LAPORAN PRAKTIKUM
Modul 2
Struktur Data

Disusun Oleh:
Aulia Jasifa Br Ginting 2311104060
SES1-07-02

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. Mengiimplementasikan fungdi dan prosedur dalam program

2. **Landasan Teori**

2.1 Array

Array dalam bahasa C++ adalah dtruktur data yang digunakan untuuk menyimpan sekumpulan elemen dengan tipe data yang sama dalam satu variabel .

2.1.1 Array Satu Dimensi

Array satu dimensi adalah jenis array paling sederhana dalam C++. Ini adalah struktur daya yang menyimpan elemen-elemen dengan tipe data yang sama dalam urutan linear.

```
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];
```

2.1.2 Array Dua Dimensi

Array dua dimensi dalam C++ berguna untuk menyimpan data yang berbentuk tabel atau matriks. Indeks pertama menunjukkan baris dan indeks kedua menunjukka kolom. Array dua dimensi sangat umum digunakan dalam pemrograman yang melibatkan data berbentuk tabel, seperti grafik, citra, atau matriks perhitungan.

Contoh:

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

	0	1	2
0			
1			
2	10		
3			

Gambar 2-1 Ilustrasi *Array* Dua Dimensi

2.1.2 Array Berdimensi Banyak

Array berdimensi banyak dalam C++ adalah array yang memiliki lebih dari dua dimensi, yaitu array yang berisi array yang juga bisa memiliki array di dalamnya, dan seterusnya. Array berdimensi banyak ini sering disebut array "multi-dimensi" atau "multi-dimensional arrays".

Cara deklarasi:

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

Contoh:

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

2.2 Pointer

Pointer dalam C++ adalah variabel yang menyimpan alamat memori dari variabel lain, memungkinkan akses langsung ke lokasi data di memori. Dengan pointer, kita bisa membaca atau menulis data di lokasi memori tertentu, yang memberi kita kendali tinggi atas alokasi dan penggunaan memori.

2.2.1 Data dan Memori

Dalam C++, pemahaman tentang data dan memori sangat penting karena bahasa ini memberi kontrol yang mendalam atas bagaimana data disimpan dan dikelola dalam memori. Mari kita bahas secara rinci mengenai konsep data dan memori di C++.

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**".

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 mencarikan cell kosong untuk dialokasikan sebagai memori variabel tersebut.

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

Gambar 2-3 Ilustrasi Alokasi *Memory*

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

2.2.2 Pointer dan Alamat

Pointer dan alamat adalah konsep fundamental dalam C++ yang memungkinkan programmer untuk mengelola memori dengan efisien dan mengakses.
Cara pendeklarasian variabel pointer adalah sebagai berikut:

```
type *nama_variabel;
```

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

```
p_int = &j;
```

Contoh

```
int *p_int;
```

/* p_int merupakan variabel pointer yang menunjukkan ke data bertipe int*/

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

int j,k;
j =10;
int *p_int;
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;
	10
	X6
	X6
	X1
	10
	
	10
		Nilai variabel j
		Alamat variabel j
		Nilai variabel p_int
		Alamat variabel p_int
		Nilai variabel yang ditunjuk p_int
		
		Nilai variabel k

Berikut 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;
}
```

```
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

Pointer dan array adalah dua konsep yang sangat penting dalam C++. Keduanya sering digunakan bersama untuk mengelola dan mengakses data dengan cara yang efisien. Mari kita bahas secara mendetail mengenai pointer dan array dalam C++.

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*

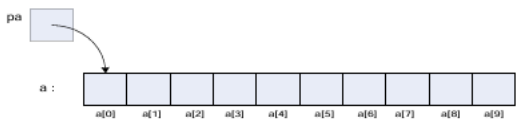
Notasi a[i] akan merujuk elemen ke-i dari array. Jika pa merupakan pointer yang menunjuk variabel bertipe integer , yang di deklarasikan sebagai berikut :

```
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].



Gambar 2-7 *Pointer* dan *Array*

Sekarang, pernyataan

```
X=*pa;
```

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

2.2.4 Pointer dan String

Dalam C++, pointer dan string saling berhubungan erat, terutama karena string sering kali diimplementasikan sebagai kumpulan karakter yang diakses melalui pointer. Mari kita lihat secara detail bagaimana pointer dan string digunakan bersama dalam C++.

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 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

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.

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 ;
}
```

2.5 Parameter Fungsi

2.5.1 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 (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 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.

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

3. Guided

1. Penjumlahan

Programnya

```cpp
#include <iostream>
#include <conio.h>

using namespace std;

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("aulia");

}
```

Outputnya:

```
hasil 8
Hello, aulia!
```

2. Array Satu Dimensi

Programnya

```cpp
//array 1 dimensi*
int main() {
 int nilai[5]={1,2,3,4,5};
cout << nilai[0];
cout << nilai[1];
cout << nilai[2];
cout << nilai[3];
cout << nilai[4];

for (int i=0; i<5; i++){
    cout << nilai[i] << endl;
}
}
```

Outputnya

```
123451
2
3
4
5
```

3. Array Dua dimensi

Programnya

```
    // array 2d
    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] << " ";
    }
    cout << endl;
}
```

Outputnya

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

4. Pointer

Programnya

```
#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;
}
```

Outputnya

```
Alamat x= 0x61ff04
Isi px= 0x61ff04
Isi X= 87
Nilai yang ditunjuk px= 87
Nilai y= 87
```

4. Unguided

1. Buatlahh program untuk menampilkan output seperti berikut dengan data diinputkan oleh user!
   Programnya

Telkom University Purwokerto

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

int main() {
    vector<int> dataArray;
    vector<int> evenNumbers;
    vector<int> oddNumbers;
    int num;

    cout << "Masukkan angka-angka (0 untuk mengakhiri input):\n";

    // Input data dari user
    while (true) {
        cin >> num;
        if (num == 0) break;
        dataArray.push_back(num);
    }

    // Memisahkan nomor genap dan ganjil
    for (int i = 0; i < dataArray.size(); i++) {
        if (dataArray[i] % 2 == 0) {
            evenNumbers.push_back(dataArray[i]);
        } else {
            oddNumbers.push_back(dataArray[i]);
        }
    }

    // Menampilkan output
    cout << "Data Array : ";
    for (int i = 0; i < dataArray.size(); i++) {
        cout << dataArray[i] << " ";
    }
    cout << endl;

    cout << "Nomor Genap : ";
    for (int i = 0; i < evenNumbers.size(); i++) {
        cout << evenNumbers[i];
        if (i < evenNumbers.size() - 1) cout << ", ";
    }
    cout << "," << endl;

    cout << "Nomor Ganjil : ";
    for (int i = 0; i < oddNumbers.size(); i++) {
        cout << oddNumbers[i];
        if (i < oddNumbers.size() - 1) cout << ", ";
    }
    cout << "," << endl;

    return 0;
}
```

Outputnya:

```
Masukkan angka-angka (0 untuk mengakhiri input):
1 2 3 4 5 6 7 8 9 10 0
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,
```

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

Programnya

```cpp
#include <iostream>

using namespace std;

int main() {
    int x, y, z;

    // input ukuran array dari user
    cout << "Masukkan ukuran array 3D (x y z): ";
    cin >> x >> y >> z;

    // membuat array 3D dengan ukuran yang diinputkan
    int*** arr = new int**[x];
    for (int i = 0; i < x; i++) {
        arr[i] = new int*[y];
        for (int j = 0; j < y; j++) {
            arr[i][j] = new int[z];
        }
    }

    // input elemen array dari usera
    cout << "Masukkan elemen array: " << endl;
    for (int i = 0; i < x; i++) {
        cout << "Lapisan " << i + 1 << ":" << endl;
        for (int j = 0; j < y; j++) {
            for (int k = 0; k < z; k++) {
                cout << arr[i][j][k] << " ";
            }
            cout << endl;
        }
        cout << endl;
    }
    // membersihkan memori
    for (int i = 0; i < x; i++) {
        for (int j = 0; j < y; j++) {
            delete[] arr[i][j];
        }
        delete[] arr[i];

        return 0;
    }
}
```

Outputnya:

```
Masukkan ukuran array 3D (x y z): 2 3 2
Masukkan elemen array:
Lapisan 1:
9468640 9437376
9468640 9437376
9468640 9437376

Lapisan 2:
9468640 9437376
9468640 9437376
9468640 9437376
```

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

Programnya

```cpp
#include <iostream>
#include <limits>
#include <iomanip>

using namespace std;

// Fungsi untuk mencari nilai maksimum
int findMax(int arr[], int size) {
    int max = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}

// Fungsi untuk mencari nilai minimum
int findMin(int arr[], int size) {
    int min = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] < min) {
            min = arr[i];
        }
    }
    return min;
}

// Fungsi untuk menghitung nilai rata-rata
double findAverage(int arr[], int size) {
    double sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum / size;
}

int main() {
    int size;
    cout << "Masukkan ukuran array: ";
    cin >> size;

    int* arr = new int[size];

    cout << "Masukkan " << size << " elemen array:" << endl;
    for (int i = 0; i < size; i++) {
        cout << "Elemen ke-" << i + 1 << ": ";
        cin >> arr[i];
    }

    int choice;
    do {
        cout << "\nMenu:" << endl;
        cout << "1. Cari nilai Maksimum" << endl;
        cout << "2. Cari nilai Minimum" << endl;
        cout << "3. Hitung nilai Rata-rata" << endl;
        cout << "4. Keluar" << endl;
        cout << "Pilih menu (1-4): ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Nilai Maksimum: " << findMax(arr, size) << endl;
                break;
            case 2:
                cout << "Nilai Minimum: " << findMin(arr, size) << endl;
                break;
            case 3:
                cout << "Nilai Rata-rata: " << fixed << setprecision(2) << findAverage(arr, size) << endl;
                break;
            case 4:
                cout << "Terima kasih, program selesai." << endl;
                break;
            default:
                cout << "Pilihan tidak valid. Silakan coba lagi." << endl;
        }
    } while (choice != 4);

    delete[] arr;  // Membersihkan memori
    return 0;
}
```

Outputnya:

```
Masukkan ukuran array: 3
Masukkan 3 elemen array:
Elemen ke-1: 9
Elemen ke-2: 7
Elemen ke-3: 5

Menu:
1. Cari nilai Maksimum
2. Cari nilai Minimum
3. Hitung nilai Rata-rata
4. Keluar
Pilih menu (1-4): 3
Nilai Rata-rata: 7.00

Menu:
1. Cari nilai Maksimum
2. Cari nilai Minimum
3. Hitung nilai Rata-rata
4. Keluar
Pilih menu (1-4): 2
Nilai Minimum: 5

Menu:
1. Cari nilai Maksimum
2. Cari nilai Minimum
3. Hitung nilai Rata-rata
4. Keluar
Pilih menu (1-4): 1
Nilai Maksimum: 9

Menu:
1. Cari nilai Maksimum
2. Cari nilai Minimum
3. Hitung nilai Rata-rata
4. Keluar
Pilih menu (1-4): 4
Terima kasih, program selesai.
```

5. **Kesimpulan**

   Lebih memahami pembagian pada bahasa C++ serta cara kerjanya