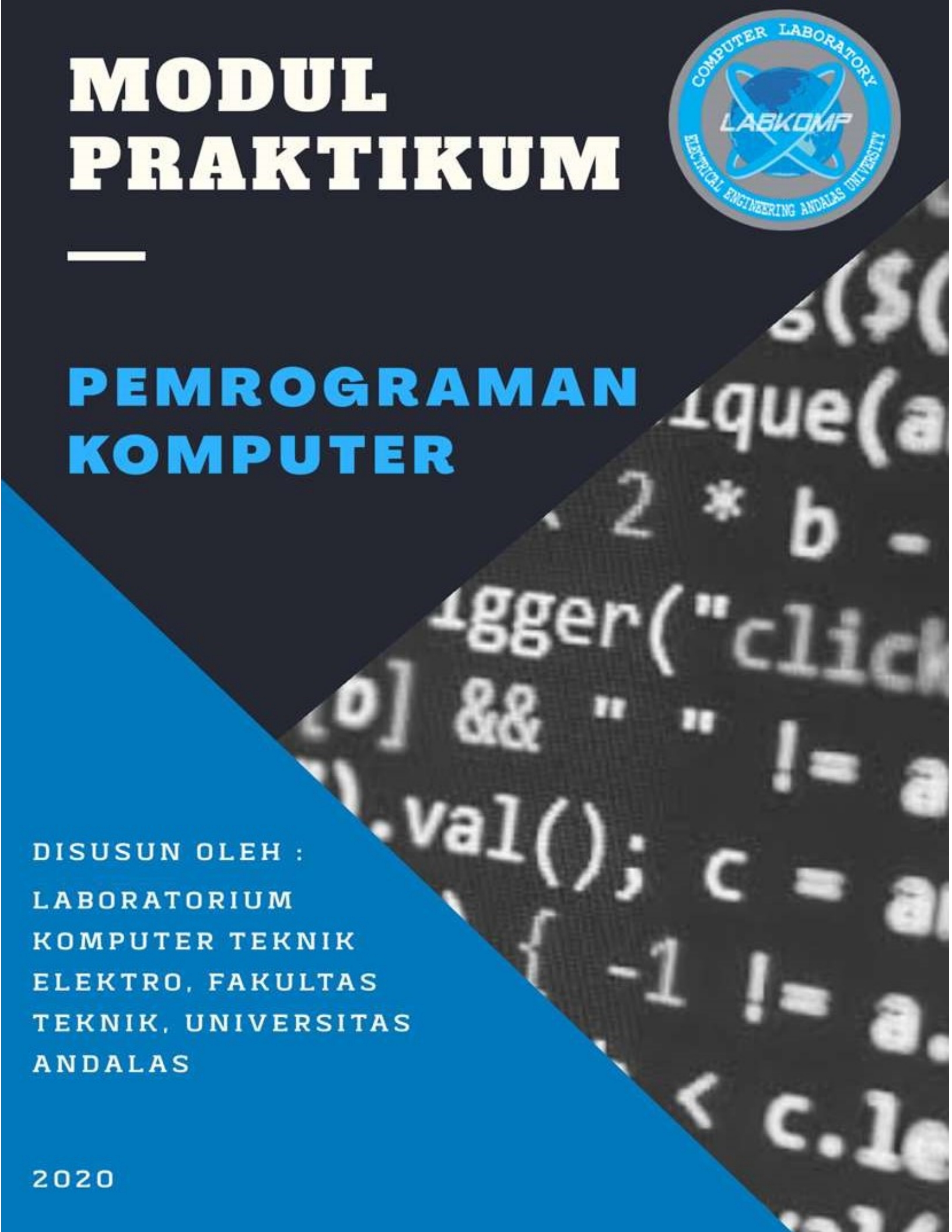
MODUL PRAKTIKUM

—

PEMROGRAMAN KOMPUTER



DISUSUN OLEH :

LABORATORIUM KOMPUTER TEKNIK ELEKTRO, FAKULTAS TEKNIK, UNIVERSITAS ANDALAS

2020

HALAMAN PENGESAHAN

BUKU PANDUAN PRAKTIKUM

Judul : Buku Panduan Praktikum Pemograman

Penulis

a. Nama Lengkap : Amirul Luthfi, S.T., M.T.
b. NIP : 199312052019031014
c. Jabatan Fungsional : Tenaga Pengajar
d. Program Studi : Teknik Elektro

Mata Kuliah Praktikum

a. Nama Mata Kuliah : Praktikum Pemograman
b. Kode Mata Kuliah : TE2117
c. Bobot SKS : 1 SKS
d. Semester ke- : 3 (Tiga)

Padang, 5 Oktober 2020

Mengetahui
Ketua Jurusan Teknik Elektro
Universitas Andalas

Penulis

Dr. Eng. Muhammad Ilhamdi Rusydi, S.T., M.T.
NIP. 198205222005011002

Amirul Luthfi, S.T., M.T.
NIP. 199312052019031014

KATA PENGANTAR

Assalamu'alaikum Wr, Wb.

Tidak ada kata-kata yang dapat kita ucapkan setelah menyelesaikan Modul Pemrograman Komputer ini selain berterima kasih kepada Allah atas Pengetahuannya yang Luas dan memberikan kesempatan bagi orang untuk mempelajari sedikit pengetahuannya.

Pemrograman adalah ilmu yang berkembang pesat di Era Dunia Digital. Saat ini, hampir semua peralatan industri baik peralatan rumah tangga maupun peralatan rumah tangga, yang bekerja dengan sistem kontrol komputer diprogram oleh komputer. Ini menjadikan ilmu pemrograman komputer menjadi peran yang sangat strategis dan signifikan. Karena itu, sebagai mahasiswa Teknik Elektro, tidak dapat dipungkiri mampu merancang dan merekayasa produk yang terkait dengan teknologi ini. Langkah pertama adalah mulai mempelajari algoritma dan pemrograman berorientasi objek.

Modul ini terdiri dari praktikum pekerjaan yang memberikan beberapa keterampilan dasar untuk Pemrograman C ++. Materi modul ini akan diperbaiki di masa depan. Semoga modul ini dapat digunakan untuk membantu orang belajar pemrograman, terutama C ++. Semoga Allah SWT memberkati untuk semua pekerjaan yang telah kami lakukan. Amiin.

Padang, Oktober 2020

Penulis

STRUKTUR ORGANISASI
LABORATORIUM KOMPUTER
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS ANDALAS

Kepala Laboratorium : Dr Eka Putra Waldi, S.T., M.Eng

Sekretaris Laboratorium : Amirul Luthfi, S.T., M.T.

TIM ASISTEN

1. Nurfadhilla 1610951027
2. Afdal 1610951029
3. Dio Okta Mandala 1610951034
4. Nelsa Mutia Sari 1610951039
5. Fallahati Alhamdi 1610952021
6. Afif Prasetya 1710951004
7. Salisa ‘Asyarina Ramadhani 1710951006
8. Nindi Anggraini 1710951007
9. Dewi Kusuma Wardani 1710951022
10. Hayatul Sukma 1710953026

PERATURAN DAN TEKNIS PRAKTIKUM

Praktikan harus melaksanakan dan menaati peraturan dan teknis praktikum yang berlaku, sebagai berikut:

a. Seluruh Praktikan wajib mengikuti Responsi umum yang dilaksanakan sebelum praktikum dimulai untuk membahas sistematika praktikum.
b. Praktikum dilaksankan secara daring/ online dengan durasi selama 90 menit, Langsung join google meet dan link untuk google meet dibuat oleh praktikan.
c. Praktikan harus mengikuti semua modul percobaan, apabila tidak mengikuti salah satunya, makan seluruh modul percobaan dianggap gagal.
d. Praktikan wajib hadir tepat waktu sesuai dengan jadwal yang telah ditentukan.
e. Praktikan membuat grup telegram per-kelompok.
f. Praktikan yang akan praktikum harus menghubungi asisten pengawas paling lambat 1 (satu) hari sebelum praktikum dan memasukkan asisten ke grup telegram kelompok.
g. Praktikan yang tidak memberikan tugas pendahuluan sebelum praktikum tidak dibenarkan untuk mengikuti praktikum.
h. Responsi awal dilakukan sebelum praktikum dimulai,.
i. Ketentuan keterlambatan saat praktikum
    1. Kurang dari 15 menit diperbolehkan ikut responsi tanpa penambahan waktu
    2. Lebih dari 15 menit, tidak bisa ikut praktikum, otomatis **gagal** pada modul tersebut.
j. Praktikan wajib mendapat nilai responsi diatas 60 agar bisa melaksanakan praktikum.
k. Apabila praktikan tidak menguasai materi percobaan pada saat responsi, maka asisten pengawas berhak mengeluarkan praktikan tersebut
l. Laporan akhir praktikum ditulis tangan dengan margin **4,3,3,3**

m. Jika sinyal praktikan bermasalah pada saat praktikum, maka praktikan dapat menyisip pada waktu yang telah ditentukan oleh asisten. Catatan : Jika sinyal bermasalah kirim bukti screenshoot setelah sinyal stabil kembali, kalau sinyal terganggu, praktikum dilanjutkan via chat.
n. Praktikan wajib membuat tugas tambahan yang diberikan asisten setelah selesai praktikum.
o. Tugas tambahan wajib diasistensikan dengan asistensi minimal sebanyak 3 (tiga) kali
p. Pengumpulan tugas tambahan dikumpulkan paling lambat seminggu setelah dilaksanakannya praktikum.

FORMAT LAPORAN AKHIR

1. Cover (print)
2. Judul (print)
3. Tujuan (print)
4. Dasar Teori (print)
5. Prosedur Percobaan (print)
6. Analisa dan Pembahasan
7. Kesimpulan dan Saran

DAFTAR ISI

CHAPTER I
Struktur Dasar Bahasa C/C++
dan
OOP (*Object Oriented Programming*)

I. Tujuan

- Praktikan bisa memahami struktur dasar bahasa C/C++
- Praktikan bisa membuat program sederhana bahasa C/C++
- Praktikan bisa mengetahui apa itu OOP

II. Teori

a. Tipe Data

Tipe		Signed		Unsigned	
		Short	Long	Short	Long
Float	Ukuran dalam bit	32			
	Jangkauan	+/- 3.4e +/- 38 (~7 digits)			
Double	Ukuran dalam bit	64			
	Jangkauan	+/- 1.7e +/- 308 (~15 digits)			
Int	Ukuran dalam bit	16	32	16	32
	Jangkauan	-32,768 s/d 32,767	-2,147,483,648 s/d 2,147,483,647	0 s/d 65,535	0 s/d 4,294,967,295
Char	Ukuran dalam bit	8		8	
	Jangkauan	-128 s/d 127		0 s/d 255	

b. Konstanta

Konstanta adalah nilai yang tidak dapat diubah selama program berjalan.Nilai konstanta selalu tetap.Konstanta harus didefinisikan terlebih dahulu diawal program. Konstanta bisa berupa *integer*, *fraction*, *character* dan *string*.Contohnya:

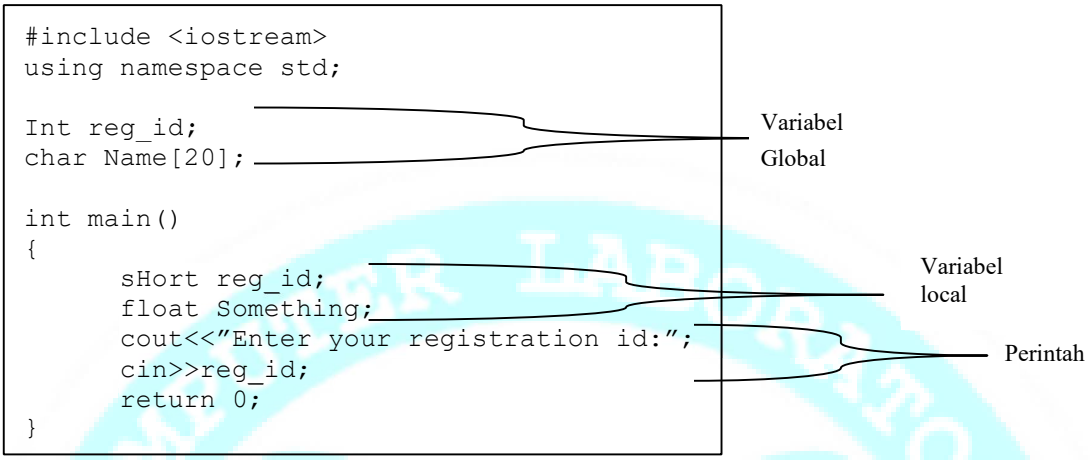
```
constant int Age = 15;

const char Grade = 'A';
#define PHI 3.14159
```

c. Variabel

1. Terdiri dari kombinasi huruf dan angka yang karakter awal nya harus berupa huruf.Bahasa C adalah suatu bahasa *case-sensitive*, dimana huruf besar dan huruf kecil merupakan sesuatu yang berbeda.Contohnya nim, NIM dan Nim.
2. Tidak boleh menggunakan spasi.
3. Tidak boleh menggunakan simbol kecuali *underscore* atau garis bawah (_). Simbol yang tidak boleh digunakan diantaranya : $, ?, %, #, !, &, *, (,), -, +, = dan lain-lain.
4. Panjang karakter nya tidak terbatas, namun hanya 32 karakter pertama yang digunakan.
5. Contoh penulisan variabel yang benar :
   : NIM,a, x, Student_name, f3089, f4, Value, Sandi dan lain-lain.
6. Contoh penulisan variabel yang salah :
   : %Student_point, 80student, student name, urgent! dan lain-lain.

Ruang lingkup variabel.

Semua variabel yang digunakan dalam suatu program harus memiliki tipe data yang dideklarasikan sebelum nama variabel. Ruang lingkup suatu variabel dapat berupa lingkup global atau lokal.



d. Deklarasi

Bentuk umum dari pendeklarasian variabel yaitu :
tipe_data nama_variable;
Contoh :

```
int x;
char y, huruf, nim[10];  float nilai;
double beta; char array[5][4];  int *p;
```

e. Operator

Pada bahasa C++ terdapat operator yang digunakan untuk mengoperasikan variabel dan konstanta dalam operasi matematika.Diantaranya :

- Assignment (=) : memberi nilai ke suatu variabel
- Arithmetic operator (+, -, *, /, %) : penjumlahan, pengurangan, perkalian, pembagian dan modulus.
- Compound assignment (+=, -=. *=, /=, %=, >>=, <<=, &=, ^=, |=)
- Increase and decrease (++, --)
- Relational and equality operators (==, !=, >, <, >=, <=)
- Logical operators (!,&&, ||)
- Conditional operator (?)
- Comma operator (,)
- Bitwise operators (&, |, ^, ~, <<, >>)

f. Standar Output

Secara default, output dari program C ++ ditampilkan di jendela konsol.

Untuk mengakses standar output, digunakan “cout” dan diikuti oleh operator “<<” (Stream insertion).
Contohnya :

```
std::cout<<”this is display on the screen” <<std::endl;
std::cout<<”and this is on the second line”;
```

atau

```
cout<<”this is display on the screen”<<endl;
cout<<”and this is on the second line”;
```

Jika telah menggunakan “using namespace std;” **setelah header.**

g. Standar Input

Perangkat standar input adalah keyboard. Untuk mengkakses standard input, digunakan “cin” dan ikuti oleh operator ">>" (Stream extraction). Pertama, deklarasikan variabel sebelum menggunakan perintah input. Sehingga ketika nilai dimasukkan, nilai tersebut bisa disimpan dalam suatu variabel. Contohnya :

```
int age;
std::cin>>age;          atau
int age;
cin>>age;
```

jika telah menggunakan “using namespace std;” setelah header.

h. OOP (Object Oriented Programming)

OOP (*Object Oriented Programming*) adalah suatu metode pemrograman yang berorientasi kepada objek. Tujuan dari OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari.

Pada OOP, Fungsi dan variabel **dibungkus** dalam sebuah **objek** atau *class* yang dapat saling brinteraksi, sehingga membentuk sebuah program.

8. Kode ASCII

ASCII (American Standard Code for Information Interchange) merupakan Kode Standar Amerika untuk Pertukaran Informasi atau sebuah standar internasional dalam pengkodean huruf dan simbol seperti Unicode dan Hex tetapi ASCII lebih bersifat universal.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

III. Prosedur Percobaan

3.1 Percobaan Penggunaan Tipe Data dan Operators

1. Ketik listing program berikut pada aplikasi C++ . .

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main()
5  {
6  //Experimental Data Types
7  string Univ;
8  int a, b;
9  float c;
10 double d;
11 char e;
12
13 cout<<"Enter The value of a : ";
14 cin>> a;
15 cout<<"Enter The value of b : ";
16 cin>> b;
17 cout<<"Enter The value of c : ";
18 cin>> c;
19 cout<<"Enter The value of d : ";
20 cin>> d;
21 cout<<"Enter The value of e : ";
22 cin>> e;
23
24 cout<<"Enter your University : ";
25 cin>>Univ;
26 cout<<endl<<endl;
27
28 cout<<" Your University is : "<<Univ<<endl;
29 cout<<" The value of a is "<<a<<endl;
30 cout<<" The value of b is "<<b<<endl;
31 cout<<" The value of c is "<<c<<endl;
32 cout<<" The value of d is "<<d<<endl;
33 cout<<" The value of e is "<<e<<endl<<endl;
```

```
34
35  //arithmetic Operator
36  cout<<" The value of a+b =" <<a+b<<endl;
37  cout<<" The value of a%b =" <<a%b<<endl<<endl;
38
39  b = 0;
40  a = 1;
41  b += a;
42  cout<<" a = 1"<<endl;
43  cout<<" b += a"<<endl;
44  cout<<" The value of b is = "<<b<<endl;
45  b=b+a;
46  cout<<" b=b+a" <<endl;
47  cout<<" The value of b is = "<<b<<endl;
48  b-=a;
49  cout<<" b-=a" <<endl;
50  cout<<" The value of b is = "<<b<<endl;
51  b=b-a;
52  cout<<" b=b-a" <<endl;
53  cout<<" The value of b is = "<<b<<endl;
54  cout<<endl<<endl;
55  cout<<" The value of a is = " <<a<<endl;
56  b=++a;
57  cout<<" b = ++a"<<endl;
58  cout<<" The value of a is = "<<a<<endl;
59  cout<<" The value of b is = "<<b<<endl;
60  b=a++;
61  cout<<" b = a++"<<endl;
62  cout<<" The value of a is = "<<a<<endl;
63  cout<<" The value of b is = "<<b<<endl;
64  b=--a;
65  cout<<" b = --a"<<endl;
66  cout<<" The value of a is = "<<a<<endl;
67  cout<<" The value of b is = "<<b<<endl;
68  b= a--;
69  cout<<" b = a--"<<endl;
```

```
70  cout<<" The value of a is = "<<a<<endl;
71  cout<<" The value of b is = "<<b<<endl;
72
73  system("PAUSE");
74  return 0;
    }
```

Jalankan listing program dan analisa!

3.2 Percobaan Penggunaan Konstanta.

1. Ketik listing program berikut pada aplikasi C++.

```
1   #include <iostream>
2   using namespace std;
3
4   const double phi = 3.14;
5
6   int main()
7   {
8   int r;
9
10    cout<<"Circle Rectangle"<<endl;
11   cout<<"Enter the radius of the circle : ";
12       cin>>r;
13   cout<<"The Circle Rectangle is : "<<phi*r*r<<endl;
14       cout<<endl;
15       system("PAUSE");
16       return 0;
17  }
```

Jalankan listing program dan analisa!

3.3 Percobaan Pengunaan Operator Logika.

Ketik listing program berikut pada aplikasi C++.

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int z;
7       bool x, y;
```

```
8       cout << "x  : ";
9       cin >> x;
10      cout << "y  : ";
11      cin >> y;
12      z = (x || y) && !(x && y);
13      cout<<x<<" XOR "<<y<< " = "<< z << endl;
14
15      system("PAUSE");
16      return 0;
        }
```

Jalankan listing program dan analisa!

IV. Tugas Pendahuluan

1. Jelaskan tentang tipe data yang Anda ketahui dan perbedaannya masing-masing!
2. Jelaskan dan tunjukkan perbedaan konstanta dan variabel dalam program C ++!
3. Jelaskan tentang arti modulus dalam program C ++!
4. "a = ++ b" dan "a = b ++", apa perbedaan antara kedua deklarasi tersebut!
5. Jelaskan perbedaan char dan string dalam suatu program!
6. Jelaskan library dan header program C ++ yang Anda tahu! (min 3)
7. Jelaskan cara menggunakan ekspresi logis dari AND, OR, NOT, buat setiap sintaks dan buat tabel di dalamnya dalam Program C ++!
8. Jelaskan apa "bool" dalam prosedur 3? Mengapa tidak menggunakan tipe data lain saja?
9. Jelaskan apa itu OOP !
10. Jelaskan apa itu Alogrithm dan flowchart? Apa bedanya? Dan jelaskan penggunaan setiap simbol diagram alur yang Anda ketahui.
11. Buat program sederhana yang dapat mendeklarasikan tipe data (min. 3 tipe data)! (jangan mengambil dari modul!)

CHAPTER II
PERCABANGAN
(*CONDITIONAL DAN SELECTIVE STRUCTURE*)

1. Tujuan

- Memahami konsep dari conditional dan selective structure pada pemograman C++
- Dapat mengaplikasikan conditional dan selective structure pada program.
- Dapat menggunakan *if* /*else* dan *switch*/*case* pada penyelesaian masalah.

2. Teori

➢ Conditional structure

Struktur ini digunakan untuk memilih sebuah tindakan dari beberapa kondisi yang diberikan. Bentuk umum dari conditional structure adalah *if statement*. Formatnya sebagai berikut :

if (*condition***)**
statement1;
else
statement2;

Jika *condition* **True** atau benar, maka statement1 akan tereksekusi. Selain itu, maka statement2 langsung tereksekusi. *Condition* merupakan sebuah ekspresi atau deklarasi dengan inisialisasi yang mengatur aliran program. Contohnya sebagai berikut :

```
if (angka >= 80)
cout<< "angka besar sama dari 80";
else
cout<< "angka kecil dari 80";
```

Ketika *condition* angka bernilai besar sama dari 80, maka “angka besar sama dari 80” akan tertampilkan. Untuk angka yang bernilai lebih kecil dari 80, maka “angka kecil dari 80” yang akan ditampilkan.

Jika program memiliki lebih dari 1 pilihan, maka opsi tersebut dijalankan ketika condition sebelumnya **False**. Formatnya adalah :

```
if (condition){
   statement1;}
else if (condition2){
   statement2;}
else {
   statement3;}
```

Beberapa kasus membutuhkan lebih dari satu *conditional*, maka dapat digunakan *nested if* untuk mempermudah. Formatnya sebagai berikut :

```
if (condition A)
{
        if (condition1)
          statement1;
        else
         statement2;
}
else
statement A;
```

- Operator (?)

Operator (?) memiliki fungsi yang sama dengan *if/else condition*, dalam bentuk lebih singkat. Berikut formatnya :

*condition***?***result1***:***result2*

Jika *condition* bernilai **True**, maka ekspresi akan mereturn result1, jika tidak maka result2 yang akan direturn.

Contoh :

```
int angka;
cout<<"Masukkan angka = "<<endl;
cin>>angka;
(angka>=80)? cout<<"Angka besar sama dari 80":cout<<"Angka kecil dari 80";
```

Ketika *condition* angka bernilai besar sama dari 80, maka "angka besar sama dari 80" akan tertampilkan. Untuk angka yang bernilai lebih kecil dari 80, maka "angka kecil dari 80" yang akan ditampilkan.

- Selective Structure

Bentuk dari *selective structure* adalah *switch statement*. Syntaxnya sedikit berbeda dengan *if/else* statement, dengan tujuan untuk memeriksa beberapa kemungkinan nilai konstan untuk sebuah ekspresi. Formatnya sebagai berikut :

switch (*expression***)**
{ case *constant1***:** *//case harus bernilai bilangan bulat (tidak boleh decimal) dan huruf*
*statements1***;**
break;
case *constant2***:**
*statements2***;**
break;
default:
default *statements***;}**

Program akan memeriksa nilai dari ekspresi, dan jika nilainya sama dengan *constant1* maka *statements1* akan dieksekusi, atau jika nilainya ternyata sama dengan *constant2*, maka hanya *statements2* yang akan tereksekusi. Dan apabila nilai tidak ada pada **case**, maka program akan langsung menuju ke pernyataan default. Setiap **case** harus diikuti dengan *break*, tujuannya untuk mengidentifikasi satu **case** telah selesai.

Berikut merupakan salah satu contoh penggunaan *selective structure* :

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

void main()
{
int angka;
cout<< "Choose one number from 1 to 3 :" << endl;
cin>> angka;
switch(angka)
{
      case 1:
      cout<< "your number is ONE";
      break;
```

```
case 2:
cout<< "your number is TWO";
break;
case 3:
cout<< "your number is THREE";
break;
default:
cout<< "number is Out Of Range!";
}
system ("PAUSE");
}
```

3. Prosedur Percobaan

3.1 Conditional : if else

1. Ketik listing program berikut pada aplikasi C++.

code

```
1  // Program if Condition
2  #include<iostream>
3  using namespace std;
4
5  int main()
6  {
7  int tahun;
8
9     cout<<"Masukkan Tahun Kelahiran Anda = ";
10    cin >>tahun;
11
12    tahun=2020-tahun;
13
14 if(tahun>=19)
15 {
16    cout<< "Umur anda besar sama dari 19 tahun";
17 }
18
19 else
20 {  cout<<"Umur anda kurang dari 19 tahun"<<endl; }
21
```

3.2 Conditional : else if

1. Ketik listing program berikut pada aplikasi C++.

code

```
1  // Program else if Condition
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7  char nilai;
8
9     cout<<"Masukkan Nilai Anda (A - E) = ";
10    cin >>nilai;
11
```

```
12      if((int)nilai==65){
13          cout<<"Sangat Memuaskan";
14      }
15      else if ((int)nilai>65 && (int)nilai<=67){
16          cout<<"Cukup";
17      }
18      else if ((int)nilai>67 && (int)nilai<=69){
19          cout<<"Kurang Memuaskan";
20      }
21      else {
22          cout<<"Nilai Yang Anda Masukan Salah";
23      }
24
25  }
```

3.3 Conditional : nested if

1. Ketik listing program berikut pada aplikasi C++.

code

```
1   z`
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7   int umur;
8
9       cout<<"Masukkan Umur Anda = ";
10      cin >>umur;
11
12      if(umur<20){
13          if(umur<11 && umur>1){
14              cout<<"Anak-anak";
15          }
16          else if (umur>=11){
17              cout<<"remaja";
18          }
19          else{
20              cout<<"Bayi";
21          }
22      }
23
24      if(umur>=20){
```

```
25          if(umur >= 60){
26              cout<<"Lanjut Usia";
27          }
28          else{
29              cout<<"Dewasa";
30          }
31      }
```

3.4 Selective: switch case

1. Ketik listing program berikut pada aplikasi C++.

code

```
1   // Program switch case
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7   int nilai;
8   char predikat = 'Z';
9
10  cout << "Masukkan nilai anda : " << endl;
11  cin >> nilai;
12
13  nilai = (nilai > 100) ? -10 : nilai;
14
15  switch (nilai/10)
16  {
17  case 10: cout << "Sempurna";
18  predikat = 'A';
19  break;
20  case 9: cout << "Sangat Baik";
21  predikat = 'A';
22  break;
23  case 8: cout << "Baik";
24  predikat = 'B';
25  break;
26  case 7: cout << "Cukup";
27  predikat = 'C';
28  break;
29  case 6: cout << "Lebih Giat Lagi";
30  predikat = 'D';
```

```
31  break;
32  case 5:
33  case 4:
34  case 3:
35  case 2:
36  case 1:
37  case 0:
38  cout << " Maaf Anda Belum Lulus";
39  predikat = 'F';
40  break;
41  default: cout << " Nilai Yang Anda Masukkan Salah" << endl;
42  e   }
43  cout << "Nilai Anda adalah :  " << predikat <<  endl;
44
45  return 0;
46  }
```

IV. Tugas Pendahuluan :

1. Apa perbedaan antara ***if/else***,***else if***, ***nested if***,***switch case***, dan operator (**?**) ?
2. Apa fungsi dari break pada program **switch/case**?
3. Buatlah program sederhana untuk mengurutkan huruf dari A hingga Z (minimal 3 karakter), dan jelaskan programnya!
4. Buatlah program sederhana dengan menggunakan operator (**?**) dan ubah program tersebut ke bentuk ***if else***, dan jelaskan programnya! (**jangan diambil dari modul**)
5. Buatlah sebuah program menggunakan ***if/else***,***else if***, ***nested if***,***switch case***, dan operator (**?**), dan jelaskan program tersebut! (**jangan diambil dari modul**)

CHAPTER III
PERULANGAN (*LOOPS*)

I. Tujuan

- Memahami tujuan dari perulangan (*loops*) dalam pemrograman.
- Mampu menerapkan perulangan (*loops*) ***for***, ***while***, dan ***do-while*** dalam memecahkan masalah.
- Mampu menerapkan pernyataan lompatan (***continue***, ***break***, ***goto***) dalam program.

II. Teori

Perulangan (*loops*) memiliki tujuan untuk mengulangi suatu pernyataan beberapa kali atau ketika suatu kondisi telah terpenuhi. Dalam pemrograman C++, ada beberapa struktur yang dapat digunakan sebagai perulangan (*loops*), seperti *for*, *while*, dan *do-while*.

Perulangan (*loops*) **for**

Bentuk umum dari for terdiri atas inisialisasi, kondisi, dan *increase/decrease.*

for(*inisialisasi* **;** *kondisi* **;** *increase/decrease***)***pernyataan* **;**

Pertambahan/Increment adalah bagian yang digunakan untuk memproses variabel agar bisa memenuhi kondisi akhir perulangan. Umumnya nilai variable tersebut bertambah (i++) / berkurang (i--) 1 (satu).

Kondisi/Condition adalah kondisi yang harus dipenuhi agar perulangan dijalankan. Selama kondisi ini terpenuhi, maka C++ akan terus melakukan perulangan.

Pertama, inisialisasi dieksekusi oleh komputer, inisialilasi disini dapat digunakan untuk menginisialisasi variabel yang digunakan dalam proses perulangan (*loops*). Kemudian kondisinya dievaluasi oleh komputer. Jika kondisi itu bernilai benar, pernyataan dieksekusi, lalu kenaikan dievaluasi, dan program berjalan kembali ke awal perulangan (*loops*) for. Setelah itu program berjalan kembali namun proses inisialisasi tidak lagi dieksekusi karena proses inisialisasi hanya dieksekusi satu kali pada saat program perulangan (*loops*) dimulai. Iterasi ini berlanjut sampai kondisi bernilai salah, di mana program akan beralih ke pernyataan berikutnya.

Contohnya, untuk mencetak angka 1 hingga b. Nilai b ditentukan sebelumnya

```
int a,b;
cout <<"Enter the biggest number =";
cin >> b;

for (a=1;a<=b;a++)
{
      cout << a << endl;
}
```

Perulangan (*loops*) **while**

Bentuk umum :

while(kondisi) pernyataan;

Pada perulangan (*loops*) *while*, fungsinya hanya untuk mengulangi pernyataan jika kondisi yang telah ditetapkan bernilai benar atau terpenuhi. Sebagai contoh, untuk membuat program untuk hitung mundur dengan menggunakan perulangan *while* :

```
int n;
cout << "Enter the starting number =
";
cin >> n;

  while (n>0)
{
cout << n << ", ";
--n;
}
```

Perulangan (*loops*) ***do-while***

Do-while adalah bentuk lain dari perulangan *while*. *While* melakukan pengecekkan kondisi di awal perulangan , tetapi do-while melakukan pengecekkan kondisi di akhir. Jadi do-while selalu menjalankan program setidaknya sekali.

Bentuk umum :

```
do
pernyataan
while (kondisi );
```

Pertama, pernyataan akan dieksekusi sekali, dan kemudian dilakukan pengecekkan kondisi. Pernyataan akan terus dieksekusi sampai kondisi bernilai salah. Contohnya pada program di bawah ini, program meminta untuk memasukkan nomor berkali-kali dan berhenti ketika angka nol dimasukkan.

```
unsigned long n;
  do {
    cout << "Enter number (0 to STOP): ";
    cin >> n;
    cout << "You entered: " << n << "\n";
  } while (n != 0);
system ("PAUSE");
  return 0;
```

Penyataan **Continue**

Pernyataan **Continue** membuat program melewati suatu perulangan (*loops*) dan melompat ke iterasi berikutnya sampai perulangan (*loops*) selesai.

Penyataan **Break**

Kebalikan dari continue adalah pernyataan break. Break akan menghentikan iterasi saat ini meskipun perulangan (*loops*) belum selesai.

Penyataan **Goto**

Goto membuat lompatan mutlak ke titik lain dalam suatu program. Titik tujuan diidentifikasi oleh label, yang kemudian digunakan sebagai argumen untuk pernyataan goto. Label dibuat dari pengidentifikasi yang valid diikuti oleh titik dua (:).

III. Prosedur Percobaan

3.1 Perulangan *For, While,* and *Do-While*

- **Perulangan *For***

1. Ketik listing program ini pada aplikasi C ++

```
1  // Program factorial
2        #include <iostream>
3        using namespace std;
4
5        int main()
6        {
7           int n, i, factorial;
8                cout << "Enter the number :";
9                cin >>n;
10               cout<<"n! = "<<n<<"! = ";
11       factorial=1;
12               for (i=n;i>=1;i--)
13               {
14                  factorial*=i;
15                  if (i!=1)
16                       {cout<<i<<"x";}
17                  Else
18                       {cout<<i<<" = ";}
19                }
20                cout << factorial << endl<<endl;
21
22            system ("PAUSE");
23            return 0;
24       }
```

2. Jalankan listing program and analisa.

3.2. Pernyaaan Continue, Break, and Goto pada Array 1 Dimensi

- **Pernyaaan Continue**

1. Ketik listing program berikut ini pada aplikasi C ++

code

```
1  // Program Incrament Multiple
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7  int n,i,j, num[100];
```

```
8        cout<<"Enter number for looping program = ";
9        cin>>n;
10       for (i=1;i<=n;i++)
11       {
12  cout<<"Enter the number "<<i<<endl;
13  cout << "Increment Multiple = ";
14  cin >> num[i];
15  cout << endl;
16
17  for (j=1;j<=100;j++)
18  {
19       if (j%num[i]!=0)continue;
20       cout<< j <<"\t";
21  }
22            cout<<endl<<endl;
23        }
24  system ("PAUSE");
25  return 0;
26  }
```

2. Jalankan listing program dan analisis!

IV. Tugas Pendahuluan

1. Apa perbedaan dari perulangan While dan do-while?
2. Buat program sederhana menggunakan perulangan for atau while atau do-while! (jangan ambil dari modul!)
3. Apa gunanya pernyataan break, continue, dan goto dalam **program 3.2**?
4. Apa perbedaan antara kedua program ini? Jelaskan!

int i=1;
while(i>1){
cout<<i;
i++;
}
	int i=1;
	do{
	cout<<i;
	i++;
	}while(i>1);

CHAPTER IV
FUNGSI DAN ARRAY

I. Tujuan

- Memahami apa itu fungsi.
- Mengetahui cara meneruskan setiap parameter aktual dalam suatu fungsi.
- Mengetahui perbedaan fungsi, prosedur, dan fungsi rekursif.
- Mengetahui cara menerapkan konsep fungsi dalam program komputer.
- Memahami struktur array
- Dapat mengakses, membaca dan memodifikasi nilai array
- Mampu menggunakan array multidimensi dalam memecahkan suatu masalah

II. Teori

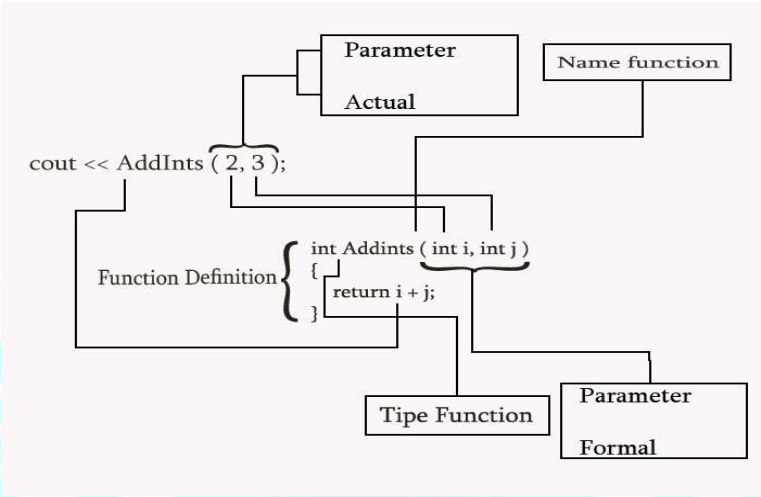
A. Fungsi

1. Apa itu fungsi?

Fungsi berisi serangkaian kode blok, yang berisi pernyataan, yang dikemas pada nama dan memiliki tujuan tertentu. Nama ini selain dapat dipanggil beberapa kali di beberapa tempat dalam program. Suatu fungsi menerima beberapa parameter aktual dan kemudian mengembalikannya berdasarkan pernyataan dalam fungsi itu.

2. Tujuan penggunaan fungsi

1. Mudah mengembangkan program. Program dibagi menjadi beberapa sub-program, jadi ini bisa lebih mudah untuk membuat program terstruktur.
2. Menghemat ukuran program, karena setiap perintah yang sama yang akan dijalankan untuk beberapa waktu pada program dapat dibuat dengan suatu fungsi, dan kemudian fungsi tersebut dapat dipanggil berulang kali.



3. Struktur Fungsi

Setiap fungsi terdiri dari tiga bagian utama:

```
tipe_pengembalian functionName(parameter_formal)
{
    Function body;
}
```

1. Tipe_pengembalian = tipe function /tipe yang dikembalikan oleh fungsi
2. FunctionName = nama function
3. Parameter_formal = daftar parameter/variable yang ingin dilewati
4. Function body = terdiri dari blok pernyataan antara kurung kurawal {}

- Prototype Function

  Pendeklarasian fungsi pada umumnya adalah membuat fungsi baru diatas fungsi utama atau main(), namun, ada cara lain yang bisa digunakan untuk mendeklarasikan fungsi yang ingin dideklarasikan dibawah fungsi main(), yaitu dengan prototype function.

  Format penulisan prototype function adalah :
  tipe_return namafungsi (argumen1, argumen2);

4. Cara penggunaan function

1. Pertama mendefinisikan prototipe function. (opsional)
2. Menetapkan **function structure**.
3. **Memanggil function** , tanpa menambahkan tipe function.

Contoh program :

```
1.  #include <iostream>
2.  using namespace std;
3.  int labkomp(int i, int j);                    --> Prototype Function
4.  int main()
5.  {
6.      int n = 2;
7.      int k = 3;
8.      cout<<labkomp(n,k);                      --> Summon Function
9.      system("PAUSE");
10. }
11. int labkomp(int i, int j)
12. {                                            --> Function Structure
13.     return i+j;
14. }
```

Penggunaan variabel dalam fungsi memiliki keterbatasan akses, karena variabel yang terdeklarasi dalam tubuh fungsi hanya bisa diakses oleh fungsi tersebut, atau disebut dengan variabel local.

5. Penggunaan void pada function

Function yang menggunakan void, tidak memiliki parameter. Hanya digunakan saat ingin menampilkan pesan dilayar. Tidak memerlukan nilai kembali seperti fungsi pada umumnya, biasanya dikenal dengan nama prosedur.

Contoh program :

```
#include <iostream>
using namespace std;
void labkomp()
{
```

```
    cout<<"Hallo saya sebuah fungsi";
    system("PAUSE");
}
int main ()
{
    labkomp();
}
```

6. Cara Melewatkan Argumen Fungsi

Ada dua macam parameter aktual ,diantaranya:

1. Memanggil dengan nilai (passed by value)
2. Memanggil dengan referensi (passed by reference)

Sejauh pembahasan fungsi yang telah dipelajari, parameter aktual yang diteruskan pada suatu fungsi diteruskan oleh nilai, yang berarti, ketika kita memanggil suatu fungsi, nilai dari parameter aktual disalin ke parameter formal, sedangkan yang dilewatkan oleh referensi adalah upaya untuk mengirimkan alamat dari variabel ke suatu fungsi, dalam hal ini x, y, z

Contoh program : **Parameter aktual passed by value**

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

int penjumlahan (int a, int b)

z = penjumlahan (5 , 3);

Contoh program : **Parameter aktual passed by reference**

```
#include <iostream>
using namespace std;
void duplicate(int& a,int& b,int& c)
{
     a=a*2;
     b=b*2;
     c=c*2;
}
int main ()
```

```
{
    int x=1,y=3,z=7;
    duplicate (x,y,z);
    cout<<"x="<<x<<"y="<<y<<"z="<<z<<endl;
    system("PAUSE");
}
```

Menjadi : x=2,y=6,z=14

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

duplicate(x , y , z)

Simbol Ampersand (&) dalam program menunjukkan parameter yang akan dikirimkan bukan hanya salinan dari nilai alamat variabel, dengan kata lain perubahan yang terjadi pada variabel a, b, dan c pada fungsi duplikat mempengaruhi nilai yang akan diberikan. x, y, dan z secara berurutan di luar.
Metode ini digunakan untuk mengubah isi suatu variabel di luar fungsi dengan implementasi konversi dilakukan di dalam fungsi.

7. Nilai Default dalam Argument

Pada saat fungsi dideklarasikan, parameter yang ada dapat diberikan nilai default, dengan tujuan saat pemanggilan fungsi tanpa ada pemberian nilai parameter nilai default dapat digunakan. Saat pemanggilan fungsi disertai dengan pemberian nilai pada parameter, maka nilai default akan digantikan dengan nilai yang diberikan.

contoh :

```
#include<iostream.h>
#include<conio.h>
void bagi (int a, int b=2)
{
int r;
r=a/b;
return (r);
}
int main()
{
cout<<bagi(12);
```

```
cout<<endl;
cout<<bagi(20,4);
return 0;
}
```

Maka hasil dari program itu adalah 6 dan 5.

8. Fungsi Rekursif

Fungsi rekursif adalah suatu fungsi yang memanggil dirinya sendiri. Artinya, fungsi tersebut dipanggil didalam tubuh fungsi itu sendiri. Fungsi rekursif banyak digunakan pada pengurutan data, atau menghitung nilai faktorial dari suatu bilangan, contoh :

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

Jika dimasukkan bilangan 9, maka yang ditampilkan pada console adalah 9! = 362880

B. Array

1. Konsep Dasar Array

Array adalah kumpulan dari nilai-nilai data bertipe sama dalam urutan tertentu yang menggunakan sebuah nama yang sama. Nilai-nilai data di suatu array disebut dengan elemen-elemen array. Letak urutan dari elemen-elemen array ditunjukkan oleh suatu *subscript* atau indeks. Pada saat pendeklarasian array, kompiler mengalokasikan memori yang cukup untuk menampung semua elemen sesuai dengan yang dideklarasikan.

Sama seperti variabel lain, sebuah array harus dideklarasikan tipenya sebelum dieksekusi. Sebagai contoh :

int bilangan [5] = {1,3,6,12,24};

Dalam memori dapat direpresentasikan sebagai :

bilangan	1	3	6	12	24
index	**0**	**1**	**2**	**3**	**4**

Nilai → bilangan [0] = 1;
bilangan [1] = 3;
bilangan [4] = 24; dan lain-lain.

Cara memasukkan nilai pada array terbagi atas dua, yaitu:

- Secara langsung
  Contoh : int angka[3] = {1, 2, 3};
- Secara tidak langsung
  Contoh :
  int angka[3];
  angka[0] = 1;
  angka[1] = 2;
  angka[2] = 3;

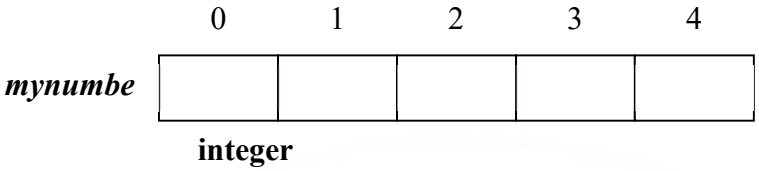
2. Jenis-Jenis Array

Adapun array dibedakan menjadi dua macam, yaitu array berdimensi satu, dan array berdimensi banyak.

- Array 1 Dimensi

Array memiliki beberapa nilai dari tipe data yang sama. Nilainya ditempatkan di lokasi memori yang berdekatan yang dapat dirujuk secara individual dengan menambahkan sebuah indeks.

Sebagai contoh, saya memiliki beberapa nilai yang disebut sebagai *mynumber*. Dan saya bisa menyimpan 5 nilai integer dalam array tanpa harus mendeklarasikan masing-masing. Jadi, bisa direpresentasikan seperti berikut :



Setiap ruang kosong dapat menyimpan sebuah angka integer. Dan array ini diberi nomor dari 0 hingga 4 karena dalam array indeks pertama selalu 0, terlepas dari panjangnya.

- Array 2 dimensi

Array dua dimensi sering digambarkan sebagai sebuah matriks, merupakan perluasand ari array satu dimensi. Jika array satu dimensi hanya terdiri dari sebuah baris dan beberapa kolom elemen, maka array dua dimensi terdiri dari beberapa baris dan beberapa kolom elemen yang bertipe sama sehingga dapat digambarkan sebagai berikut:

int y [2][2];

y[0][0]	y[0][1]
y[1][0]	y[1][1]

- Array Multidimensi

Array Multidimensi merupakan array yang serupa dengan array satu dimensi maupun array dua dimensi, namun array multidimensi dapat memiliki memori yang lebih besar. Biasanya array multidimensi digunakan untuk menyebut array dengan dimensi lebih dari dua atau array yang mempunyai lebih dari dua subskrip, seperti untuk menyebut array tiga dimensi, empat dimensi, lima dimensi dan seterusnya.

Ilustrasi Array Tiga dimensi dengan 24 elemen

III. Prosedur Percobaan

1. Buat listing program berikut ke compiler c++

Code

```
1  // Prototype Functions
2  #include <iostream>
3  using namespace std;
4  int perkalian (int &a, int &b);
5  int main ()
6  {
7  int z=5; int y=6;
8  cout<<perkalian(y,z);
9  return 0;
10 }
11
12 int perkalian (int &a, int &b)
13 {
14 a=a*3;
15 b=b*3;
16 int r;
17 r=a*b;
18 return (r);
19 }
```

Jalankan listing program dan analisis.

2. Ketik listing program berikut ini pada aplikasi C ++

Code

```
1  //Program metric 3x4
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     int i, j;
8     int metricA[100][100];
9
10    for (i=1;i<=3;i++)
11    {   for (j=1;j<=4;j++)
12     {cout <<"A " << i << "," << j <<" = ";cin >>
   metricA[i][j];}
13    }
14
```

```
15    cout << endl;
16
17    for (i=1;i<=3;i++)
18    {
19        for (j=1;j<=4;j++)
20        { cout << metricA[i][j] <<"\t";}
21 cout<<endl;
22    }
23
24    system ("PAUSE");
25    return 0;
26 }
```

Jalankan listing program dan analisis.

3. Buat listing program berikut ke compiler c++

Code

```
1  #include <iostream>
2  #include <math.h>
3  using namespace std;
4  float s;
5  int range(float v, float t, float a)
6  {
7  s=(v*t)-((0.5*a)*pow(t,2));
8  return(s);
9  }
10
11 int bil(float x)
12 { x=s;
13 if(s < 0)
14 cout<<"Nilai s merupakan bilangan negatif"<<endl;
15 else
16 cout<<"Nilai s merupakan bilangan positif"<<endl;
17 return 0;
18 }
19
20 int main()
21 {
22 float r,u,nilai[3];
23 cout<<"Masukkan nilai pertama = ";
24 cin>>nilai[0];
25 cout<<endl;
26 cout<<"Masukkan nilai kedua = ";
```

```
27  cin>>nilai[1];
28  cout<<endl;
29  cout<<"Masukkan nilai ketiga = ";
30  cin>>nilai[2];
31  cout<<endl;
32
33  r=range(nilai[0],nilai[1],nilai[2]);
34  cout<<"Nilai dari s adalah : "<<r <<endl;
35  u=bil(r);
36  cout<<endl<<endl;
37  //system ("pause");
38  return 0;
39  }
```

IV. Tugas Pendahuluan

1. Jelaskan pengertian array?
2. Sebutkan macam - macam array yang digunakan dalam pemrograman C++ dan jelakan bagaimana cara mendeklarasikannya?
3. Buat program sederhana menggunakan array dengan matriks (3x3) dimasukkan oleh user!
4. Buatlah program penjumlahan dan pengurangan array! Jelaskan proses pengurangan dan penjumlahannya!
5. Apa itu fungsi?
6. Buat program sederhana dengan penggunaan fungsi!
7. Bisakah kita membuat fungsi dengan tipe data int tanpa menggunakan return? Jelaskan!
8. Jelaskan perbedaan antara parameter formal dan aktual!
9. Apa perbedaan parameter pass by value dan pass by reference? Jelaskan dengan contoh program (**jangan diambil dari modul**)
10. Buatlah satu contoh program rekursif, dan jelaskan jalan programnya (**jangan diambil dari modul**)

CHAPTER V
POINTER DAN STRUCT

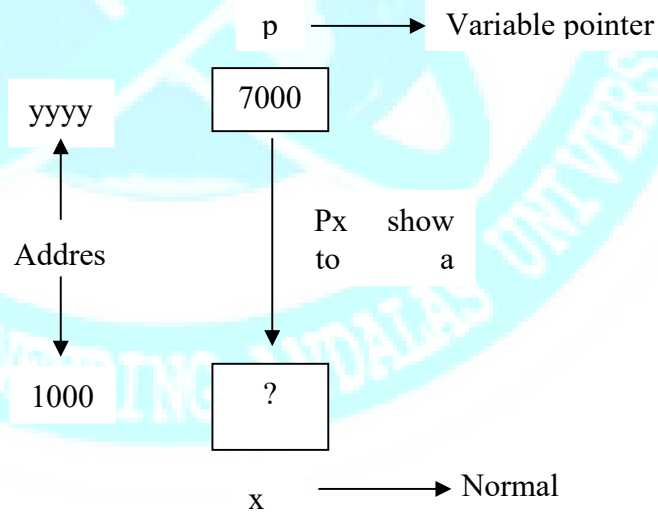
I. Tujuan

- Mampu memahami konsep dari pointer.
- Dapat membedakan berbagai jenis pointer.
- Dapat mengaplikasikan pointer pada program komputer.
- Mampu memahami konsep dari struct
- Mampu mengaplikasikan struct pada program komputer.

II. Teori

a. Apa itu pointer ?

Pointer adalah variabel yang berisi alamat memori dari variabel lain, sering merujuk sebagai variabel yang menunjuk ke objek / variabel lain. Sebagai contoh, px adalah pointer dan x adalah variabel yang ditunjukkan oleh px. Jika x adalah 7000 jam pada alamat memori, maka px juga diisi / dinilai 7000 jam. tipe data pointer dan variabel runcing harus sama.



b. Tujuan pointer:

Membuat variabel dinamis, yang diizinkan untuk menggunakan memori bebas (memori yang tidak digunakan) selama eksekusi program.

c. Jenis Pointer

i. Deference Operator (&)
Variabel yang dideklarasikan akan disimpan dalam sebuah lokasi memori dan pengguna tidak mengetahui dimana letak alamat data itu disimpan. Untuk mengetahui dimana alamat data disimpan, digunakan tanda ampersand (&), contoh :

Bil1 = &Bil2;
Berarti : isi variabel Bil1 sama dengan alamat dari variabel Bil2.

ii. Reference Operator (*)
Operator ini digunakan untuk mengakses nilai dari sebuah alamat yang ditunjuk oleh variabel pointer. Contoh :

Bil1=*Bil2;
Maka, nilai Bil1 sama dengan nilai yang ditunjuk oleh Bil2.

d. Cara penggunaan pointer

```
Data type *variable_name ;
```

Contoh :

```
int *px, *ptotal,x,y;
px = &x;
y = *px
```

Pernyataan pertama, px dan ptotal adalah variabel pointer yang menunjuk ke data / nilai tipe data int, dan pernyataan kedua px berisi alamat variabel x. pernyataan ketiga y berisi data / nilai x.

Contoh :

```
#include<iostream>
using namespace std;
int main()
{
int *px,x,y;
x = 87;
px = &x;
y = *px;
cout << "Alamat x = " <<&x << endl;
cout << "isi px = " << px << endl;
cout << "isi x = " << x << endl;
cout << "Nilai yang ditunjuk oleh px = " << *px << endl;
cout << "Nilai y = " << *px << endl;
system ("PAUSE");
return 0;
}
```

Menghasilkan : Alamat x = 0012FF78
isi px = 0012FF78
isi x = 87
Nilai yang ditunjukkan oleh px = 87
Nilai y = 87

e. Apa itu Struct?

Struct adalah salah satu fitur bahasa pemrograman C/C++, merupakan keyword yang memungkinkan kita untuk membuat sebuah deklarasi untuk membuat pengelompokan variabel dengan tipe data yang berbeda.Dalam kata lain, structure dapat menyimpan variabel-variabel dalam satu nama.

f. Cara Pendeklarasian Struct

```
struct nama_struct{

    tipe_data variable_1;

    tipe_data variabel_2;

    ...

    tipe_data variabel n;
}nama_object;
```

nama_struct : merupakan identitas dari struct tersebut.

nama_object : merupakan deklarasi yang menggunakan struct tersebut menjadi tipe data dari deklarasi tersebut. kita dapat mendirikan banyak object di tempat tersebut, masing-masing dipisahkan dengan tanda koma , . Object selalu diletakan setelah penutup block dan sebelum semicolon ;

Contoh :

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

struct mahasiswa{
    int nim ;
    string nama;
    float nilai ;
};

int main(){
    siswa budi, dio; //deklarasi object di luar struct

    budi.nim = 1;
    budi.nama = "Budi Januar";
    budi.nilai = 75.5;

    dio.nim = 2;
    dio.nama = "Dio Laksono";
    dio.nilai = 89.9;

    cout<<budi.nama<<" mendapatkan nilai "<<budi.nilai<<endl;
    cout<<dio.nama<<" mendapatkan nilai "<<dio.nilai<<endl;

    return 0;
```

}

III. Prosedur Percobaan

a. Type this listing program on your C++ source file.

```
1  // Program Struktur & Pointer
2  #include <iostream>
3  using namespace std;
4
5  struct number{
6          int value;
7          }one,two,three;
8
9  int formula (int &i, int &o,int &p)
10 {     int r;
11       r = (i + o) * p;
12       return (r);
13 }
14
15 int main()
16 {
17        int *px,x,y;
18 cout<< "Insert your number with three step..!" << endl;
19        cout << "number 1 = ";
20        cin >>one.value;
21        cout << "number 2 = ";
22        cin >>two.value;
23        cout << "number 3 = ";
24        cin >>three.value;
25        cout << endl;
26        x = formula (one.value , two.value, three.value);
27     px = &x;
28        y = *px;
29        cout << "The address of x = " <<&x << endl;
30        cout << "Content of px = "   << px << endl;
31        cout << "Content of x = "    << x << endl;
32        cout << "The value in the px = " << *px << endl;
33        cout << "The value of y = " << y << endl;
34        system("PAUSE");
35        return 0;
36 }
```

IV. TUGAS PENDAHULUAN

1. Jelaskan tentang program ini!

```
#include <iostream>
using namespace std;
int main()
{
int new1 = 6;
int *p_new1;
p_new1 = &new1;
cout << "value: new1 = " <<new1;
cout << ", *p_new1 = " << *p_new1<< "\n";
cout << "address new1 = " <<&new1;
cout << ", p_new1 = " << p_new1<< "\n";
*p_new1 = *p_new1 + 1;
cout << "The present new1 = " <<new1<< "\n";
system ("PAUSE");
return 0;
}
```

2. Apa itu pointer? Tujuan pointer dan Sintaks?
3. Buat contoh program menggunakan deference dan reference operator, dan jelaskan programnya! (**jangan diambil dari modul**)
4. Apakah pointer (*) dapat mengakses variabel pointer (*) lainnya? Coba jelaskan!
5. Apakah nilai pointer dapat diubah? Jika iya, bagaimana cara mengubahnya? Jelaskan dengan contoh program sederhana. (**jangan diambil dari modul**)

CHAPTER VI
OOP 1
(Class, Enkapsulasi, Constructor dan Destructor)

I. Tujuan

- Praktikan dapat mengerti perbedaan class tipe *public, private dan protected* dan cara penggunaanya
- Praktikan dapat mengerti apa itu constructor dan destructor
- Praktikan dapat mempraktekan konsep penggunaan class di program komputer.

II. Teori

a. Classes

Class adalah konsep struktur data yang diperluas. Sama seperti struktur data, class dapat beranggotakan data, dan juga bisa diisi dengan fungsi. Kelas dalam C++ akhir-akhir ini diperkenalkan sebagai pendukung program berorientasi objek. Kelas dapat mengatur data dan fungsi dalam struktur yang sama dengan cara mendeklarasikannya menggunakan keyword ***class***, yang tujuannya mirip dengan ***struct*** pada modul sebelumnya, namun kelas memiliki kemungkinan untuk memiliki anggota selain data.

Bentuk umum dan contoh ***class***:

class *Class_name***{**
permission_label_1:
*member1***;**
permission_label_2:
*member2***;**
...
} *object_name***;**

```
class mahasiswa{
public:
    int nim;
    string nama;
    float nilai;
private:
    void printData(){
        cout<<"NIM \t= "<<nim<<endl;
        cout<<"Nama\t= "<<nama<<endl;
        cout<<"Nilai \t= "<<nilai<<endl;
}
}anton;
```

Dimana **class_name** merupakan nama kelas yang ditentukan, dan **object_name** merupakan daerah pilihan atau sebuah objek pengenal. Pada bagian deklarasi bisa berisi sebuah ***members,*** dimana bisa berupa deklarasi data atau sebuah fungsi serta sebuah **permission_label** yang bisa menjadi salah satu dari ***private:***, ***public:***, or ***protected:.***

b. **Enkapsulasi**

Encapsulation merupakan suatu cara pembungkusan data dan method yang menyusun suatu kelas sehingga kelas dapat dipandang sebagai suatu modul dan cara bagaimana menyembunyikan informasi detail dari suatu class (information hiding). Dalam OOP,

Enkapsulasi sangat penting untuk keamanan serta menghindari kesalahan permrograman, pembungkus disini dimaksudkan untuk menjaga suatu proses program agar tidak dapat diakses secara sembarangan atau di intervensi oleh program lain.

Jenis dari Enkapsulasi

1. **Private** adalah sebuah *permission label* di mana anggota kelas hanya dapat diakses di kelasnya sendiri atau dari kelas "teman".

   Anggota kelas private mempunyai kendali akses yang paling ketat. Dalam bagian private, hanya fungsi anggota dari kelas itu yang dapat mengakses anggota private atau kelas yang dideklarasikan sebagai teman (friend).

2. **Protected** merupakan sebuah *permission label* dimana anggoata kelas hanya dapat diakses dikelasnya sendiri atau dari kelas “teman” dan anggota kelas turunan.

   Suatu kelas dapat dibuat berdasarkan kelas lain. Kelas baru ini mewarisi sifat-sifat dari kelas dasarnya. Dengan cara ini bisa dibentuk kelas turunan dari beberapa tingkat kelas dasar. Bila pada kelas dasar mempunyai anggota dengan hak akses terproteksi, maka anggota kelas ini akan dapat juga diakses oleh kelas turunannya. Anggota kelas terproteksi dibentuk dengan didahului kata kunci protected. Pada bagian protected, hanya fungsi anggota dari kelas dan kelas-kelas turunannya yang dapat mengakses anggota.

3. **Public** merupakan sebuah *permission label* dimana anggota kelas dapat diakses dari kelas apapun atau program utama dan fungsi lain.

Suatu kelas agar bisa diakses dari luar kelas, misalnya dalam fungsi main(), perlu mempunyai hak akses publik. Hak akses ini yang biasanya digunakan sebagai perantara antara kelas dengan dunia luar.

```
class Persegi
{
private:
  double panjang;
  double lebar;
public:
  double hitungLuas(){
    return panjang*lebar;
  }
  double hitungKeliling(){
    return 2.0*panjang+lebar;
  }
};
```

c. **Constructors and Destructors**

Konstruktor digunakan untuk menginisialisasi variabel atau memori dinamis saat sebuah objek dibuat. Pada sebuah kelas, fungsi dari konstruktor menggunakan nama yang sama dengan nama kelasnya. Sebuah konstruktor akan otomatis dibuat ketika sebuah objek pada sebuah kelas dibuat.

Biasanya konstruktor dipakai untuk inisialisasi anggota data dan melakukan operasi lain seperti membuka file dan melakukan alokasi memori secara dinamis. Meskipun konstruktor tidak harus ada di dalam kelas, tetapi jika diperlukan konstruktor dapat lebih dari satu.

Destruktor akan secara otomatis terpanggil atau dieksekusi ketika program telah selesai dijalankan. Destruktor memiliki nama yang sama dengan kelas, namun diawalnya terdapat tilde (~), dan tidak memiliki nilai return.

contoh:

```
class Mobil
{
    private :
      int roda, pintu;
      charwarna[10];
```

```
public :
  Mobil(); //Konstruktor
  ~Mobil(); //Destruktor
  void info();
  }object1;
```

III. Prosedur Percobaan

1.1 Class ,Constructor and destructor

```
1  #include<iostream>
2
3  using namespace std;
4
5  class Rectangle{
6  public:
7  int *length, *width;
8
9  public :
10 Rectangle (int a, int b){
11           length = new int;
12           width = new int;
13
14 *length = a;
15 *width = b;
16           }
17 int arectangle(){
18 return (*length * *width);
19 }
20 ~Rectangle (){cout <<"\nThe result before destructor executed is
   : "<<(*length * *width)<<endl;
21 cout <<"The address of length is : "<<&length << endl;
22 cout <<"The address of width is  : "<<&width<< endl<<endl;
23 system("pause");
24       delete length;
25       delete width;
26 cout <<"\nThe result after destructor executed is : "<<(*length
   * *width)<<endl;
27 cout <<"The address of length is : "<<&length << endl;
28 cout <<"The address of width is  : "<<&width << endl<<endl;
29 system("pause");
30 }
31 };
32
33 int main(){
34     Rectangle square1 (3,4);
35     cout  <<"The  rectangle  of  the  square1  is  :  "<<
   square1.arectangle() << endl<<endl;
36     }
```

1.2 Buatlah listing program berikut

```
1  // OOP Sederhana
2  #include <iostream>
3  using namespace std;
4  class lingkaran{
5  private:
6       float radius;
7       float pi = 3.14;
8  public:
9       lingkaran(float r){
10            radius = r;
11 }
12 float luas(){
13       return (pi*radius*radius);
14 }
15 float keliling(){
16       return (2*pi*radius);
17 }};
18 int main (){
19 lingkaran lingkaran1(12);
20 cout<< "Luas :" <<lingkaran1.luas() <<endl;
21 cout<< "keliling :" <<lingkaran1.keliling()<<endl;
22 return 0;
23 }
```

IV. Tugas Pendahuluan

1. Apa perbedaan antara *struct* dan *class?*
2. Buatlah program sederhana tentang class! **Jangan ambil dari modul**
3. Jelaskan apa yang dimaksud dari constructor dan destructor!
4. Jelaskan perbedaan class public, private dan protected!
5. Buatlah syntax dari class, destructor dan constructor!
6. Apa fungsi dari **delete** dari percobaan 1?
7. Apa fungsi dari ~ (tilde) dari program percobaan 1?
8. Dapatkah kita menggunakan lebih dari satu objek dari satu kelas ? buatlah program sederhana tentang itu!

CHAPTER VII
OOP 2
(Polimorfisme dan *Inheritance*)

I. Tujuan

- Praktikan dapat mengetahui apa itu polimorfisme
- Praktikan dapat mempraktekan konsep penggunaan polimorfisme
- Praktikan dapat mengerti apa itu inheritance dan cara penggunaanya.

II. Teori

a. Inheritance

Inheritance merupakan subkelas dari kelas utama, ketika sebuah objek atau kelas berdasarkan kelas atau objek yang lain, dan menggunakan implementasi yang sama (mewarisi dari kelas utama) atau menentukan implementasi untuk mempertahankan antarmuka atau perilaku yang sama. Sehingga kode dan ekstensi independen dari perangkat asli dapat digunakan kembali. Hubungan kelas atau objek melalui inheritance merupakan bentuk dari hierarki. Sebuah kelas bisa menjadi superclass untuk kelas lainnya yang disebut dngan multilevel inheritance.

Dalam kelas pada pemrograman C++, terdapat mekanisme keturunan (*inheritance*), dimana dalam mekanisme ini terdapat istilah kelas induk dan kelas anak. Kelas anak akan menginduk pada kelas induk. Keistimewaan dari metode *inheritance* adalah, kelas anak memiliki sifat-sifat atau fungsi dari kelas induk, namun dengan "batasan" tertentu. Dengan mekanisme keturunan ini, member kelas anak akan bertambah yaitu member kelas anak sendiri ditambah dengan member kelas induk.

Bentuk penulisan pewarisan:

class <nama_kelas_turunan> : <penentu_pewarisan> <nama_kelas_dasar>

Penentu pewarisan	Hak akses sebelumnya di kelas dasar	Hak akses baru di kelas turunan
private	**private** **protected** **public**	tidak diwariskan **private** **private**

protected	**private** **protected** **public**	tidak diwariskan **protected** (tetap) **protected**
public	**private** **protected** **public**	tidak diwariskan **protected** (tetap) **public** (tetap)

Tabel Penentu Warisan

Sebagai contoh, kelas anak memiliki member a dan b, sedangkan kelas induk memiliki member c. Apabila kelas anak adalah keturunan dari kelas induk, maka member kelas anak adalah a, b, dan c. Namun di sini kelas induk tidak berhak akan member anak yaitu a dan b.

contoh:

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

class induk{
protected:
int sisi_a,sisi_b;
public:
void input(float panjang, float lebar);
};

void induk::input(float panjang, float lebar){
sisi_a = panjang;
sisi_b = lebar;
}

class anak:public induk{
public:
float luas(){return sisi_a*sisi_b;};
};

int main(){
anak o;
o.input(7,8);
```

```
cout<<"Nilai dari o.luas() adalah = "<<o.luas()<<endl;
return 0;
}Daughter.say("Pemrograman Komputer");
system("PAUSE");
return 0;
}
```

b. Polimorfisme

Polimorfisme merupakan suatu konsep yang menyatakan bahwa sesuatu yang sama dapat memiliki berbagai bentuk dan perilaku yang berbeda. Dalam hal ini polimorfisme merupakan suatu sifat menyandarkan pada kesamaan nama dalam program. Pengenal data, instans, dan metode, bahkan nama fungsi dapat dibuat dengan nama yang sama untuk kegunaan yang berbeda.

Salah satu bentuk polimorfisme pada C++ dapat digunakan pada fungsi atau operator dan dikenal sebagai istilah **overloading**. **Overloading** terhadap fungsi akan memungkinkan sebuah fungsi dapat menerima bermacam-macam tipe dan memberikan nilai balik yang bervariasi pula.

```
#include <iostream>
#include <conio.h>
using namespace std;
class balok {
double p,l,t;
public : double Luaspermukaan() {
return(2*p*l)+(2*p*t)+(2*l*t);
}

public : double volume() {
return(p*l*t);
}
return(p*l*t);
}
```

```
public : double volume(double t) {
return(p*l*t);
}
public : double volume(double t, double l) {
public : double volume(double t, double l, double p) {
return(p*l*t);
}
void setpanjang(double pan){
p = pan;
}
void setlebar(double leb){
l = leb;
}
void settinggi(double tin){
t = tin;
}
};
int main(){
int pa,le,ti,pp,ll,tt;
balok tes;
cout<<"Balok"<<endl;
cout<<"Input Panjang : "; cin>>pa;
cout<<"Input Lebar : "; cin>>le;
cout<<"Input Tinggi : "; cin>>ti;
tes.setpanjang(pa);
tes.setlebar(le);
tes.settinggi(ti);

cout<<"Luas Permukaan Balok = "<<tes.Luaspermukaan()<<endl;
cout<<"Volume Balok = "<<tes.volume()<<endl;
cout<<"Tinggi baru = "; cin>>tt;
cout<<"Volume Balok 1 parameter = "<<tes.volume(tt)<<endl;
cout<<"Lebar baru = "; cin>>ll;
cout<<"Volume Balok 2 parameter = "<<tes.volume(tt, ll)<<endl;
cout<<"Panjang baru = "; cin>>pp;
cout<<"Volume Balok 3 parameter = "<<tes.volume(tt, ll, pp)<<endl;

getch();
}
```

Dari contoh program diatas, terdapat fungsi volume yang di pakai secara berulang kali dengan parameter yang berbeda-beda.

Polimorfism pada class terjadi ketika terdapat class yang mendeklarasikan atau mewarisi fungsi virtual.Fungsi Virtual –> fungsi yang mendukung adanya polymorphic function, artinya fungsi tersebut dapat di difinisikan ulang pada kelas turunanya. Fungsi virtual biasanya terdapat pada kelas dasar.

Salah satu fitur utama dari pewarisan kelas adalah bahwa pointer ke kelas turunan kompatibel dengan tipe dengan penunjuk ke kelas dasarnya. Polimorfisme adalah seni memanfaatkan fitur yang sederhana namun kuat dan serbaguna ini.

Contoh programnya

```
1  // virtual members#include <iostream>using namespace std;
2  class Polygon {
3    protected:
4      int width, height;
5    public:
6      void set_values (int a, int b)
7        { width=a; height=b; }
8      virtual int area ()
9        { return 0; }
10 };
11 class Rectangle: public Polygon {
12   public:
13     int area ()
14       { return width * height; }
15 };
16 class Triangle: public Polygon {
17   public:
18     int area ()
19       { return (width * height / 2); }
20 };
21 int main () {
22   Rectangle rect;
23   Triangle trgl;
24   Polygon poly;
25   Polygon * ppoly1 = &rect;
26   Polygon * ppoly2 = &trgl;
27   Polygon * ppoly3 = &poly;
28   ppoly1->set_values (4,5);
```

```
29    ppoly2->set_values (4,5);
30    ppoly3->set_values (4,5);
31    cout << ppoly1->area() << '\n';
32    cout << ppoly2->area() << '\n';
33    cout << ppoly3->area() << '\n';
34    return 0;
35  }
```

Output :

20

10

0

Fungsi main mendeklarasikan tiga pointer ke Polygon (bernama ppoly1, ppoly2 dan ppoly3). Ini diberi alamat rect, trgl dan poly, masing-masing, yang merupakan objek dari Rectangle, Triangle dan Polygon. Rectangle dan Triangle adalah kelas yang diturunkan dari Poligon.

Dereferensi ppoly1 dan ppoly2 (dengan * ppoly1 dan * ppoly2) valid dan memungkinkan kita untuk mengakses anggota objek yang mereka tunjuk. Dapat dilihat pada line 28 dan line 29

Dalam contoh ini, ketiga kelas (Poligon, Rectangle dan Triangle) memiliki anggota yang sama: lebar, tinggi, dan fungsi set_values dan area.

Fungsi area anggota telah dideklarasikan sebagai virtual di kelas dasar karena kemudian didefinisikan ulang di setiap kelas turunan. Anggota non-virtual juga dapat didefinisikan ulang dalam kelas turunan, tetapi anggota non-virtual dari kelas turunan tidak dapat diakses melalui referensi kelas dasar: yaitu, jika virtual dihapus dari deklarasi area dalam contoh di atas, ketiga panggilan tersebut to area akan mengembalikan nol, karena dalam semua kasus, versi kelas dasar akan dipanggil sebagai gantinya.

Oleh karena itu, pada dasarnya, apa yang dilakukan kata kunci virtual adalah mengizinkan anggota kelas turunan dengan nama yang sama dengan yang ada di kelas dasar untuk dipanggil dengan tepat dari penunjuk, dan lebih tepatnya bila jenis penunjuk adalah penunjuk ke kelas dasar yang menunjuk ke suatu objek dari kelas turunan, seperti pada contoh di atas.

Kelas yang mendeklarasikan atau mewarisi fungsi virtual disebut class polimorfik.

Perhatikan bahwa terlepas dari virtualitas salah satu anggotanya, Polygon adalah kelas biasa, yang bahkan sebuah objek telah dipakai (poli), dengan definisi sendiri dari area anggota yang selalu mengembalikan 0.

III. Prosedur Percobaan

1.3 Inheritance

```
1  #include <iostream>
2  using namespace std;
3  class basic
4  {
5  public :
6  int X;
7  public :
8  basic ()
9  {X=10;}
10 };
11
12 class inherit : public basic
13 {
14 int Y, result;
15 public :
16 void setY (int YY)
17 {Y=YY;}
18 void timesXY ()
19 {result=X*Y;}
20 int get_result()
21 {return result;}
22 };
23
24 int main ()
25 {
26 basic A;
27 inherit B;
28        cout<<" x : " <<endl;
29 cin>>A.X;
30 B.setY(5);
31 B.timesXY();
32 cout<<"The result of X times Y is : "<<B.get_result()<<endl;
33 system("PAUSE");
34 return 0;
35 }
```

1.2 Polimorfisme

```
1  #include <stdio.h>
2  #include <iostream>
3  using namespace std;
4  // Deklarasi kelas-kelas
5  class Kotak
6  {
7  protected :
8  int panjang, lebar;
9  int luas, keliling;
10 public :
11 Kotak (int Panjang, int Lebar);
12 int ambil_panjang();
13 int ambil_lebar();
14 virtual void hitung(); // metode virtual
15 void ambil (int& Luas, int& Keliling);
16 };
17 class Balok : public Kotak
18 {
19 private :
20 int tinggi;
21 int volume;
22 public :
23 Balok (int Panjang, int Lebar, int Tinggi);
24 int ambil_tinggi();
25 virtual void hitung(); // metode virtual
26 void ambil (int& Luas, int& Keliling, int& Volume);
27 };
28
29 // Definisi metode-metode kelas Kotak
30
31 Kotak::Kotak (int Panjang, int Lebar)
32
33 {
34 panjang = Panjang;
35 lebar = Lebar;
36 }
37 int Kotak::ambil_panjang()
38 {
39 return panjang;
40 }
41
```

```
42  int Kotak::ambil_lebar()
43  {
44  return lebar;
45
46  }
47  void Kotak::hitung()
48  {
49  luas = panjang * lebar;
50  keliling = 2 * (panjang + lebar);
51  }
52
53  void Kotak::ambil (int& Luas, int& Keliling)
54  {
55  Luas = luas;
56  Keliling = keliling;
57  }
58
59  // Definisi metode-metode kelas Balok
60
61  Balok::Balok (int Panjang, int Lebar, int Tinggi):Kotak
    (Panjang, Lebar)
62  {
63  tinggi = Tinggi;
64  }
65
66  int Balok::ambil_tinggi()
67  {
68  return tinggi;
69  }
70  void Balok::hitung()
71  {
72  Kotak::hitung();
73  volume = panjang*lebar*tinggi;
74  }
75
76  void Balok::ambil(int& Luas, int& Keliling, int& Volume)
77  {
78  Kotak::ambil(Luas, Keliling);
79  Volume = volume;
80  }
81
82  // Program Utama
```

```
83  int main ()
84  {
85  Kotak A (12, 3);
86  Balok B (12, 3, 8);
87  cout << endl;
88  cout << "Menghitung luas, keliling, dan volume bangun" <<
    endl;
89  cout << "================================================"
    << endl;
90  A.hitung();
91
92  int l, k;
93  A.ambil(l, k);
94  cout << "Kotak sisi-sisinya : " << endl;
95  cout << "Panjang = " << A.ambil_panjang() << endl;
96  cout << "Lebar = " << A.ambil_lebar() << endl;
97  cout << "Memiliki luas " << l <<" dan keliling "<< k <<
    endl << endl;
98
99  int v;
100 B.hitung();
101 B.ambil(l, k, v);
102 cout << "Balok dengan rusuk-rusuknya : " << endl;
103 cout << "Panjang = " << B.ambil_panjang() << endl;
104 cout << "Lebar = " << B.ambil_lebar() << endl;
105 cout << "Tinggi = " << B.ambil_tinggi() << endl;
106 cout << "Memiliki luas " << l <<", keliling "<< k <<",
    dan volume " << v <<endl;
107
108 cout << endl;
109 }
```

IV. Tugas Pendahuluan

1. Jelaskan apa yang dimaksud dengan polimorfisme!
2. Jelaskan apa fungsi dari polimorfisme!
3. Buatlah contoh program sederhana dengan polimorfisme! **Jangan diambil dari modul!**
4. Jelaskan apa yang dimaksud inheritance dan buat program sederhana ! **jangan ambil dari modul**
5. Bisakah inheritance menggunakan class public, private, dan protected? Jelaskan!