LEMBAR KERJA PRAKTIKUM 2 STRUKTUR DATA

VECTOR & ARRAY

A. Tujuan Praktikum

- Mahasiswa dapat membedakan penggunaan array dan vector.
- Mahasiswa dapat mengimplementasikan array, array dinamis, dan vector.

B. Teori

Struktur data merupakan cara untuk menyimpan, mengorganisasi, dan mengelola data agar dapat digunakan secara efisien. Pemilihan struktur data yang tepat sangat berpengaruh terhadap kemudahan implementasi program serta efisiensi operasi seperti penambahan, penghapusan, dan pencarian data. Pada praktikum ini, struktur data yang dibahas adalah array dan vector, yang keduanya sering digunakan dalam pemrograman namun memiliki karakteristik yang berbeda.

1. Array

Array adalah struktur data linier yang digunakan untuk menyimpan sekumpulan elemen data dengan tipe data yang sama. Elemen-elemen array disimpan secara kontinyu di dalam memori, sehingga setiap elemen dapat diakses secara langsung menggunakan indeks. Indeks array biasanya dimulai dari nol, sehingga elemen pertama berada pada indeks ke-0.

Keunggulan utama array adalah kemudahan dan kecepatan akses elemen, karena sistem dapat langsung menuju alamat memori yang sesuai berdasarkan indeks. Oleh karena itu, operasi akses elemen pada array bersifat konstan. Namun, array memiliki keterbatasan dalam hal fleksibilitas ukuran. Pada array statis, ukuran array harus ditentukan sejak awal dan tidak dapat diubah selama program berjalan. Hal ini dapat menyebabkan pemborosan memori atau keterbatasan kapasitas jika kebutuhan data berubah.

2. Array Dinamis

Array dinamis merupakan pengembangan dari array statis yang memungkinkan alokasi memori dilakukan secara dinamis saat program dijalankan. Dengan array dinamis, ukuran array dapat ditentukan berdasarkan kebutuhan dan dapat diubah selama eksekusi program menggunakan mekanisme alokasi dan dealokasi memori.

Penggunaan array dinamis memberikan fleksibilitas yang lebih tinggi dibandingkan array statis, karena memori dapat digunakan sesuai kebutuhan. Namun, array dinamis memerlukan pengelolaan memori secara manual, sehingga programmer harus berhati-hati dalam melakukan alokasi dan dealokasi untuk menghindari kesalahan seperti kebocoran memori. Secara konsep, cara akses elemen pada array dinamis tetap sama seperti array biasa, yaitu menggunakan indeks.

3. Vector

Vector merupakan struktur data yang disediakan dalam bahasa C++ sebagai bagian dari Standard Template Library (STL). Secara konsep, vector dapat dipandang sebagai array dinamis yang pengelolaan memorinya dilakukan secara otomatis oleh sistem. Vector mampu menyesuaikan ukurannya secara otomatis ketika elemen ditambahkan atau dihapus.

Seperti array, elemen pada vector dapat diakses menggunakan indeks. Selain itu, vector menyediakan berbagai fungsi bawaan untuk menambahkan, menghapus, dan menelusuri elemen, sehingga mempermudah pengembangan program. Penambahan elemen di akhir vector umumnya efisien, namun penyisipan atau penghapusan elemen di tengah atau di awal vector tetap memerlukan pergeseran elemen lain, sehingga jumlah langkah yang dibutuhkan bergantung pada banyaknya data.

4. Perbandingan Array dan Vector

Perbedaan utama antara array dan vector terletak pada fleksibilitas ukuran dan pengelolaan memori. Array statis bersifat sederhana dan cepat diakses, tetapi tidak fleksibel. Array dinamis memberikan fleksibilitas ukuran, namun memerlukan pengelolaan memori manual. Vector menggabungkan fleksibilitas array dinamis dengan kemudahan penggunaan karena pengelolaan memorinya dilakukan secara otomatis.

Melalui praktikum ini, mahasiswa diharapkan dapat memahami kapan sebaiknya menggunakan array dan kapan menggunakan vector, serta mampu mengimplementasikan keduanya sesuai dengan kebutuhan permasalahan yang dihadapi.

C. Latihan : Implementasikan Program di bawah ini

PROGRAM 1. Ketikkan program di bawah ini, dan simpan hasil implementasi anda pada file **Prog1_NIM.cpp**

```
1. #include <stdio.h>
2. #include <stdlib.h>

3. int main() {

4. // Deklarasi dan alokasi array dinamis
5. int *dynamicArray = (int *)malloc(5 * sizeof(int));

6. // Inisialisasi elemen array dinamis
7. for (int i = 0; i < 5; ++i) {
        dynamicArray[i] = i + 1;
8. }

9. // Akses dan cetak elemen array
10.   for (int i = 0; i < 5; ++i) {
        printf("%d ", dynamicArray[i]);
11.   }
```

```
12.   // Dealokasi array dinamis
13.   free(dynamicArray);
14.   return 0;
15.   }
```

PROGRAM 2. Simpan hasil implementasi anda pada file **Prog2_NIM.cpp**

Diantara baris 11 dan 12 pada PROGRAM 1 di atas, sisipkan potongan baris berikut ini:

```
    // Penambahan alokasi array di tengah program
    int newSize = 8;
    dynamicArray = (int *)realloc(dynamicArray, newSize * sizeof(int));

    // Inisialisasi elemen tambahan
    for (int i = 5; i < newSize; ++i) {
        dynamicArray[i] = i + 1;
    }

    // Akses dan cetak setelah penambahan
    printf("\nArray setelah penambahan : ");
    for (int i = 0; i < newSize; ++i) {
        printf("%d ", dynamicArray[i]);
    }
```

PROGRAM 3. Simpan hasil implementasi anda pada file **Prog3_NIM.cpp**

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

int main() {

int n, i;
int *a, *b;
scanf("%d", &n);

a = (int*) malloc (n * sizeof(int));
b = (int*) malloc (n * sizeof(int));

for (i = 0; i<n; i++) {
      a[i] = rand()%10;
      b[n-i-1] = a[i];
}

for (i = 0; i<n; i++) printf("%d ", a[i]);
for (i = 0; i<n; i++) printf("%d ", b[i]);
```

```
free(a);
free(b);
return 0;
}
```

PROGRAM 4. Simpan hasil implementasi anda pada file **Prog4_NIM.cpp**

```
#include <bits/stdc++.h>
using namespace std;

int* getValues()
{

   int *arr = (int *) malloc(10*sizeof(int));
   for (int i = 0; i < 10; i++)
         arr[i] = i + 1;

   return arr;
}

int main()
{

   int* array;
   array = getValues();

   for (int i = 0; i < 10; i++) {
         cout << "*(array + " << i << ") : ";
         cout << *(array + i) << endl;
   }

   return 0;
}
```

PROGRAM 5. Simpan hasil implementasi anda pada file **Prog5_NIM.cpp**

```
#include <bits/stdc++.h>
using namespace std;

vector<int> getValues()
{
    vector<int> v;
    for (int i = 0; i < 10; i++)
```

```
        v.push_back(i + 1);
    return v;
}

int main()
{
    vector<int> get;
    get = getValues();

    // Menggunakan iterator untuk mengakses elemen vektor
    vector<int>::iterator it;
    for (it = get.begin(); it != get.end(); ++it)
        cout << *it << " ";

    return 0;
}
```

D. Diskusikan hal-hal berikut ini bersama asisten dan teman-teman di kelas praktikum kalian (hasil diskusi tidak perlu dituliskan) :

Program 1

1. Apakah output dari Program 1?
2. Apakah yang terjadi jika baris 13 dihapus?

Program 2

1. Apakah output Program 2?
2. Apakah isi array yang sudah ada sebelumnya berubah?
3. Apa yang dapat Anda simpulkan mengenai penggunaan malloc dan realloc?

Program 3

1. Apakah bagian berwarna kuning bisa diubah menggunakan pointer style untuk mengakses array?
2. Bagaimana mengubahnya?

Program 4 dan 5

1. Jalankan PROGRAM 4 dan PROGRAM 5. Apakah outputnya sama?
2. Apakah it, begin() dan end() pada PROGRAM 5?
3. Apa yang dapat Anda simpulkan mengenai perbedaan array dan vector dari kedua program di atas?
4. Pada PROGRAM 5, tambahkan beberapa baris perintah untuk melakukan penghapusan seluruh elemen sebelum return 0, menggunakan erase() atau pop_back(). Simpan hasilnya pada PROGRAM 5 (**`Prog5_NIM.cpp`).**

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

Setelah Anda menjalankan program-program di atas, 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

Kasus 1 – Data Nilai Praktikum

Sebuah program digunakan untuk menyimpan nilai praktikum 20 mahasiswa dalam satu kelas.
Jumlah mahasiswa sudah pasti dan tidak akan berubah selama program berjalan.
Data hanya dibaca dan ditampilkan, tidak ada penambahan atau penghapusan.

Pertanyaan Diskusi:

1. Struktur data apa yang paling tepat digunakan pada kasus ini?
2. Apakah penggunaan vector memberikan keuntungan signifikan dibanding array?
3. Apa potensi kerugian jika ukuran array dibuat terlalu besar “untuk jaga-jaga”?
4. Kapan pendekatan seperti ini tidak lagi cocok?

Kasus 2 – Input Data dari User

Sebuah program menerima jumlah data dari input user, misalnya jumlah sensor atau jumlah mahasiswa, yang baru diketahui saat program dijalankan. Di tengah eksekusi, ternyata diperlukan penambahan data baru.

Pertanyaan Diskusi:

1. Mengapa array statis kurang sesuai untuk kasus ini?
2. Apa kelebihan penggunaan array dinamis dibanding array statis?
3. Berdasarkan Program 2, apa risiko yang muncul jika realloc tidak dikelola dengan benar?
4. Menurut Anda, bagian mana dari pengelolaan array dinamis yang rawan kesalahan?

Kasus 3 – Aplikasi Pengolah Data Sementara (Program 3)

Sebuah aplikasi melakukan:

- membaca N data acak
- membuat salinan data tersebut dalam urutan terbalik
- data hanya dipakai sementara, lalu program selesai

Pertanyaan Diskusi:

1. Mengapa program ini menggunakan array dinamis, bukan array statis?
2. Apakah vector bisa menggantikan peran a dan b pada Program 3?

3. Jika menggunakan vector, bagian mana dari kode yang menjadi lebih sederhana?
4. Dalam konteks ini, apakah penggunaan array dinamis masih relevan?

Kasus 4 – Fungsi yang Mengembalikan Kumpulan Data (Program 4 vs 5)

Sebuah fungsi bertugas menghasilkan sekumpulan data dan mengembalikannya ke main() untuk diproses lebih lanjut.
Bandingkan dua pendekatan:

- PROGRAM 4: fungsi mengembalikan pointer array
- PROGRAM 5: fungsi mengembalikan vector

Pertanyaan Diskusi:

1. Apa risiko penggunaan array dinamis pada PROGRAM 4?
2. Mengapa PROGRAM 5 relatif lebih aman dari kesalahan memori?
3. Dari sisi programmer pemula, pendekatan mana yang lebih mudah dipahami?
4. Dalam proyek besar, pendekatan mana yang lebih mudah dirawat (maintainable)?

Kasus 5 – Sistem yang Terus Bertambah Datanya

Sebuah aplikasi mencatat riwayat aktivitas pengguna.
Setiap aktivitas baru akan ditambahkan ke akhir data, dan jumlah aktivitas tidak dapat diprediksi.

Pertanyaan Diskusi:

1. Struktur data apa yang paling sesuai untuk kasus ini?
2. Mengapa penggunaan vector lebih tepat dibanding array dinamis?
3. Apa konsekuensi performa jika data dihapus di tengah-tengah vector?
4. Dalam kondisi apa vector tidak menjadi pilihan terbaik?