LEMBAR KERJA PRAKTIKUM 1 STRUKTUR DATA

SEMESTER GENAP 2025-2026

REVIEW : POINTER, ARRAY, STRUCT, STL, OPERASI FILE

A. Tujuan Praktikum

- Mahasiswa dapat memahami penggunaan pointer, array, dan struct pada C++
- Mahasiswa dapat menuliskan kode program dengan menggunakan pointer, array, dan struct pada C++
- Mahasiswa dapat memahami STL dan FILE pada C++

B. Teori

Bahasa C++

Bahasa C++ dibangun berdasarkan bahasa C dan dilengkapi dengan banyak fitur baru seperti dukungan terhadap pemrograman berorientasi objek, template, dan exception. C++ adalah bahasa yang handal dan fleksibel dan cocok digunakan untuk mempelajari cara bekerja struktur data dasar, baik dalam bentuk kontainer yang disediakan oleh STL maupun dengan membuat sendiri kontainer tersebut dengan menggunakan pointer.

Mata kuliah Struktur Data hanya akan mengenalkan fitur C++ yang berhubungan dengan struktur data. Anda diharapkan menguasai konsep pointer, struct, dan alokasi memori yang telah dipelajari pada Algoritme dan Dasar Pemrograman. Keduanya akan menjadi hal dasar yang digunakan di hampir seluruh program yang Anda buat pada mata kuliah ini.

Walaupun sintaks program C++ sekilas terlihat mirip dengan bahasa C, terdapat beberapa perbedaan mendasar yang perlu Anda ketahui. Potongan program berikut, program sederhana untuk menghitung jumlah dua bilangan, akan menunjukkan beberapa perbedaan tersebut.

```
1  /*
2   * Program 1: Menghitung jumlah dua buah bilangan bulat
3   */
4  #include <cstdlib>
5  #include <iostream>


6  int main(){
7     int x, y;
8     std::cout << “Masukkanlah dua buah angka yang akan dijumlahkan:”;
9     std::cin >> x >> y; //Menerima input dari pengguna
10    int sum = x + y;    //Operasi penjumlahan
11
12    //Mencetak hasil penjumlahan
13    std::cout << “Jumlah kedua bilangan tersebut ialah: ” << sum <<
14                 std::endl;
```

```
15      return EXIT_SUCCESS; //Program berakhir dengan sukses
16  }
```

Bagi Anda yang telah membuat program dengan C, beberapa perbedaan muncul pada potongan kode di atas. *Header file* **cstdlib**[1] yang digunakan pada baris ke-5 berisi beberapa fungsi utilitas dan definisi standar (misal: EXIT_SUCCESS). *Header file* **iostream**[2] berisi fungsi *input*/*output* standar. Kedua *header file* ini akan sering Anda gunakan saat membuat program C++;

Baris 8 merupakan perintah untuk mencetak suatu string ke layar dengan menggunakan kata kunci **cout** dan operator keluaran "**<<**". Baris 9 merupakan perintah untuk menerima masukan dari layar, dengan menggunakan kata kunci **cin** dan operator masukan ">>". Akan tetapi, kedua perintah tersebut lebih lambat dibanding fungsi `scanf()` dan `printf()` pada bahasa C.

Awalan **`std::`** mengindikasikan bahwa `cout` dan `cin` merupakan objek dalam pustaka standar (std). **`std::endl`** setara dengan karakter *newline* (\n) untuk menandakan baris yang baru. Untuk mempersingkat program, Anda dapat menambahkan pernyataan **`using`** seperti berikut:

```
1  #include <cstdlib>
2  #include <iostream>
3
4  using namespace std;
5  int main(){
6     ...
7     cout << "Masukkanlah dua buah angka yang akan dijumlahkan:";
8     cin >> x >> y;
9     ...
```

Ketika dijalankan, program di atas akan mengeluarkan *output* berupa:

```
Masukkanlah dua buah angka yang akan dijumlahkan:1 20
Jumlah kedua bilangan tersebut ialah 21
```

Bagi Anda yang ingin mempelajari fitur C++ lebih lanjut, kami menyarankan Anda untuk membaca referensi C++ yang dapat diakses pada alamat www.cplusplus.com/reference.

[1] http://www.cplusplus.com/reference/cstdlib/
[2] http://www.cplusplus.com/reference/iostream/

Pointer

Setiap variabel memiliki tempat penyimpanan di memori. Pointer adalah variabel yang memuat alamat memori. Dua buah operator utama digunakan dalam pointer, yaitu & untuk mendapatkan alamat dari suatu variabel (misal: &x untuk mendapatkan alamat dari x) serta * untuk mendapatkan nilai yang disimpan pada alamat tersebut.

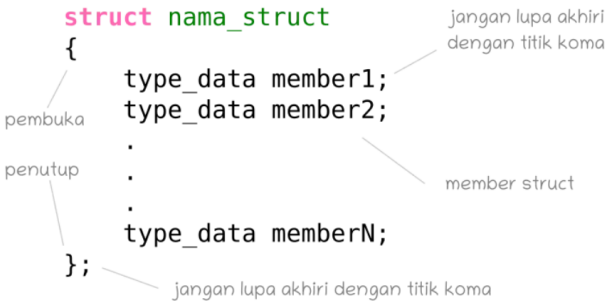
Pointer sangat berguna saat membangun sebuah struktur data, terutama untuk menghubungkan objek pada struktur data. Hal ini akan dibahas lebih lanjut pada bab-bab selanjutnya. Sebagai pengingat, potongan program berikut memberikan contoh penggunaan pointer:

```
char ch = 'A';
char *p = &ch;      // p memuat alamat dari ch
cout << p;          // mencetak alamat dari ch
cout << &ch;        // mencetak alamat dari ch
cout << *p;         // mencetak nilai dari ch, yaitu 'A'
ch = 'B';           // ch sekarang berisi 'B'
cout << *p;         // mencetak nilai ch, yaitu 'B'
*p = 'C';           // ch sekarang berisi 'C'
cout << ch;         // mencetak nilai ch, yaitu 'C'

char c[] = {'I', 'P', 'B'};
char *p = c;                // p menunjuk ke c[0]
char *q = &c[0];            // q menunjuk ke c[0]
cout << c[0] << c[1] << c[2]; // keluaran: IPB
```

Struct

struct merupakan tipe data bentukan yang memungkinkan kita untuk membuat pengelompokan variabel dari tipe data yang berbeda.



Sumber : https://www.petanikode.com/c-struct/

Contoh deklarasi struct :

```
struct siswa{
    int no_induk ;
    std::string nama;
    float nilai ;
}
```

Struct siswa di atas, terdiri dari variable no_induk, nama, dan nilai, yang masing-masing memiliki tipe data yang berbeda. Struct siswa ini selanjutnya akan digunakan sebagai tipe data baru. Tipe data baru ini selanjutnya dapat kita gunakan untuk mendeklarasikan sebuah variable baru bertipe struct siswa, seperti pada contoh berikut :

```
struct siswa{
    int no_induk ;
    std::string nama;
    float nilai ;
};
siswa budi;
```

Variabel budi sekarang adalah sebuah variable bertipe siswa, di mana siswa adalah sebuah struct yang didalamnya berisi no_induk, nama, dan nilai.

STL (Standard Template Library)

STL (Standard Template Library) adalah kumpulan template kelas dan fungsi dalam bahasa C++ yang menyediakan struktur data dan algoritma umum sehingga programmer tidak perlu membuatnya dari nol. STL mendukung konsep generic programming, artinya struktur data dan algoritma dapat digunakan untuk berbagai tipe data. STL bertujuan untuk mempercepat proses pengembangan program, mengurangi kesalahan implementasi struktur data, serta membuat kode lebih ringkas, rapi, dan efisien.

STL terdiri dari empat komponen utama: Container, Iterator, Algorithm, dan Function Object

1. Container

Container adalah struktur data yang digunakan untuk menyimpan sekumpulan data.

Contoh container STL:

Container	Keterangan
vector	Array dinamis
list	Double linked list
deque	Double-ended queue

Container	Keterangan
stack	Struktur data LIFO
queue	Struktur data FIFO
set	Data unik dan terurut
map	Pasangan key–value

2. Iterator

Iterator berfungsi seperti pointer untuk mengakses elemen dalam container. Iterator digunakan untuk menelusuri isi container dan menghubungkan container dengan algoritma STL

Contoh:

vector<int>::iterator it;

3. Algorithm

Algorithm adalah kumpulan fungsi untuk memproses data dalam container.

Contoh algoritma STL:

- sort()
- find()
- count()
- reverse()
- max_element()

Algorithm bekerja menggunakan iterator, bukan container secara langsung.

4. Function Object (Functor)

Functor adalah objek yang diperlakukan seperti fungsi, biasanya digunakan untuk kustomisasi algoritma STL.

Contoh sederhana:

greater<int>()

Contoh Penggunaan Container

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

int main() {
   vector<int> data;

   data.push_back(10);
   data.push_back(20);
   data.push_back(30);

   for(int i = 0; i < data.size(); i++) {
      cout << data[i] << " ";
   }
   return 0;
}
```

vector bersifat dinamis (ukuran bisa bertambah otomatis).

Contoh Penggunaan Iterator

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

int main() {
   vector<int> data = {5, 10, 15};

   for(vector<int>::iterator it = data.begin(); it != data.end(); it++) {
      cout << *it << " ";
   }
   return 0;
}
```

Contoh Penggunaan Algorithm

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

int main() {
   vector<int> data = {30, 10, 20};
```

```
    sort(data.begin(), data.end());

    for(int x : data) {
        cout << x << " ";
    }
    return 0;
}
```

sort() mengurutkan data secara ascending.

Contoh Penggunaan Container map

```
#include <iostream>
#include <map>
using namespace std;

int main() {
    map<string, int> nilai;

    nilai["Ani"] = 85;
    nilai["Budi"] = 90;

    for(auto it = nilai.begin(); it != nilai.end(); it++) {
        cout << it->first << " : " << it->second << endl;
    }
    return 0;
}
```

map menyimpan data dalam bentuk key–value dan otomatis terurut berdasarkan key.

FILE

Operasi file di **C++** biasanya menggunakan pustaka <fstream> yang menyediakan kelas-kelas berikut:

1. **ofstream** untuk menulis ke file.
2. **ifstream** untuk membaca dari file.
3. **fstream** untuk operasi baca dan tulis sekaligus.

Perhatikan contoh program berikut, lalu coba pahami langkah per-langkahnya.

```
1. #include <iostream>
2. #include <fstream>
3. #include <string>
```

```
4. using namespace std;

5. int main() {
6. // 1. Membuat file
7. ofstream file("contoh.txt");
8. if (file.is_open()) {
9. cout << "File berhasil dibuat.\n";
10.      file.close();
11.      } else {
12.      cout << "Gagal membuat file.\n";
13.      }

14.      // 2. Menulis ke file
15.      file.open("contoh.txt", ios::out);
16.      if (file.is_open()) {
17.      file << "Ini adalah baris pertama.\n";
18.      file << "Ini adalah baris kedua.\n";
19.      cout << "Data berhasil ditulis ke file.\n";
20.      file.close();
21.      } else {
22.      cout << "Gagal membuka file untuk menulis.\n";
23.      }

24.      // 3. Membaca file
25.      ifstream readFile("contoh.txt");
26.      if (readFile.is_open()) {
27.      string line;
28.      cout << "Isi file:\n";
29.      while (getline(readFile, line)) {
30.      cout << line << endl;
31.      }
32.      readFile.close();
33.      } else {
34.      cout << "Gagal membuka file untuk membaca.\n";
35.      }

36.      return 0;
37.      }
```

C. Tugas

Task 1. Ketik dan jalankan program di bawah ini secara mandiri, pahami bagaimana masing-masing komponen pada program ini bekerja :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <vector>
```

```
// Struktur data
struct Mahasiswa {
   char nama[50];
   int nim;
   float ipk;
};

int main() {
   // Menggunakan POINTER
   int angka = 10;
   int *ptrAngka = &angka;

   printf("Nilai angka: %d\n", *ptrAngka);

   // Menggunakan ARRAY
   int arrayAngka[5] = {1, 2, 3, 4, 5};

   printf("Elemen-elemen array: ");
   for (int i = 0; i < 5; ++i) {
      printf("%d ", arrayAngka[i]);
   }
   printf("\n");

   // Menggunakan STRUCT
   struct Mahasiswa mhs1;
   strcpy(mhs1.nama, "John Doe");
   mhs1.nim = 12345;
   mhs1.ipk = 3.75;

   printf("Data Mahasiswa:\n");
   printf("Nama: %s\n", mhs1.nama);
   printf("NIM: %d\n", mhs1.nim);
   printf("IPK: %.2f\n", mhs1.ipk);

   // Menggunakan STL (vector)
   std::vector<int> vectorAngka;
   for (int i = 0; i < 5; ++i) {
```

```
    vectorAngka.push_back(i * 2);
  }

  printf("Elemen-elemen vector: ");
  for (int i = 0; i < vectorAngka.size(); ++i) {
    printf("%d ", vectorAngka[i]);
  }
  printf("\n");

  // Modifikasi program
  // 1. Modifikasi nilai variabel angka menggunakan pointer
  *ptrAngka = 20;

  // 2. Modifikasi IPK mahasiswa mhs1 menggunakan pointer
  float *ptrIpk = &mhs1.ipk;
  *ptrIpk = 3.90;

  // 3. Tambahkan dua elemen baru ke dalam vectorAngka
  vectorAngka.push_back(12);
  vectorAngka.push_back(15);

  // Tampilkan hasil setelah modifikasi
  printf("Setelah dimodifikasi:\n");
  printf("Nilai angka setelah dimodifikasi: %d\n", *ptrAngka);
  printf("IPK mahasiswa setelah dimodifikasi: %.2f\n", mhs1.ipk);
  printf("Elemen-elemen vector setelah dimodifikasi: ");
  for (int i = 0; i < vectorAngka.size(); ++i) {
    printf("%d ", vectorAngka[i]);
  }
  printf("\n");

  return 0;
}
```

Task 2. Video dan Hasil Diskusi (dikumpulkan selambatnya satu minggu setelah praktikum ini)

Setelah Anda menjalankan program di task 1, diskusikan tugas di bawah ini dengan kelompok Anda, lalu buat video diskusinya dan tuliskan hasil diskusinya (PDF), masukkan ke gdrive, lalu kumpulkan link gdrive pada form berikut: https://forms.gle/JCWa8XJy9n7Vnc7r7

1. Modifikasilah program agar dapat menyimpan informasi lebih dari satu mahasiswa menggunakan array dari struct data Mahasiswa. Berikan penjelasan mengenai modifikasi yg Anda lakukan.
2. Tambahkan sebuah fungsi untuk mencetak semua mahasiswa beserta informasi mereka menggunakan pointer. Apa kelebihan dan kekurangan cara ini dibandingkan penggunaan index array?
3. Gantilah array yang digunakan untuk menyimpan mahasiswa dengan vector dari struct data Mahasiswa. Apa kelebihan dan kekurangan penggunaan array dan vector?
4. Tambahkan dua mahasiswa baru ke dalam vector tersebut menggunakan fungsi dari STL. Bagaimana cara kerja fungsi dari STL ini?
5. Implementasikan fungsi atau metode untuk mencari dan menampilkan mahasiswa dengan NIM tertentu.

Selamat Mengerjakan