LAPORAN PRAKTIKUM
MODUL 2
PENGENALAN BAHASA C++ BAGIAN KEDUA

Disusun Oleh:

Satria Putra Dharma Prayudha - 21104036

SE07-02

Dosen :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

A. Tujuan

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

B. Landasan Teori

Landasan teori ini berdasarkan pada modul pembelajaran praktikum struktur data kali ini

2.1 *Array*

Array adalah struktur data yang terdiri dari kumpulan elemen dengan tipe data yang sama. Setiap elemen dalam *array* diakses menggunakan indeks, yang dimulai dari 0.

2.1.1 *Array* Satu Dimensi

Array satu dimensi adalah *array* yang hanya memiliki satu indeks dan digunakan untuk menyimpan data dalam satu baris. Elemen dalam *array* satu dimensi dapat diakses menggunakan indeks yang mengidentifikasi posisi elemen tersebut dalam *array*. Data dalam *array* disimpan secara berurutan dalam memori.

2.1.2 *Array* Dua Dimensi

Array dua dimensi mirip dengan tabel yang memiliki baris dan kolom. Elemen-elemen dalam *array* dua dimensi diakses dengan menggunakan dua indeks: satu untuk baris dan satu untuk kolom. *Array* ini digunakan untuk menyimpan data yang memiliki dua dimensi, seperti matriks.

2.1.3 *Array* Berdimensi Banyak

Array berdimensi banyak adalah pengembangan dari *array* dua dimensi yang memiliki lebih dari dua indeks. Ini memungkinkan penyimpanan data yang lebih kompleks dengan banyak dimensi. *Array* berdimensi banyak lebih sulit dibayangkan dan digunakan untuk penyimpanan data yang memerlukan banyak dimensi.

2.2 Pointer

Pointer adalah variabel yang menyimpan alamat memori dari variabel lain. Dengan menggunakan pointer, kita dapat mengakses dan memanipulasi data yang tersimpan di alamat memori tersebut.

2.2.1 Data dan Memori

Semua data yang digunakan oleh program disimpan dalam memori komputer. Setiap lokasi memori memiliki alamat unik, yang dapat diakses melalui *pointer*. Sistem operasi mengalokasikan ruang memori untuk setiap variabel yang digunakan dalam program.

2.2.2 *Pointer* dan Alamat

Pointer menyimpan alamat memori dari variabel lain. Dengan menggunakan pointer, program dapat mengakses dan memanipulasi data dari variabel yang ditunjuk. *Pointer* diinisialisasi dengan alamat dari variabel lain menggunakan operator '&'.

2.2.3 *Pointer* dan *Array*

Array dan *pointer* memiliki keterkaitan yang kuat. Dalam banyak kasus, operasi yang dilakukan pada *array* dapat dilakukan menggunakan pointer. Nama *array* sebenarnya adalah *pointer* yang menunjuk ke elemen pertama dari *array*.

2.2.4 *Pointer* dan String

String dalam C adalah *array* karakter yang diakhiri dengan karakter null ('\0'). *Pointer* dapat digunakan untuk mengakses elemen-elemen string ini. Dengan menggunakan pointer, program dapat mengelola dan memanipulasi string secara efisien.

2.3 Fungsi

Fungsi adalah blok kode yang dirancang untuk melaksanakan tugas khusus. Fungsi menerima parameter sebagai input, melakukan operasi tertentu,

dan mengembalikan hasil sebagai *output*. Fungsi memudahkan pembuatan program yang terstruktur dan modular.

2.4 Prosedur

Prosedur adalah fungsi yang tidak mengembalikan nilai. Dalam C++, prosedur dideklarasikan dengan kata kunci *void*, yang menunjukkan bahwa prosedur tidak memiliki nilai balik. Prosedur digunakan untuk menjalankan operasi tertentu tanpa menghasilkan nilai kembali.

2.5 Parameter Fungsi

Parameter adalah data yang diteruskan ke fungsi untuk diolah. Parameter formal adalah variabel yang dideklarasikan dalam definisi fungsi, sedangkan parameter aktual adalah nilai yang diberikan saat fungsi dipanggil.

2.5.1 Parameter Formal dan Parameter Aktual

Parameter formal adalah variabel yang digunakan di dalam definisi fungsi, sedangkan parameter aktual adalah nilai atau variabel yang digunakan saat memanggil fungsi. Parameter formal menerima nilai dari parameter aktual saat fungsi dipanggil.

2.5.2 Cara Melewatkan Parameter

Ada beberapa cara untuk melewatkan parameter ke fungsi:

- Call by Value: Nilai dari parameter aktual disalin ke parameter formal, sehingga perubahan pada parameter formal tidak mempengaruhi nilai asli dari parameter aktual.
- Call by Pointer: Alamat dari parameter aktual dilewatkan ke fungsi, sehingga perubahan pada parameter formal akan mengubah nilai asli dari parameter aktual.
- Call by Reference: Mirip dengan call by pointer, tetapi lebih mudah digunakan karena tidak memerlukan dereferensi pointer. Parameter formal langsung mereferensikan parameter aktual, sehingga perubahan pada parameter formal akan mengubah nilai

asli dari parameter aktual.

C. Guided

a. *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.

1. *Array* Satu Dimensi

Code :

```cpp
#include <iostream>

using namespace std;

// Array adalah kumpulan data yang memiliki tipe data yang sama
int main(){
    int nilai [5]={1,2,3,4,5};

    for (int i = 0; i < 5; i++){
        cout << "Nilai ke-" << i << " adalah " << nilai[i] << endl;
    }
}
```

Output :

```
PS D:\Kuliah\Struktur Data\Github\02_Pengenalan_CPP_Bagian_2\Guided>
Nilai ke-0 adalah 1
Nilai ke-1 adalah 2
Nilai ke-2 adalah 3
Nilai ke-3 adalah 4
Nilai ke-4 adalah 5
PS D:\Kuliah\Struktur Data\Github\02_Pengenalan_CPP_Bagian_2\Guided>
```

Penjelasan : Program ini mendeklarasikan sebuah *array* satu dimensi nilai yang berisi 5 elemen integer. Kemudian menggunakan *for-loop* untuk mencetak setiap elemen *array* beserta indeksnya. *Array* adalah kumpulan elemen dengan tipe data yang sama, dalam hal ini tipe int. Indeks *array* dimulai dari 0 hingga 4.

2. *Array* 2 Dimensi

Code :

```cpp
#include <iostream>

using namespace std;

// Array 2 Dimensi
int main(){
   int nilai [3][4] = {
       {1, 2, 3, 4},
       {5, 6, 7, 8},
       {9, 10, 11, 12}
   };

    cout << "Outputnya adalah :"<< endl;

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

    cout << endl;
    }
}
```

Output :

```
PS D:\Kuliah\Struktur Data\Github\02_Pengenalan_CPP_Bagian_2\Guided>
e } ; if ($?) { .\tempCodeRunnerFile }
Outputnya adalah :
1 2 3 4
5 6 7 8
9 10 11 12
PS D:\Kuliah\Struktur Data\Github\02_Pengenalan_CPP_Bagian_2\Guided>
```

Penjelasan: Program ini mengimplementasikan *array* dua dimensi dengan ukuran 3x4. Data diatur dalam bentuk matriks, dan ditampilkan melalui dua *for-loop* bersarang (nested loop), dimana loop pertama mengakses baris, dan loop kedua mengakses kolom.

b. Pointer

Code :

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

using namespace std;

// Pointer
int main(){
    int x,y;
    int *px;
    px = &x;
    y = *px;

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

Output :

```
PS D:\Kuliah\Struktur Data\Github\02_Pengenalan_CPP_Bagian_2\Guided> c
e } ; if ($?) { .\tempCodeRunnerFile }
Alamat x= 0x61ff04
isi px= 0x61ff04
Alamat x= 4201200
Nilai yang ditunjuk px= 4201200
Nilai y= 4201200
PS D:\Kuliah\Struktur Data\Github\02_Pengenalan_CPP_Bagian_2\Guided>
```

Penjelasan: Program ini memperkenalkan konsep **pointer**. *Pointer* px menyimpan alamat variabel x. Program ini mencetak alamat dari x, isi dari px (yang berupa alamat x), serta nilai yang ditunjuk oleh px. Variabel y kemudian menyimpan nilai yang ditunjuk oleh *pointer* px.

c. **Fungsi dan Prosedur**

Code:

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

using namespace std;

// 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 Penjumlahan adalah :" << hasil <<
endl;

    greet("Yudha");
}
```

Output:

```
PS D:\Kuliah\Struktur Data\Github\02_Pengenalan_CPP_Bagian_2\Guided>
e } ; if ($?) { .\tempCodeRunnerFile }
Hasil Penjumlahan adalah :8
Hello, Yudha
PS D:\Kuliah\Struktur Data\Github\02_Pengenalan_CPP_Bagian_2\Guided>
```

Penjelasan: Program ini memperkenalkan dua konsep penting: **fungsi** dan **prosedur**. Fungsi penjumlahan mengembalikan hasil penjumlahan dua angka, sementara prosedur *greet* hanya menampilkan pesan ke layar tanpa mengembalikan nilai apapun. Pada fungsi main, hasil dari penjumlahan dua angka (5 dan 3) ditampilkan, diikuti dengan pemanggilan prosedur *greet* yang menampilkan pesan sapaan.

D. Unguided

a. Program Pengecekan Bilangan Ganjil dan Genap di Dalam Array

Code:

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

using namespace std;

//Program untuk menampilkan Output seperti berikut dengan
data yang diinputkan yaitu Array menjadi bilangan ganjil
dan genap

int main(){
    int n;
    cout << "Masukkan jumlah data : ";
    cin >> n;

    int data[n];
    for (int i = 0; i < n; i++){
        cout << "Masukkan data ke-" << i+1 << " : ";
        cin >> data[i];
    }

    cout << endl;

    cout << "Data yang dimasukkan : ";
    for (int i = 0; i < n; i++){
        cout << data[i] << " ";
    }

    cout << endl;


    cout << "Data ganjil : ";
    for (int i = 0; i < n; i++){
        if (data[i] % 2 != 0){
            cout << data[i] << " ";
        }
    }

    cout << endl;

    cout << "Data genap : ";
    for (int i = 0; i < n; i++){
        if (data[i] % 2 == 0){
            cout << data[i] << " ";
        }
    }

    cout << endl;

    getch();
    return 0;
}
```

Output:

```
PS D:\Kuliah\Struktur Data\Github\02_Pengenalan_CPP_Bagian_2\Guided>
ile } ; if ($?) { .\tempCodeRunnerFile }
Masukkan jumlah data : 8
Masukkan data ke-1 : 2
Masukkan data ke-2 : 4
Masukkan data ke-3 : 7
Masukkan data ke-4 : 1
Masukkan data ke-5 : 61
Masukkan data ke-6 : 12
Masukkan data ke-7 : 52
Masukkan data ke-8 : 8

Data yang dimasukkan : 2 4 7 1 61 12 52 8
Data ganjil : 7 1 61
Data genap : 2 4 12 52 8
```

Penjelasan: Program ini memungkinkan untuk menginput sejumlah data ke dalam *array* dan kemudian memisahkan serta menampilkan data berdasarkan kategori bilangan ganjil dan genap. Setelah pengguna memasukkan jumlah data yang diinginkan, program akan meminta input untuk setiap elemen yang akan disimpan dalam *array*. Kemudian, program menampilkan semua data yang dimasukkan sebelum melakukan pengecekan untuk menentukan mana yang termasuk bilangan ganjil (sisa bagi 2 tidak sama dengan 0) dan mana yang termasuk bilangan genap (sisa bagi 2 sama dengan 0). Dengan cara ini, dapat diketahui perbedaan antara bilangan ganjil dan genap dari data yang sudah di masukkan.

b. **Program Array Tiga Dimensi dengan Input**

Code:

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

using namespace std;

// Input array tiga dimensi tetapi jumlah atau ukuran
elemennya diinputkan

int main (){
    int x, y, z;
    cout << "Masukkan jumlah elemen x : ";
    cin >> x;
    cout << "Masukkan jumlah elemen y : ";
    cin >> y;
    cout << "Masukkan jumlah elemen z : ";
    cin >> z;

    int data[x][y][z];

    for (int i = 0; i < x; i++){
        for (int j = 0; j < y; j++){
            for (int k = 0; k < z; k++){
                cout << "Masukkan data ke-" << i+1 << " :
";
                cin >> data[i][j][k];
            }
        }
    }

    cout << endl;

    for (int i = 0; i < x; i++){
        for (int j = 0; j < y; j++){
            for (int k = 0; k < z; k++){
                cout << data[i][j][k] << " ";
            }
            cout << endl;
        }
        cout << endl;
    }

    getch();
    return 0;
}
```

Output:

```
PS D:\Kuliah\Struktur Data\Github\02_Pengenalan_CPP_Bagian_2\Unguided> cd "d:\
nguided_02_Array_Tiga_Dimensi } ; if ($?) { .\Unguided_02_Array_Tiga_Dimensi }
Masukkan jumlah elemen x : 2
Masukkan jumlah elemen y : 3
Masukkan jumlah elemen z : 3
Masukkan data ke-1 : 1
Masukkan data ke-1 : 1
Masukkan data ke-1 : 1
Masukkan data ke-1 : 1
Masukkan data ke-1 : 1
Masukkan data ke-1 : 1
Masukkan data ke-1 : 1
Masukkan data ke-1 : 1
Masukkan data ke-1 : 1
Masukkan data ke-2 : 1
Masukkan data ke-2 : 1
Masukkan data ke-2 : 1
Masukkan data ke-2 : 1
Masukkan data ke-2 : 1
Masukkan data ke-2 : 1
Masukkan data ke-2 : 1
Masukkan data ke-2 : 1
Masukkan data ke-2 : 1

1 1 1
1 1 1
1 1 1

1 1 1
1 1 1
1 1 1
```

Penjelasan: Program ini memungkinkan untuk menyimpan data dalam struktur 3 Dimensi. Pada program ini diminta untuk memasukkan ukuran untuk setiap dimensi (x, y, dan z), yang menentukan jumlah elemen di masing-masing dimensi. Setelah itu, program menggunakan *loop nest* untuk memasukkan data yang di *Input* ke setiap elemen dalam *array* tiga dimensi. Setelah semua data diinputkan, program mencetak elemen-elemen dari *array* dengan format yang sudah disesuaikan, untuk outputnya adalah data dalam bentuk tiga dimensi.

c. **Program Mencari Nilai Maksimum, Minimum, dan Rata-Rata**

Code:

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

using namespace std;

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

int main(){
    int n;
    cout << "Masukkan jumlah data : ";
    cin >> n;

    int data[n];
    for (int i = 0; i < n; i++){
        cout << "Masukkan data ke-" << i+1 << " : ";
        cin >> data[i];
    }

    cout << endl;

    cout << "Data yang dimasukkan : ";
    for (int i = 0; i < n; i++){
        cout << data[i] << " ";
    }

    cout << endl;

    int max = data[0];
    int min = data[0];
    int sum = 0;
    for (int i = 0; i < n; i++){
        if (data[i] > max){
            max = data[i];
        }
        if (data[i] < min){
            min = data[i];
        }
        sum += data[i];
    }

    cout << "Nilai maksimum : " << max << endl;
    cout << "Nilai minimum : " << min << endl;
    cout << "Nilai rata-rata : " << (float)sum/n << endl;

    getch();
    return 0;
}
```

Output:

```
PS D:\Kuliah\Struktur Data\Github> cd "d:\Kuliah\Struktur Data\Github\02_Pengenalan_CPP_Bagian_2\Ung
le }
Masukkan jumlah data : 4
Masukkan data ke-1 : 3
Masukkan data ke-2 : 7
Masukkan data ke-3 : 10
Masukkan data ke-4 : 23

Data yang dimasukkan : 3 7 10 23
Nilai maksimum : 23
Nilai minimum : 3
Nilai rata-rata : 10.75
```

Penjelasan: Program ini bertujuan untuk mencari nilai maksimum, minimum, dan rata-rata dari sekumpulan data yang sudah *diinputkan*. Setelah memasukkan jumlah data yang dinginkan, program selanjutnya akan meminta *input* untuk setiap elemen dan menyimpannya dalam *array*. Setelah itu, program mencetak semua data yang dimasukkan untuk konfirmasi. Selanjutnya, program menghitung nilai maksimum dan minimum dengan membandingkan setiap elemen, serta menghitung total nilai dari semua elemen untuk mendapatkan rata-rata.

E. Kesimpulan

Dalam modul ini telah diketahui tentang berbagai konsep dasar dalam pemrograman menggunakan bahasa C++, khususnya yang berkaitan dengan *array*, *pointer*, fungsi, dan prosedur. Penggunaan *array* satu dimensi dan dua dimensi memberikan pemahaman tentang cara menyimpan dan mengakses kumpulan data yang terstruktur. *Array* juga dapat diperluas ke dalam dimensi yang lebih tinggi, seperti *array* tiga dimensi, yang menunjukkan kemampuan bahasa C++ untuk menangani data yang lebih kompleks dan beragam.

Selain itu, konsep *pointer* yang diperkenalkan memberikan pengetahuan yang lebih dalam mengenai cara program berinteraksi dengan memori. Dengan memahami bagaimana *pointer* berfungsi, kita dapat melakukan manipulasi data yang lebih efisien dan mengoptimalkan penggunaan memori dalam program. Fungsi dan prosedur, sebagai elemen penting dalam pemrograman terstruktur, memudahkan dalam pembuatan kode yang modular dan lebih mudah dikelola. Secara keseluruhan, modul ini sangat bermanfaat dalam memberikan dasar yang kuat untuk pemrograman lebih lanjut di C++.