LAPORAN PRAKTIKUM
Modul 2.
PENGENALAN BAHASA C++ (BAGIAN KEDUA)

Disusun Oleh:
Zhafir Zaidan Avail
S1-SE-07-2

Dosen :
Wahyu Andi Saputra, S.Pd., M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. Tujuan

1. Memahami penggunaan pointer dan alamat memori
2. Mengimplementasikan fungsi dan prosedur dalam program

2. Landasan Teori

❖ Array

Array merupakan kumpulan data dengan nama yang sama dan setiap elemen bertipe data sama. Untuk mengakses setiap komponen / elemen array berdasarkan indeks dari setiap elemen.

➢ Array 1 Dimensi

Adalah array yang hanya terdiri dari satu larik data saja. Cara pendeklarasian array satu dimensi:

```
tipe_data nama_var[ukuran]
```

Keterangan:

```
Tipe_data → menyatakan jenis elemen array (int, char, float, dll).
Ukuran → menyatakan jumlah maksimum array.
```

Contoh:

```
int nilai[10];
```

Menyatakan bahwa array nilai mengandung 10 elemen dan bertipe integer.
Dalam C++ data array disimpan dalam memori pada lokasi yang berurutan. Elemen pertama memiliki indeks 0 dan selemen selanjutnya memiliki indeks 1 dan seterusnya. Jadi jika terdapat array dengan 5 elemen maka elemen pertama memiliki indeks 0 dan elemen terakhir memiliki indeks 4.

```
nama_var[indeks]
```

nilai[5] → elemen ke-5 dari array nilai. Contoh memasukkan data ke dalam array:

```
nilai[4] = 90; /*memasukkan 90 ke dalam array nilai indeks ke-4*/
cin << nilai[4] /*membaca input-an dari keyboard*/ 2
```

- **Array 2 Dimensi**

  Bentuk array dua dimensi ini mirip seperti tabel. Jadi array dua dimensi bisa digunakan untuk menyimpan data dalam bentuk tabel. Terbagi menjadi dua bagian, dimensi pertama dan dimensi kedua. Cara akses, deklarasi, inisialisasi, dan menampilkan data sama dengan array satu dimensi, hanya saja indeks yang digunakan ada dua. Contoh:

```
int data_nilai[4][3];
nilai[2][0] = 10;
```

	0	1	2
0			
1			
2	10		
3			

- **Array Berdimensi Banyak**

  Merupakan array yang mempunyai indeks banyak, lebih dari dua. Indeks inilah yang menyatakan dimensi array. Array berdimensi banyak lebih susah dibayangkan, sejalan dengan jumlah dimensi dalam array.
  Cara deklarasi:

```
tipe_data nama_var[ukuran1][ukuran2]…[ukuran-N];
```

  Contoh:

```
int data_rumit[4][6][6];
```

  Array sebenarnya masih banyak pengembangannya untuk penyimpanan berbagai betuk data, pengembangan array misalnya untuk array tak berukuran.

❖ Pointer

- **Data dan Memori**

  Semua data yang ada digunakan oleh program komputer disimpan di dalam memori (RAM) komputer. Memori dapat digambarkan sebagai sebuah array 1 dimensi yang berukuran sangat besar. Seperti layaknya array, setiap cell memory memiliki “indeks” atau “alamat” unik yang berguna untuk identitas yang biasa kita sebut sebagai “address”

  Saat program berjalan, Sistem Operasi (OS) akan mengalokasikan space memory untuk setiap variabel, objek, atau array yang kita buat. Lokasi pengalokasian memori bisa sangat teracak sesuai proses yang ada di dalam OS masing-masing.

Pada saat komputer pertama kali berjalan keadaan memori adalah kosong. Saat variabel dideklarasikan, OS akan mencarikan cell kosong untuk dialokasikan sebagai memori variabel tersebut.

```
char a;
int j;
char arr[6];
arr[3] = 'b'
a = 'u'
```

Pada contoh di atas variabel a dialokasikan di memory alamat x6, variabel j dialokasi kan di alamat x10-13, dan variabel arr dialokasiakan di alamat x17-22. Nilai variabel yang ada di dalam memori dapat dipanggil menggunakan alamat dari cell yang menyimpannya. Untuk mengetahui alamat memori tempat di mana suatu variabel dialokasikan, kita bisa menggunakan keyword "&" yang ditempatkan di depan nama variabel yang ingin kita cari alamatnya.

C++	Output	Keterangan
Cout << a << endl; Cout << &a << endl; Cout << j << endl; Cout << &j << endl; Cout << &(arr[4]) << endl;	'u' x6 0 X10 X21	Nilai variabel a Alamat varibel j Nilai variabel j Alamat varibel j Alamat variabel arr[4]

➢ Pointer dan Alamat

Variabel pointer merupakan dasar tipe variabel yang berisi integer dalam format heksadesimal. Pointer digunakan untuk menyimpan alamat memori variabel lain sehingga pointer dapat mengakses nilai dari variabel yang alamatnya ditunjuk.

Cara pendeklarasian variabel pointer adalah sebagai berikut:

```
type *nama_variabel;
```

Contoh:

```
int *p_int;
/* p_int merupakan variabel pointer yang menunjuk ke data bertipe int */
```

Agar suatu pointer menunjuk ke variabel lain, mula-mula pointer harus diisi dengan alamat memori yang ditunjuk.

```
p_int = &j;
```

Pernyataan di atas berarti bahwa p_int diberi nilai berupa alamat dari variabel j. Setelah pernyataan tersebut di eksekusi maka dapat dikatakan bahwa p_int menunjuk ke variabel j. Jika suatu variabel sudah ditunjuk oleh pointer. Maka, variabel yang ditunjuk oleh pointer dapat diakses melalui variabel itu sendiri ataupun melalui pointer.

Untuk mendapatkan nilai dari variabel yang ditunjuk pointer, gunakan tanda * di depan nama variabel pointer

Pointer juga merupakan variabel, karena itu pointer juga akan menggunakan space memory dan memiliki alamat sendiri

```
int *p_int;
```

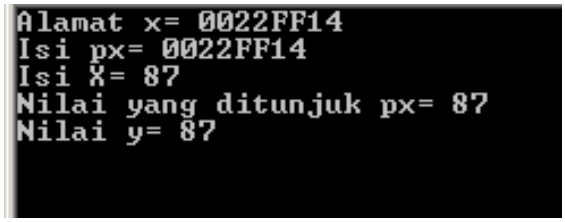
C++	Output	Keterangan
int j,k; j =10; int *p_int;	10	Nilai variabel j

	p_int = &j;
	cout<< j << endl;
	cout<< &j << endl;
	cout<< p_int << endl;
	cout<< &p_int << endl;
	cout<< *p_int << endl;
	k = *p_int;
	cout << k << endl;
		X6
		X6
		
		X1
		
		10
		
		10
			Alamat variabel j
			Nilai variabel p_int
			
			Alamat variabel p_int
			
			Nilai variabel yang ditunjuk p_int
			Nilai variabel k

Berikut ini contoh program sederhana menggunakan pointer:

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

int main(){
int x,y; //x dan y bertipe int
int *px; //px merupakan variabel pointer menunjuk ke variabel int
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 px= "<<*px<<endl;
cout<<"Nilai y= "<<y<<endl;
getch();
return 0;
}
```



➢ Poinger dan Array

Ada keterhubungan yang kuat antara *array* dan *pointer*. Banyak operasi yang bisa dilakukan dengan *array* juga bisa dilakukan dengan *pointer*. Pendeklarasian *array*: int a[10];

Mendefinisikan *array* sebesar 10, kemudian blok dari objek *array* tersebut diberi nama

a[0],a[1],a[2],…….a[9].

Notasi a[i] akan merujuk elemen ke-i dari *array*. Jika pa merupakan *pointer* yang menunjuk variabel bertipe *integer* , yang di deklarasikan sebagai berikut :

```
int *pa;
```

maka pernyataan :

```
pa=&a[0];
```

akan membuat pa menunjuk ke alamat dari elemen ke-0 dari variabel a. Sehingga, pa akan mengandung alamat dari a[0].

Sekarang, pernyataan :

```
x=*pa;
```

Akan menyalinkan isi dari a[0] ke variabel x.

Jika pa akan menunjuk ke elemen tertentu dari *array*, maka pendefinisian pa + 1 akan menunjuk elemen berikutnya, pa + i akan menunjuk elemen ke-i setelah pa, sedangkan pa − i akan menunjuk elemen ke-i sebelum pa sehingga jika pa menunjuk ke a[0] maka ∗ (pa + 1) akan mengandung isi elemen ke a[1] . pa + i merupakan alamat dari $a[i]$, dan ∗ (pa + i) akan mengandung isi dari elemen $a[i]$.

```
#include <iostream>
#include <conio.h>
#define MAX 5
using namespace std;

int main(){
int i,j;
float nilai_total, rata_rata;
float nilai[MAX];
static int nilai_tahun[MAX][MAX]=
{      {0,2,2,0,0},
{0,1,1,1,0},
{0,3,3,3,0},
{4,4,0,0,4},
{5,0,0,0,5}
};
/*inisialisasi array dua dimensi */
for (i=0; i<MAX; i++){
cout<<"masukkan nilai ke-"<<i+1<<endl;
cin>>nilai[i];
}
cout<<"\ndata nilai siswa :\n";

/*menampilkan array satu dimensi */
for (i=0; i<MAX; i++)
cout<<"nilai k-"<<i+1<<"=" <<nilai[i]<<endl;
cout<<"\n nilai tahunan : \n";

/* menampilkan array dua dimensi */
for(i=0; i<MAX; i++){
for(j=0; j<MAX; j++)
cout<<nilai_tahun[i][j];
cout<<"\n";
}
getch();
return 0;
}
```

- **Pointer dan String**

A. String

String merupakan bentuk data yang sering digunakan dalam bahasa pemrograman

untuk mengolah data teks atau kalimat. Dalam bahasa C pada dasarnya *string* merupakan kumpulan dari karakter atau *array* dari karakter.

Deklarasi variabel *string*:

```
char nama[50];
50 à menyatakan jumlah maksimal karakter dalam string.
```

Memasukkan data *string* dari *keyboard*:

```
gets(nama_array);
```

Menampilkan *string* bisa nggunakan **puts()** atau **cout()** :

```
puts(nama);
cout << nama;
```

Untuk mengakses data *string* sepertihalnya mengakses data pada *array*, pengaksesan dilakukan perkarakter sesuai dengan indeks setiap karakter dalam *string*.

Contoh :

```
Cout<<nama[3]; /*menampilkan karakter ke-3 dari string*/
```

B. Pointer dan String

Merupakan array dari karakter. Dalam representasi internal, array diakhiri dengan karakter '\0' sehingga program dapat menemukan akhir dari program. Panjang dari storage merupakan panjang
dari karakter yang ada dalam tanda petik dua ditambah satu. Ketika karakter string tampil dalam sebuah program maka untuk mengaksesnya digunakan pointer karakter. Standar input/output akan menerima pointer dari awal karakter array sehingga konstanta string akan diakses oleh pointer mulai dari elemen pertama.

Jika pmessage di deklarasikan :

```
char *pmessage ;
```

Maka pernyataan berikut :

```
pmessage = "now is the time";
```

Akan membuat *pmessage* sebagai *pointer* pada karakter *array*. Ini bukan copy *string*, hanya *pointer*
yang terlibat. C tidak menyediakan operator untuk memproses karakter *string* sebagai sebuah unit. Ada perbedaan yang sangat penting diantara pernyataan berikut :

```
char amessage[]="now is the time"; //merupakan array
char *pmessage= "now is the time"; //merupakan pointer
```

Variabel amessage merupakan sebuah *array*, hanya cukup besar untuk menampung karakter-karakter sequence tersebut dan karakter null '\0' yang menginisialisasinya. Tiap-tiap karakter dalam *array* bisa saja berubah tapi variabel amessage akan selalu menunjuk apada *storage* yang sama. Di sisi lain, pmessage merupakan *pointer*, diinisialisasikan menunjuk konstanta *string*, *pointer* bisa di modifikasi untuk menunjuk kemanapun, tapi hasilnya tidak akan terdefinisi jika kamu mencoba untuk mengubah isi *string*.

❖ Fungsi

Fungsi merupakan blok dari kode yang dirancang untuk melaksanakan tugas khusus dengan

tujuan:

1. Program menjadi terstruktur, sehingga mudah dipahami dan mudah dikembangkan. Program dibagi menjadi beberapa modul yang kecil.
2. Dapat mengurangi pengulangan kode (duplikasi kode) sehingga menghemat ukuran program.

Pada umumnya fungsi memerlukan masukan yang dinamakan sebagai parameter. Masukan ini selanjutnya diolah oleh fungsi. Hasil akhir fungsi berupa sebuah nilai (nilai balik fungsi).
Bentuk umum sebuah fungsi:

```
tipe_keluaran nama_fungsi(daftar_parameter) {
blok pernyataan fungsi ;
}
```

Jika penentu_tipe fungsi merupakan tipe dari nilai balik fungsi, bila tidak disebutkan maka akan dianggap (default) sebagai int.

Algoritma	C++

Program coba_fungsi
Kamus
x,y,z : integer
function max3(input:
a,b,c : integer) : integer
 	
Algoritma
input(x,y,z)
output(max3(x,y,z))
function max3(input:a,b,c :
integer) : integer
kamus
temp_max : integer algoritma
temp_max □ a
if (b>temp_max) then
temp_max □ b
if (c>temp_max) then temp_max □ c
→ temp_max
	#include <conio.h>
	#include <iostream>
	#include <stdlib.h>
	using namespace std;
	int maks3(int a, int b, int c);
	/*mendeklarasikan prototype fungsi
	*/
	int main(){
	system("cls"); int x,y,z;
	cout<<"masukkan nilai bilangan ke-1
	=";
	cin>>x;
	cout<<"masukkan nilai bilangan ke-2
	=";
	cin>>y;
	cout<<"masukkan nilai bilangan ke-3
	=";
	cin>>z;
	cout<<"nilai maksimumnya adalah =
	<<maks3(x,y,z);
	getche();
	return 0;
	}
	/*badan fungsi */
	int maks3(int a, int b, int c){
	/* deklarasi variabel lokal dalam
	fungsi */
	Int temp_max =a;
	if(b>temp_max)
	temp_max=b;
	if(c>temp_max)
	temp_max=c;
	return (temp_max);
	}

❖ Prosedur

Dalam C sebenarnya tidak ada prosedur, semua berupa fungsi, termasuk main() pun adalah sebuah fungsi. Jadi prosedur dalam C merupakan fungsi yang tidak mengembalikan nilai, biasa diawali dengan kata kunci void di depan nama prosedur.
Bentuk umum sebuah prosedur:

```
void nama_prosedur (daftar_parameter) { blok pernyataan prosedur ;
}
```

Algoritma	C++
Program coba_procedur	#include <iostream>

Kamus
jum : integer
procedure tulis(input: x: integer)
Algoritma
input(jum) tulis(jum)
 	
procedure tulis(input: x: integer)
kamus
i : integer algoritma
i traversal [1..x] output("baris ke-",i+1)
	#include <conio.h> #include <stdlib.h>
	using namespace std;
	/*prototype fungsi */ void tulis(int x);
	int main()
	{
	System("cls");
	int jum;
	cout << " jumlah baris kata=";
	cin >> jum;
	tulis(jum); getche();
	return 0;
	}
	/*badan prosedur*/ void tulis(int x){
	for (int i=0;i<x;i++)
	cout<<"baris ke-"<<i+1<<endl;
	}

❖ Parameter Fungsi

➢ Paramater Formal dan Parameter Aktual

Parameter formal adalah variabel yang ada pada daftar paramerter ketika mendefinisikan fungsi. Pada fungsi maks3() contoh diatas, a, b dan merupakan parameter formal.

```
float perkalian (float x, float y) { return (x*y);
}
```

Pada contoh di atas x dan y adalah parameter formal.
Adapun parameter aktual adalah parameter (tidak selamanya menyatakan variabel) yang dipakai untuk memanggil fungsi.

```
X = perkalian(a, b);
Y = perkalian(10,30);
```

Dari pernyataan diatas a dan b merupakan parameter aktual, begitu pula 10 dan 30. parameter aktual tidak harus berupa variabel, melainkan bisa berupa konstanta atau ungkapan.

➢ Cara melewatkan Parameter

A. Pemanggilan dengan Nilai (call by value)

Pada pemanggilan dengan nilai, nilai dari parameter aktual akan disalin kedalam parameter formal, jadi parameter aktual tidak akan berubah meskipun parameter formalnya berubah. Untuk lebih jelasnya perhatikan contoh berikut:

Algoritma	C++

Program coba_parameter_by_value
Kamus
a,b : integer
procedure tukar(input: x,y : integer)
Algoritma
a 4
b 6 output(a,b) tukar(a,b)
output(a,b)
 	
procedure tukar(input:x,y : integer)
kamus
temp : integer
algoritma
temp x x y
y temp output(x,y)
	#include <iostream>
	#include <conio.h>
	#include <stdlib.h>
	using namespace std;
	/*prototype fungsi */ void tukar(int x, int y);
	int main () { int a, b;
	system("cls"); a=4; b=6;
	cout << "kondisi sebelum ditukar \n"); cout << " a = "<<a<<" b = "<<b<<endl;
	tukar(a,b);
	printf("kondisi setelah ditukar \n"); cout << " a = "<<a<<" b = "<<b<<endl;
	getch();
	return 0;
	}

	void tukar (int x, int y) {
	int temp;
	temp = x; x = y;
	y = temp;
	cout << "nilai akhir pada fungsi tukar \n"; cout << " x = "<<x<<" y = "<<y<<endl;
	}

Hasil eksekusi :
Kondisi sebelum tukar
a = 4, b = 6
Nilai akhir pada fungsi tukar
x = 6, y = 4
Kondisi setelah tukar
a = 4, b = 6
Jelas bahwa pada pemanggilan fungsi tukar, yang melewatkan variabel a dan b tidak merubah nilai dari variabel tersebut. Hal ini dikarenakan ketika pemanggilan fungsi tersebut nilai dari a dan b disalin ke variabel formal yaitu x dan y.

B. Pemanggilan dengan Pointer (call by pointer)

Pemanggilan dengan pointer merupakan cara untuk melewatkan alamat suatu variabel ke dalam suatu fungsi. Dengan cara ini dapat merubah nilai dari variabel aktual yang dilewatkan ke dalam fungsi. Jadi cara ini dapat merubah variabel yang ada diluar fungsi.
Cara penulisan :

```
tukar(int *px, int *py) {
int temp;
temp = *px;
*px = *py;
*py = temp;
… …
}
```

Cara pemanggilan:

```
tukar(&a, &b);
```

Pada ilustrasi tersebut, *px merupakan suatu variabel pointer yang menunjuk ke suatu variabel interger. Pada pemanggilan fungsi tukar(), &a dan &b menyatakan "alamat a" dan "alamat b". dengan cara diatas maka variabel yang diubah dalam fungsi tukar() adalah variabel yang dilewatkan dalam fungsi itu juga, karena yang dilewatkan dalam fungsi adalah alamat dari variabel tersebut, jadi bukan sekedar disalin.

C. Pemanggilan dengan Referensi (Call by Reference)

Pemanggilan dengan referensi merupakan cara untuk melewatkan alamat suatu variabel kedalam suatu fungsi. Dengan cara ini dapat merubah nilai dari variabel aktual yang dilewatkan ke dalam fungsi. Jadi cara ini dapat merubah variabel yang ada diluar fungsi.
Cara penulisan :

```
tukar(int &px, int &py) { int temp;
temp = px; px = py; py = temp;
… …
}
```

Algoritma	C++
Program coba_parameter_by_reference	#include <iostream> #include <conio.h>

Kamus
a,b : integer
procedure tukar(input/output: x,y
: integer)
 	
Algoritma
a ← 4
b ← 6 output(a,b)
tukar(a,b) output(a,b)
procedure tukar(input/output
:x,y : integer) kamus
temp : integer
algoritma
temp ← x
x ← y
y ← temp
output(x,y)
	#include <stdlib.h>
	
	using namespace std;
	
	/*prototype fungsi */
	void tukar(int &x, int &y);
	int main () {
	int a, b; system(“cls”); a=4;
	b=6;
	cout << “kondisi sebelum
	ditukar \n”); cout << ” a =
	“<<a<<” b = “<<b<<endl;
	tukar(a,b);
	printf(“kondisi setelah
	ditukar \n”); cout << ” a =
	“<<a<<” b = “<<b<<endl;
	getch();
	return 0;
	}
	
	void tukar (int &x, int &y) {
	int temp;
	temp = x; x = y;
	y = temp;
	cout<< “nilai akhir pada
	fungsi tukar \n”; cout << ” x
	= “<<x<<” y = “<<y<<endl;
	}

3. Guided

❖ Greeting

```
int penjumlahan(int a, int b){
    return a + b;
}

void greet(string name){
    cout<<"Halo "<<name<<"!"<<endl;
}
int main(){
    int hasil = penjumlahan(5, 3);
    cout<<"Hasil "<<hasil<<endl;

    greet("PANTEK!!!!!!!!!!!!");
```

1. Fungsi Penjumlahan

```
int penjumlahan(int a, int b){
    return a + b;
}
```

Fungsi ini bernama penjumlahan dan menerima dua parameter bertipe int, yaitu a dan b.
Fungsi ini mengembalikan hasil penjumlahan dari kedua parameter tersebut.

2. Fungsi Greet

```
void greet(string name){
    cout<<"Halo "<<name<<"!"<<endl;
}
```

Fungsi ini bernama greet dan menerima satu parameter bertipe string, yaitu name.
Fungsi ini tidak mengembalikan nilai (void), tetapi mencetak pesan sapaan ke layar, yang menggabungkan kata "Halo" dengan nama yang diberikan.

3. Fungsi Main

```
int main(){
    int hasil = penjumlahan(5, 3);
    cout<<"Hasil "<<hasil<<endl;

    greet("PANTEK!!!!!!!!!!!!!");
}
```

Ini adalah fungsi utama dari program, yang akan dijalankan saat program dieksekusi.
Di dalam fungsi ini, pertama-tama program memanggil fungsi penjumlahan dengan argumen 5 dan 3, dan hasilnya disimpan dalam variabel hasil.
Kemudian, program mencetak hasil penjumlahan tersebut ke layar dengan format "Hasil [nilai]".
Terakhir, program memanggil fungsi greet dengan argumen "PANTEK!!!!!!!!!!!!!", yang akan mencetak sapaan ke layar.

4. Output

```
Hasil 8
Halo PANTEK!!!!!!!!!!!!!
```

❖ Array 1 Dimensi

```
int nilai[5]={1,2,3,4,5};
cout << nilai[0]<< endl;
cout << nilai[1]<< endl;
cout << nilai[2]<< endl;
cout << nilai[3]<< endl;
cout << nilai[4]<< endl;
```

1. Deklarasi dan Inisialisasi Array

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

Di sini, sebuah array bernama nilai dideklarasikan dengan ukuran 5. Ini berarti array tersebut dapat menyimpan 5 elemen.
Tipe data array adalah int, sehingga setiap elemen dalam array adalah bilangan bulat.
Array diinisialisasi dengan lima nilai: 1, 2, 3, 4, dan 5. Elemen-elemen ini akan berada pada indeks 0 hingga 4 (indeks mulai dari 0 di C++).

2. Mencetak Elemen

```
cout << nilai[0] << endl;
cout << nilai[1] << endl;
cout << nilai[2] << endl;
cout << nilai[3] << endl;
cout << nilai[4] << endl;
```

Setiap baris cout mencetak elemen array nilai pada indeks yang berbeda.
nilai[0] akan mencetak 1, nilai[1] akan mencetak 2, nilai[2] akan mencetak 3, nilai[3] akan mencetak 4, dan nilai[4] akan mencetak 5.
endl digunakan untuk membuat baris baru setelah setiap elemen dicetak.

3. Output

```
1
2
3
4
5
```

❖ **Array 2 Dimensi**

```
   int nilai[3][4]={
       {1,2,3,4},
       {5,6,7,8},
       {9,10,11,12}
   };
   for(int i=0; i<3; i++){
       for(int j=0; j<4; j++){
           cout<<nilai[i][j]<<""<<endl;
       }
   }
   cout<<endl;
```

1. Deklarasi dan Inisialisasi Array Dua Dimensi

```
int nilai[3][4] = {
    {1, 2, 3, 4},
    {5, 6, 7, 8},
    {9, 10, 11, 12}
};
```

Deklarasi:
Array nilai dideklarasikan dengan 3 baris dan 4 kolom.
Artinya, array ini dapat menyimpan total 12 elemen.

Inisialisasi:
Elemen-elemen array diisi dengan nilai-nilai dari 1 hingga 12, terorganisir dalam 3 baris.
Misalnya, baris pertama berisi {1, 2, 3, 4}, baris kedua {5, 6, 7, 8}, dan baris ketiga {9, 10, 11, 12}.

2. Perulangan

```
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 4; j++) {
        cout << nilai[i][j] << "" << endl;
    }
}
cout << endl;
```

Looping:
Dua loop for digunakan di sini:
Loop luar (i) iterasi dari 0 hingga 2, yang mewakili indeks baris.
Loop dalam (j) iterasi dari 0 hingga 3, yang mewakili indeks kolom.

Mencetak Elemen:
Pada setiap iterasi dari loop dalam, elemen yang sesuai (nilai[i][j]) dicetak di layar, diikuti dengan endl, yang mencetak baris baru setelah setiap elemen.
Ini berarti setiap elemen dari array akan dicetak di baris yang terpisah.

3. Output

```
1
2
3
4
5
6
7
8
9
10
11
12
```

❖ **Pointer**

```
int x,y;
    int *px;
    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 px = "<<*px<<endl;
    cout<<"nilai y = "<<y<<endl;
    getch();
    return 0;
```

int x, y;: Mendeklarasikan dua variabel bertipe integer, yaitu x dan y.
int *px;: Mendeklarasikan sebuah pointer ke tipe integer dengan nama px. Tanda * menunjukkan bahwa px adalah sebuah pointer, artinya ia akan menyimpan alamat memori dari sebuah variabel bertipe integer.
x = 87;: Mengisi variabel x dengan nilai 87.
px = &x;: Simbol &x mendapatkan alamat memori dari variabel x. Alamat ini disimpan ke dalam pointer px, sehingga sekarang px menunjuk ke variabel x.
y = *px;: Simbol *px melakukan dereferensi, yaitu mengakses nilai yang ada di alamat memori yang ditunjuk oleh px. Karena px menunjuk ke x, maka nilai x (yaitu 87) disalin ke dalam variabel y.

Output:

```
cout << "Alamat x = " << &x << endl;
```

&x akan menghasilkan alamat memori di mana variabel x disimpan. Ini akan mencetak alamat memori dari x.

```
cout << "Isi px = " << px << endl;
```

px menyimpan alamat dari x, jadi ini akan mencetak alamat yang sama dengan hasil dari &x.

```
cout << "Isi x = " << x << endl;
```

x menyimpan nilai 87, sehingga ini akan mencetak nilai 87.

```
cout << "Nilai yang ditunjuk px = " << *px << endl;
```

*px adalah dereferensi dari pointer px, yang mengembalikan nilai yang disimpan di alamat yang ditunjuk oleh px (yang merupakan nilai x, yaitu 87). Jadi ini juga akan mencetak 87.

```
cout << "nilai y = " << y << endl;
```

y telah diberi nilai ***px**, yang sama dengan nilai x (yaitu 87). Jadi ini juga akan mencetak 87.

4. Unguided

1. Pemisah Ganjil-Genap

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<int> angka;
    vector<int> genap;
    vector<int> ganjil;
    int input;
```

```
    cout << "Masukkan angka (akhiri dengan angka negatif): ";

    while (true) {
        cin >> input;
        if (input < 0) {
            break;
        }
        angka.push_back(input);
    }

    for (int num : angka) {
        if (num % 2 == 0) {
            genap.push_back(num);
        } else {
            ganjil.push_back(num);
        }
    }

    cout << "Data Array: ";
    for (int num : angka) {
        cout << num << " ";
    }
    cout << endl;

    cout << "Nomor Genap: ";
    for (int num : genap) {
        cout << num << " ";
    }
    cout << endl;

    cout << "Nomor Ganjil: ";
    for (int num : ganjil) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

Output:

```
PS D:\Coding\C++\Pertemuan3\output> & .\'pisahganjilgenap.exe'
Masukkan angka (akhiri dengan angka negatif): 1 2 3 5 6 7 8 9 10 -1
Data Array: 1 2 3 5 6 7 8 9 10
Nomor Genap: 2 6 8 10
Nomor Ganjil: 1 3 5 7 9
PS D:\Coding\C++\Pertemuan3\output>
```

2. Array 3 Dimensi

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    int baris, kolom, kedalaman;

    cout << "Masukkan jumlah baris: ";
    cin >> baris;
    cout << "Masukkan jumlah kolom: ";
    cin >> kolom;
    cout << "Masukkan jumlah kedalaman: ";
    cin >> kedalaman;
```

```
    // Membuat array tiga dimensi menggunakan vector of vectors
    vector<vector<vector<int>>> array3D(baris,
vector<vector<int>>(kolom, vector<int>(kedalaman)));

    // Mengisi elemen-elemen array
    for (int i = 0; i < baris; i++) {
        for (int j = 0; j < kolom; j++) {
            for (int k = 0; k < kedalaman; k++) {
                cout << "Masukkan nilai untuk elemen [" << i << "][" <<
j << "][" << k << "]: ";
                cin >> array3D[i][j][k];
            }
        }
    }

    // Menampilkan isi array
    cout << "Isi array 3D:\n";
    for (int i = 0; i < baris; i++) {
        for (int j = 0; j < kolom; j++) {
            for (int k = 0; k < kedalaman; k++) {
                cout << array3D[i][j][k] << " ";
            }
            cout << endl;
        }
        cout << endl;
    }

    return 0;
}
```

Output:

```
PS D:\Coding\C++\Pertemuan3\output> & .\'array3D.exe'
Masukkan jumlah baris: 1
Masukkan jumlah kolom: 2
Masukkan jumlah kedalaman: 3
Masukkan nilai untuk elemen [0][0][0]: 1
Masukkan nilai untuk elemen [0][0][1]: 2
Masukkan nilai untuk elemen [0][0][2]: 3
Masukkan nilai untuk elemen [0][1][0]: 4
Masukkan nilai untuk elemen [0][1][1]: 5
Masukkan nilai untuk elemen [0][1][2]: 6
Isi array 3D:
1 2 3
4 5 6

PS D:\Coding\C++\Pertemuan3\output>
```

3. Searching Nilai Max-Min-Avg

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<int> angka;
```

```
    int pilihan, nilai, jumlah = 0;
    double rata_rata;

    do {
        cout << "\nMenu:\n";
        cout << "1. Masukkan angka\n";
        cout << "2. Tampilkan nilai maksimum\n";
        cout << "3. Tampilkan nilai minimum\n";
        cout << "4. Hitung rata-rata\n";
        cout << "5. Keluar\n";
        cout << "Masukkan pilihan: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                cout << "Masukkan angka (masukkan angka negatif untuk
berhenti): ";
                while (true) {
                    cin >> nilai;
                    if (nilai < 0) {
                        break;
                    }
                    angka.push_back(nilai);
                    jumlah += nilai;
                }
                break;
            case 2:
                if (angka.empty()) {
                    cout << "Array masih kosong.\n";
                } else {
                    int maks = angka[0];
                    for (int i = 1; i < angka.size(); i++) {
                        if (angka[i] > maks) {
                            maks = angka[i];
                        }
                    }
                    cout << "Nilai maksimum: " << maks << endl;
                }
                break;
            case 3:
                if (angka.empty()) {
                    cout << "Array masih kosong.\n";
                } else {
                    int min = angka[0];
                    for (int i = 1; i < angka.size(); i++) {
                        if (angka[i] < min) {
                            min = angka[i];
                        }
                    }
                    cout << "Nilai minimum: " << min << endl;
                }
                break;
            case 4:
                if (angka.empty()) {
                    cout << "Array masih kosong.\n";
                } else {
                    rata_rata = (double)jumlah / angka.size();
                    cout << "Rata-rata: " << rata_rata << endl;
                }
                break;
            case 5:
                cout << "Terima kasih!\n";
                break;
            default:
                cout << "Pilihan tidak valid.\n";
        }
```

```
    } while (pilihan != 5);

    return 0;
}
```

Output Inputan Array:

```
PS D:\Coding\C++\Pertemuan3\output> & .\'max-min-avg.exe'

Menu:
1. Masukkan angka
2. Tampilkan nilai maksimum
3. Tampilkan nilai minimum
4. Hitung rata-rata
5. Keluar
Masukkan pilihan: 1
Masukkan angka (masukkan angka negatif untuk berhenti): 1 2 3 4 5 6 7 8 9 10 -1
```

Output nilai Maximum:

```
Menu:
1. Masukkan angka
2. Tampilkan nilai maksimum
3. Tampilkan nilai minimum
4. Hitung rata-rata
5. Keluar
Masukkan pilihan: 2
Nilai maksimum: 10
```

Output nilai Minimum:

```
Menu:
1. Masukkan angka
2. Tampilkan nilai maksimum
3. Tampilkan nilai minimum
4. Hitung rata-rata
5. Keluar
Masukkan pilihan: 3
Nilai minimum: 1
```

Output nilai Rata-rata:

```
Menu:
1. Masukkan angka
2. Tampilkan nilai maksimum
3. Tampilkan nilai minimum
4. Hitung rata-rata
5. Keluar
Masukkan pilihan: 4
Rata-rata: 5.5
Nilai minimum: 1
```

Output Exit:

```
Menu:
1. Masukkan angka
1. Masukkan angka
2. Tampilkan nilai maksimum
3. Tampilkan nilai minimum
4. Hitung rata-rata
5. Keluar
Masukkan pilihan: 5
Terima kasih!
PS D:\Coding\C++\Pertemuan3\output>
```

5. **Kesimpulan**

Pointer, array, dan fungsi merupakan konsep fundamental dalam pemrograman C++ yang saling terkait. Pointer adalah variabel yang menyimpan alamat memori dari suatu variabel lain, memungkinkan kita untuk mengakses dan memanipulasi data secara langsung. Array adalah kumpulan data dengan tipe yang sama yang disimpan secara berurutan dalam memori, dan seringkali diakses menggunakan pointer. Fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu, dan dapat menerima parameter serta mengembalikan nilai. Kombinasi dari ketiganya memungkinkan kita membangun struktur data yang kompleks, melakukan operasi pada data secara efisien, dan membuat program yang lebih modular dan terorganisir. Pointer digunakan untuk mengakses elemen array secara fleksibel, memodifikasi nilai variabel dalam fungsi, dan membangun struktur data seperti linked list dan tree. Array menyediakan cara untuk menyimpan kumpulan data yang berukuran tetap, sementara fungsi memungkinkan kita untuk membagi program menjadi bagian-bagian yang lebih kecil dan dapat digunakan kembali. Dengan memahami konsep-konsep ini, kita dapat menulis program C++ yang lebih baik dan efisien.