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

Disusun Oleh:
Jauhar Fajar Zuhair - 2311104072
S1SE – 07 -02

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. **Tujuan**
   - Memahami penggunaan pointer dan alamat memori.
   - Mengimplementasikan fungsi dan prosedur dalam program.
2. **Landasan Teori**

Array

Array adalah struktur data yang menyimpan sejumlah elemen bertipe data yang sama dalam satu kesatuan. Setiap elemen dalam array diakses melalui indeks yang dimulai dari nol. Array memungkinkan penyimpanan data dalam blok memori yang berurutan, sehingga akses ke

elemen-elemen dalam array menjadi lebih efisien. Array satu dimensi digunakan untuk menyimpan data linear, sedangkan array dua dimensi menyerupai tabel dan dapat digunakan untuk menyimpan data dalam bentuk matriks. Selain itu, array berdimensi banyak memungkinkan penyimpanan data dalam beberapa lapisan, meskipun lebih kompleks untuk dikelola.

Pointer

Pointer adalah variabel yang menyimpan alamat memori dari variabel lain. Penggunaan pointer memungkinkan akses langsung ke alamat memori, sehingga memungkinkan program untuk memodifikasi nilai variabel secara tidak langsung. Pointer juga memungkinkan pemrosesan data yang lebih efisien, terutama dalam hal manipulasi array, struktur, dan objek dinamis. Sebuah pointer dapat menunjuk ke tipe data apa pun, termasuk tipe dasar seperti integer, karakter, dan float. Pointer juga memainkan peran penting dalam manajemen memori dan struktur data tingkat lanjut, seperti linked list, tree, dan graph.

Pointer dan Array

Pointer dan array memiliki hubungan yang erat dalam bahasa C++. Sebuah array sebenarnya adalah pointer yang menunjuk ke elemen pertama dari array tersebut. Hal ini berarti, operasi yang dilakukan dengan array juga dapat dilakukan menggunakan pointer. Dalam konteks ini, sebuah pointer yang menunjuk ke elemen array dapat digunakan untuk mengakses atau memodifikasi nilai-nilai dalam array. Dengan memanfaatkan aritmetika pointer, kita dapat menavigasi melalui elemen-elemen array dengan cara yang lebih fleksibel dibandingkan akses indeks tradisional.

Pointer dan String

Dalam C++, string secara teknis merupakan array dari karakter yang diakhiri dengan karakter null (`\0`). String dapat dikelola menggunakan pointer, di mana pointer karakter menunjuk ke lokasi memori dari elemen pertama string. Hal ini memudahkan manipulasi string menggunakan pointer, seperti iterasi karakter per karakter, serta memungkinkan operasi-operasi lain yang berkaitan dengan manajemen memori dan efisiensi program.

Fungsi

Fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu. Fungsi membantu memecah program menjadi bagian-bagian kecil yang terstruktur, sehingga program menjadi lebih mudah dipahami dan dikelola. Fungsi umumnya menerima input berupa parameter, dan mengembalikan hasil setelah memproses data yang diberikan. Salah satu keuntungan utama menggunakan fungsi adalah pengurangan duplikasi kode, karena kita dapat memanggil fungsi yang sama berulang kali di berbagai bagian program.

Prosedur

Prosedur dalam bahasa C++ adalah fungsi yang tidak mengembalikan nilai. Biasanya, prosedur digunakan untuk melakukan operasi tertentu yang tidak memerlukan hasil balik, seperti menampilkan data ke layar atau memproses input dari pengguna. Prosedur umumnya

didefinisikan dengan tipe kembalian `void`, dan memungkinkan program menjadi lebih modular dan terstruktur. Meskipun tidak mengembalikan nilai, prosedur tetap dapat menerima parameter input untuk menjalankan tugasnya.

Parameter dalam Fungsi

Parameter dalam fungsi dibagi menjadi dua jenis: parameter formal dan parameter aktual. Parameter formal adalah variabel yang digunakan dalam deklarasi fungsi, sedangkan parameter aktual adalah nilai yang dikirimkan ketika fungsi dipanggil. Ada beberapa cara untuk melewatkan parameter ke dalam fungsi: pemanggilan dengan nilai (call by value), pemanggilan dengan pointer (call by pointer), dan pemanggilan dengan referensi (call by reference). Pada call by value, nilai dari parameter aktual disalin ke parameter formal, sehingga perubahan pada parameter formal tidak memengaruhi parameter aktual. Sebaliknya, pada call by pointer dan call by reference, fungsi dapat mengubah nilai dari variabel yang diteruskan karena yang dilewatkan adalah alamat memori atau referensi dari variabel tersebut.

3. Guided

3.1 Array

Array merupakan kumpulan data yang memiliki nama yang sama dengan elemen-elemen bertipe data yang serupa. Setiap elemen array diakses menggunakan indeks tertentu.

3.1.1 Array Satu Dimensi

Array satu dimensi adalah array yang hanya memiliki satu larik data. Deklarasi array ini berbentuk:

```
tipe_data nama_var[ukuran];
```

Contohnya, `int nilai[10];` mendeklarasikan sebuah array bernama "nilai" yang berisi 10 elemen bertipe integer. Elemen pertama array memiliki indeks 0, sehingga elemen ke-5 ditulis sebagai `nilai[4]`.

3.1.2 Array Dua Dimensi

Array dua dimensi menyerupai tabel dan digunakan untuk menyimpan data dalam bentuk tabel. Deklarasinya mirip dengan array satu dimensi, namun melibatkan dua indeks:

```
int data_nilai[4][3];
```

Setiap elemen dapat diakses menggunakan dua indeks, seperti `nilai[2][0] = 10;`.

3.1.3 Array Berdimensi Banyak

Array berdimensi banyak memiliki lebih dari dua indeks. Deklarasinya sebagai berikut:

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

Contohnya, `int data_rumit[4][6][6];` adalah array berdimensi tiga.

3.2 Pointer

3.2.1 Data dan Memori

Dalam program komputer, data disimpan di memori (RAM), yang dapat dianggap sebagai array satu dimensi dengan setiap elemen memori memiliki alamat yang unik. Setiap variabel yang dideklarasikan akan mendapatkan alokasi di lokasi memori tertentu, yang bisa diakses menggunakan alamat memori variabel tersebut.

3.2.2 Pointer dan Alamat

Pointer adalah variabel yang menyimpan alamat memori variabel lain. Deklarasi pointer:

```
tipe *nama_variabel;
```

Contoh, `int *p_int;` menunjukkan bahwa `p_int` adalah pointer yang menunjuk ke variabel bertipe integer. Nilai alamat variabel dapat disimpan dalam pointer menggunakan simbol "&", seperti `p_int = &j;`.

3.2.3 Pointer dan Array

Array dan pointer sangat erat hubungannya. Deklarasi array:

```
int a[10];
```

Pointer bisa digunakan untuk mengakses elemen array, misalnya `pa = &a[0];` membuat pointer `pa` menunjuk ke elemen pertama array.

3.2.4 Pointer dan String

String dalam C++ adalah kumpulan karakter yang diperlakukan sebagai array karakter. Pointer dapat digunakan untuk mengakses string dengan cara yang serupa dengan array. Misalnya:

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

Menginisialisasi pointer untuk menunjuk ke string "now is the time".

3.3 Fungsi

Fungsi adalah blok kode yang melakukan tugas tertentu. Keuntungan penggunaan fungsi adalah membuat program lebih terstruktur dan menghindari duplikasi kode. Fungsi umumnya menerima input melalui parameter dan mengembalikan nilai hasil.

Format umum fungsi:

```
tipe_keluaran nama_fungsi(daftar_parameter) {
    // blok kode
}
```

Contoh fungsi yang mengembalikan nilai maksimum dari tiga bilangan:

```
int maks3(int a, int b, int c) {
    int temp_max = a;
    if (b > temp_max) temp_max = b;
    if (c > temp_max) temp_max = c;
    return temp_max;
}
```

3.4 Prosedur

Dalam bahasa C, tidak ada istilah prosedur khusus, karena semuanya adalah fungsi, termasuk `main()`. Prosedur dalam C++ adalah fungsi yang tidak mengembalikan nilai dan biasanya menggunakan kata kunci `void`.

Format prosedur:

```
void nama_prosedur(daftar_parameter) {
    // blok kode
}
```

Contoh prosedur:

```
void tulis(int x) {
    for (int i = 0; i < x; i++) {
        cout << "baris ke-" << i + 1 << endl;
    }
}
```

2.5 Parameter Fungsi

2.5.1 Parameter Formal dan Aktual

Parameter formal adalah variabel yang disebutkan dalam deklarasi fungsi, sementara parameter aktual adalah nilai atau variabel yang dilewatkan saat pemanggilan fungsi.

2.5.2 Cara Melewatkan Parameter

Ada tiga metode utama untuk melewatkan parameter ke dalam fungsi:

1. **Call by Value**: Parameter formal menerima salinan dari parameter aktual, sehingga perubahan pada parameter formal tidak mempengaruhi parameter aktual.
2. **Call by Pointer**: Menggunakan alamat variabel untuk memungkinkan perubahan parameter aktual melalui parameter formal. Contoh:

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

3. **Call by Reference**: Menggunakan referensi agar parameter formal merujuk langsung ke parameter aktual, memungkinkan perubahan parameter aktual secara langsung.

4. Unguided

Telkom University Purwokerto

1.

```cpp
#include <iostream>

int main() {
#include <iostream>

    using namespace std;

    int main(); {
        int arr[10];
        int i;

        cout << "Masukkan 10 angka (dipisahkan spasi): ";
        for (i = 0; i < 10; i++) {
            cin >> arr[i];
        }

        cout << "Data Array : ";
        for (i = 0; i < 10; i++) {
            cout << arr[i] << " ";
        }
        cout << endl;

        cout << "Nomor Genap : ";
        for (i = 0; i < 10; i++) {
            if (arr[i] % 2 == 0) {
                cout << arr[i] << ", ";
            }
        }
        cout << endl;

        cout << "Nomor Ganjil : ";
        for (i = 0; i < 10; i++) {
            if (arr[i] % 2 != 0) {
                cout << arr[i] << ", ";
            }
        }
        cout << endl;
        return 0;
    }
```

```
Masukkan 10 angka (dipisahkan spasi):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,
```

2.

```cpp
#include <iostream>
    using namespace std;
    int main() {
        int x, y, z;
        cout << "Masukkan dimensi array (x y z): ";
        cin >> x >> y >> z;
        int array3D[x][y][z];
        cout << "Masukkan elemen array:\n";
        for (int i = 0; i < x; ++i) {
            for (int j = 0; j < y; ++j) {
                for (int k = 0; k < z; ++k) {
                    cin >> array3D[i][j][k];
                }
            }
        }
        cout << "Isi array tiga dimensi:\n";
        for (int i = 0; i < x; ++i) {
            for (int j = 0; j < y; ++j) {
                for (int k = 0; k < z; ++k) {
                    cout << array3D[i][j][k] << " ";
                }
                cout << endl;
            }
            cout << endl;
        }
        return 0;
    }
```

Telkom University Purwokerto

```
Masukkan dimensi array (x y z):2
2
2
 Masukkan elemen array:
1
1
1
1
1
1
1
1
Isi array tiga dimensi:
1 1
1 1

1 1
1 1


Process finished with exit code 0
```

Telkom University Purwokerto

3.

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

using namespace std;

int main() {
    int n;

    cout << "Masukkan jumlah elemen array: ";
    cin >> n;

    int arr[n];

    cout << "Masukkan elemen array:\n";
    for (int i = 0; i < n; ++i) {
        cin >> arr[i];
    }

    int maks = INT_MIN;
    int min = INT_MAX;
    int total = 0;

    for (int i = 0; i < n; ++i) {
        if (arr[i] > maks) {
            maks = arr[i];
        }
        if (arr[i] < min) {
            min = arr[i];
        }
        total += arr[i];
    }

    float rata_rata = (float)total / n;

    cout << "Nilai Maksimum: " << maks << endl;
    cout << "Nilai Minimum: " << min << endl;
    cout << "Nilai Rata-rata: " << rata_rata << endl;

    return 0;
}
```

```
Masukkan jumlah elemen array:3
 Masukkan elemen array:
3
2
1
Nilai Maksimum: 3
Nilai Minimum: 1
Nilai Rata-rata: 2

Process finished with exit code 0
```

5. Kesimpulan

Dari praktikum ini, dapat disimpulkan bahwa penggunaan array dan pointer dalam bahasa C++ sangat penting dalam manajemen data dan memori. Array memungkinkan penyimpanan dan akses elemen-elemen data secara terstruktur, baik dalam bentuk satu dimensi, dua dimensi, maupun multidimensi. Penggunaan pointer semakin memperkuat fleksibilitas dalam pemrograman karena mampu mengakses dan memanipulasi data langsung melalui alamat memori.

Pointer dan array memiliki keterkaitan erat, di mana pointer dapat digunakan untuk mengakses elemen-elemen array secara langsung. Hal ini meningkatkan efisiensi dalam operasi manipulasi data. Selain itu, konsep string dalam C++ yang merupakan array karakter dapat dikelola secara efektif menggunakan pointer.

Penggunaan fungsi dan prosedur dalam program memungkinkan pembagian tugas yang terstruktur, sehingga membuat program lebih modular, mudah dikembangkan, dan menghindari duplikasi kode. Parameter dalam fungsi bisa dilewatkan melalui berbagai metode, yaitu call by value, call by pointer, dan call by reference, yang masing-masing memiliki kegunaannya tergantung pada kebutuhan program. Dengan memahami penggunaan parameter ini, program dapat dirancang lebih efektif dan efisien.

Secara keseluruhan, praktikum ini memberikan pemahaman mendalam tentang cara kerja dan implementasi struktur data dasar dalam C++, terutama array, pointer, dan fungsi, yang merupakan fondasi penting dalam pemrograman berorientasi objek dan pemrograman efisien.