

LAPORAN PRAKTIKUM
PERTEMUAN 2
PENGENALAN BAHASA C++(BAGIAN 2)



Nama :

Alvin Bagus Firmansyah - 2311104070

Dosen :

Wahyu Andi Saputra, S.Pd, M.Eng

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024

I. TUJUAN

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

II. TOOL

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

III. LANDASAN TEORI

2.1 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*/
```

IV. GUIDED

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:

```
type_data nama_var[ukuran1][ukuran2]...[ukuran-N];
```

Contoh:

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

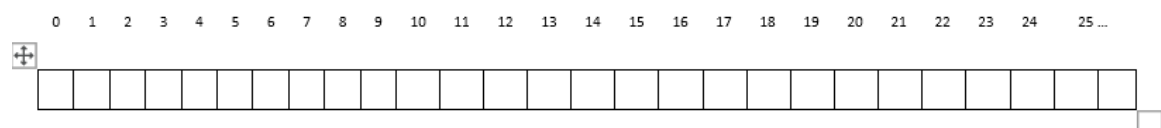
Array sebenarnya masih banyak pengembangannya untuk penyimpanan berbagai betuk data, pengembangan *array* misalnya untuk *array* tak berukuran.

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



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'
```



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
<code>Cout << a << endl;</code>	'u'	Nilai variabel a
<code>Cout << &a << endl;</code>	x6	Alamat variabel a
<code>Cout << j << endl;</code>	0	Nilai variabel j
<code>Cout << &j << endl;</code>	X10	Alamat variabel j
<code>Cout << &(arr[4]) << endl;</code>	X21	Alamat variabel arr[4]

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



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

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 }

```

```

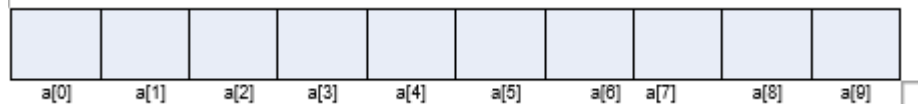
Alamat x= 0022FF14
Isi px= 0022FF14
Isi X= 87
Nilai yang ditunjuk px= 87
Nilai y= 87

```

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]`.



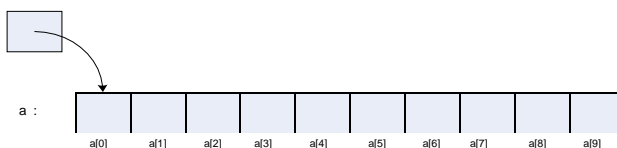
Notasi `a[i]` akan merujuk elemen ke-*i* dari *array*. Jika *pa* merupakan *pointer* yang menunjuk variabel bertipe *integer*, yang di deklarasikan sebagai berikut :

`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
7  int main() {
8      int i, j;
9      float nilai_total, rata_rata;
10     float nilai[MAX];
11     static int nilai_tahun[MAX][MAX] =
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];

```

```

21     }
22
23
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 }

```

Fakultas Informatika
School of Computing
Telkom University



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* seperti halnya 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*.

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 :

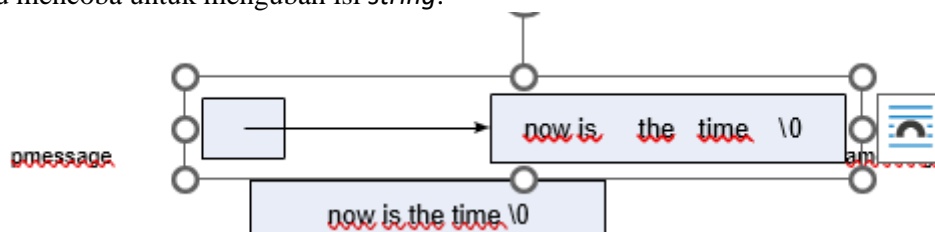
```
pmessage = "now is the time";
```

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++
<p>Program <u>coba fungsi</u></p> <p>Kamus</p> <p><u>x,y,z</u> : integer</p> <p><u>function</u> <u>max3</u>(<u>input</u>: <u>a,b,c</u> : integer) : integer</p> <p>Algoritma</p> <p><u>input</u>(<u>x,y,z</u>) <u>output</u>(<u>max3</u>(<u>x,y,z</u>))</p> <p><u>function</u> <u>max3</u>(<u>input</u>:<u>a,b,c</u> : integer) : integer kamus <u>temp_max</u> : integer <u>algoritma</u> <u>temp_max</u> ← <u>a</u> <u>if</u> (<u>b</u>><u>temp_max</u>) <u>then</u></p>	<pre>#include <conio.h> #include <iostream> #include <stdlib.h> using namespace std; int maks3(int a, int b, int c); /*mendeklarasikan prototype fungsi */ int m a i n () { a y a t e m (" C l e ") ; i n t x y z ; cout<<"masukkan nilai bilangan ke-1 = " ; C i n g > > x ; cout<<"masukkan nilai bilangan ke-2 = " ; C</pre>

```

temp_max ← b
if (c > temp_max) then temp_max ← c
temp_max

```

```

i
p
>
>
y
;
cout<<"masukkan nilai
bilangan ke-3 =";
cin>>z;
cout<<"nilai
maksimumnya adalah ="
<
<maks
3(x,y
,z);
getch
e();
retur
n 0;
}
/*badan fungsi */
int maks3(int a, int
b, int c){ /*
deklarasi variabel
lokal dalam fungsi
*/
Int temp_max
=a;
if (b>temp_ma
x)
temp_max=b;
if (c>temp_ma
x)
temp_max
=c;
return (temp_max);
}

```

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_procedur Kamus jum : integer procedure tulis(input: x: integer)	<pre> #include <iostream> #include <conio.h> #include <stdlib.h> using names pace std; /*pro totyp e fungs i */ </pre>
Algoritma input(jum) tulis(jum) procedure tulis(input: x: integer) kamus i : integer algoritma i traversal [1..x] output("baris ke-",i+1)	<pre> void tulis (int x); int main() { Sy st em (" cl s"); in t ju m; cout << " jumlah baris kata="; cin >> jum; tulis (jum) ; getch e(); return 0; } /*badan prosedur*/ void tulis(int x){ for (int i=0;i<x;i++) cout<<"baris ke- "<<i+1<<endl; } </pre>

PRATIKUM GUIDED

1.

```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6
7      int nilai[5]={1,2,3,4,5};
8      cout << nilai[0];
9      cout << nilai[1];
10     cout << nilai[2];
11     cout << nilai[3];
12     cout << nilai[4];
13
14
15 }
16
```

Hasilnya;

```
"C:\Users\alvin\OneDrive\Do... x + v
12345
Process returned 0 (0x0)   execution time : 1.703 s
Press any key to continue.
```

2.

```
21
22  #include <iostream> // Include iostream for cout
23  using namespace std; // Allows using cout without std::
24
25  int main() {
26      // Declare array once
27      int nilai[5] = {1, 2, 3, 4, 5};
28
29      // Loop through the array and print values
30      for (int i = 0; i < 5; i++) {
31          cout << nilai[i] << endl;
32      }
33
34      return 0; // Indicate successful program termination
35  }
36
```

Hasilnya:

```
"C:\Users\alvin\OneDrive\Do... x + v
1
2
3
4
5

Process returned 0 (0x0)    execution time : 1.088 s
Press any key to continue.
|
```

3.

```
44
45  #include <iostream> // Include iostream for cout
46  using namespace std; // Allows us to use cout without std::
47
48  int main() {
49      // 2D array initialization
50      int nilai[3][4] = {
51          {1, 2, 3, 4},
52          {5, 6, 7, 8},
53          {9, 10, 11, 12}
54      };
55
56      // Loop through the rows
57      for (int i = 0; i < 3; i++) {
58          // Loop through the columns
59          for (int j = 0; j < 4; j++) {
60              cout << nilai[i][j] << " "; // Print the elements
61          }
62          cout << endl; // New line after each row
63      }
64
65      return 0; // Indicate successful program termination
66  }
67
68
```

Hasilnya:

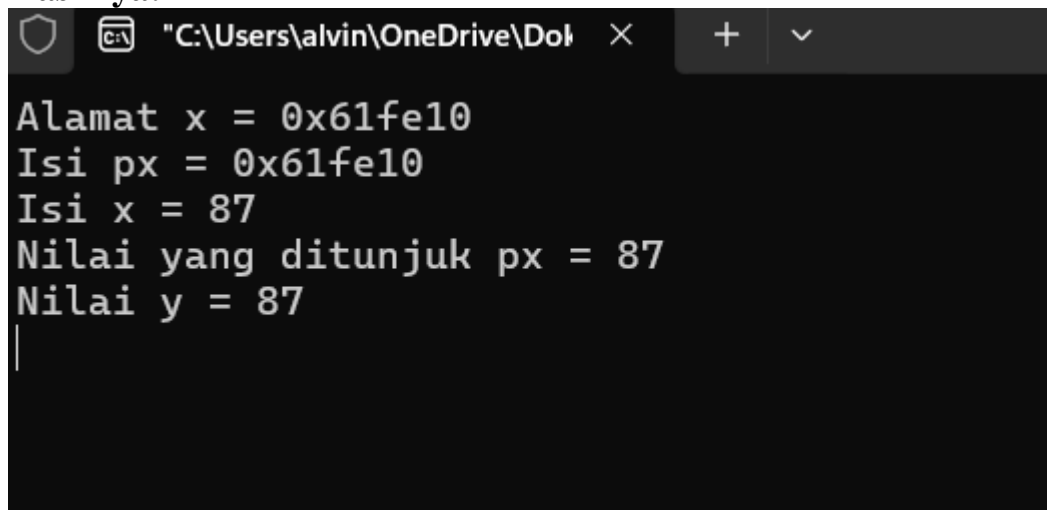
```
"C:\Users\alvin\OneDrive\Do... x + v
1 2 3 4
5 6 7 8
9 10 11 12

Process returned 0 (0x0)    execution time : 0.819 s
Press any key to continue.
|
```

4.

```
78 #include <iostream> // Include iostream for cout
79 using namespace std;
80
81 int main() {
82     int x, y;
83     int *px;
84
85     x = 87;
86     px = &x;
87
88     y = *px;
89     // Print the results
90     cout << "Alamat x = " << &x << endl;
91     cout << "Isi px = " << px << endl;
92     cout << "Isi x = " << x << endl;
93     cout << "Nilai yang ditunjuk px = " << *px << endl;
94     cout << "Nilai y = " << y << endl;
95
96     cin.get();
97     return 0;
98 }
99
100
```

Hasilnya:



```
Alamat x = 0x61fe10
Isi px = 0x61fe10
Isi x = 87
Nilai yang ditunjuk px = 87
Nilai y = 87
|
```

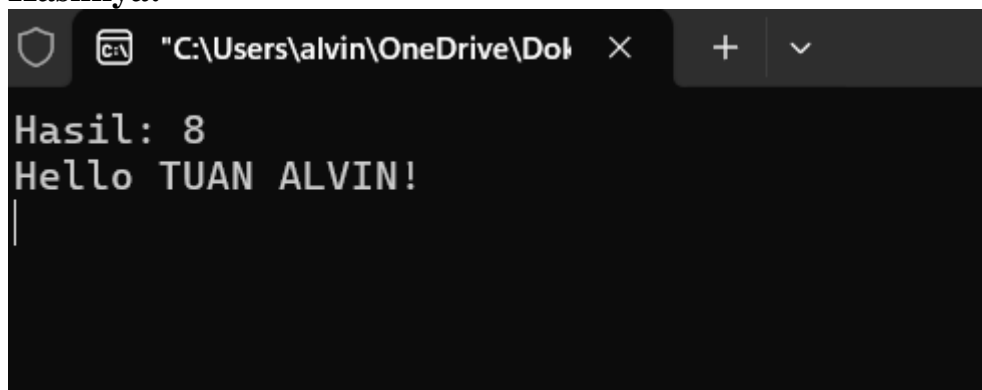
5.

```

102  #include <iostream>
103  using namespace std;
104
105  int penjumlahan(int a, int b) {
106      return a + b;
107  }
108
109  void greet(string name) {
110      cout << "Hello " << name << "!" << endl;
111  }
112
113  int main() {
114      int hasil = penjumlahan(5, 3);
115      cout << "Hasil: " << hasil << endl;
116
117      greet("TUAN ALVIN");
118
119      cin.get();
120
121      return 0; // Baris ini sudah benar
122  }
123
124
125

```

Hasilnya:



```

Hasil: 8
Hello TUAN ALVIN!

```

V. UNGUIDED

1. Buatlah program untuk menampilkan Output seperti berikut dengan data yang diinputkan oleh user!

```

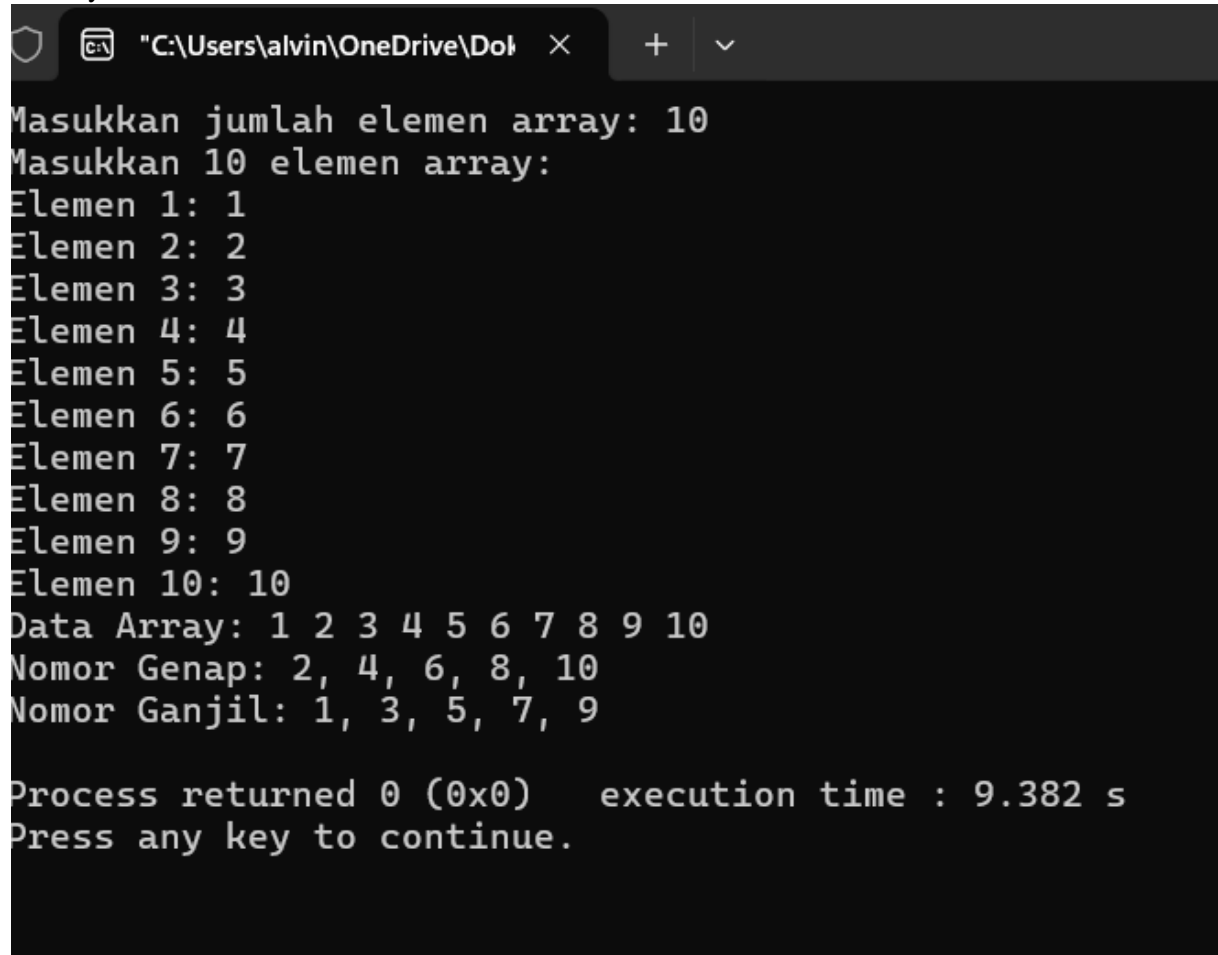
Data Array : 1 2 3 4 5 6 7 8 9 10
Nomor Genap : 2, 4, 6, 8, 10,
Nomor Ganjil : 1, 3, 5, 7, 9,

```

Jawabanya:

```
main.cpp X
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 int main() {
7     int n;
8
9     // Meminta pengguna untuk memasukkan jumlah elemen array
10    cout << "Masukkan jumlah elemen array: ";
11    cin >> n;
12
13    vector<int> dataArray(n); // Membuat array dinamis
14
15    // Meminta input untuk setiap elemen array
16    cout << "Masukkan " << n << " elemen array: " << endl;
17    for (int i = 0; i < n; i++) {
18        cout << "Elemen " << (i + 1) << ": ";
19        cin >> dataArray[i];
20    }
21
22    // Menampilkan Data Array
23    cout << "Data Array: ";
24    for (int i = 0; i < n; i++) {
25        cout << dataArray[i] << " ";
26    }
27    cout << endl;
28
29    // Menampilkan Nomor Genap
30    cout << "Nomor Genap: ";
31    bool isFirst = true; // Untuk menghindari koma di awal
32    for (int i = 0; i < n; i++) {
33        if (dataArray[i] % 2 == 0) {
34            if (!isFirst) {
35                cout << ", ";
36            }
37            cout << dataArray[i];
38            isFirst = false;
39        }
40    }
```

Hasilnya:



```
"C:\Users\alvin\OneDrive\Do... X + v
Masukkan jumlah elemen array: 10
Masukkan 10 elemen array:
Elemen 1: 1
Elemen 2: 2
Elemen 3: 3
Elemen 4: 4
Elemen 5: 5
Elemen 6: 6
Elemen 7: 7
Elemen 8: 8
Elemen 9: 9
Elemen 10: 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 : 9.382 s
Press any key to continue.
```

3. Buatlah program Input array tiga dimensi tetapi jumlah atau ukuran elemennya diinputkan oleh user!

Jawabanya:


```
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 array dari pengguna
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    // Membuat array 3 dimensi secara dinamis
16    int ***array_3d = new int**[x];
17    for (int i = 0; i < x; i++) {
18        array_3d[i] = new int*[y];
19        for (int j = 0; j < y; j++) {
20            array_3d[i][j] = new int[z];
21        }
22    }
23
24    // Meminta input nilai untuk setiap elemen array
25    for (int i = 0; i < x; i++) {
26        for (int j = 0; j < y; j++) {
27            for (int k = 0; k < z; k++) {
28                cout << "Masukkan nilai untuk elemen [" << i << "][" << j << "][" << k << "]: ";
29                cin >> array_3d[i][j][k];
30            }
31        }
32    }
33
34    // Menampilkan array 3D yang diinputkan
35    cout << "\nArray 3D yang diinputkan:\n";
36    for (int i = 0; i < x; i++) {
37        for (int j = 0; j < y; j++) {
38            for (int k = 0; k < z; k++) {
39                cout << "Elemen [" << i << "][" << j << "][" << k << "] = " << array_3d[i][j][k] << endl;
40            }
41        }
42    }
43
44    // Membersihkan memori yang dialokasikan secara dinamis
45    for (int i = 0; i < x; i++) {
46        for (int j = 0; j < y; j++) {
47
48
49            // Membersihkan memori yang dialokasikan secara dinamis
50            for (int i = 0; i < x; i++) {
51                for (int j = 0; j < y; j++) {
52                    delete[] array_3d[i][j];
53                }
54                delete[] array_3d[i];
55            }
56            delete[] array_3d;
57
58            return 0;
59        }
60    }
```

Hasilnya:

```
"C:\Users\alvin\OneDrive\Dot
Masukkan ukuran dimensi pertama (x): 3
Masukkan ukuran dimensi kedua (y): 2
Masukkan ukuran dimensi ketiga (z): 1
Masukkan nilai untuk elemen [0][0][0]: 1 2 3
Masukkan nilai untuk elemen [0][1][0]: Masukkan nilai untuk elemen [1][0][0]: Masukkan nilai untuk elemen [1][1][0]: 4 5
6
Masukkan nilai untuk elemen [2][0][0]: Masukkan nilai untuk elemen [2][1][0]:
Array 3D yang diinputkan:
Elemen [0][0][0] = 1
Elemen [0][1][0] = 2
Elemen [1][0][0] = 3
Elemen [1][1][0] = 4
Elemen [2][0][0] = 5
Elemen [2][1][0] = 6

Process returned 0 (0x0)   execution time : 41.548 s
Press any key to continue.
```

3. Buatlah program menu untuk mencari nilai Maksimum, Minimum dan Nilai rata – rata dari suatu array dengan input yang dimasukan oleh user!

Jawabanya:

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

Hasilnya:

```
"C:\Users\alvin\OneDrive\Do... X + v
Masukkan jumlah elemen array: 9
Masukkan nilai elemen array:
Elemen ke-1: 1
Elemen ke-2: 2
Elemen ke-3: 3
Elemen ke-4: 4
Elemen ke-5: 5
Elemen ke-6: 6
Elemen ke-7: 7
Elemen ke-8: 8
Elemen ke-9: 9

Menu Pilihan:
1. Cari Nilai Maksimum
2. Cari Nilai Minimum
3. Cari Nilai Rata-Rata
4. Keluar
Masukkan pilihan (1-4): 2
Nilai minimum: 1

Menu Pilihan:
1. Cari Nilai Maksimum
2. Cari Nilai Minimum
3. Cari Nilai Rata-Rata
4. Keluar
Masukkan pilihan (1-4): 3
Nilai rata-rata: 5

Menu Pilihan:
1. Cari Nilai Maksimum
2. Cari Nilai Minimum
3. Cari Nilai Rata-Rata
4. Keluar
Masukkan pilihan (1-4): 1
Nilai maksimum: 9

Menu Pilihan:
1. Cari Nilai Maksimum
2. Cari Nilai Minimum
3. Cari Nilai Rata-Rata
4. Keluar
Masukkan pilihan (1-4): 4
Keluar dari program.

Process returned 0 (0x0)   execution time : 22.615 s
Press any key to continue.
```

VI. KESIMPULAN

Kesimpulan pengenalan bahasa C++ secara ringkas:

1. Bahasa Berorientasi Objek: Mendukung konsep seperti *class*, *object*, dan *inheritance* untuk membuat kode yang modular dan dapat digunakan ulang.
2. Kinerja Tinggi: C++ menghasilkan program dengan performa efisien, cocok untuk aplikasi yang membutuhkan kecepatan seperti game dan sistem operasi.
3. Kontrol Memori: Memungkinkan kontrol penuh atas manajemen memori dengan alokasi dinamis melalui *new* dan *delete*.

4. **Kompilasi Statis:** Memungkinkan deteksi kesalahan saat kompilasi, mengurangi bug saat runtime.
 5. **Kompatibilitas dengan C:** C++ mendukung kode C, membuatnya mudah diadopsi oleh programmer C.
 6. **Pustaka Standar:** Pustaka standar (STL) menyediakan struktur data dan algoritma siap pakai.
 7. **Penggunaan Luas:** Digunakan di berbagai aplikasi, termasuk perangkat lunak sistem, game, dan aplikasi perusahaan.
- C++ menawarkan kekuatan dan fleksibilitas untuk mengembangkan perangkat lunak skala besar dengan performa tinggi.