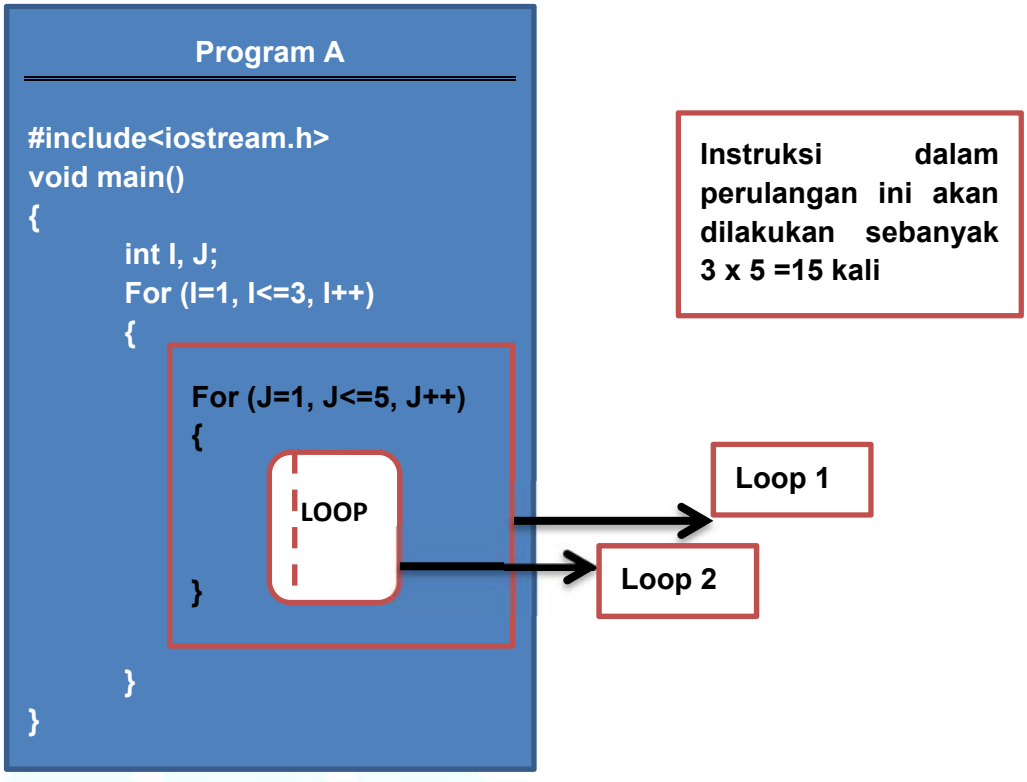
Konsep Dasar



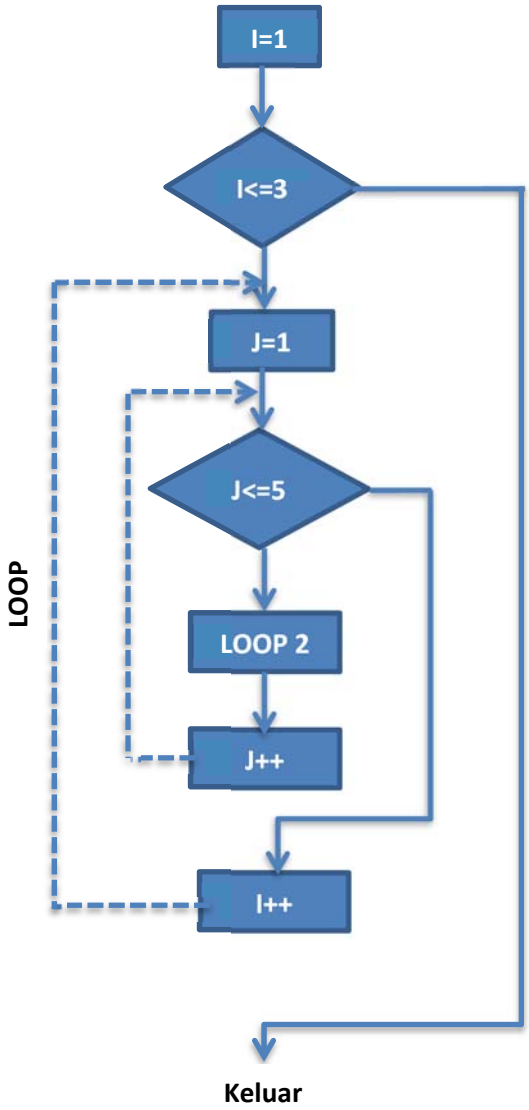


Pada program A ada perulangan yang dikerjakan sebanyak 3 kali, dan potongan program B terdapat perulangan yang dikerjakan sebanyak 5 kali.

Jika Program B dimasukkan kedalam program A, maka terdapat perulangan didalam perulangan yang disebut NESTED LOOP.



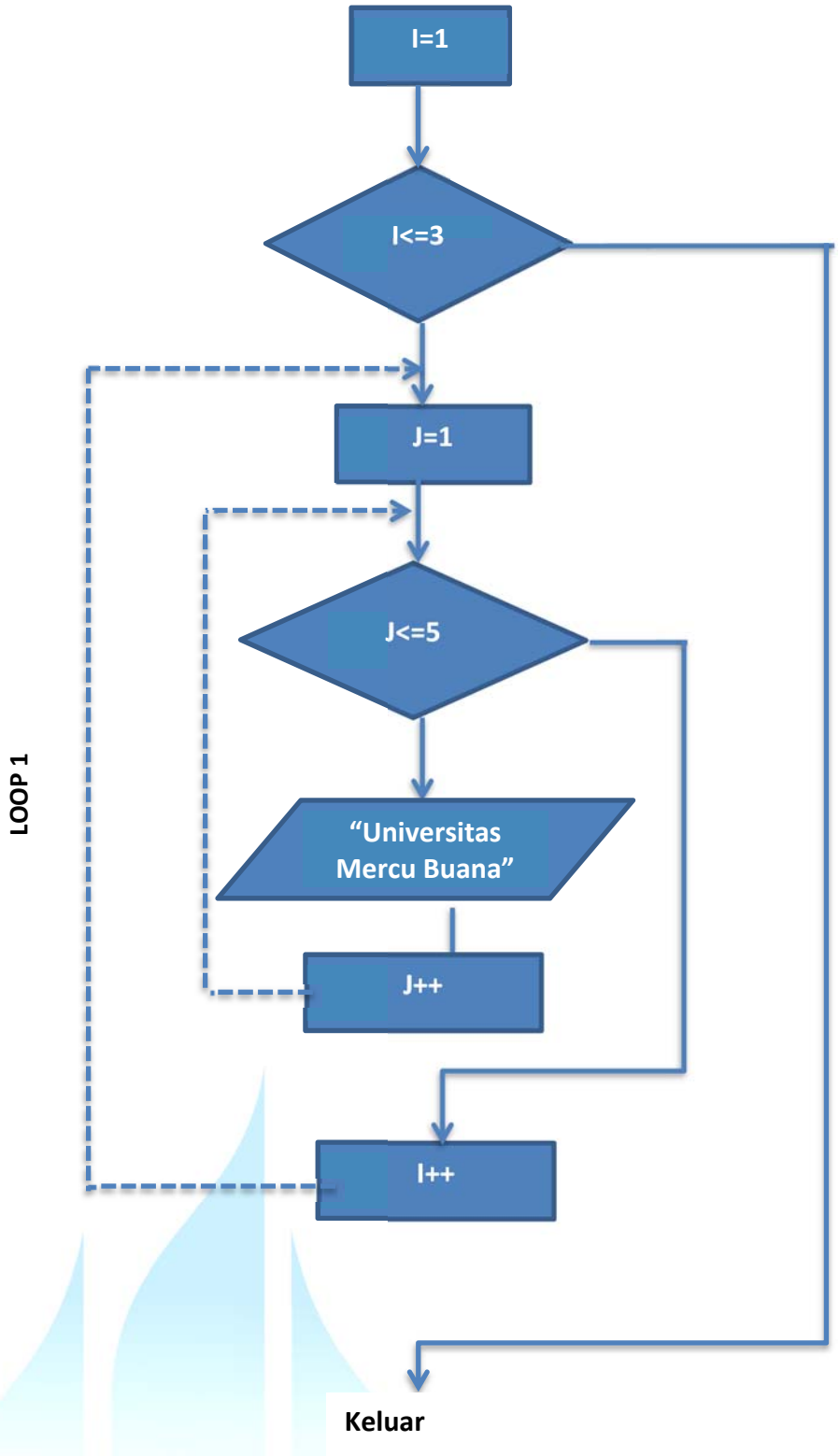
Flowchart



Contoh Penggunaan NESTED LOOP

Mencetak tulisan Universitas Mercu Buana 15 kali dengan menggunakan 2 perulangan

Flowchart



Program

```
#include<iostream.h>
void main()
{
        int I, J;
        for (I=1, I<=3, I++)
        {
           for (J=1, J<=5, J++)
           {
            cout<<"Universitas Mercu Buana";
        }
}
```

Analisis

Nilai I	Nilai J	Output
1	1	Universitas Mercu Buana
	2	Universitas Mercu Buana
	3	Universitas Mercu Buana
	4	Universitas Mercu Buana
	5	Universitas Mercu Buana
2	1	Universitas Mercu Buana
	2	Universitas Mercu Buana
	3	Universitas Mercu Buana
	4	Universitas Mercu Buana
	5	Universitas Mercu Buana
3	1	Universitas Mercu Buana
	2	Universitas Mercu Buana
	3	Universitas Mercu Buana
	4	Universitas Mercu Buana
	5	Universitas Mercu Buana

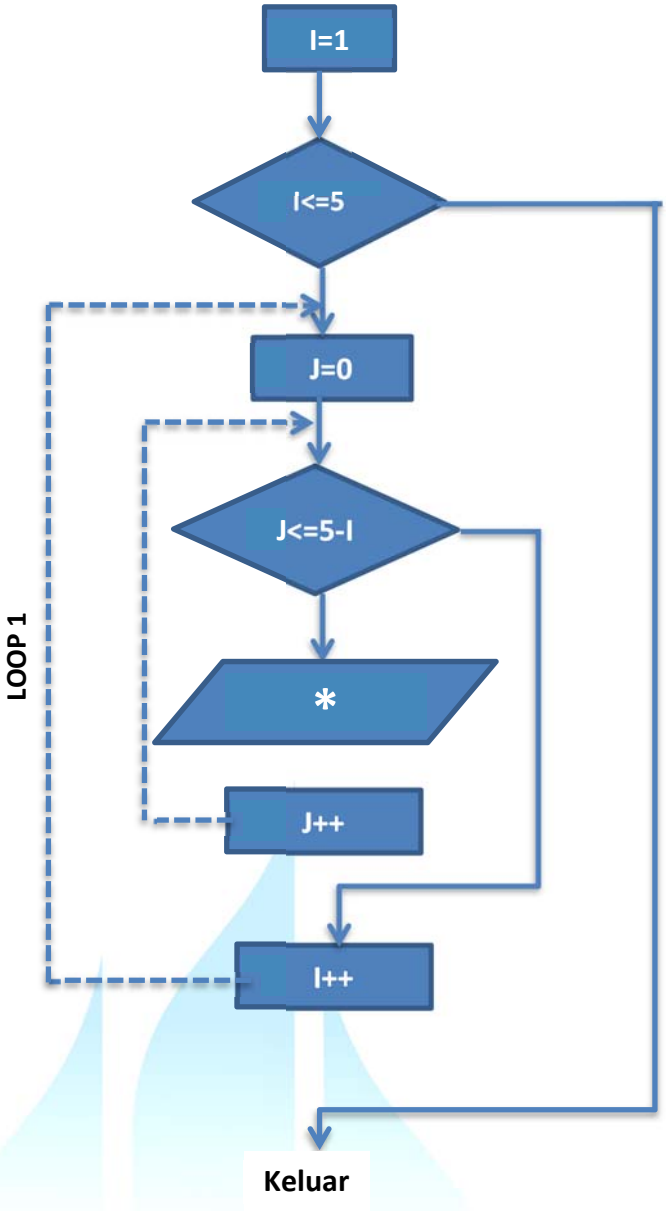
Contoh

1. Menampilkan layout seperti berikut

```
*****
****
***
**
*
```

Jawab

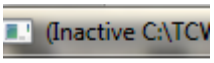
Flowchart



Program

```
#include<iostream.h>
void main()
{
int I, J;
        for (I=1; I<=5; I++)
        {
                for (J=0; J<=5-I; J++)
                {
                 cout<<"*";
                }
                cout<<endl;
        }
}
```

Hasil



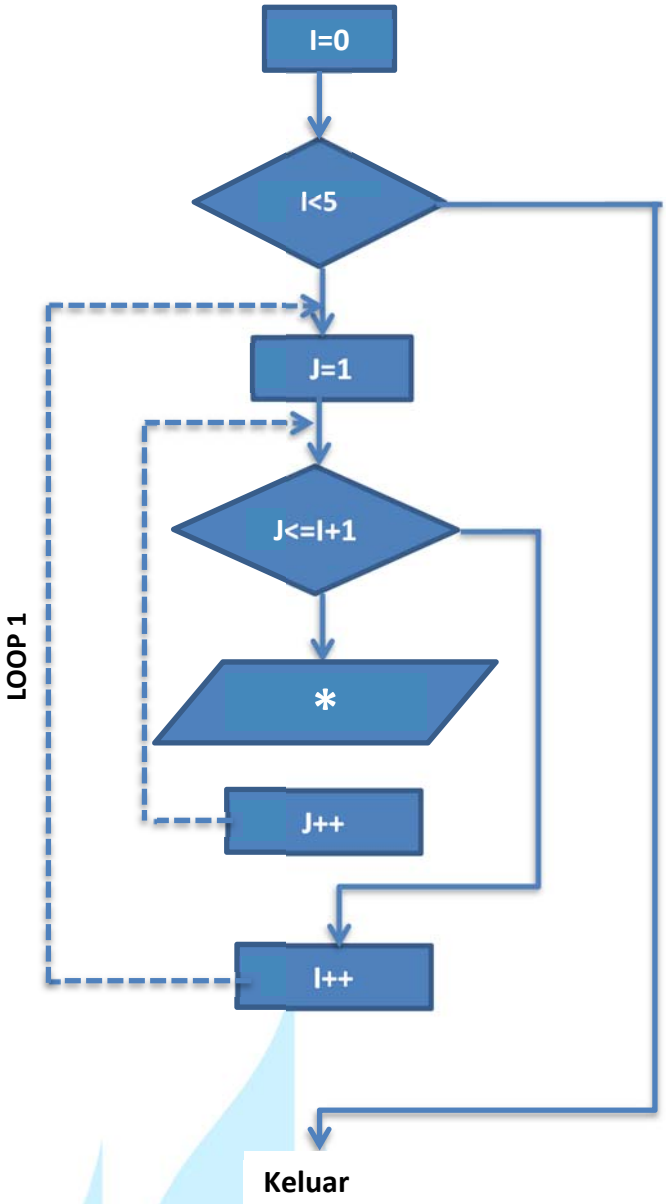

```
*****
****
***
**
*
```

2. Menampilkan layout seperti berikut

```
*
**
***
****
*****
```

Jawab

Flowchart



Program

```
#include<iostream.h>
void main()
{
int I, J;
        for (I=0; I<5; I++)
        {
                for (J=1; J<=I+1; J++)
                {
                 cout<<"*";
                }
                cout<<endl;
        }
}
```

Hasil

DaftarPustaka

1. Kristianto. Andri, Algoritma dan Pemrograman dengan C++ Edisi 3, Yokyakarta, Graha Ilmu, 2013
2. Munir. Rinaldi, Algoritma dan Pemrograman Dalam Bahasa Pascal dan C, Bandung Informatika, 2007
3. Sjukani. Moh Algoritma (Algoritma dan Struktur Data1) dengan C, C++, dan Java, Jakarta, Mitra Kencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial, diakses tanggal 10 September 2013

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

Aplikasi Dasar Loop

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	12	MK87001	Tim Dosen

Abstract

Modul ini berisi tentang contoh-contoh penggunaan looping dalam berbagai kasus

Kompetensi

Diharapkan mahasiswa dapat:

- Memahami dan membuat program untuk kasus-kasus terkait perulangan

n

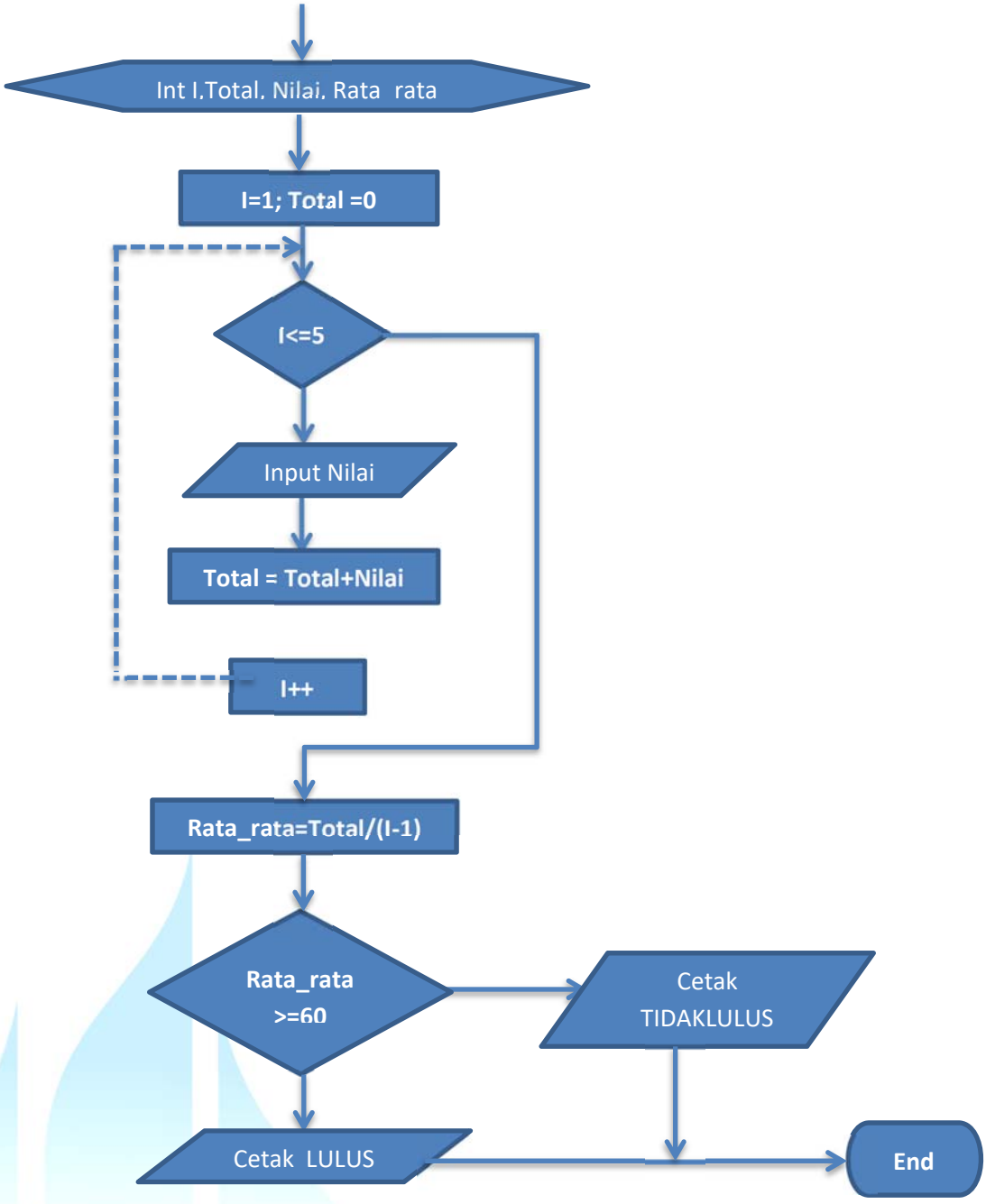
Contoh 1

Mencetak keterangan Lulus atau tidak berdasarkan nilai rata-rata dari 5 nilai yang diinput. Jika nilai rata-rata nya >= 60 maka dinyatakan Lulus, jika <60 maka dinyatakan tidak lulus.

Jawab

---

1. **Flowchart**



2. **Program**

```
#include<iostream.h>
void main()
{
int I, nilai,total,rata_rata;
total=0;
        for (I=1; I<=5; I++)
        {
                cout<<"Input nilai ke-"<<I<<":";
                cin>>nilai;
                total=total+nilai  ;
        }
        cout<<"\nNilai Total yang diperoleh adalah :"<<total;
        rata_rata=total/(I-1);
        cout<<"\nNilai Rata-rata= "<<rata_rata;
        if (rata_rata >=60)
        cout<<"\n\n Keterangan LULUS"  ;
        else
        cout<<"\n\n Keterangan TIDAK LULUS";
}
```

3. **Analisis**

I	I<=5	Nilai Input	Total	I++	Rata rata	Rata_rata>=60	Keterangan
1	Ya (looping)	70	70	2	-	-	-
2	Ya (looping)	60	130	3	-	-	-
3	Ya (looping)	65	195	4	-	-	-
4	Ya (looping)	55	250	5	-	-	-
5	Ya (looping)	70	320	6	-	-	-
6	**Tidak (Exit looping)**		**320**	**6**	320/6-1= 64	Ya	LULUS

4. **Output**

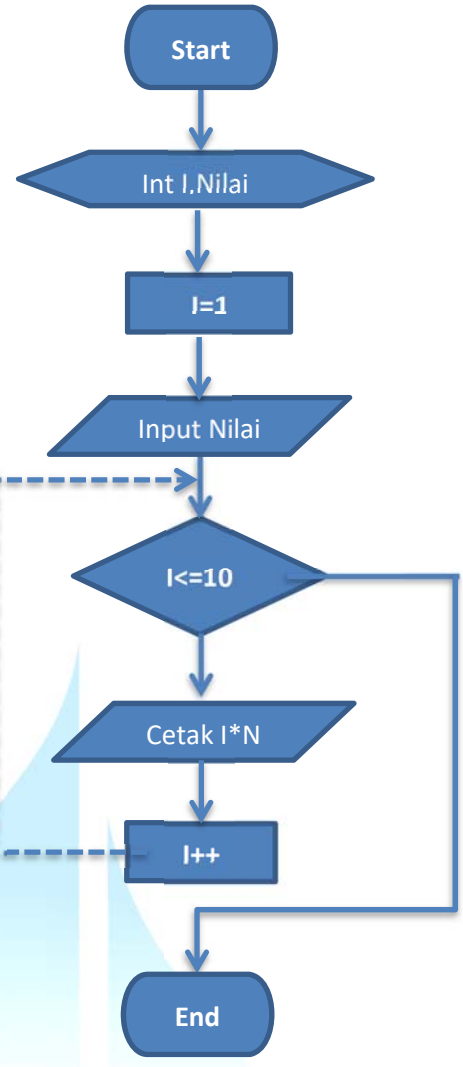
```
(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
Input nilai ke-1:70
Input nilai ke-2:60
Input nilai ke-3:65
Input nilai ke-4:55
Input nilai ke-5:70

Nilai Total yang diperoleh adalah :320
Nilai Rata-rata= 64

 Keterangan LULUS
```

Contoh 2

Mencetak deret perkalian 1 – 10, dari bilangan yang diinputkan

Jawab

1. **Flowchart**



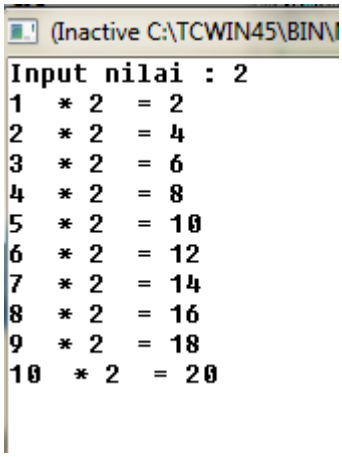
2. Program

```
#include<iostream.h>
void main()
{
int I, nilai;

 cout<<"Input nilai : ";
 cin>>nilai;

        for (I=1; I<=10; I++)
        {
                cout<<I<<"  * "<<nilai<<"  = "<<I*nilai;
                cout<<endl;

        }

 }
```

3. Analisis

Input Nilai	I	I<=10	Output	I++
2	1	Ya (looping)	1 * 2=2	2
	2	Ya (looping)	2 * 2=4	3
	3	Ya (looping)	3 * 2=6	4
	4	Ya (looping)	4 * 2=8	5
	5	Ya (looping)	5 * 2=10	6
	6	Ya (looping)	6 * 2=12	7
	7	Ya (looping)	7 * 2=14	8
	8	Ya (looping)	8 * 2=16	9
	9	Ya (looping)	9 * 2=18	10
	10	Ya (looping)	10* 2=20	11
	11	Tidak (Exit Looping)		

4. **Output**

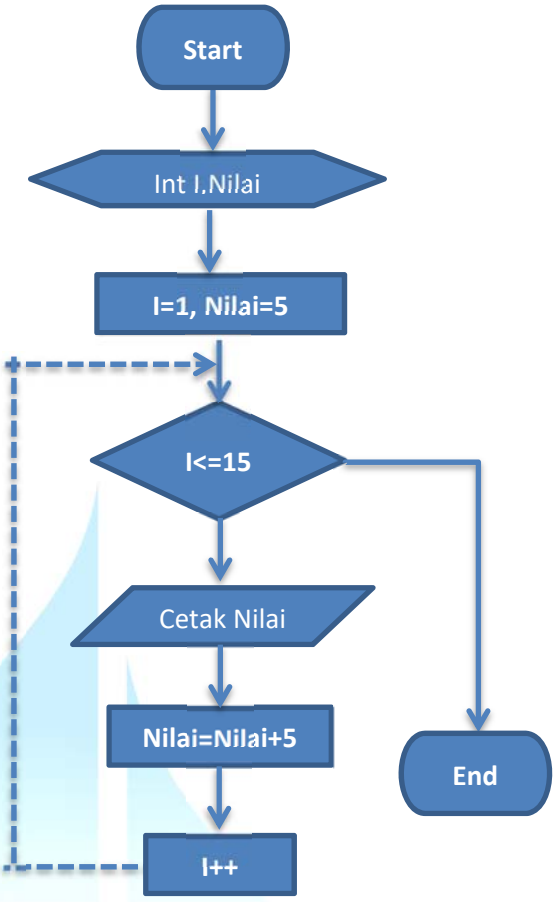


Contoh 3

Mencetak 15 deret matematika pertama kelipatan 5 : 5 10 15 20 .....

Jawab

1. **Flowchart**



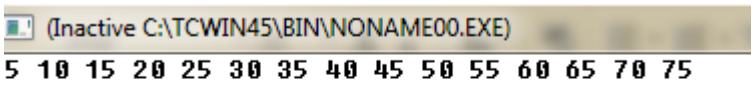
2. Program

```
#include<iostream.h>
void main()
{
int I, nilai;

        nilai = 5;
        for (I=1; I<=15; I++)
        {
                cout<<nilai<<" ";
                nilai=nilai+5;
        }

}
```

3. Analisis

Nilai	I	I<=15	Cetak Nilai	Nilai = Nilai+5	I++
5	1	Ya (looping)	5	5+5=10	2
10	2	Ya (looping)	10	10+5=15	3
15	3	Ya (looping)	15	15+5=20	4
20	4	Ya (looping)	20	20+5=25	5
25	5	Ya (looping)	25	25+5=30	6
30	6	Ya (looping)	30	30+5=35	7
35	7	Ya (looping)	35	35+5=40	8
40	8	Ya (looping)	40	40+5=45	9
45	9	Ya (looping)	45	45+5=50	10
50	10	Ya (looping)	50	50+5=55	11
55	11	Ya (looping)	55	55+5=60	12
60	12	Ya (looping)	60	60+5=65	13
65	13	Ya (looping)	65	65+5=70	14
70	14	Ya (looping)	70	70+5=75	15
75	15	Ya (looping)	75	75+5=80	16
80	16	Tidak (Exit Looping			

4. Output


(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
5 10 15 20 25 30 35 40 45 50 55 60 65 70 75

DaftarPustaka

1. Kristianto. Andri, Algoritma dan Pemrograman dengan C++ Edisi 3, Yokyakarta, Graha Ilmu, 2013
2. Munir. Rinaldi, Algoritma dan Pemrograman Dalam Bahasa Pascal dan C, Bandung Informatika, 2007
3. Sjukani. Moh Algoritma (Algoritma dan Struktur Data1) dengan C, C++, dan Java, Jakarta, Mitra Kencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial, diakses tanggal 10 September 2013

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

Aplikasi gabungan multi kondisi dan looping

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	13	MK87001	Tim Dosen

Abstract

Modul ini berisi tentang contoh-contoh penggunaan multi kondisi dan looping dalam berbagai kasus

Kompetensi

Diharapkan mahasiswa dapat:

- Memahami dan membuat program untuk kasus-kasus terkait multi kondisi dan perulangan

Contoh Kasus

Buatlah program untuk kasus berikut:

Jenis Susu		Jenis Ukuran	
Kode	Jenis Susu	Kode	Jenis Ukuran
1	Dancow	A	Besar
2	Bendera	B	Sedang
3	SGM	C	Kecil

Kode	Harga		
	A	B	C
1	10000	20000	30000
2	25000	50000	75000
3	20000	40000	60000

Total = Jumlah * Harga

Jawab

1. **Layout**

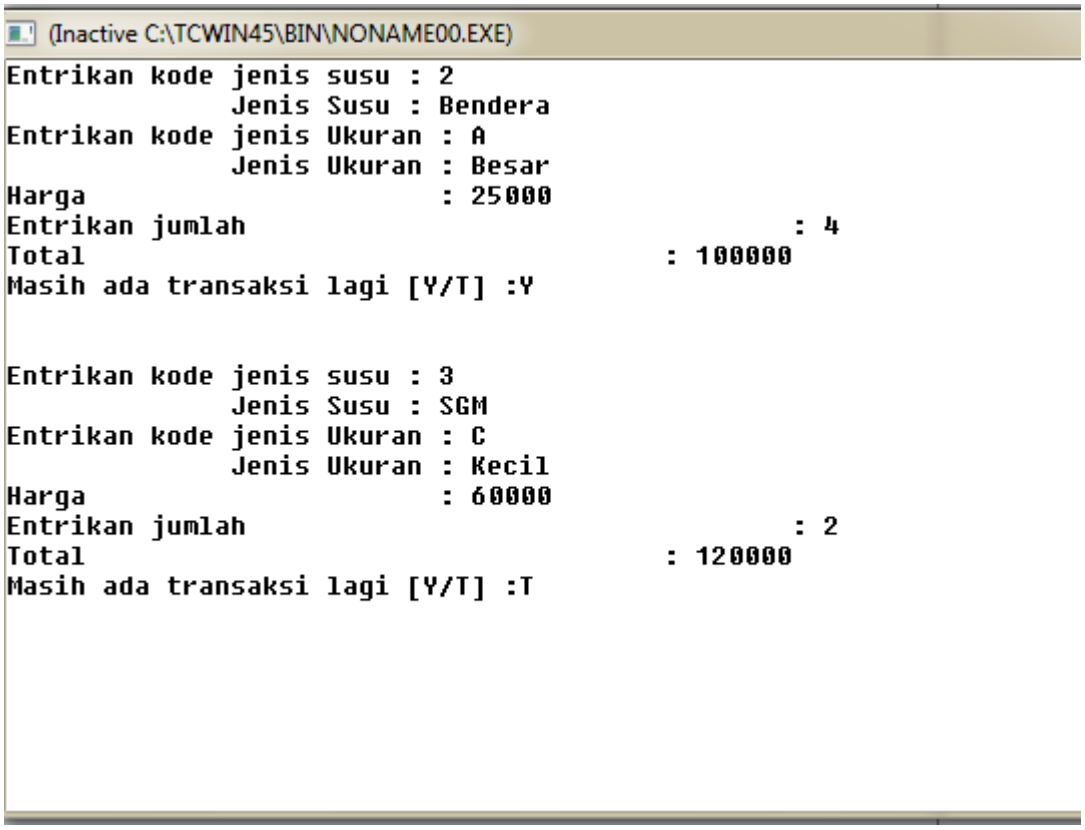
Entrikan Kode Jenis Susu :
Jenis Susu :
Entrikan Kode Jenis Ukuran :
Jenis Ukuran :
Harga :
Jumlah beli :
Total :
Masih ada transaksi lain [Y/T] :

2. **Program**

```
#include<iostream.h>
void main()
{
//inisialisasi variabel
char menu, KU;
int KJ, jumlah;
double harga,total;

//setting awal nilai variabel
harga=0;
total=0;
jumlah=0;

//PERULANGAN
do
{
  //PROSES KODE JENIS SUSU
        cout<<"Entrikan kode jenis susu : ";
        cin>>KJ;

        if (KJ==1)
        cout<<"          Jenis Susu : Dancow \n";
        else if (KJ==2)
        cout<<"          Jenis Susu : Bendera \n";
        else if (KJ==3)
        cout<<"          Jenis Susu : SGM \n";

        //PROSES KODE JENIS UKURAN
        cout<<"Entrikan kode jenis Ukuran : ";
        cin>>KU;

        if (KU=='A')
        cout<<"          Jenis Ukuran : Besar \n";
        else if (KU=='B')
        cout<<"          Jenis Ukuran : Sedang \n";
        else if (KU=='C')
        cout<<"          Jenis Ukuran : Kecil \n";
```

```
//PROSES HARGA
        if ((KJ==1) && (KU=='A'))
                harga = 10000;
        else if ((KJ==1) && (KU=='B'))
                harga = 20000;
        else if ((KJ==1) && (KU=='C'))
                harga = 30000;

        else if ((KJ==2) && (KU=='A'))
                harga = 25000;
        else if ((KJ==2) && (KU=='B'))
                harga = 50000;
        else if ((KJ==2) && (KU=='C'))
                harga = 75000;

        else if ((KJ==3) && (KU=='A'))
                harga = 20000;
        else if ((KJ==3) && (KU=='B'))
                harga = 40000;
        else if ((KJ==3) && (KU=='C'))
                harga = 60000;

        cout<<"Harga                : "<<harga<<endl;
        cout<<"Entrikan jumlah                          : ";
        cin>>jumlah;

        total=harga*jumlah;
        cout<<"Total                                    : "<<total<<endl;

        cout<<"Masih ada transaksi lagi [Y/T] :";
        cin>>menu;

        cout<<endl<<endl;

 }
 while(menu=='Y');
}
```

3. Output

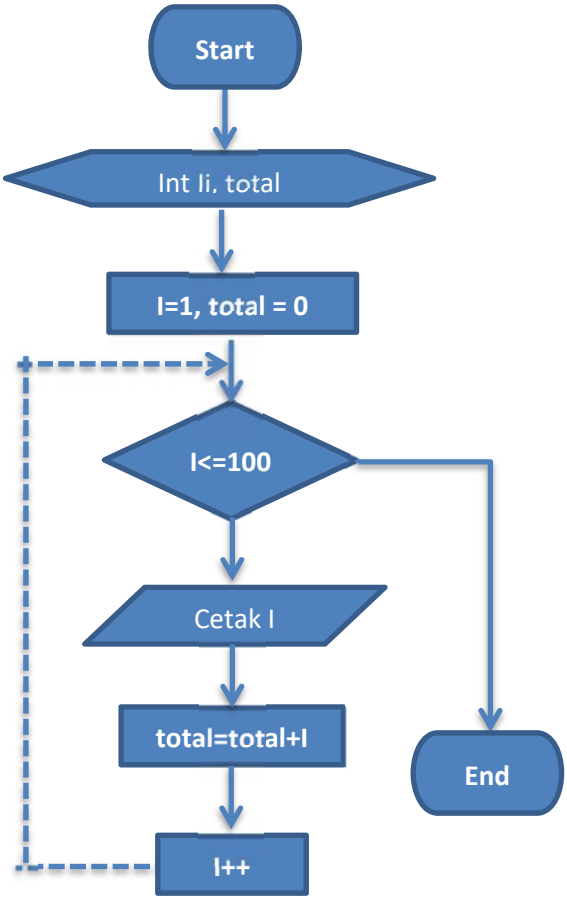

(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
Entrikan kode jenis susu : 2
Jenis Susu : Bendera
Entrikan kode jenis Ukuran : A
Jenis Ukuran : Besar
Harga : 25000
Entrikan jumlah : 4
Total : 100000
Masih ada transaksi lagi [Y/T] :Y
Entrikan kode jenis susu : 3
Jenis Susu : SGM
Entrikan kode jenis Ukuran : C
Jenis Ukuran : Kecil
Harga : 60000
Entrikan jumlah : 2
Total : 120000
Masih ada transaksi lagi [Y/T] :T

Contoh 2

Mencetak nilai total dari 1 sampai 100

Jawab

1. **Flowchart**



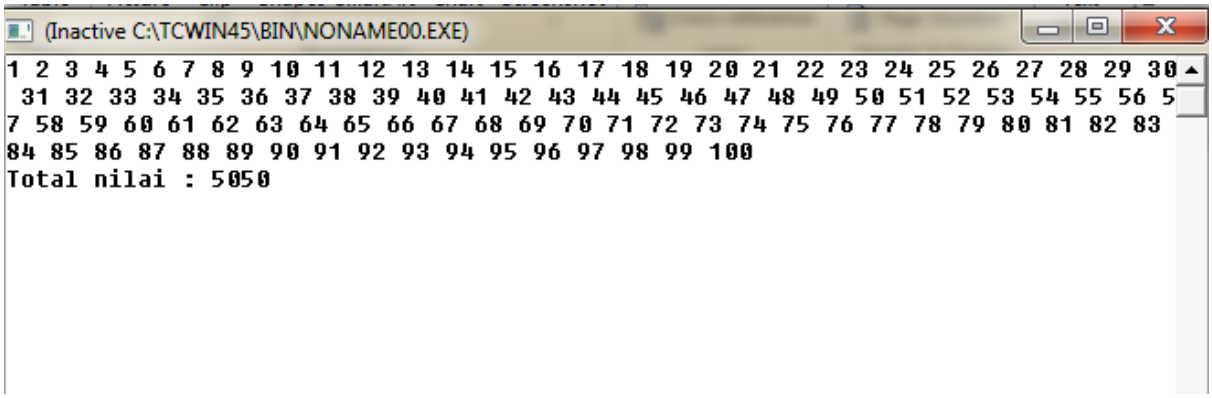
2. **Program**

```
#include<iostream.h>
void main()
{
int I, total;

        total = 0;
        for (I=1; I<=100; I++)
        {
                cout<<I<<" ";
                total=total+I;
        }
        cout<<endl<<"Total nilai : "<<total;
}
```

3. **Analisis**

I	I<=100	Cetak I	Total	I++
1	Ya (looping)	1	0+1=1	2
2	Ya (looping)	2	1+2=3	3
3	Ya (looping)	3	3+3=6	4
4	Ya (looping)	4	6+4=10	5
5	Ya (looping)	5	10+5=15	6
6	Ya (looping)	6	15+6=21	7
7	Ya (looping)	7	21+7=28	8
8	Ya (looping)	8	28+8=36	9
9	Ya (looping)	9	36+9=45	10
10	Ya (looping)	10	45+10=55	11
Dst....				

4. Output


(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 5
7 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83
84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
Total nilai : 5050

DaftarPustaka

1. Kristianto. Andri, Algoritma dan Pemrograman dengan C++ Edisi 3, Yokyakarta, Graha Ilmu, 2013
2. Munir. Rinaldi, Algoritma dan Pemrograman Dalam Bahasa Pascal dan C, Bandung Informatika, 2007
3. Sjukani. Moh Algoritma (Algoritma dan Struktur Data1) dengan C, C++, dan Java, Jakarta, Mitra Kencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial, diakses tanggal 10 September 2013

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Dasar Pemrograman

FUNGSI

Fakultas	Program Studi	TatapMuka	Kode MK	DisusunOleh
IlmuKomputer	TeknikInformatika	14	MK87001	Tim Dosen

Abstract

Modul ini membahas tentang fungsi

Kompetensi

Diharapkan mahasiswa dapat:

- Memahami dan membuat program dengan menggunakan fungsi

Fungsi

Fungsi memberikan izin untuk mengelompokkan program dalam segmen segmen kode untuk melaksanakan tugas-tugas individu.

Sintaks.

```
type name ( parameter1, parameter2, ...)
{
        statements
}
```

Keterangan:

- Type

  Jenis dari nilai yang akan dikembalikan oleh fungsi.
- Name

  Pengenal dari fungsi, dimana fungsi dipanggil dengan nama nya.
- Parameter

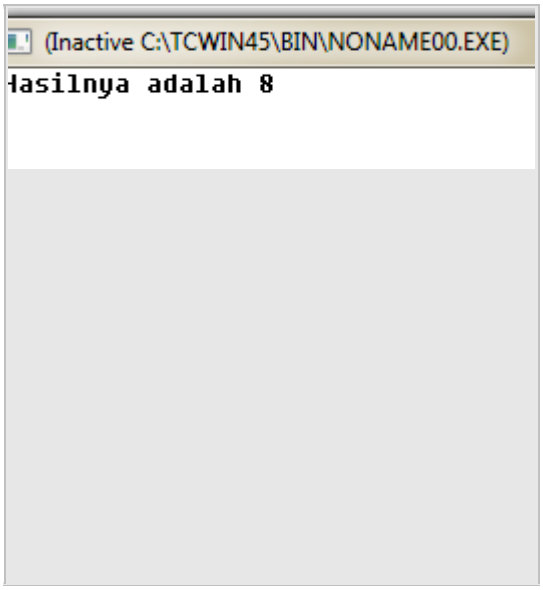
  Jumlah dari sebuah parameter tergantung kepada kebutuhan. Setiap parameter terdiri dari type dan diikuti dengan sebuah pengenal. Antara satu parameter dengan parameter lainnya dipisahkan dengan sebuah koma. Setiap parameter seperti deklarasi variabel biasa (contoh, int X).

  Tujuan parameter adalah melewatkan argumen-argumen ke fungsi dari lokasi dimana argumen tersebut berasal.
- Statement

  Statement merupakan tubuh dari fungsi. Meruakan sebuah blok statement yang diawali { dan diakhiri }.

Contoh 1

```
#include <iostream.h>

//Fungsi Jumlah
int jumlah (int a, int b)
{
  int r;
  r=a+b;
  return r;
}

//Fungsi Utama
void main ()
{
  int z;
  z = jumlah (5,3);                    //panggil fungsi jumlah
  cout << "Hasilnya adalah " << z;
}
```



Contoh 1 diatas merupakan sebuah program yang terdiri dari 2 fungsi yaitu fungsi jumlah dan fungsi utama.

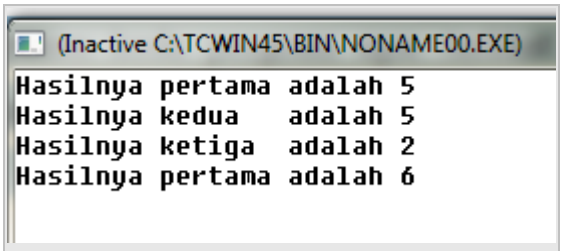
Posisi dari fungsi tidak akan dipermasalahkan, karen C++ akan memulai dengan memanggil fungsi utama.

Alur program

1. Fungsi utama dimulai dengan mendeklarasikan variabel z dengan tipe data int.
2. Pemanggilan fungsi jumlah.
    a. Kontrol program berpindah ke fungsi jumlah, dan fungsi utama dihentikan eksekusinya.
    b. Kedua nilai (5 dan 3) akan dicopykan ke variabel lokal a dan b dalam fungsi jumlah.
3. Dalam fungsi jumlah
    a. Sebuah variabel lokal dideklarasikan yaitu int r.
    b. Proses r=a+b;
    c. r bernilai 8 (5+3)
    d. return r menyatakan akhir dari fungsi jumlah, dan kontrol program pindah ke fungsi utama dengan mengembalikan nilai 8.
4. Sehingga z=8

Contoh 2

```
#include <iostream.h>

int kurang (int a, int b)
{
  int r;
  r=a-b;
  return r;
}

void main ()
{
  int z,x=
rang(75, y=3;
  z = kurang (7,2);
  cout << "Hasilnya pertama adalah " << z<<endl;
  cout << "Hasilnya kedua   adalah " << ku,2)<<endl;
  cout << "Hasilnya ketiga  adalah " << kurang(x,y)<<endl;
  z=4+kurang(x,y);
  cout << "Hasilnya pertama adalah " << z<<endl;

}
```



Fungsi tanpa tipe data, menggunakan void.

Sintaks.

```
type name ( parameter1, parameter2, ...)
{
        statements
}
```

Sebuah fungsi memerlukan deklarasi tipe data. Tipe data dari nilai yang dikembalikan oleh fungsi. Tetapi jika sebuah fungsi tidak memerlukan tipe data, maka digunakan adalah void.

```
// void function example
#include <iostream>
using namespace std;

void printmessage ()
{
  cout << "I'm a function!";
}

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

```
I'm a function!
```

void dapat juga digunakan dalam parameter fungsi dengan ditulis secara jelas untuk menyatakan bahwa fungsi tidak memiliki nilai kembali.
Contoh adalah fungsi printmessage :

```
void printmessage (void)
{
  cout << "I'm a function!";
}
```

The return value of main

Well, there is a catch: If the execution of main ends normally without encountering a return statement the compiler assumes the function ends with an implicit return statement:

```
return 0;
```

Note that this only applies to function main for historical reasons. All other functions with a return type shall end with a proper return statement that includes a return value, even if this is never used.

When main returns zero (either implicitly or explicitly), it is interpreted by the environment as that the program ended successfully. Other values may be returned by main, and some environments give access to that value to the caller in some way, although this behavior is not required nor necessarily portable between platforms. The values for main that are guaranteed to be interpreted in the same way on all platforms are:

value	description
0	The program was successful
EXIT_SUCCESS	The program was successful (same as above). This value is defined in header <cstdlib>.
EXIT_FAILURE	The program failed. This value is defined in header <cstdlib>.

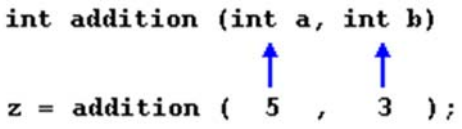
Because the implicit return 0; statement for main is a tricky exception, some authors consider it good practice to explicitly write the statement.

Arguments passed by value and by reference

In the functions seen earlier, arguments have always been passed *by value*. This means that, when calling a function, what is passed to the function are the values of these arguments on the moment of the call, which are copied into the variables represented by the function parameters. For example, take:

```
1 int x=5, y=3, z;
2 z = addition ( x, y );
```

In this case, function addition is passed 5 and 3, which are copies of the values of x and y, respectively. These values (5 and 3) are used to initialize the variables set as parameters in the function's definition, but any modification of these variables within the function has no effect on the values of the variables x and y outside it, because x and y were themselves not passed to the function on the call, but only copies of their values at that moment.



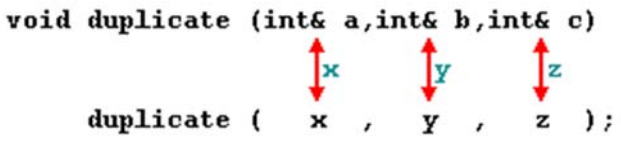
In certain cases, though, it may be useful to access an external variable from within a function. To do that, arguments can be passed *by reference*, instead of *by value*. For example, the function duplicate in this code duplicates the value of its three arguments, causing the variables used as arguments to actually be modified by the call:

```
// passing parameters by reference
#include <iostream>
using namespace std;

void duplicate (int& a, int& b, int& c)
{
  a*=2;
  b*=2;
  c*=2;
}

int main ()
{
  int x=1, y=3, z=7;
  duplicate (x, y, z);
  cout << "x=" << x << ", y=" << y << ", z=" << z;
  return 0;
}
```

```
x=2, y=6, z=14
```

To gain access to its arguments, the function declares its parameters as *references*. In C++, references are indicated with an ampersand (&) following the parameter type, as in the parameters taken by duplicate in the example above.

When a variable is passed *by reference*, what is passed is no longer a copy, but the variable itself, the variable identified by the function parameter, becomes somehow associated with the argument passed to the function, and any modification on their corresponding local variables within the function are reflected in the variables passed as arguments in the call.



In fact, a, b, and c become aliases of the arguments passed on the function call (x, y, and z) and any change on awithin the function is actually modifying variable x outside the function. Any change on b modifies y, and any change on c modifies z. That is why when, in the example, function duplicate modifies the values of variables a, b, and c, the values of x, y, and z are affected.

If instead of defining duplicate as:

```
void duplicate (int& a, int& b, int& c)
```

Was it to be defined without the ampersand signs as:

```
void duplicate (int a, int b, int c)
```

The variables would not be passed *by reference*, but *by value*, creating instead copies of their values. In this case, the output of the program would have been the values of x, y, and z without being modified (i.e., 1, 3, and 7).

Efficiency considerations and const references

Calling a function with parameters taken by value causes copies of the values to be made. This is a relatively inexpensive operation for fundamental types such as int, but if the parameter is of a large compound type, it may result on certain overhead. For example, consider the following function:

```
1 string concatenate (string a, string b)
2 {
3   return a+b;
4 }
```

This function takes two strings as parameters (by value), and returns the result of concatenating them. By passing the arguments by value, the function forces a and b to be copies of the arguments passed to the function when it is called. And if these are long strings, it may mean copying large quantities of data just for the function call.

But this copy can be avoided altogether if both parameters are made *references*:

```
1 string concatenate (string& a, string& b)
2 {
3   return a+b;
4 }
```

Arguments by reference do not require a copy. The function operates directly on (aliases of) the strings passed as arguments, and, at most, it might mean the transfer of certain pointers to the function. In this regard, the version of concatenate taking references is more efficient than the version taking values, since it does not need to copy expensive-to-copy strings.

On the flip side, functions with reference parameters are generally perceived as functions that modify the arguments passed, because that is why reference parameters are actually for.

The solution is for the function to guarantee that its reference parameters are not going to be modified by this function. This can be done by qualifying the parameters as constant:

```
1 string concatenate (const string& a, const string& b)
2 {
3   return a+b;
4 }
```

By qualifying them as const, the function is forbidden to modify the values of neither a nor b, but can actually access their values as references (aliases of the arguments), without having to make actual copies of the strings.

Therefore, const references provide functionality similar to passing arguments by value, but with an increased efficiency for parameters of large types. That is why they are extremely popular in C++ for arguments of compound types. Note though, that for most fundamental types, there is no noticeable difference in efficiency, and in some cases, const references may even be less efficient!

Inline functions

Calling a function generally causes a certain overhead (stacking arguments, jumps, etc...), and thus for very short functions, it may be more efficient to simply insert the code of the function where it is called, instead of performing the process of formally calling a function.

Preceding a function declaration with the inline specifier informs the compiler that inline expansion is preferred over the usual function call mechanism for a specific function. This does not change at all the behavior of a function, but is merely used to suggest to the compiler that the code generated by the function body be inserted at each point the function is called, instead of being invoked with a regular function call.

For example, the concatenate function above may be declared inline as:

```
inline string concatenate (const string& a, const string& b)
{
  return a+b;
}
```

This informs the compiler that when concatenate is called, the program prefers the function to be expanded inline, instead of performing a regular call. inline is only specified in the function declaration, not when it is called.

Note that most compilers already optimize code to generate inline functions when they see an opportunity to improve efficiency, even if not explicitly marked with the inline specifier. Therefore, this specifier merely indicates the compiler that inline is preferred for this function, although the compiler is free to not inline it, and optimize otherwise. In C++, optimization is a task delegated to the compiler, which is free to generate any code for as long as the resulting behavior is the one specified by the code.

Default values in parameters

In C++, functions can also have optional parameters, for which no arguments are required in the call, in such a way that, for example, a function with three parameters may be called with only two. For this, the function shall include a default value for its last parameter, which is used by the function when called with fewer arguments. For example:

```
// default values in functions
#include <iostream>
using namespace std;

int divide (int a, int b=2)
{
  int r;
  r=a/b;
  return (r);
}

int main ()
{
  cout << divide (12) << '\n';
  cout << divide (20,4) << '\n';
  return 0;
}
```

```
6
5
```

In this example, there are two calls to function divide. In the first one:

```
divide (12)
```

The call only passes one argument to the function, even though the function has two parameters. In this case, the function assumes the second parameter to be 2 (notice the function definition, which declares its second parameter as int b=2). Therefore, the result is 6.

In the second call:

```
divide (20,4)
```

The call passes two arguments to the function. Therefore, the default value for b (int b=2) is ignored, and b takes the value passed as argument, that is 4, yielding a result of 5.

Declaring functions

In C++, identifiers can only be used in expressions once they have been declared. For example, some variable xcannot be used before being declared with a statement, such as:

```
int x;
```

The same applies to functions. Functions cannot be called before they are declared. That is why, in all the previous examples of functions, the functions were always defined before the main function, which is the function from where the other functions were called. If main were defined before the other functions, this would break the rule that functions shall be declared before being used, and thus would not compile.

The prototype of a function can be declared without actually defining the function completely, giving just enough details to allow the types involved in a function call to be known. Naturally, the function shall be defined somewhere else, like later in the code. But at least, once declared like this, it can already be called.

The declaration shall include all types involved (the return type and the type of its arguments), using the same syntax as used in the definition of the function, but replacing the body of the function (the block of statements) with an ending semicolon.

The parameter list does not need to include the parameter names, but only their types. Parameter names can nevertheless be specified, but they are optional, and do not need to necessarily match those in the function definition. For example, a function called protofunction with two int parameters can be declared with either of these statements:

```
1 int protofunction (int first, int second);
2 int protofunction (int, int);
```

Anyway, including a name for each parameter always improves legibility of the declaration.

```
// declaring functions prototypes
#include <iostream>
using namespace std;

void odd (int x);
void even (int x);

int main()
{
  int i;
  do {
    cout << "Please, enter number (0 to exit): ";
    cin >> i;
    odd (i);
  } while (i!=0);
  return 0;
}

void odd (int x)
{
  if ((x%2)!=0) cout << "It is odd.\n";
  else even (x);
}

void even (int x)
{
  if ((x%2)==0) cout << "It is even.\n";
  else odd (x);
}
```

```
Please, enter number (0 to exit): 9
It is odd.
Please, enter number (0 to exit): 6
It is even.
Please, enter number (0 to exit): 1030
It is even.
Please, enter number (0 to exit): 0
It is even.
```

This example is indeed not an example of efficiency. You can probably write yourself a version of this program with half the lines of code. Anyway, this example illustrates how functions can be declared before its definition:

The following lines:

```
1 void odd (int a);
2 void even (int a);
```

Declare the prototype of the functions. They already contain all what is necessary to call them, their name, the types of their argument, and their return type (void in this case). With these prototype declarations in place, they can be called before they are entirely defined, allowing for example, to place the function from where they are called (main) before the actual definition of these functions.

But declaring functions before being defined is not only useful to reorganize the order of functions within the code. In some cases, such as in this particular case, at least one of the declarations is required, because odd and evenare mutually called; there is a call to even in odd and a call to odd in even. And, therefore, there is no way to structure the code so that odd is defined before even, and even before odd.

Recursivity

Recursivity is the property that functions have to be called by themselves. It is useful for some tasks, such as sorting elements, or calculating the factorial of numbers. For example, in order to obtain the factorial of a number (n!) the mathematical formula would be:

n! = n * (n-1) * (n-2) * (n-3) ... * 1
More concretely, 5! (factorial of 5) would be:
5! = 5 * 4 * 3 * 2 * 1 = 120

And a recursive function to calculate this in C++ could be:

```
// factorial calculator
#include <iostream>
using namespace std;

long factorial (long a)
{
  if (a > 1)
   return (a * factorial (a-1));
  else
   return 1;
}

int main ()
{
  long number = 9;
  cout << number << "! = " << factorial (number);
  return 0;
}
```

```
9! = 362880
```

Notice how in function factorial we included a call to itself, but only if the argument passed was greater than 1, since, otherwise, the function would perform an infinite recursive loop, in which once it arrived to 0, it would continue multiplying by all the negative numbers (probably provoking a stack overflow at some point during runtime).

DaftarPustaka

1. Kristianto. Andri, Algoritma dan Pemrograman dengan C++ Edisi 3, Yokyakarta, Graha Ilmu, 2013
2. Munir. Rinaldi, Algoritma dan Pemrograman Dalam Bahasa Pascal dan C, Bandung Informatika, 2007
3. Sjukani. Moh Algoritma (Algoritma dan Struktur Data1) dengan C, C++, dan Java, Jakarta, Mitra Kencana Media, 2010
4. http://www.cplusplus.com/doc/tutorial, diakses tanggal 10 September 2013