LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL I
PENGENALAN CODE BLOCKS

Disusun Oleh :

Andera Singgih Pratama 2211104007

Dosen

Diah Septiani S.Kom M.Cs

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Pengajaran pemrograman untuk pemula merupakan area penelitian yang mendapat perhatian luas, mencakup isu-isu seperti pemilihan bahasa yang sesuai, strategi pengajaran, serta desain kurikulum yang mendukung pembelajaran bertahap. Kajian literatur menunjukkan bahwa pemahaman konteks riset ini sangat penting agar pendidik dapat menyusun pengalaman belajar yang efektif sekaligus berlandaskan bukti empiris [1].

Dalam penyusunan materi ajar, dua komponen kunci yang sering menjadi fokus adalah Abstract Data Type (ADT) dan fungsi. ADT, yang dapat direpresentasikan melalui *struct* dalam C++, memberikan kerangka untuk mengorganisasi data agar lebih terstruktur. Di sisi lain, fungsi berperan dalam membagi persoalan kompleks menjadi bagian-bagian yang lebih kecil, terukur, dan dapat digunakan kembali. Kedua konsep ini berfungsi sebagai instrumen penting dalam mengurangi kompleksitas pemrograman, namun sering kali justru menjadi tantangan awal bagi mahasiswa yang baru belajar [2].

Selain itu, penguasaan struktur kontrol seperti percabangan (*conditional statement*) merupakan fondasi utama logika pemrograman. Penggunaan *if-else* memungkinkan mahasiswa mengatur alur program berdasarkan kondisi tertentu. Sejumlah penelitian melaporkan bahwa kesulitan transisi dari sekadar memahami sintaks ke penerapan logika percabangan untuk menyelesaikan masalah nyata adalah salah satu hambatan terbesar dalam tahap awal pembelajaran pemrograman [3].

B. Guided

Guided 1

```cpp
#include <iostream>


using namespace std;

int main()
{
    int x, y; // variabel
    int *px;  // pointer
    x = 87;   // x menampung nilai 87
    px = &x;  // pointer px menyimpan alamat dari x
    y = *px;  // y menampung nilai yang ditunjuk oleh px (nilai tidak terdefinisi karena px belum diinisialisasi)

    cout << "Alamat x = " << &x << endl;  // & adalah operator address of
    cout << "Isi px = " << px << endl;    // px menyimpan alamat dari x
    cout << "Isi x = " << x << endl;      // x menampilkan nilai 87
    cout << "Nilai px = " << *px << endl; // * adalah operator dereference
    cout << "Nilai y = " << y << endl;    // y menampilkan nilai 87

    getchar(); // fungsi untuk menahan layar
    return 0;
}
```

Screenshots Output

```
lovinpeace@LVNPC-MAC Guided % ./program1
Alamat x = 0x16bde67f8
Isi px = 0x16bde67f8
Isi x = 87
Nilai px = 87
Nilai y = 87
```

Program ini menunjukkan cara kerja *pointer* dalam C++. Variabel x diisi dengan nilai 87, kemudian alamatnya disimpan dalam pointer px. Dengan menggunakan operator *dereference* (*px), nilai dari x dapat diakses melalui *pointer* dan disalin ke variabel y. Program juga menampilkan alamat memori x, isi dari pointer px (alamat x), serta nilai dari x, *px, dan y.

Guided 2

```cpp
#include <iostream>

#define MAX 5 // Ukuran array ditetapkan sebagai 5

using namespace std;

int main()
{
    int i, j;
    float nilai[MAX];   // Array 1 dimensi
    static int nilai_tahun[MAX][MAX] = {    // Array 2 dimensi (5x5)
        {0, 2, 2, 0, 0},
        {0, 1, 1, 1, 0},
        {0, 3, 3, 3, 0},
        {4, 4, 0, 0, 4},
        {5, 0, 0, 0, 5}};

    // Input data array 1 dimensi
    for (i = 0; i < MAX; i++)
    {
        cout << "Masukkan nilai ke-" << i + 1 << " ";
        cin >> nilai[i];
    }

    // Menampilkan isi array 1 dimensi
    cout << "\nData nilai siswa:\n";
    for (i = 0; i < MAX; i++)
    {
        cout << "Nilai ke-" << i + 1 << " = " << nilai[i] << endl;
    }

    // Menampilkan isi array 2 dimensi
    cout << "\nNilai tahunan:\n";
    for (i = 0; i < MAX; i++)
    {
        for (j = 0; j < MAX; j++)
        {
            cout << nilai_tahun[i][j] << " ";
        }
        cout << endl; // Tambahkan baris baru setelah menampilkan array 2D
    }
    getchar(); // fungsi untuk menahan layar
    return 0;
}
```

Screenshots Output

```
lovinpeace@LVNPC-MAC Guided % ./program2
Masukkan nilai ke-1 90
Masukkan nilai ke-2 78
Masukkan nilai ke-3 89
Masukkan nilai ke-4 89
Masukkan nilai ke-5 89

Data nilai siswa:
Nilai ke-1 = 90
Nilai ke-2 = 78
Nilai ke-3 = 89
Nilai ke-4 = 89
Nilai ke-5 = 89

Nilai tahunan:
0 2 2 0 0
0 1 1 1 0
0 3 3 3 0
4 4 0 0 4
5 0 0 0 5
```

Program ini mendemonstrasikan penggunaan array satu dimensi dan array dua dimensi dalam C++. Array satu dimensi *nilai* diisi melalui input pengguna, lalu ditampilkan kembali

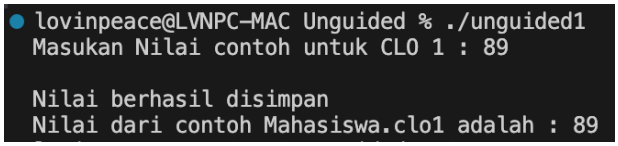
sebagai data nilai siswa. Sementara itu, array dua dimensi *nilai_tahun* yang sudah berisi data bawaan dalam kode ditampilkan dalam bentuk matriks menggunakan *nested loop* (perulangan bersarang).

C. Unguided

Unguided 1

```cpp
#include <iostream>
#include <string>

using namespace std;

struct NilaiSTD {
    float clo1;
    float clo2;
    float clo3;
    float clo4;

    float nilaiAkhir;
    string indeks;
};

int main(){

    //contoh untuk mengisi nilai mahasiswa
    NilaiSTD mahasiswa_contoh;
    cout<< "Masukan Nilai untuk CLO 1 : ";
    cin >> mahasiswa_contoh.clo1;
    cout <<"\nNilai berhasil disimpan" << endl;

    cout<<"Nilai CLO 1 : " << mahasiswa_contoh.clo1 << end
l;
    return 0;
}
```

Screenshots Output



Program ini memperlihatkan penggunaan *struct* dalam C++ untuk mengelola data nilai mahasiswa. Sebuah *struct* bernama *NilaiSTD* didefinisikan dengan beberapa atribut, yaitu *clo1*, *clo2*, *clo3*, *clo4*, *nilaiAkhir*, dan *indeks*. Pada fungsi main, dibuat sebuah variabel *struct* bernama mahasiswa_contoh yang digunakan untuk menyimpan nilai *CLO 1* melalui *input* pengguna. Nilai yang sudah dimasukkan kemudian ditampilkan kembali ke layar.

Unguided 2

```cpp
#include <iostream>
#include <string>

using namespace std;

struct NilaiSTD {
    float clo1;
    float clo2;
    float clo3;
    float clo4;

    float nilaiAkhir;
    string indeks;
};

int main(){

    //contoh untuk mengisi nilai mahasiswa
    NilaiSTD mahasiswa_contoh;
    cout<< "Masukan Nilai untuk CLO 1 : ";
    cin >> mahasiswa_contoh.clo1;
    cout <<"\nNilai berhasil disimpan" << endl;

    cout<<"Nilai CLO 1 : " << mahasiswa_contoh.clo1 << end
l;
    return 0;
}
```

Screenshots Output

```
lovinpeace@LVNPC-MAC Unguided % ./unguided2
Masukan Nilai akhir untuk mahasiswa satu : 78
Masukan Nilai akhir untuk mahasiswa dua : 89

Nilai berhasil disimpan
Nilai akhir mahasiswa 1 : 78
Nilai akhir mahasiswa 2 : 89
```

Program ini mendemonstrasikan penggunaan *struct* dalam C++ untuk menyimpan data nilai mahasiswa. Sebuah *struct* bernama *NilaiSTD* memiliki beberapa atribut, yaitu *clo1*, *clo2*, *clo3*, *clo4*, *nilaiAkhir*, dan *indeks*. Pada fungsi main, dibuat sebuah variabel *struct* bernama mahasiswa_contoh. Program meminta pengguna memasukkan nilai untuk *CLO 1*, kemudian nilai tersebut disimpan ke dalam atribut clo1 dan ditampilkan kembali ke layar sebagai output.

Unguided 3

```cpp
#include <iostream>

using namespace std;

int main() {
    float clo1_val, clo2_val, clo3_val, clo4_val;

    cout << "Masukkan nilai CLO-1 : ";
    cin >> clo1_val;
    cout << "Masukkan nilai CLO-2 : ";
    cin >> clo2_val;
    cout << "Masukkan nilai CLO-3 : ";
    cin >> clo3_val;
    cout << "Masukkan nilai CLO-4 : ";
    cin >> clo4_val;

    cout << "\nNilai yang Anda masukkan adalah:" << endl;
    cout << "CLO 1: " << clo1_val << endl;
    cout << "CLO 2: " << clo2_val << endl;
    cout << "CLO 3: " << clo3_val << endl;
    cout << "CLO 4: " << clo4_val << endl;

    return 0;
}
```

Screenshots Output

```
lovinpeace@LVNPC-MAC Unguided % ./unguided3
Masukkan nilai CLO-1 : 90
Masukkan nilai CLO-2 : 98
Masukkan nilai CLO-3 : 78
Masukkan nilai CLO-4 : 70

Nilai yang Anda masukkan adalah:
CLO 1: 90
CLO 2: 98
CLO 3: 78
CLO 4: 70
```

Program ini digunakan untuk menerima *input* nilai dari empat *Course Learning Outcome (CLO)* mahasiswa. Empat variabel bertipe *float* dideklarasikan untuk menyimpan nilai masing-masing *CLO*. Program meminta pengguna memasukkan nilai satu per satu melalui perintah cin, kemudian menampilkannya kembali di layar menggunakan cout. Dengan demikian, program memperlihatkan proses dasar *input* dan *output* data pada C++ menggunakan tipe data numerik.

Unguided 4,5

```cpp
#include <iostream>
#include <string>

using namespace std;

struct NilaiSTD {
    float clo1;
    float clo2;
    float clo3;
    float clo4;
    float nilaiAkhir;
    string indeks;
};

NilaiSTD gabungNilaiStd(float c1, float c2, float c3, float c4) {
    NilaiSTD temp;
    temp.clo1 = c1;
    temp.clo2 = c2;
    temp.clo3 = c3;
    temp.clo4 = c4;
    temp.nilaiAkhir = 0.0f;
    temp.indeks = "N/A";
    return temp;
}

int main() {
    float c1, c2, c3, c4;

    cout << "--- Input 4 Nilai CLO ---" << endl;
    cout << "CLO 1: "; cin >> c1;
    cout << "CLO 2: "; cin >> c2;
    cout << "CLO 3: "; cin >> c3;
    cout << "CLO 4: "; cin >> c4;

    // Panggil fungsi dengan data dari input user
    NilaiSTD mahasiswa = gabungNilaiStd(c1, c2, c3, c4);

    cout << "\nNilai Berhasil Disimpan" << endl;
    cout << "Data CLO 1 yang tersimpan di dalam struct: " << mahasiswa.clo1 << endl;

    return 0;
}
```

Screenshots Output

```
lovinpeace@LVNPC-MAC Unguided % ./unguided4,5
--- Input 4 Nilai CLO ---
CLO 1: 89
CLO 2: 70
CLO 3: 87
CLO 4: 60

Nilai Berhasil Disimpan
Data CLO 1 yang tersimpan di dalam struct: 89
```

Program ini menunjukkan penggunaan *struct* dan fungsi dalam C++. Sebuah *struct* bernama *NilaiSTD* didefinisikan untuk menyimpan nilai empat *Course Learning Outcome (CLO)*, nilai akhir, serta indeks huruf. Fungsi gabungNilaiStd dibuat untuk menerima empat nilai *CLO* sebagai parameter, lalu mengembalikan sebuah objek *NilaiSTD* dengan nilai-nilai tersebut. Pada fungsi main, pengguna diminta memasukkan empat nilai *CLO*. Data tersebut kemudian diproses oleh fungsi gabungNilaiStd dan hasilnya disimpan dalam variabel

mahasiswa. Program menampilkan pesan bahwa nilai berhasil disimpan, sekaligus menampilkan salah satu nilai *CLO* yang tersimpan dalam *struct*.

Unguided 6

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

using namespace std;

struct NilaiSTD {
    float clo1;
    float clo2;
    float clo3;
    float clo4;
    float nilaiAkhir;
    string indeks;
};

float hitungNilaiAkhir(NilaiSTD dataMhs) {
    return (0.30 * dataMhs.clo1) + (0.30 * dataMhs.clo2) + (0.20 * dataMhs.clo3) + (0.20 * dataMhs.clo4);
}

int main() {
    NilaiSTD mahasiswa;

    cout << "--- Input 4 Nilai CLO untuk Dihitung Nilai Akhirnya ---" << endl;
    cout << "CLO 1: "; cin >> mahasiswa.clo1;
    cout << "CLO 2: "; cin >> mahasiswa.clo2;
    cout << "CLO 3: "; cin >> mahasiswa.clo3;
    cout << "CLO 4: "; cin >> mahasiswa.clo4;

    // Panggil fungsi untuk menghitung nilai akhirnya
    float nilai_akhir_hasil = hitungNilaiAkhir(mahasiswa);

    cout << fixed << setprecision(2);
    cout << "\nHasil perhitungan nilai akhir: " << nilai_akhir_hasil << endl;

    return 0;
}
```

Screenshots Output

```
lovinpeace@LVNPC-MAC Unguided % ./unguided6
--- Input 4 Nilai CLO untuk Dihitung Nilai Akhirnya ---
CLO 1: 70
CLO 2: 90
CLO 3: 80
CLO 4: 89

Hasil perhitungan nilai akhir: 81.80
```

Program ini memperlihatkan penggunaan *struct* dan fungsi untuk menghitung nilai akhir mahasiswa berdasarkan bobot masing-masing *Course Learning Outcome (CLO)*. Sebuah *struct* bernama *NilaiSTD* didefinisikan dengan empat atribut *CLO*, nilai akhir, serta indeks huruf. Fungsi hitungNilaiAkhir menerima parameter berupa objek *NilaiSTD*, kemudian menghitung nilai akhir dengan rumus: 30% untuk *CLO 1*, 30% untuk *CLO 2*, 20% untuk *CLO 3*, dan 20% untuk *CLO 4*. Pada fungsi main, pengguna diminta untuk memasukkan empat nilai *CLO* melalui *input*. Data tersebut diproses oleh fungsi hitungNilaiAkhir dan hasilnya ditampilkan ke layar

dengan format dua angka di belakang koma. Program ini sekaligus menunjukkan penerapan *struct*, fungsi, operasi aritmatika berbobot, serta format keluaran numerik dalam C++.

Unguided 7

```cpp
#include <iostream>
#include <string>

using namespace std;

string tentukanIndeks(float skorAkhir) {
    if (skorAkhir > 80) return "A";
    else if (skorAkhir > 70) return "AB";
    else if (skorAkhir > 65) return "B";
    else if (skorAkhir > 60) return "BC";
    else if (skorAkhir > 50) return "C";
    else if (skorAkhir > 40) return "D";
    else return "E";
}

int main() {
    float skor_input;

    cout << "Masukkan nilai akhir : ";
    cin >> skor_input;

    // Panggil fungsi dengan nilai dari input user
    string indeks_hasil = tentukanIndeks(skor_input);

    cout << "Untuk nilai " << skor_input << ", indeksnya adalah: " << indeks_hasil << endl;

    return 0;
}
```

Screenshots Output

```
lovinpeace@LVNPC-MAC Unguided % ./unguided7
Masukkan nilai akhir : 90
Untuk nilai 90, indeksnya adalah: A
```

Program ini menunjukkan penggunaan *struct* dan fungsi dalam C++. Sebuah *struct* bernama *NilaiSTD* dibuat untuk menyimpan empat nilai *Course Learning Outcome (CLO)*, nilai akhir, serta indeks huruf. Fungsi gabungNilaiStd berfungsi untuk menerima empat nilai *CLO* sebagai parameter, kemudian mengembalikan sebuah objek *NilaiSTD* dengan data tersebut, sementara nilai akhir diinisialisasi dengan 0 dan indeks dengan “N/A”. Pada fungsi main, pengguna diminta untuk memasukkan nilai empat *CLO* melalui *input*. Data yang diperoleh diproses oleh fungsi gabungNilaiStd dan hasilnya disimpan dalam variabel mahasiswa. Program menampilkan pesan bahwa nilai berhasil disimpan, serta menampilkan kembali salah satu nilai *CLO* yang tersimpan di dalam *struct*.

C. Kesimpulan

Praktikum ini berhasil memperlihatkan penerapan konsep dasar pemrograman dalam C++ melalui penggunaan *struct*, fungsi, array, pointer, serta struktur kontrol. Dengan memanfaatkan *struct* sebagai representasi *Abstract Data Type (ADT)*, data dapat dikelola secara lebih sistematis dan mudah diakses. Penerapan fungsi seperti gabungNilaiStd dan hitungNilaiAkhir membuktikan efektivitas pendekatan modular dalam memecah persoalan kompleks menjadi bagian yang lebih sederhana dan dapat digunakan kembali. Selain itu, penguasaan input-output, operasi aritmatika berbobot, hingga penggunaan *looping* untuk array satu dan dua dimensi menunjukkan pentingnya keterampilan teknis dalam membangun program yang terstruktur. Secara keseluruhan, praktikum ini mendukung pemahaman mahasiswa terhadap konsep fundamental pemrograman sekaligus melatih logika penyelesaian masalah secara bertahap

D. Referensi

[1] Robins, A., Rountree, J., & Rountree, N. (2003). *Learning and Teaching Programming: A Review and Discussion*. Computer Science Education, 13(2), 137–172.

[2] Lister, R. (2011). *Concrete and Other Neo-Piagetian Forms of Reasoning in the Novice Programmer*. Proceedings of the Thirteenth Australasian Computing Education Conference, 9–18.

[3] Bennedsen, J., & Caspersen, M. E. (2007). *Failure Rates in Introductory Programming*. ACM SIGCSE Bulletin, 39(2), 32–36.