

**LAPORAN PRAKTIKUM**  
**Modul 2**  
**“PENGENALAN BAHASA C++ (BAGIAN KEDUA)”**



**Disusun Oleh:**  
**Fahmi Hasan Asagaf -2311104074**  
**S1 SE 07 02**

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

**PROGRAM STUDI S1 SOFTWARE ENGINEERING**  
**FAKULTAS INFORMATIKA**  
**TELKOM UNIVERSITY**  
**PURWOKERTO**  
**2024**

## 1. Tujuan Praktikum

Memahami penggunaan pointer dan alamat memori

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

#### 2.1.1 Array Satu 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 elemen 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*/
```

#### 2.1.2 Array Dua 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			

### 2.1.3 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];
```

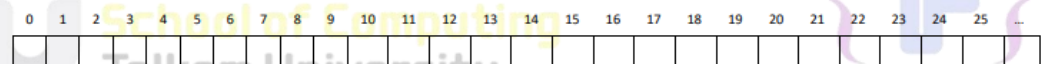
*Array* sebenarnya masih banyak pengembangannya untuk penyimpanan berbagai betuk data, pengembangan *array* misalnya untuk *array* tak berukuran.

## 2.2 Pointer

### 2.2.1 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*”

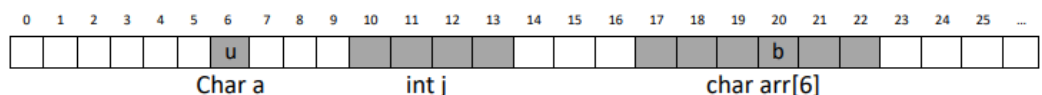
Saat program berjalan, Sistem Operasi (OS) akan mengalokasikan *space memory* untuk setiap variabel, objek, atau *array* yang kita buat. Lokasi pengalokasian memori bisa sangat teracak sesuai proses yang ada di dalam OS masing-masing. Perhatikan ilustrasi berikut



Gambar 2-2 Ilustrasi *Memory*

Digambarkan sebuah *memory* diasumsikan setiap *cell* menyimpan 1 *byte* data. Pada saat komputer pertama kali berjalan keadaan memori adalah kosong. Saat variabel dideklarasikan, OS akan mencari *cell* kosong untuk dialokasikan sebagai memori variabel tersebut.

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



Char a

int j

char arr[6]

Gambar 2-3 Ilustrasi Alokasi *Memory*

Pada contoh di atas variabel a dialokasikan di *memory* alamat x6, variabel j dialokasikan di alamat x10-13, dan variabel arr dialokasikan 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
<pre>Cout &lt;&lt; a &lt;&lt; endl; Cout &lt;&lt; &amp;a &lt;&lt; endl; Cout &lt;&lt; j &lt;&lt; endl; Cout &lt;&lt; &amp;j &lt;&lt; endl; Cout &lt;&lt; &amp;(arr[4]) &lt;&lt; endl;</pre>	<pre>'u' x6 0 x10 x21</pre>	<pre>Nilai variabel a Alamat variabel a Nilai variabel j Alamat variabel j Alamat variabel arr[4]</pre>

## 2.2.2 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;
```



Gambar 2-4 Ilustrasi Alokasi Pointer

C++	Output	Keterangan
<pre>int j,k; j =10; int *p_int; p_int = &amp;j; cout&lt;&lt; j &lt;&lt; endl; cout&lt;&lt; &amp;j &lt;&lt; endl; cout&lt;&lt; p_int &lt;&lt; endl; cout&lt;&lt; &amp;p_int &lt;&lt; endl; cout&lt;&lt; *p_int &lt;&lt; endl; k = *p_int; cout &lt;&lt; k &lt;&lt; endl;</pre>	<pre>10 X6 X6 X1 10 10</pre>	<pre>Nilai variabel j Alamat variabel j Nilai variabel p_int Alamat variabel p_int Nilai variabel yang ditunjuk p_int Nilai variabel k</pre>

Berikut ini contoh program sederhana menggunakan *pointer*:

```

1  #include <iostream>
2  #include <conio.h>
3  using namespace std;
4
5  int main() {
6      int x,y; //x dan y bertipe int
7      int *px; //px merupakan variabel pointer menunjuk ke variabel int
8      x =87;
9      px=&x;
10     y=*px;
11     cout<<"Alamat x= "<<&x<<endl;
12     cout<<"Isi px= "<<&px<<endl;
13     cout<<"Isi X= "<<x<<endl;
14     cout<<"Nilai yang ditunjuk px= "<<*px<<endl;
15     cout<<"Nilai y= "<<y<<endl;
16     getch();
17     return 0;
18 }

```

```

Alamat x= 0022FF14
Isi px= 0022FF14
Isi X= 87
Nilai yang ditunjuk px= 87
Nilai y= 87

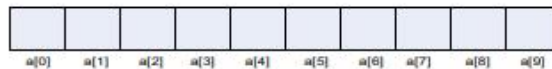
```

Gambar 2-5 Output Pointer

### 2.2.3 Pointer 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]`.



Gambar 2-6 Array

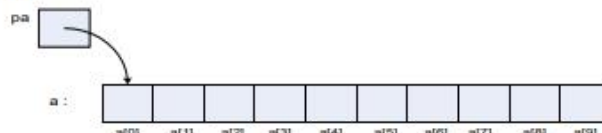
Notasi `a[i]` akan merujuk elemen ke-*i* dari *array*. Jika *pa* merupakan *pointer* yang menunjuk variabel bertipe *integer*, yang di deklarasikan sebagai berikut :

```
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  $a[i]$ .

```
1  #include <iostream>
2  #include <conio.h>
3  #define MAX 5
4  using namespace std;
5
6  int main() {
7      int i,j;
8      float nilai_total, rata_rata;
9      float nilai[MAX];
10     static int nilai_tahun[MAX][MAX]=
11     {
12         {0,2,2,0,0},
13         {0,1,1,1,0},
14         {0,3,3,3,0},
15         {4,4,0,0,4},
16         {5,0,0,0,5}
17     };
18     /*inisialisasi array dua dimensi */
19     for (i=0; i<MAX; i++){
20         cout<<"masukkan nilai ke-"<<i+1<<endl;
21         cin>>nilai[i];
22     }
23     cout<<"ndata nilai siswa : \n";
24     /*menampilkan array satu dimensi */
25     for (i=0; i<MAX; i++){
26         cout<<"nilai k-"<<i+1<<"=" <<nilai[i]<<endl;
27     }
28     cout<<"\n nilai tahunan : \n";
29
30     /* menampilkan array dua dimensi */
31     for(i=0; i<MAX; i++){
32         for(j=0; j<MAX; j++){
33             cout<<nilai_tahun[i][j];
34         }
35         cout<<"\n";
36     }
37     getch();
38     return 0;
39 }
```



## 2.2.4 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 → menyatakan jumlah maksimal karakter dalam *string*.

Memasukkan data *string* dari keyboard:

```
gets(nama_array);
```

contoh: `gets (nama) ;`

jika menggunakan `cin ()` :

contoh: `cin>>nama;`

Inisialisasi *string*:

```
char nama[] = {'s','t','r','u','k','d','a','t','\0'};
```

Merupakan variabel nama dengan isi data *string* "strukdat".

Bentuk inisialisasi yang lebih singkat:

```
char nama[] = "strukdat";
```

Menampilkan *string* bisa menggunakan `puts ()` atau `cout ()` :

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

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

Contoh :

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

## B. Pointer dan String

Sesuai dengan penjelasan di atas , misalkan ada *string* :

"I am 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 :

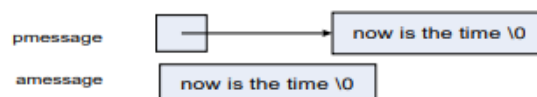
```
pmessage = "now is the time";
```

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*.



## 2.3 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	<pre>#include &lt;conio.h&gt; #include &lt;iostream&gt; #include &lt;stdlib.h&gt; using namespace std; int maks3(int a, int b, int c); /*mendeklarasikan prototype fungsi */ int main() {     system("cls");     int x,y,z;     cout&lt;&lt;"masukkan nilai bilangan ke-1     =";     cin&gt;&gt;x;     cout&lt;&lt;"masukkan nilai bilangan ke-2     =";     cin&gt;&gt;y;     cout&lt;&lt;"masukkan nilai bilangan ke-3     =";     cin&gt;&gt;z;     cout&lt;&lt;"nilai maksimumnya adalah ="     &lt;&lt;maks3(x,y,z);     getch();     return 0; } /*badan fungsi */ int maks3(int a, int b, int c){     /* deklarasi variabel lokal dalam     fungsi */     Int temp_max =a;     if(b&gt;temp_max)         temp_max=b;     if(c&gt;temp_max)         temp_max=c;     return (temp_max); }</pre>



## 2.4 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_procedure 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)	<pre>#include &lt;iostream&gt; #include &lt;conio.h&gt; #include &lt;stdlib.h&gt;  using namespace std; /*prototype fungsi */ void tulis(int x); int main() {     System("cls");     int jum;     cout &lt;&lt; " jumlah baris kata=";     cin &gt;&gt; jum;     tulis(jum);     getch();     return 0; } /*badan prosedur*/ void tulis(int x){     for (int i=0;i&lt;x;i++)         cout&lt;&lt;"baris ke-"&lt;&lt;i+1&lt;&lt;endl; }</pre>

## 2.5 Parameter Fungsi

### 2.5.1 Parameter Formal dan Parameter Aktual

Parameter formal adalah variabel yang ada pada daftar parameter 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.

## 2.5.2 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)	<pre> #include &lt;iostream&gt; #include &lt;conio.h&gt; #include &lt;stdlib.h&gt;  using namespace std; /*prototype fungsi */ void tukar(int x, int y);  int main () {     int a, b;     system("cls");     a=4; b=6;     cout &lt;&lt; "kondisi sebelum ditukar \n";     cout &lt;&lt; " a = "&lt;&lt;a&lt;&lt;" b = "&lt;&lt;b&lt;&lt;endl;     tukar(a,b);     printf("kondisi setelah ditukar \n");     cout &lt;&lt; " a = "&lt;&lt;a&lt;&lt;" b = "&lt;&lt;b&lt;&lt;endl;     getch();     return 0; }  void tukar (int x, int y) {     int temp;     temp = x;     x = y;     y = temp;     cout &lt;&lt; "nilai akhir pada fungsi tukar \n";     cout &lt;&lt; " x = "&lt;&lt;x&lt;&lt;" y = "&lt;&lt;y&lt;&lt;endl; } </pre>

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 *integer*. 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;
    ... ..
}

```

Cara pemanggilan:

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

untuk melewatkan nilai dengan referensi, argumen dilalui ke fungsi seperti nilai lain. Jadi pada akhirnya, harus mendeklarasikan di parameter awal, serta untuk pemanggilan tidak perlu menggunakan paramter tambahan seperti pada *call by pointer*.

Algoritma	C++
Program coba_parameter_by_reference 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)	<pre> #include &lt;iostream&gt; #include &lt;conio.h&gt; #include &lt;stdlib.h&gt;  using namespace std; /*prototype fungsi */ void tukar(int &amp;x, int &amp;y); int main () {     int a, b;     system("cls");     a=4; b=6;     cout &lt;&lt; "kondisi sebelum ditukar \n";     cout &lt;&lt; " a = "&lt;&lt;a&lt;&lt;" b = "&lt;&lt;b&lt;&lt;endl;     tukar(a,b);     printf("kondisi setelah ditukar \n");     cout &lt;&lt; " a = "&lt;&lt;a&lt;&lt;" b = "&lt;&lt;b&lt;&lt;endl;     getch();     return 0; }  void tukar (int &amp;x, int &amp;y) {     int temp;     temp = x;     x = y;     y = temp;     cout&lt;&lt; "nilai akhir pada fungsi tukar \n";     cout &lt;&lt; " x = "&lt;&lt;x&lt;&lt;" y = "&lt;&lt;y&lt;&lt;endl; </pre>

Call By  
Reference

	}
--	---

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

Call by  
Pointer



### 3. Guided

#### Array 1 dimensi

```
10 //array 1 dimensi
11 int nilai[5]={1,2,3,4,5};
12
13 for(int i=0; i<5; i++){
14     cout << nilai[i] << endl;
15 }
16 }
```

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

- Deklarasi Array: Ini membuat sebuah array (deretan) dengan nama nilai yang dapat menampung 5 buah bilangan bulat (tipe data int).
- Inisialisasi: Segera setelah dibuat, array nilai langsung diisi dengan nilai-nilai 1, 2, 3, 4, dan 5.

Baris 13-15:

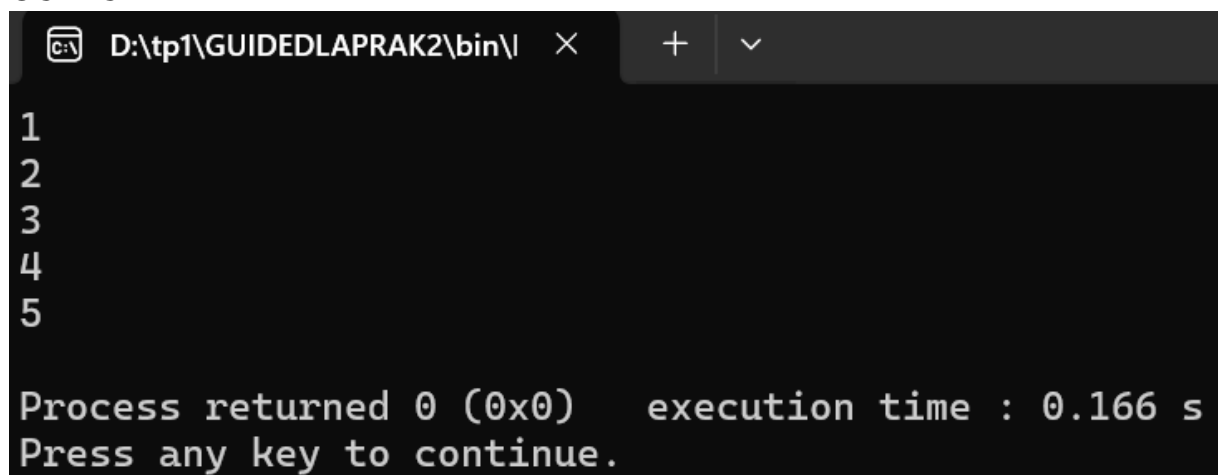
`for (int i = 0; i < 5; i++) {`

- Perulangan: Ini adalah sebuah perulangan for yang akan menjalankan kode di dalamnya sebanyak 5 kali.
- i adalah sebuah variabel yang akan digunakan sebagai indeks untuk mengakses elemen-elemen dalam array. Nilai i akan dimulai dari 0 dan bertambah 1 setiap kali perulangan.

`cout << nilai[i] << endl;`

- Mencetak ke Layar: Setiap kali perulangan, kode ini akan mencetak nilai dari elemen array nilai pada indeks ke-i ke layar, diikuti dengan pindah ke baris baru (endl).

#### OUTPUT



```
D:\tp1\GUIDEDLAPRAK2\bin\I
1
2
3
4
5
Process returned 0 (0x0)   execution time : 0.166 s
Press any key to continue.
```



## Array 2 dimensi

```
19
20 //array 2 dimensi
21
22 int nilai[3][4] = {
23     {1,2,3,4},
24     {5,6,7,8},
25     {9,10,11,12}
26 };
27
28 for (int i=0; i<3; i++) {
29     for (int j=0; j<4; j++) {
30         cout << nilai[i][j] << " ";
31     }
32     cout << endl;
33 }
34
```

```
int nilai[3][4] = {...};
```

- **Deklarasi Array 2D:** Ini membuat sebuah array dua dimensi (matriks) dengan nama nilai yang memiliki 3 baris dan 4 kolom. Setiap elemen dalam array ini bertipe bilangan bulat (int).
- **Inisialisasi:** Array langsung diisi dengan nilai-nilai yang diberikan dalam kurung kurawal. Setiap baris mewakili satu baris dalam matriks.

```
for (int i=0; i<3; i++) {
```

- **Perulangan Baris:** Perulangan ini akan mengulang setiap baris dalam matriks. Variabel i akan digunakan sebagai indeks untuk mengakses baris.

```
for (int j=0; j<4; j++) {
```

- **Perulangan Kolom:** Di dalam setiap iterasi baris, perulangan ini akan mengulang setiap kolom dalam baris tersebut. Variabel j akan digunakan sebagai indeks untuk mengakses kolom.

```
cout << nilai[i][j] << " ";
```

- **Mencetak Elemen:** Kode ini akan mencetak nilai elemen pada baris ke-i dan kolom ke-j ke layar, diikuti dengan spasi.

```
cout << endl;
```

- **Pindah Baris:** Setelah semua elemen dalam satu baris dicetak, kode ini akan pindah ke baris baru

## output

```
D:\tp1\GUIDEDLAPRAK2\bin\l  X  +  v
1 2 3 4
5 6 7 8
9 10 11 12

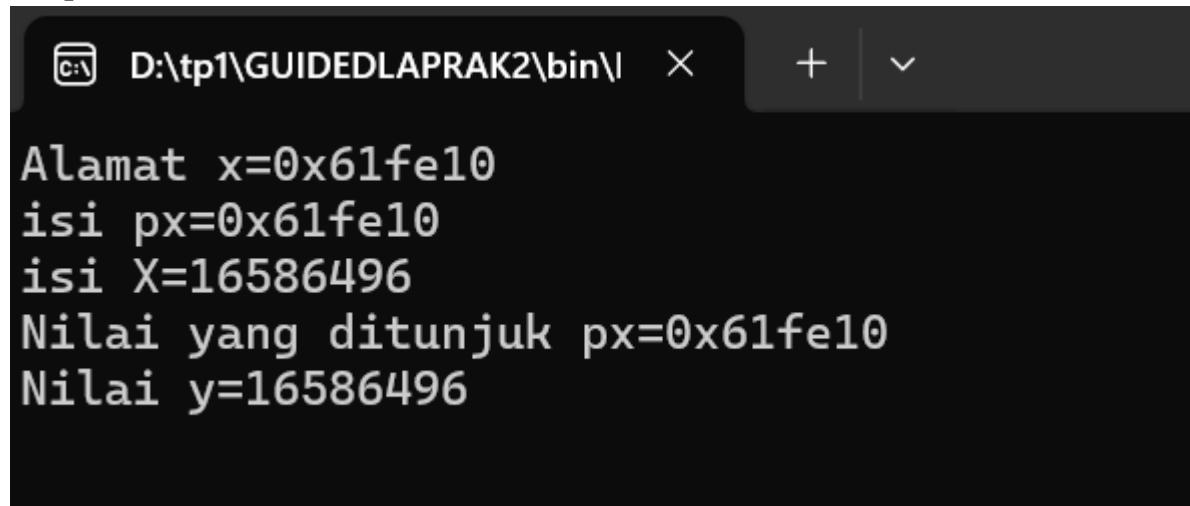
Process returned 0 (0x0)    execution time : 0.184 s
Press any key to continue.
|
```

## Pointer

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

cout << "Alamat x=" << &x <<endl;
cout << "isi px=" << px <<endl;
cout << "isi X=" << x <<endl;
cout << "Nilai yang ditunjuk px=" << &x <<endl;
cout << "Nilai y=" << y <<endl;
getch();
}
```

Kode C++ ini menjelaskan konsep pointer. Pointer px berfungsi sebagai petunjuk alamat ke variabel x. Dengan kata lain, px menunjuk ke lokasi di memori komputer tempat nilai x disimpan. Nilai yang ditunjuk oleh px kemudian disalin ke variabel y. Jadi, pada dasarnya, kode ini menunjukkan bagaimana kita bisa menggunakan pointer untuk mengakses dan memanipulasi data secara tidak langsung. Ini mirip seperti memiliki sebuah peta (pointer) yang menunjuk ke sebuah rumah (variabel). Dengan peta itu, kita bisa mengetahui lokasi rumah dan apa yang ada di dalamnya tanpa harus mencari-cari rumah tersebut secara langsung.



```
D:\tp1\GUIDEDLAPRAK2\bin\l  X  +  v

Alamat x=0x61fe10
isi px=0x61fe10
isi X=16586496
Nilai yang ditunjuk px=0x61fe10
Nilai y=16586496
```

### Fungsi dan Pointer

```
//fungsi dan prosedur
int penjumlahan(int a, int b){
    return a + b;
}

void greet(string name){
    cout << "Hello," << name << "!" << endl;
}

int main(){
    int hasil = penjumlahan(5,3);

    cout << "hasil " << hasil << endl;

    greet("Fahmi");
}
}
```

1. Fungsi penjumlahan(5,3) akan mengembalikan hasil dari 5 + 3, yaitu 8.
2. Fungsi greet("Fahmi") akan mencetak Hello, Fahmi!.

## Output

```
hasil 8  
Hello, Fahmi!
```

## 4. Unguided

### Soal 1

```
1  #include <iostream>  
2  #include <vector>  
3  
4  using namespace std;  
5  
6  int main() {  
7      int n;  
8  
9      cout << "Masukkan jumlah elemen dalam array: ";  
10     cin >> n;  
11  
12     vector<int> arr(n);  
13     cout << "Masukkan elemen array: ";  
14     for (int i = 0; i < n; i++) {  
15         cin >> arr[i];  
16     }  
17  
18     cout << "Data Array: ";  
19     for (int i = 0; i < n; i++) {  
20         cout << arr[i] << " ";  
21     }  
22     cout << endl;  
23  
24     cout << "Nomor Genap: ";  
25     for (int i = 0; i < n; i++) {  
26         if (arr[i] % 2 == 0) {  
27             cout << arr[i] << " ";  
28         }  
29     }  
30     cout << endl;  
31  
32     cout << "Nomor Ganjil: ";  
33     for (int i = 0; i < n; i++) {  
34         if (arr[i] % 2 != 0) {  
35             cout << arr[i] << " ";  
36         }  
37     }  
38     cout << endl;  
39  
40     return 0;  
41 }  
42  
43
```

## Output

```
"C:\codeblocks file\soal1\bin\Debug\soal1.exe"

Masukkan jumlah elemen dalam array: 10
Masukkan elemen array: 1 2 3 4 5 6 7 8 9 10
Data Array: 1 2 3 4 5 6 7 8 9 10
Nomor Genap: 2, 4, 6, 8, 10,
Nomor Ganjil: 1, 3, 5, 7, 9,

Process returned 0 (0x0)   execution time : 9.827 s
Press any key to continue.
```

## Soal 2

```
*main.cpp x main.cpp x main.cpp x
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int x, y, z;
6
7      // Meminta input ukuran dimensi array
8      cout << "Masukkan ukuran dimensi pertama (x): ";
9      cin >> x;
10     cout << "Masukkan ukuran dimensi kedua (y): ";
11     cin >> y;
12     cout << "Masukkan ukuran dimensi ketiga (z): ";
13     cin >> z;
14
15     // Mendeklarasikan array tiga dimensi
16     int array3D[x][y][z];
17
18     // Mengisi array tiga dimensi dengan nilai urut atau input dari user
19     cout << "Masukkan nilai untuk setiap elemen array:\n";
20     for (int i = 0; i < x; i++) {
21         for (int j = 0; j < y; j++) {
22             for (int k = 0; k < z; k++) {
23                 cout << "Nilai untuk array[" << i << "][" << j << "][" << k << "]: ";
24                 cin >> array3D[i][j][k];
25             }
26         }
27     }
28
29     // Menampilkan isi array tiga dimensi
30     cout << "\nArray Tiga Dimensi:\n";
31     for (int i = 0; i < x; i++) {
32         for (int j = 0; j < y; j++) {
33             for (int k = 0; k < z; k++) {
34                 cout << "array[" << i << "][" << j << "][" << k << "] = " << array3D[i][j][k] << endl;
35             }
36         }
37     }
38
39     return 0;
40 }
41
```



## Output

```
"C:\codeblocks file\soal2laprak2\bin\Debug\soal2laprak2.exe"
Masukkan ukuran dimensi pertama (x): 2
Masukkan ukuran dimensi kedua (y): 2
Masukkan ukuran dimensi ketiga (z): 2
Masukkan nilai untuk setiap elemen array:
Nilai untuk array[0][0][0] : 1
Nilai untuk array[0][0][1] : 2
Nilai untuk array[0][1][0] : 3
Nilai untuk array[0][1][1] : 4
Nilai untuk array[1][0][0] : 5
Nilai untuk array[1][0][1] : 6
Nilai untuk array[1][1][0] : 7
Nilai untuk array[1][1][1] : 8

Array Tiga Dimensi:
array[0][0][0] = 1
array[0][0][1] = 2
array[0][1][0] = 3
array[0][1][1] = 4
array[1][0][0] = 5
array[1][0][1] = 6
array[1][1][0] = 7
array[1][1][1] = 8

Process returned 0 (0x0)   execution time : 19.436 s
Press any key to continue.
```

## Soal 3

```
*main.cpp X main.cpp X main.cpp X

1  #include <iostream>
2  using namespace std;
3
4  // Fungsi untuk mencari nilai maksimum
5  int cariMaksimum(int array[], int n) {
6      int maksimum = array[0];
7      for (int i = 1; i < n; i++) {
8          if (array[i] > maksimum) {
9              maksimum = array[i];
10         }
11     }
12     return maksimum;
13 }
14
15 // Fungsi untuk mencari nilai minimum
16 int cariMinimum(int array[], int n) {
17     int minimum = array[0];
18     for (int i = 1; i < n; i++) {
19         if (array[i] < minimum) {
20             minimum = array[i];
21         }
22     }
23     return minimum;
24 }
25
26 // Fungsi untuk mencari nilai rata-rata
27 float cariRataRata(int array[], int n) {
28     int jumlah = 0;
29     for (int i = 0; i < n; i++) {
30         jumlah += array[i];
31     }
32     return (float)jumlah / n;
33 }
34
35 int main() {
36     int n;
37
38     // Meminta input jumlah elemen array
39     cout << "Masukkan jumlah elemen array: ";
40     cin >> n;
41
42     int array[n];
43
44     // Meminta input nilai dari pengguna untuk setiap elemen array
45     cout << "Masukkan nilai elemen array:\n";
46     for (int i = 0; i < n; i++) {
47         cout << "Elemen " << i + 1 << ": ";
48         cin >> array[i];
49     }
50
51     int pilihan;
52     do {
53         // Menampilkan menu pilihan
54         cout << "\nPilih Operasi:\n";
55         cout << "1. Cari Nilai Maksimum\n";
56         cout << "2. Cari Nilai Minimum\n";
57         cout << "3. Cari Nilai Rata-rata\n";
58         cout << "4. Keluar\n";
59         cout << "Masukkan pilihan Anda (1-4): ";
60         cin >> pilihan;
61
62         // Switch case berdasarkan pilihan pengguna
63         switch (pilihan) {
64             case 1:
65                 cout << "Nilai Maksimum: " << cariMaksimum(array, n) << endl;
66                 break;
67             case 2:
68                 cout << "Nilai Minimum: " << cariMinimum(array, n) << endl;
69                 break;
70             case 3:
71                 cout << "Nilai Rata-rata: " << cariRataRata(array, n) << endl;
72                 break;
73             case 4:
74                 cout << "Keluar dari program.\n";
75                 break;
76             default:
77                 cout << "Pilihan tidak valid! Silakan pilih antara 1-4.\n";
78                 break;
79         }
80     } while (pilihan != 4);
81
82     return 0;
83 }
84
```

## Output

```
"C:\codeblocks file\soal 3\bin\Debug\soal 3.exe"
Masukkan jumlah elemen array: 5
Masukkan nilai elemen array:
Elemen 1: 12
Elemen 2: 8
Elemen 3: 15
Elemen 4: 10
Elemen 5: 9

Pilih Operasi:
1. Cari Nilai Maksimum
2. Cari Nilai Minimum
3. Cari Nilai Rata-rata
4. Keluar
Masukkan pilihan Anda (1-4): 1
Nilai Maksimum: 15

Pilih Operasi:
1. Cari Nilai Maksimum
2. Cari Nilai Minimum
3. Cari Nilai Rata-rata
4. Keluar
Masukkan pilihan Anda (1-4): 2
Nilai Minimum: 8

Pilih Operasi:
1. Cari Nilai Maksimum
2. Cari Nilai Minimum
3. Cari Nilai Rata-rata
4. Keluar
Masukkan pilihan Anda (1-4): 3
Nilai Rata-rata: 10.8

Pilih Operasi:
1. Cari Nilai Maksimum
2. Cari Nilai Minimum
3. Cari Nilai Rata-rata
4. Keluar
Masukkan pilihan Anda (1-4): 4
Keluar dari program.

Process returned 0 (0x0)   execution time : 74.008 s
Press any key to continue.
```

## **5. Kesimpulan**

Kesimpulan dari tujuan praktikum ini adalah untuk memahami cara kerja pointer dan alamat memori, serta bagaimana menggunakannya dalam program. Selain itu, tujuan lain adalah mengimplementasikan fungsi dan prosedur untuk membuat program lebih modular dan efisien.