

LAPORAN PRAKTIKUM

STRUKTUR DATA

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

LAPORAN AWAL

PERTEMUAN 1

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 1

TIPE DATA & HIRARKI

LANDASAN TEORI

Tipe data adalah klasifikasi nilai yang menentukan jenis data yang dapat disimpan dalam variabel, seperti integer, float, char, dan boolean. Dalam pemrograman, tipe data berfungsi agar komputer mengetahui bagaimana data diproses dan berapa besar memori yang dibutuhkan. Selain tipe data dasar, terdapat pula tipe data majemuk seperti array, record, dan pointer. Hirarki data menggambarkan tingkatan dalam struktur penyimpanan data mulai dari bit, byte, field, record, file, hingga database. Pemahaman tipe data dan hirarki data sangat penting karena menjadi dasar dalam membangun struktur data yang efisien dan sistematis.

SOAL/LATIHAN TUGAS

Latihan.1

Kerjakan soal dibawah ini dengan menggunakan Bahasa pemograman C++ dan mengujinya di compiler C++

1. Diberikan tiga nilai dari tipe data yang berbeda. Anda diminta untuk menyimpan nilai-nilai ini dalam variabel tipe data yang sesuai dan mencetaknya.

Input:

Satu-satunya baris masukan berisi tiga nilai yang dipisahkan spasi dari jenis yang berbeda.

Deskripsi tipe data dari nilai:

- 1) Nilai pertama adalah tipe integer.
- 2) Nilai kedua adalah tipe karakter.
- 3) Nilai ketiga adalah tipe float.

Output:

Output tiga baris seperti yang dijelaskan:

- 1) Baris pertama akan berisi nilai pertama.
- 2) Baris kedua akan berisi nilai kedua
- 3) Baris ketiga akan berisi nilai ketiga.

2. Tulis program yang meminta pengguna untuk mengetik lebar dan tinggi persegi panjang dan kemudian menampilkan area dan keliling persegi panjang tersebut ke layar.
3. Tulis program yang menanyakan nilai siswa. Skor adalah angka dari 0-100. Terjemahkan skor menjadi nilai sesuai dengan batas berikutnya:

```

skor >= 90 ==> "A"
skor >= 80 ==> "B"
skor >= 70 ==> "C"
skor >= 60 ==> "D"
apa pun ==> "F"

```

jika skor 100 cetak "Skor sempurna!"

4. Tulis program yang meminta pengguna mengetikkan koordinat 2 titik, A dan B (dalam bidang), lalu tulis jarak antara A dan B.
5. Tulis program yang meminta pengguna untuk mengetikkan 2 bilangan bulat A dan B dan menukar nilai A dan B

PENYELESAIAN DAN PENJELASAN TUGAS MENGGUNAKAN SOURCE CODE C++

```

1 #include <iostream>
2 #include <iomanip> // untuk format output
3 #include <cmath> // untuk fungsi sqrt()
4 #include <limits> // untuk membersihkan input error
5 using namespace std;
6
7 // Fungsi bantu untuk membersihkan input jika salah
8 void clearInput() {
9     cin.clear(); // hapus flag error
10    cin.ignore(numeric_limits<streamsize>::max(), '\n');
11 }
12
13 // SOAL 1 : Baca 3 nilai (int, char, float)
14 void soal1() {
15     cout << "\n[Soal 1] Baca tiga nilai (int, char, float)\n";
16     int a; char b; float c;
17
18     cout << "Masukkan bilangan bulat: ";
19     if (!(cin >> a)) { clearInput(); cout << "Input tidak valid!\n"; return; }
20
21     cout << "Masukkan karakter: ";
22     if (!(cin >> b)) { clearInput(); cout << "Input tidak valid!\n"; return; }
23
24     cout << "Masukkan bilangan pecahan: ";
25     if (!(cin >> c)) { clearInput(); cout << "Input tidak valid!\n"; return; }
26
27     cout << fixed << setprecision(2);
28     cout << "\nHasil:\n";
29     cout << "Bilangan bulat : " << setw(10) << a << '\n';
30     cout << "Karakter : " << setw(10) << b << '\n';

```

Library dan Namespace

#include <iostream> → Mengaktifkan fungsi input/output (cin, cout).

#include <iomanip> → Mengatur tampilan output (misalnya jumlah digit desimal, jarak teks, dll).

#include <cmath> → Menyediakan fungsi matematika seperti sqrt().

#include <limits> → Digunakan untuk menangani kesalahan input.

using namespace std; → Supaya tidak perlu menulis std::cout, std::cin, dll.

Fungsi clearInput

cin.clear() → Membersihkan status error dari input.

cin.ignore() → Mengabaikan semua karakter yang masih tersisa di buffer input (misal pengguna salah mengetik huruf di tempat angka).

Fungsi ini dipanggil setiap kali input tidak valid agar program tidak crash dan bisa lanjut berjalan.

Fungsi soal 1

Program meminta tiga input dari pengguna: int, char, float.

Setiap kali input gagal, fungsi clearInput() dipanggil agar program tidak berhenti.

fixed dan setprecision(2) → menampilkan bilangan pecahan dengan dua angka di belakang koma.

setw(10) → menata tampilan agar kolom hasil sejajar.

```
31     cout << "Bilangan pecahan: " << setw(10) << c << "\n";
32 }
33
34 // -----
35 // SOAL 2 : Luas dan keliling persegi panjang
36 void soal2() {
37     cout << "\n[Soal 2] Luas dan Keliling Persegi Panjang\n";
38     double lebar, tinggi;
39     cout << "Masukkan lebar dan tinggi: ";
40     if (!(cin >> lebar >> tinggi)) { clearInput(); cout << "Input tidak valid!\n"; return; }
41
42     double luas = lebar * tinggi;
43     double keliling = 2 * (lebar + tinggi);
44
45     cout << fixed << setprecision(2);
46     cout << "\nLuas      = " << setw(10) << luas << '\n';
47     cout << "Keliling   = " << setw(10) << keliling << "\n";
48 }
49
50 // -----
51 // SOAL 3 : Konversi skor ke huruf dan cek skor 100
52 void soal3() {
53     cout << "\n[Soal 3] Konversi Skor ke Nilai Huruf\n";
54     int skor;
55     cout << "Masukkan skor (0 - 100): ";
56     if (!(cin >> skor)) { clearInput(); cout << "Input tidak valid!\n"; return; }
57
58     if (skor < 0 || skor > 100) {
59         cout << "Skor harus antara 0 dan 100!\n";
60     }
61 }
```

Fungsi soal 2

② Program membaca dua angka (lebar dan tinggi).

Menghitung:

$$\text{luas} = \text{lebar} * \text{tinggi}$$

$$\text{keliling} = 2 * (\text{lebar} + \text{tinggi})$$

Menampilkan hasil dengan dua angka desimal.

Jika input tidak valid, tampilkan pesan kesalahan

```

61 }
62
63 if (skor == 100) {
64     cout << "Skor sempurna! Nilai: A\n";
65     return;
66 }
67
68 char nilai;
69 if (skor >= 90) nilai = 'A';
70 else if (skor >= 80) nilai = 'B';
71 else if (skor >= 70) nilai = 'C';
72 else if (skor >= 60) nilai = 'D';
73 else nilai = 'E';
74
75 cout << "Nilai huruf: " << nilai << "\n";
76 }
77
78 // -----
79 // SOAL 4 : Hitung jarak antara dua titik
80 void soal4() {
81     cout << "\n[Soal 4] Jarak antara dua titik (x1, y1) dan (x2, y2)\n";
82     double x1, y1, x2, y2;
83     cout << "Masukkan x1 y1 x2 y2: ";
84     if (!(cin >> x1 >> y1 >> x2 >> y2)) { clearInput(); cout << "Input tidak valid!\n"; return; }
85
86     double dx = x2 - x1;
87     double dy = y2 - y1;
88     double jarak = sqrt(dx*dx + dy*dy);
89
90     cout << fixed << setprecision(3);

```

Fungsi soal 3

Penjelasan:

1. Pengguna memasukkan nilai skor.
2. Jika skor tidak berada di antara 0–100 → tampil pesan kesalahan.
3. Jika skor = 100 → tampil pesan khusus “*Skor sempurna!*”
4. Kondisi if–else digunakan untuk menentukan nilai huruf:

$\geq 90 \rightarrow A$

$\geq 80 \rightarrow B$

$\geq 70 \rightarrow C$

$\geq 60 \rightarrow D$

$< 60 \rightarrow E$

Fungsi soal 4

Penjelasan:

Input empat angka untuk dua titik (x1,y1) dan (x2,y2).

Rumus jarak = $\sqrt{((x2 - x1)^2 + (y2 - y1)^2)}$

Fungsi sqrt() dari <cmath> menghitung akar kuadrat.

setprecision(3) menampilkan hasil hingga tiga angka di belakang koma.

```

91     cout << "\nJarak antara kedua titik: " << jarak << "\n";
92 }
93
94 // -----
95 // SOAL 5 : Tukar nilai dua bilangan
96 void soal5() {
97     cout << "\n[Soal 5] Tukar dua bilangan\n";
98     int A, B;
99     cout << "Masukkan dua bilangan (A B): ";
100    if (!(cin >> A >> B)) { clearInput(); cout << "Input tidak valid!\n"; return; }
101
102    cout << "Sebelum ditukar: A = " << A << ", B = " << B << '\n';
103    swap(A, B);
104    cout << "Setelah ditukar: A = " << A << ", B = " << B << "\n";
105 }
106
107 // -----
108 // MENU UTAMA
109 int main() {
110     int pilihan;
111     do {
112         cout << "\n-----\n";
113         cout << "      MENU LATIHAN / TUGAS PERTEMUAN 1      \n";
114         cout << "-----\n";
115         cout << "1. Baca 3 Nilai (int, char, float)\n";
116         cout << "2. Luas dan Keliling Persegi Panjang\n";
117         cout << "3. Konversi Skor ke Nilai Huruf\n";
118         cout << "4. Jarak antara Dua Titik\n";
119         cout << "5. Tukar Dua Bilangan\n";
120         cout << "0. Keluar\n";
121
122         cout << "-----\n";
123         cout << "Pilih menu [0-5]: ";
124
125         if (!(cin >> pilihan)) { clearInput(); cout << "Input tidak valid!\n"; continue; }
126         switch (pilihan) {
127             case 1: soal1(); break;
128             case 2: soal2(); break;
129             case 3: soal3(); break;
130             case 4: soal4(); break;
131             case 5: soal5(); break;
132             case 0: cout << "Program selesai.\n"; break;
133             default: cout << "Pilihan tidak tersedia!\n"; break;
134         }
135     } while (pilihan != 0);
136
137     return 0;
138 }

```

Fungsi soal 5

Program membaca dua bilangan (A dan B).

swap(A, B) menukar nilai kedua variabel tanpa perlu variabel tambahan.

Program menampilkan nilai sebelum dan sesudah ditukar.

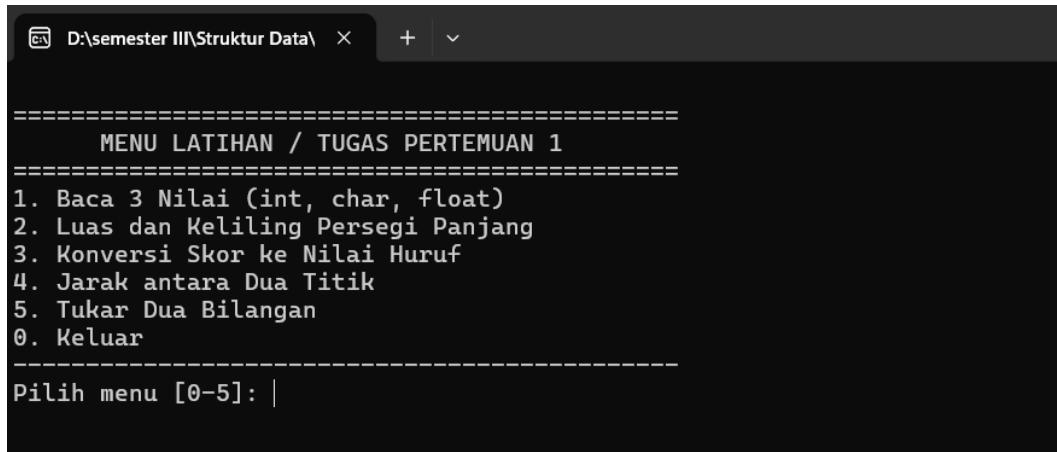
KESIMPULAN

Program ini melatih konsep dasar **struktur data dan algoritma** dalam pemrograman C++:

Penggunaan fungsi untuk modularisasi program. Validasi input agar program tahan terhadap kesalahan pengguna. Percabangan (if–else, switch) untuk

logika pemilihan. Loop (do-while) untuk interaksi berulang. Pemformatan output dengan iomanip agar hasil tampil rapi.

OUTPUT PROGRAM:



```
D:\semester III\Struktur Data\ + v
=====
 MENU LATIHAN / TUGAS PERTEMUAN 1
=====
1. Baca 3 Nilai (int, char, float)
2. Luas dan Keliling Persegi Panjang
3. Konversi Skor ke Nilai Huruf
4. Jarak antara Dua Titik
5. Tukar Dua Bilangan
0. Keluar
=====
Pilih menu [0-5]: |
```

MENU 1 — Baca 3 Nilai (int, char, float)

Program menampilkan tiga nilai yang kamu input.

MENU 2 — Luas dan Keliling Persegi Panjang

Program ini menampilkan lebar dan tinggi sesuai yang kamu input.

MENU 3 — Konversi Skor ke Nilai Huruf

Program ini akan meminta kamu memasukkan skor 0-100 misal skor yang dimasukkan Adalah 95 maka mendapat nilai huruf A

Skor 90–100 = A

80–89 = B

70–79 = C

60–69 = D

Di bawah 60 = E

MENU 4 — Jarak antara Dua Titik

Program ini akan menampilkan, Jarak antara dua titik (x1, y1) dan (x2, y2)

Contoh: masukkan x1 y1 x2 y2: 0 0 3 4 maka hasil nya akan 5.000

Rumus jarak:

$$\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$$

$$= \sqrt{(3-0)^2 + (4-0)^2} = \sqrt{9+16} = \sqrt{25} = 5$$

Ditampilkan tiga angka di belakang koma.

MENU 5 — Tukar Dua Bilangan

Program ini akan menukar dua bilangan

Contoh: masukkan bilangan A dan B ketik 10 20, lalu bilangan A 10 dan B 20 di tukar menjadi A 20 dan B 10

KESIMPULAN

Program ini menunjukkan bagaimana konsep dasar **struktur data dan algoritma** diterapkan dalam bahasa pemrograman **C++**.

Melalui lima soal yang dibuat menjadi satu program interaktif dengan menu pilihan, pengguna dapat memahami berbagai konsep penting dalam pemrograman dasar, yaitu:

1. Input dan Output (I/O):

Program menggunakan cin dan cout untuk membaca serta menampilkan data dengan berbagai tipe, seperti int, char, dan float.

Pemformatan output dilakukan dengan setw() dan setprecision() agar hasil tampil rapi.

2. Struktur Percabangan:

if-else dan switch-case digunakan untuk menentukan jalur eksekusi program berdasarkan kondisi tertentu, seperti konversi nilai huruf dan pemilihan menu.

3. Perulangan (Looping):

Struktur do...while membuat menu utama terus muncul sampai pengguna memilih keluar (0), sehingga program menjadi interaktif dan mudah digunakan.

4. Fungsi (Modularisasi):

Setiap soal dikelompokkan ke dalam fungsi terpisah (soal1() sampai soal5()), sehingga kode lebih teratur, mudah dipahami, dan mudah dikembangkan.

5. Validasi Input:

Fungsi clearInput() digunakan untuk mencegah error akibat input yang salah, menunjukkan praktik pemrograman yang aman dan profesional.

6. Penerapan Logika dan Matematika Dasar:

Soal-soal melatih kemampuan berpikir logis dan penggunaan rumus sederhana (seperti menghitung luas, keliling, dan jarak titik).

LAPORAN AWAL

PERTEMUAN 2

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PAMULANG Jl. Surya Kencana No.1 Pamulang Telp
(021)7412566, FAX. (021)741566 Tangerang Selatan

PERTEMUAN 2

ARRAY DAN POINTER

LANDASAN TEORI

Array merupakan struktur data statis yang menyimpan sejumlah elemen bertipe sama secara berurutan dalam memori. Setiap elemen diakses menggunakan indeks. Pointer adalah variabel yang menyimpan alamat memori suatu data, digunakan untuk mengakses atau memanipulasi data secara langsung. Kombinasi array dan pointer sering digunakan untuk mengoptimalkan akses data dan pemahaman struktur memori.

SOAL/LATIHAN TUGAS

Latihan 2.

Kerjakan soal dibawah ini dengan teliti.

1. Dideklarasikan Array 1 Dimensi yang dibuat dengan char x[12]. Jika diketahui &x[0] = 1000 Hexadecimal. Ditanya alamat elemen x[8] atau &x[8] =?
2. Dideklarasikan Array 1 Dimensi yang dibuat dengan int x[15]. Jika diketahui &x[3] = 1000 H, Ditanya &x[9] =...?
3. Dideklarasikan Array 2 Dimensi yang dibuat dengan float x[5][8]. Jika diketahui &x[0][0]= 1000 H, Ditanya &x[2][4]=...?
4. Dideklarasikan Array 2 Dimensi yang dibuat dengan long x[12][14]. Jika diketahui &x[0][0]= 1000 H, Ditanya &x[2][4]=...?
5. Dideklarasikan Array 3 Dimensi yang dibuat dengan int x [2][3] [5]. Jika diketahui &x [1][1][4]= 12EF H, Ditanya &x [0][0][3]=?

PENYELESAIAN DAN PENJELASAN TUGAS MENGGUNAKAN SOURCE CODE C++

Code 1

```

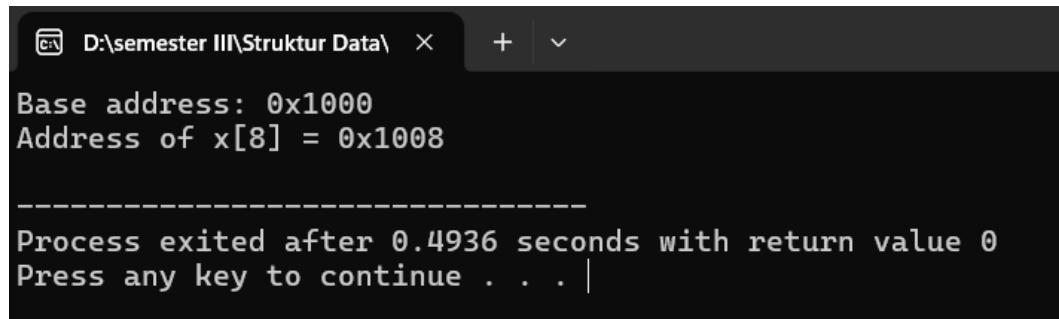
1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4
5 int main() {
6     uintptr_t base = 0x1000; // alamat &x[0]
7     size_t elemSize = 1;      // sizeof(char) = 1 byte
8     size_t index = 8;
9     uintptr_t addr = base + index * elemSize;
10    cout << "Base address: 0x" << hex << base << dec << "\n";
11    cout << "Address of x[8] = 0x" << hex << addr << dec << "\n";
12    return 0;
13 }

```

PENJELASAN CODE:

#include <iomanip> & iostream untuk cetak format.
 uintptr_t base = 0x1000; menyimpan alamat dasar dalam bentuk bilangan integer (bukan pointer nyata — simulasi).
 elemSize = 1 karena char ukuran 1 byte.
 addr = base + index * elemSize — rumus lokasi elemen satu dimensi.
 cout << hex menampilkan heksadesimal; dec kembalikan ke desimal bila perlu.

OUTPUT PROGRAM:



```

D:\semester III\Struktur Data> +
Base address: 0x1000
Address of x[8] = 0x1008
-----
Process exited after 0.4936 seconds with return value 0
Press any key to continue . . .

```

Penjelasan output:

Base address: 0x1000 menegaskan alamat awal.

Address of x[8] = 0x1008 karena $8 * 1 = 8$; $0x1000 + 0x8 = 0x1008$.

Artinya secara fisik dalam memori elemen x[8] berada 8 byte setelah x[0].

Code 2

```

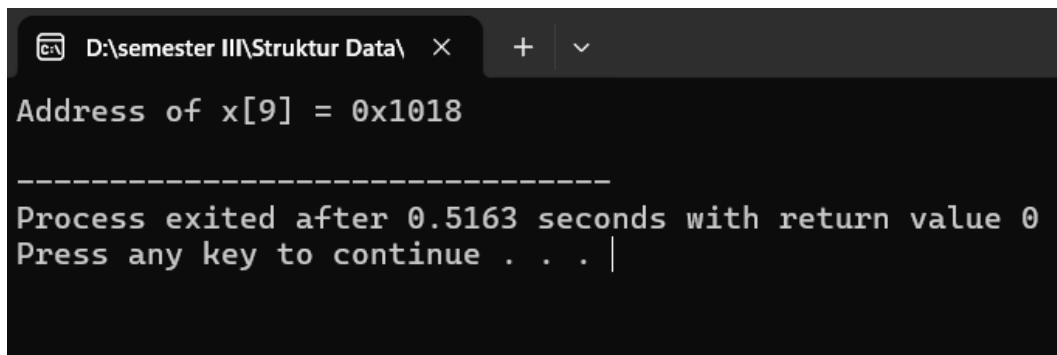
1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4
5 int main(){
6     uintptr_t addr_x3 = 0x1000; // alamat x[3]
7     size_t elemSize = 4; // sizeof(int)
8     size_t idx_given = 3;
9     size_t idx_target = 9;
10    uintptr_t base = addr_x3 - idx_given * elemSize;
11    uintptr_t addr = base + idx_target * elemSize;
12    cout << "Address of x[9] = 0x" << hex << addr << dec << "\n";
13
14 }

```

PENJELASAN CODE:

Kita diberi alamat x[3], bukan base. Rekomputasi base: base = addr_x3 - 3*4.
Lalu addr untuk x[9] dihitung dari base: base + 9*4.

OUTPUT CODE:



```

D:\semester III\Struktur Data\ Address of x[9] = 0x1018
-----
Process exited after 0.5163 seconds with return value 0
Press any key to continue . . .

```

Penjelasan output:

Jika $\&x[3] = 0x1000$, base = $0x1000 - 12 = 0x0FF4$.

$\&x[9] = \text{base} + 9*4 = 0x0FF4 + 36 = 0x1018$.

Output menampilkan 0x1018. Itu berarti x[9] berada 36 byte setelah base.

Code 3

```

1 #include <iostream>
2 using namespace std;
3 int main(){
4     uintptr_t base = 0x1000;
5     size_t elemSize = 4;
6     size_t cols = 8;
7     size_t i = 2, j = 4;
8     uintptr_t addr = base + ((i * cols) + j) * elemSize;
9     cout << hex << "Address x[2][4] = 0x" << addr << dec << "\n";
10
11 }
12

```

PENJELASAN CODE:

Untuk array 2D disimpan row-major: offset elemen $(i*cols + j)*elemSize$.

OUTPUT CODE:

```
D:\semester III\Struktur Data\ Address x[2][4] = 0x1050
-----
Process exited after 0.6814 seconds with return value 0
Press any key to continue . . . |
```

Penjelasan output:

$$\text{Offset} = (2*8 + 4) * 4 = (16+4)*4 = 20*4 = 80 \text{ (0x50).}$$

$$\text{Address} = 0x1000 + 0x50 = 0x1050.$$

Output 0x1050.

Code 4

```
1 #include <iostream>
2 using namespace std;
3 int main(){
4     uintptr_t base = 0x1000;
5     size_t elemSize = 8;
6     size_t cols = 14;
7     size_t i = 2, j = 4;
8     uintptr_t addr = base + ((i * cols) + j) * elemSize;
9     cout << hex << "Address x[2][4] = 0x" << addr << dec << "\n";
10    return 0;
11 }
```

PENJELASAN CODE & OUTPUT CODE

```
Address x[2][4] = 0x1100
-----
Process exited after 0.5527 seconds with return value 0
Press any key to continue . . . |
```

Penjelasan:

$$\text{Offset} = (2*14 + 4) * 8 = (28+4)*8 = 32*8 = 256 \text{ (0x100).}$$

$$\text{Address} = 0x1000 + 0x100 = 0x1100.$$

Output 0x1100

Code 5

```

1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4 int main(){
5     uintptr_t addr_given = 0x12EF; // &x[1][1][4]
6     size_t elem = 4;
7     size_t D2 = 3, D3 = 5;
8     size_t i_g=1,j_g=1,k_g=4;
9     uintptr_t base = addr_given - ((i_g*D2*D3 + j_g*D3 + k_g) * elem);
10    size_t i=0,j=0,k=3;
11    uintptr_t addr_target = base + ((i*D2*D3 + j*D3 + k) * elem);
12    cout << hex << "Base=0x" << base << "\n";
13    cout << "Address x[0][0][3] = 0x" << addr_target << dec << "\n";
14    return 0;
15 }

```

```

Base=0x128f
Address x[0][0][3] = 0x129b
-----
```

```

Process exited after 0.6192 seconds with return value 0
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE:

Hitung offset dari $x[1][1][4]$ ke base: $(1*3*5 + 1*5 + 4) * 4 = (15+5+4)*4 = 24*4 = 96$.

Base = $0x12EF - 96 = 0x12EF - 0x60 = 0x128F$ (hitung biner/heksasimal sesuai).

Target offset for $x[0][0][3] = 3*4 = 12$.

Address = base + 12 = $0x128F + 0xC = 0x129B$.

Output tampilkan base dan target. (Angka heksa mungkin perlu verifikasi digit, tapi langkah di atas menjelaskan proses).

KESIMPULAN

Code menunjukkan bagaimana cara menghitung dan memahami posisi data dalam memori menggunakan array dan pointer. Output program menampilkan alamat memori dari elemen-elemen array sesuai indeksnya, baik satu dimensi, dua dimensi, maupun tiga dimensi. Kesimpulannya, mahasiswa dapat memahami bahwa letak suatu elemen array dalam memori ditentukan oleh ukuran tipe data dan posisi indeksnya. Konsep ini penting karena menjadi dasar bagaimana data disimpan dan diakses secara efisien oleh komputer.

LAPORAN AKHIR

PERTEMUAN 1

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 1

TIPE DATA & HIRARKI

CONTOH CODE DAN OUTPUT PROGRAM PRAKTEK

```
1 // Program C++ Tipe data dan ukurannya
2 #include<iostream>
3 using namespace std;
4
5 int main()
6 {
7     cout << "Ukuran dari char : " << sizeof(char) << " byte" << endl;
8     cout << "Ukuran dari int : " << sizeof(int) << " bytes" << endl;
9     cout << "Ukuran dari short int : " << sizeof(short int) << " bytes" << endl;
10    cout << "Ukuran dari long int : " << sizeof(long int) << " bytes" << endl;
11    cout << "Ukuran dari signed long int : " << sizeof(signed long int) << " bytes" << endl;
12    cout << "Ukuran dari unsigned long int : " << sizeof(unsigned long int) << " bytes" << endl;
13    cout << "Ukuran dari float : " << sizeof(float) << " bytes" << endl;
14    cout << "Ukuran dari double : " << sizeof(double) << " bytes" << endl;
15    cout << "Ukuran dari wchar_t : " << sizeof(wchar_t) << " bytes" << endl;
16
17 }
18
```

```
Ukuran dari char : 1 byte
Ukuran dari int : 4 bytes
Ukuran dari short int : 2 bytes
Ukuran dari long int : 4 bytes
Ukuran dari signed long int : 4 bytes
Ukuran dari unsigned long int : 4 bytes
Ukuran dari float : 4 bytes
Ukuran dari double : 8 bytes
Ukuran dari wchar_t : 2 bytes

-----
Process exited after 0.4611 seconds with return value 0
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE

Program ini berfungsi untuk menampilkan ukuran (dalam byte) dari berbagai tipe data dasar C++ menggunakan operator sizeof().

Hasil dari output ini menunjukkan ukuran tiap tipe data di sistem dan compiler yang digunakan (kemungkinan g++/MinGW pada Windows 64-bit). Ukuran bisa sedikit berbeda jika dijalankan di sistem lain (misalnya Linux x64 biasanya long int = 8 byte, wchar_t = 4 byte).

KESIMPULAN

Operator sizeof() digunakan untuk mengetahui ukuran memori tipe data dalam byte. Ukuran setiap tipe data tergantung pada arsitektur komputer dan compiler. Program ini merupakan contoh dasar untuk memahami tipe data primitif dalam C++.

HASIL YANG DI DAPAT

1. char: 1 byte
2. int: 4 bytes
3. short int: 2 bytes
4. long int: 4 bytes
5. signed/unsigned long int: 4 bytes
6. float: 4 bytes
7. double: 8 bytes
- 8 .wchar_t: 2 bytes

LAPORAN AWAL

PERTEMUAN 3

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 3

STRUKTUR/RECORD

LANDASAN TEORI

Struct atau struktur data bentukan digunakan untuk menggabungkan beberapa variabel dengan tipe data berbeda menjadi satu kesatuan. Record memiliki konsep serupa dan digunakan untuk merepresentasikan objek nyata seperti data mahasiswa atau barang. Konsep ini memungkinkan penyimpanan data yang lebih kompleks dan terstruktur, menjadi dasar dalam pembuatan program berbasis data objek.

SOAL/LATIHAN TUGAS

Latihan.3

1. Tulis program untuk menyimpan dan mencetak No urut, Nama, usia dan nilai mahasiswa menggunakan struktur.
 2. Tulis program untuk menyimpan No urut. (mulai dari 1), nama dan umur 5 Mahasiswa kemudian cetak detail Mahasiswa dengan No urut 2.
 3. Tulis program untuk menyimpan dan mencetak no. urut, nama, umur, alamat dan nilai 15 mahasiswa menggunakan struktur.
 4. Kerjakan menu perpustakaan. Buat struktur yang berisi informasi buku seperti nomor akses, nama penulis, judul buku dan bendera untuk mengetahui apakah buku tersebut diterbitkan atau tidak.
 5. Buat menu di mana hal berikut dapat dilakukan.
 - 1 - Menampilkan informasi buku
 - 2 - Tambahkan buku baru
 - 3 - Tampilkan semua buku di perpustakaan penulis tertentu
 - 4 - Menampilkan jumlah buku dengan judul tertentu
 - 5 - Menampilkan jumlah total buku di perpustakaan
 - 6 - Terbitkan buku
- (Jika kita menerbitkan buku, maka jumlahnya akan berkurang 1 dan jika kita menambahkan buku, jumlahnya bertambah 1)

PENYELESAIAN DAN PENJELASAN TUGAS MENGGUNAKAN SOURCE CODE C++

Code 1

```

1 #include <iostream>
2 #include <vector>
3 #include <string>
4 using namespace std;
5
6 struct Mahasiswa {
7     int no;
8     string nama;
9     int usia;
10    float nilai;
11 };
12
13 int main(){
14     vector<Mahasiswa> v;
15     for(int i=1;i<=3;i++){
16         Mahasiswa m;
17         m.no = i;
18         cout << "Masukkan nama mahasiswa " << i << ": ";
19         getline(cin, m.nama);
20         cout << "Usia: "; cin >> m.usia;
21         cout << "Nilai: "; cin >> m.nilai;
22         cin.ignore(); // clear newline
23         v.push_back(m);
24     }
25     cout << "\nDaftar Mahasiswa:\n";
26     for(auto &m: v){
27         cout << m.no << ". " << m.nama << " | Usia: " << m.usia << " | Nilai: " << m.nilai << "\n";
28     }
29 }
30

```

```

Masukkan nama mahasiswa 1: budi
Usia: 20
Nilai: 78.5
Masukkan nama mahasiswa 2: siti
Usia: 21
Nilai: 82
Masukkan nama mahasiswa 3: andi
Usia: 19
Nilai: 90

Daftar Mahasiswa:
1. budi | Usia: 20 | Nilai: 78.5
2. siti | Usia: 21 | Nilai: 82
3. andi | Usia: 19 | Nilai: 90

-----
Process exited after 47.44 seconds with return value 0
Press any key to continue . . .

```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE:

Penjelasan kode

struct Mahasiswa mendefinisikan record.

vector<Mahasiswa> v; menampung banyak record secara dinamis.

Loop input 3 mahasiswa: getline untuk nama (mendukung spasi), cin untuk angka, lalu cin.ignore() untuk membersihkan newline sebelum getline berikutnya.

Menampilkan semua isi vector.

Penjelasan output

Contoh input:

Nama: Budi, Usia: 20, Nilai: 78.5

Nama: Siti, Usia: 21, Nilai: 82
Nama: Andi, Usia: 19, Nilai: 90

Code 2

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Buku {
5     int nomorAkses;
6     string penulis;
7     string judul;
8     int jumlah;
9 }
10
11 int main(){
12     vector<Buku> lib;
13     // contoh: tambah 2 buku otomatis
14     lib.push_back({101, "Taufik", "Algoritma Dasar", 3});
15     lib.push_back({102, "Rina", "Struktur Data", 2});
16     // tampil semua
17     cout << "Semua buku:\n";
18     for(auto &b: lib) cout << b.nomorAkses << " | " << b.penulis << " | " << b.judul << " | jml:" << b.jumlah << "\n";
19     // cari penulis "Rina"
20     string pencari = "Rina";
21     cout << "\nBuku penulis " << pencari << ":\n";
22     for(auto &b: lib) if(b.penulis == pencari) cout << b.judul << " (No: " << b.nomorAkses << ")\n";
23     // hitung jumlah buku berjudul "Struktur Data"
24     string jud = "Struktur Data";
25     int sum=0;
26     for(auto &b: lib) if(b.judul==jud) sum+=b.jumlah;
27     cout << "\nJumlah buku judul '"<<jud<<"' = "<<sum<<"\n";
28     // kurangi stok buku nomor 101
29     for(auto &b: lib) if(b.nomorAkses==101) { if(b.jumlah>0){ b.jumlah--; cout << "\nBuku '"<<b.judul<<"' diterbitkan. Sisa: "<<b.jumlah<<"\n"; } else cout << "Stok kosong\n"; }
30     return 0;
31 }
```

```
Semua buku:
101 | Taufik | Algoritma Dasar | jml:3
102 | Rina | Struktur Data | jml:2
```

```
Buku penulis Rina:
Struktur Data (No: 102)
```

```
Jumlah buku judul 'Struktur Data' = 2
```

```
Buku Algoritma Dasar diterbitkan. Sisa: 2
```

```
-----
Process exited after 0.8232 seconds with return value 0
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE:

Penjelasan code

vector<Buku> lib; menyimpan koleksi buku.

Menambahkan contoh data statis untuk demonstrasi.

Operasi pencarian dan penghitungan sederhana menggunakan loop for.

Mengurangi stok memeriksa b.jumlah > 0.

Penjelasan output

Menampilkan daftar buku.

Menampilkan buku penulis tertentu akan mencetak judul dan nomor akses.
Pengurangan stok memperlihatkan sisa stok buku setelah operasi.

KESIMPULAN

Code yang dibuat menggunakan struct dan record berhasil menampilkan kumpulan data yang lebih kompleks seperti data mahasiswa atau buku dalam sistem perpustakaan. Output menunjukkan data yang tersimpan dapat ditampilkan dengan rapi sesuai atributnya. Kesimpulannya, struct memudahkan pengelolaan data yang memiliki banyak jenis informasi berbeda, serta mempermudah representasi objek dunia nyata ke dalam bentuk data terstruktur.

LAPORAN AKHIR

PERTEMUAN 2

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 2

ARRAY DAN POINTER

CONTOH CODE DAN OUTPUT PROGRAM PRAKTEK

Contoh program 1

```
1 // Program C++ untuk mencetak elemen array dua dimensi
2 #include<iostream>
3 using namespace std;
4
5 int main()
6 {
7     // array dengan 4 baris dan 3 kolom.
8     int x[4][3] = {{0,1,2}, {3,4,5}, {6,7,8}, {9,10,11}};
9
10    // keluarkan nilai setiap elemen array
11    for (int i = 0; i < 4; i++)
12    {
13        for (int j = 0; j < 3; j++)
14        {
15            cout << "Elemen di x[" << i << "][" << j << "]: ";
16            cout << x[i][j] << endl;
17        }
18    }
19    return 0;
20 }
```

```
Elemen di x[0][0]: 0
Elemen di x[0][1]: 1
Elemen di x[0][2]: 2
Elemen di x[1][0]: 3
Elemen di x[1][1]: 4
Elemen di x[1][2]: 5
Elemen di x[2][0]: 6
Elemen di x[2][1]: 7
Elemen di x[2][2]: 8
Elemen di x[3][0]: 9
Elemen di x[3][1]: 10
Elemen di x[3][2]: 11

-----
Process exited after 0.5813 seconds with return value 0
Press any key to continue . . . |
```

Contoh program 2

```
1 // Program C++ untuk mencetak elemen pada Array Tiga Dimensi
2 #include<iostream>
3 using namespace std;
4
5 int main()
6 {
7     // menginisialisasi array 3 dimensi
8     int x[3][2][3] =
9     {
10         { {0,1,2}, {3,4,5} },
11         { {6,7,8}, {9,10,11} },
12         { {12,13,14}, {15,16,17} }
13     };
14
15     // keluarkan nilai setiap elemen
16     for (int i = 0; i < 3; ++i)
17     {
18         for (int j = 0; j < 2; ++j)
19         {
20             for (int k = 0; k < 3; ++k)
21             {
22                 cout << "Elemen di x[" << i << "][" << j << "][" << k << "] = "
23                 << x[i][j][k] << endl;
24             }
25         }
26     }
27     return 0;
28 }
```

```
Elemen di x[0][0][0] = 0
Elemen di x[0][0][1] = 1
Elemen di x[0][0][2] = 2
Elemen di x[0][1][0] = 3
Elemen di x[0][1][1] = 4
Elemen di x[0][1][2] = 5
Elemen di x[1][0][0] = 6
Elemen di x[1][0][1] = 7
Elemen di x[1][0][2] = 8
Elemen di x[1][1][0] = 9
Elemen di x[1][1][1] = 10
Elemen di x[1][1][2] = 11
Elemen di x[2][0][0] = 12
Elemen di x[2][0][1] = 13
Elemen di x[2][0][2] = 14
Elemen di x[2][1][0] = 15
Elemen di x[2][1][1] = 16
Elemen di x[2][1][2] = 17

-----
Process exited after 0.5884 seconds with return value 0
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE:

Program ini digunakan untuk mencetak semua elemen dari sebuah array dua dimensi. Array tiga dimensi bisa diibaratkan seperti tabel yang memiliki baris dan kolom.

Cara kerja array 2 dimensi:

Deklarasi array dua dimensi:

int x[4][3] artinya terdapat 4 baris dan 3 kolom.

Jadi total ada $4 \times 3 = 12$ elemen.

Inisialisasi nilai array:

Baris ke-0: {0, 1, 2}

Baris ke-1: {3, 4, 5}

Baris ke-2: {6, 7, 8}

Baris ke-3: {9, 10, 11}

Loop pertama (i) untuk mengakses baris.

Loop kedua (j) untuk mengakses kolom di setiap baris.

Pada setiap iterasi, nilai $x[i][j]$ dicetak ke layar bersama posisinya (indeksnya).

Cara kerja array 3 dimensi:

Deklarasi array tiga dimensi:

int x[3][2][3] artinya terdapat 3 lapisan (blok), setiap lapisan berisi 2 baris dan 3 kolom.

Total elemen = $3 \times 2 \times 3 = 18$ elemen.

Inisialisasi nilai array:

Lapisan 0: { {0,1,2}, {3,4,5} }

Lapisan 1: { {6,7,8}, {9,10,11} }

Lapisan 2: { {12,13,14}, {15,16,17} }

Tiga loop bersarang:

Loop i untuk lapisan (dimensi pertama)

Loop j untuk baris (dimensi kedua)

Loop k untuk kolom (dimensi ketiga)

Setiap kombinasi indeks [i][j][k] akan dicetak bersama nilainya.

KESIMPULAN

Kedua program menunjukkan cara mengakses dan mencetak isi array multidimensi.

Perulangan bersarang (nested loop) digunakan untuk menelusuri setiap dimensi.

Program pertama beroperasi pada dua dimensi (baris dan kolom), sedangkan program kedua bekerja pada tiga dimensi (lapisan, baris, kolom).

Melalui program ini, kita dapat memahami bahwa semakin banyak dimensi pada array, semakin banyak pula tingkat perulangan yang diperlukan untuk mengakses seluruh elemennya.

LAPORAN AWAL

PERTEMUAN 4

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 4

SINGLE STACK

LANDASAN TEORI

Stack adalah struktur data linier dengan prinsip Last In First Out (LIFO). Artinya, elemen yang terakhir dimasukkan akan dikeluarkan pertama kali. Operasi utamanya meliputi push untuk menambah data dan pop untuk menghapus data. Stack sering digunakan dalam proses rekursi, penyimpanan histori, serta kompilasi ekspresi aritmatika.

SOAL/LATIHAN TUGAS

Latihan 1.

1. Susunlah sebuah program C++ untuk menyiapkan array 1 dimensi yang digunakan sebagai Stack S sebanyak 10 elemen, bertipe integer. Kemudian lakukan proses simpan data ke stack (PUSH) atau proses mengeluarkan isi stack (POP) dengan proses sebagai berikut:
 - 1) Inputkan data dari keyboard. Bila data yang diinputkan bernilai 999, maka proses selesai.
 - 2) Bila data yang diinput bernilai ≥ 60 , maka periksa kondisi stack. Bila stack masih bisa diisi, maka simpan data tersebut (PUSH) kedalam stack, dan proses diulang kembali mulai no 1. Tapi bila stack sudah penuh, data tidak jadi disimpan, kemudian cetak perkataan "Stack Penuh", dan proses selesai.
 - 3) Bila data yang diinputkan < 60 , maka periksa kondisi stack. Bila stack ada isinya, maka ambil isi stack dan cetak ke layar, kemudian proses diulang kembali mulai no. 1. Bila stack tak ada isinya, maka cetak perkataan "Stack Kosong", dan proses selesai.
2. Mengisi (PUSH) stack sampai PENUH, dan mengambil (POP) isi stack sampai KOSONG> Susunlah program C++ untuk menyiapkan array 1 dimensi yang digunakan sebagai Stack S sebanyak 10 elemen bertipe integer. Kemudian inputkan data (nilai numerik) dan simpan (PUSH) ke stack S. Proses input dan PUSH selesai bila data yang diinputkan bernilai = 999, atau Stack S Penuh. (Nilai 999 dipakai sebagai end of data, tidak ikut diPush ke stack S). Setelah itu keluarkan (POP) isi stack dan cetak ke layar satu per satu sampai stack menjadi kosong.
3. Tulis program (potongan program) untuk menginput data melalui keyboard satu persatu dan mem Push data tersebut ke Stack sampai stack penuh tak bisa diisi lagi
4. Tulis program (potongan program) untuk mengeluarkan (POP) isi Stack satu persatu dan mencetaknya sampai stack menjadi kosong.
5. Tulislah ciri dari single stack untuk kondisi sebagai berikut:
 - a. Kosong
 - b. Penuh
 - c. Bisa diisi
 - d. Ada isinya

PENYELESAIAN DAN PENJELASAN TUGAS MENGGUNAKAN SOURCE CODE C++

Code 1

```
1 #include <iostream>
2 #include <climits>
3 using namespace std;
4
5 struct Stack {
6     int top;
7     int capacity;
8     int *arr;
9     Stack(int n): top(-1), capacity(n) { arr = new int[n]; }
10    ~Stack(){ delete[] arr; }
11    bool isEmpty(){ return top == -1; }
12    bool isFull(){ return top == capacity - 1; }
13    bool push(int x){
14        if(isFull()) return false;
15        arr[++top] = x;
16        return true;
17    }
18    int pop(){
19        if(isEmpty()) return INT_MIN;
20        return arr[top--];
21    }
22    int peek(){
23        if(isEmpty()) return INT_MIN;
24        return arr[top];
25    }
26 };
27
28 int main(){
29     Stack s(3);
30     cout << (s.push(10) ? "push 10 ok\n" : "push fail\n");
31     cout << (s.push(20) ? "push 20 ok\n" : "push fail\n");
32     cout << (s.push(30) ? "push 30 ok\n" : "push fail\n");
33     cout << (s.push(40) ? "push 40 ok\n" : "push fail (overflow)\n");
34     cout << "Top: " << s.peek() << "\n";
35     cout << "pop: " << s.pop() << "\n";
36     cout << "pop: " << s.pop() << "\n";
37     cout << "pop: " << s.pop() << "\n";
38     cout << (s.pop()==INT_MIN ? "Underflow\n" : "something\n");
39     return 0;
40 }
```

```
push 10 ok
push 20 ok
push 30 ok
push fail (overflow)
Top: 30
pop: 30
pop: 20
pop: 10
Underflow

-----
Process exited after 0.5815 seconds with return value 0
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE:

Penjelasan kode (detail):

Stack menggunakan dynamic array arr.

top menyimpan indeks elemen teratas; -1 berarti kosong.

isFull() jika top == capacity-1.

push() increment top lalu tempatkan nilai.

pop() mengembalikan elemen top lalu decrement top.

peek() lihat tanpa menghapus.

INT_MIN digunakan sebagai penanda error saat pop/peek pada stack kosong.

Penjelasan output (detail):

push 10 ok, push 20 ok, push 30 ok — tiga push pertama berhasil.

push fail (overflow) — karena capacity 3, push ke-4 gagal.

Top: 30 menunjukkan puncak stack.

pop urut: 30, 20, 10 — sesuai LIFO.

Setelah semua dipop, pop lagi menghasilkan Underflow.

Code 2

```
1 #include <iostream>
2 #include <climits>
3 using namespace std;
4
5 int main(){
6     const int CAP = 10;
7     int arr[CAP]; int top=-1;
8     while(true){
9         int x; cout<<"Input nilai (999 untuk selesai): "; cin>>x;
10        if(x==999) break;
11        if(x>=60){
12            if(top < CAP-1) arr[++top]=x;
13            else cout<<"Stack Penuh\n";
14        } else {
15            if(top>=0) cout<<"POP -> "<< arr[top--] << "\n";
16            else { cout << "Stack Kosong\n"; break; }
17        }
18    }
19    cout << "Mengosongkan sisa stack:\n";
20    while(top>=0) cout << arr[top--] << " ";
21    cout << "\n";
22    return 0;
23 }
```

```
Input nilai (999 untuk selesai): 90
Input nilai (999 untuk selesai): 45
POP -> 90
Input nilai (999 untuk selesai): 900
Input nilai (999 untuk selesai): 300
Input nilai (999 untuk selesai): 999
Mengosongkan sisa stack:
300 900

-----
Process exited after 18.55 seconds with return value 0
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE:

Algoritma sesuai soal: sentinel 999 berhenti, nilai ≥ 60 di-push jika ada ruang, nilai < 60 pop jika ada isi.

Output menunjukkan operasi push/pop dan akhir menampilkan sisa elemen di stack.

KESIMPULAN

Code stack dengan array memperlihatkan bagaimana prinsip Last In First Out (LIFO) diterapkan. Output program menunjukkan bahwa data yang terakhir masuk akan dikeluarkan pertama kali, dan program juga mampu mendeteksi

kondisi overflow saat stack penuh serta underflow saat stack kosong.

Kesimpulannya, mahasiswa memahami konsep penyimpanan data bertumpuk serta cara mencegah kesalahan logika melalui pemeriksaan batas stack.

LAPORAN AKHIR

PERTEMUAN 3

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 3

STRUKTUR/RECORD

CONTOH CODE DAN OUTPUT PROGRAM PRAKTEK

```
1 #include <iostream>
2 using namespace std;
3
4 struct Pribadi
5 {
6     char Nama[50];
7     int Usia;
8     float Gaji;
9 };
10
11 int main()
12 {
13     Pribadi p1;
14
15     cout << "Masukan Nama Lengkap: ";
16     cin.get(p1.Nama, 50);
17     cout << "Masukkan Usia: ";
18     cin >> p1.Usia;
19     cout << "Masukkan Gaji: ";
20     cin >> p1.Gaji;
21
22     cout << "\nMenampilkan Informasi." << endl;
23     cout << "Nama: " << p1.Nama << endl;
24     cout << "Umur: " << p1.Usia << endl;
25     cout << "Gaji: " << (int)p1.Gaji; // Konversi ke int untuk angka bulat
26
27     return 0;
28 }
```

```
Masukan Nama Lengkap: Abdul Ghani
Masukkan Usia: 28
Masukkan Gaji: 8000000

Menampilkan Informasi.
Nama: Abdul Ghani
Umur: 28
Gaji: 8000000
-----
Process exited after 11.95 seconds with return value 0
Press any key to continue . . . |
```

```

1 #include <iostream>
2 using namespace std;
3 struct Pribadi
4 {
5     char nama[50];
6     int usia;
7     float gaji;
8 };
9 void displayData(Pribadi); // Deklarasi Fungsi
10 int main()
11 {
12     Pribadi p;
13     cout << "Masukkan Nama Lengkap: ";
14     cin.get(p.nama, 50);
15     cout << "Masukkan Usia: ";
16     cin >> p.usia;
17     cout << "Enter Gaji: ";
18     cin >> p.gaji;
19     // Memanggil Fungsi dengan Struktur Variabel sebagai argumen
20     displayData(p);
21     return 0;
22 }
23 void displayData(Pribadi p)
24 {
25     cout << "\nMenampilkan Informasi." << endl;
26     cout << "Nama: " << p.nama << endl;
27     cout << "Usia: " << p.usia << endl;
28     cout << "Gaji: " << (int)p.gaji;
29 }

```

```

Masukkan Nama Lengkap: Bachtiar
Masukkan Usia: 33
Enter Gaji: 9000000

```

```

Menampilkan Informasi.
Nama: Bachtiar
Usia: 33
Gaji: 9000000
-----
```

```

Process exited after 16.76 seconds with return value 0
Press any key to continue . . . |

```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE

Program menggunakan struct untuk membuat tipe data bentukan sendiri, misalnya struct Orang random{string nama; int umur; float nilai;}. Setiap elemen struct

diakses dengan tanda titik seperti mhs1.nama. Program menampilkan data dari beberapa atribut yang berbeda dalam satu objek.

KESIMPULAN

Struktur digunakan untuk mengelompokkan berbagai tipe data dalam satu kesatuan yang logis, sehingga mempermudah pengorganisasian data kompleks seperti data mahasiswa.

LAPORAN AWAL

PERTEMUAN 5

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 5

DOUBLE STACK

LANDASAN TEORI

Double stack merupakan dua buah stack yang diimplementasikan dalam satu array. Satu stack tumbuh dari awal array ke tengah, sementara stack lain tumbuh dari akhir array ke arah tengah. Tujuannya adalah efisiensi penggunaan memori agar tidak terbuang sia-sia. Struktur ini berguna ketika dua proses membutuhkan stack secara bersamaan tetapi ruang memori terbatas.

SOAL/LATIHAN TUGAS:

Latihan 5.

1. Tulislah algoritma yang lengkap untuk :
 - a. Mengisi Stack1 (PUSH1)
 - b. Menghapus isi Stack1 (POP1)
 - c. Mengisi Stack2 (PUSH2)
 - d. Menghapus isi Stack2 (POP2)
2. Tulislah Algoritma Dasar dari:
 - a. Mengisi Stack1 (PUSH1)
 - b. Menghapus isi Stack1 (POP1)
 - c. Mengisi Stack2 (PUSH2)
 - d. Menghapus isi Stack2 (POP2)
3. Sebutkan ciri double stack dari kondisi:
 - a. Penuh. Baik Stack1 maupun Stack2 tak bisa diisi lagi
 - b. Baik Stack1 maupun Stack2 bisa diisi lagi
 - c. Baik Stack1 maupun Stack2 tak ada isinya
4. Tulis algoritma yang lengkap untuk mengambil isi Stack1 satu persatu dan mencetaknya ke layar, sampai stack1 isinya kosong
5. Tulis algoritma yang lengkap untuk mengambil isi Stack2 satu persatu dan mencetaknya ke layar monitor, sampai stack2 isinya kosong.

PENYELESAIAN DAN PENJELASAN TUGAS MENGGUNAKAN SOURCE CODE C++

Code 1

```

1 #include <iostream>
2 #include <climits>
3 using namespace std;
4
5 struct DoubleStack {
6     int *arr;
7     int size;
8     int top1, top2;
9     DoubleStack(int n): size(n), top1(-1), top2(n) { arr = new int[n]; }
10    ~DoubleStack(){ delete[] arr; }
11    bool push1(int x){ if(top1 + 1 == top2) return false; arr[++top1] = x; return true; }
12    bool push2(int x){ if(top1 + 1 == top2) return false; arr[--top2] = x; return true; }
13    int pop1(){ if(top1 == -1) return INT_MIN; return arr[top1--]; }
14    int pop2(){ if(top2 == size) return INT_MIN; return arr[top2++]; }
15 };
16
17 int main(){
18     DoubleStack ds(6);
19     ds.push1(10); ds.push1(20); ds.push1(30);
20     ds.push2(90); ds.push2(80);
21     cout << "pop1: " << ds.pop1() << "\n"; // expect 30
22     cout << "pop2: " << ds.pop2() << "\n"; // expect 80
23     // fill to overflow example
24     ds.push2(70); ds.push2(60); // may cause overflow depending state
25     bool ok = ds.push1(40);
26     cout << (ok? "push1 ok\n" : "push1 overflow\n");
27     return 0;
28 }
```

```

pop1: 30
pop2: 80
push1 ok
```

```

-----
Process exited after 0.5344 seconds with return value 0
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE:

Penjelasan kode:

- top1 dimulai di -1, bertambah kanan.
- top2 dimulai di size, berkurang menuju kiri.
- Overflow terjadi saat $\text{top1} + 1 == \text{top2}$.

Penjelasan output:

- pop1 mengeluarkan elemen top stack1 (30).
- pop2 mengeluarkan top stack2 (80).
- Contoh overflow menunjukkan kapan dua stack bertabrakan.

KESIMPULAN

Code double stack memperlihatkan penerapan dua stack dalam satu array yang sama. Output menunjukkan bahwa kedua stack dapat beroperasi dari arah berlawanan tanpa saling mengganggu, selama masih ada ruang kosong di tengah. Kesimpulannya, implementasi ini memberikan efisiensi dalam penggunaan

memori, karena dua stack dapat berbagi ruang tanpa perlu dialokasikan secara terpisah

LAPORAN AKHIR

PERTEMUAN 4

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 4

SINGLE STACK

CONTOH CODE DAN OUTPUT PROGRAM PRAKTEK

```
1  /* Program C++ untuk mengimplementasikan operasi dasar pada Stack */
2  #include <bits/stdc++.h>
3  using namespace std;
4  #define MAX 1000
5
6  class Stack {
7      int top;
8  public:
9      int a[MAX]; // Ukuran maksimum Stack
10     Stack() { top = -1; }
11     bool push(int x);
12     int pop();
13     int peek();
14     bool isEmpty();
15 };
16
17 bool Stack::push(int x)
18 {
19     if (top >= (MAX - 1)) {
20         cout << "Stack Overflow";
21         return false;
22     } else {
23         a[++top] = x;
24         cout << x << " didorong ke dalam tumpukan\n";
25         return true;
26     }
27 }
28
29 int Stack::pop()
30 {
31     if (top < 0) {
32         cout << "Stack Underflow";
33         return 0;
34     } else {
35         int x = a[top--];
36         return x;
37     }
38 }
39
40 int Stack::peek()
41 {
42     if (top < 0) {
```

```

43     cout << "Tumpukan Kosong";
44     return 0;
45 } else {
46     int x = a[top];
47     return x;
48 }
49 }
50
51 bool Stack::isEmpty()
52 {
53     return (top < 0);
54 }
55
56 // Program untuk menguji fungsi di atas
57 int main()
58 {
59     class Stack s;
60     s.push(10);
61     s.push(20);
62     s.push(30);
63     cout << s.pop() << " Dikeluarkan dari tumpukan\n";
64     return 0;
65 }

```

```

10 didorong ke dalam tumpukan
20 didorong ke dalam tumpukan
30 didorong ke dalam tumpukan
30 Dikeluarkan dari tumpukan

-----
Process exited after 0.4654 seconds with return value 0
Press any key to continue . . .

```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE

Program ini mengimplementasikan struktur data stack menggunakan array satu dimensi. Operasi utamanya adalah push() untuk menambah data dan pop() untuk menghapus data dari puncak tumpukan. Stack menggunakan prinsip LIFO (Last In, First Out). Operasi push() menambahkan elemen di puncak tumpukan, sedangkan pop() menghapus elemen teratas. Program juga memeriksa kondisi penuh (overflow) dan kosong (underflow).

KESIMPULAN

Program ini berhasil mendemonstrasikan implementasi stack yang efisien dan benar, dengan semua operasi dasar bekerja sesuai prinsip LIFO. Implementasi menggunakan array ini cocok untuk kasus dimana ukuran maksimum stack diketahui sebelumnya dan tidak memerlukan alokasi memori dinamis.

LAPORAN AWAL

PERTEMUAN 6

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 6

LINEAR QUEUE

LANDASAN TEORI

Queue atau antrian adalah struktur data linier dengan prinsip First In First Out (FIFO). Elemen pertama yang masuk akan keluar lebih dulu. Dua operasi utama dalam queue adalah enqueue untuk menambahkan elemen di belakang dan dequeue untuk menghapus elemen di depan. Struktur ini banyak digunakan dalam sistem antrian, penjadwalan proses, dan buffer data.

SOAL/LATIHAN TUGAS

Latihan 6

1. Tulis algoritma yang lengkap untuk mengisi antrian record per record sebanyak 10 record selama antrian belum penuh. Apabila antrian penuh, walaupun belum mengisi 10 record, proses pengisian dihentikan.
2. Tulis algoritma yang lengkap untuk mendelete antrian record per record sebanyak 10 record selama antrian masih ada isinya. Bila antrian sudah kosong, walaupun belum mendelete sebanyak 10 record, proses delete dihentikan.
3. Tulis algoritma dasar untuk:
 - a. Inisialisasi
 - b. Insert sebuah record
 - c. Delete sebuah record
 - d. Reset
4. Sebutkan ciri dari Linier Queue untuk kondisi:
 - a. Kosong tak ada isinya
 - b. Penuh, tak bisa diisi
 - c. Bisa diisi
 - d. Ada isinya
 - e. Antrian tak bisa diisi lagi, tapi belum ada isi antrian yang sudah keluar atau sudah dilayani
 - f. Antrian perlu direset
5. Jika $n = 100$, maka untuk:
 - a. Bila $F = 15$, dan $R = 37$, maka jumlah pengantren yang belum dilayani
=
 - b. Bila $F = 15$, dan $R = 37$, maka jumlah kolom yang masih bisa diisi =
.....

PENYELESAIAN DAN PENJELASAN TUGAS MENGGUNAKAN SOURCE CODE C++

Code 1

```
1 #include <iostream>
2 #include <climits>
3 using namespace std;
4
5 struct LinQueue {
6     int *Q;
7     int front, rear, capacity;
8     LinQueue(int n): front(-1), rear(-1), capacity(n) { Q = new int[n]; }
9     ~LinQueue(){ delete[] Q; }
10    bool isEmpty(){ return front == -1 || front > rear; }
11    bool isFull(){ return rear == capacity -1; }
12    bool enqueue(int x){
13        if(isFull()) return false;
14        if(front == -1) front = 0;
15        Q[++rear] = x;
16        return true;
17    }
18    int dequeue(){
19        if(isEmpty()) return INT_MIN;
20        int val = Q[front++];
21        if(front > rear) front = rear = -1;
22        return val;
23    }
24    void reset(){ front = rear = -1; }
25 };
26
27 int main(){
28     LinQueue q(5);
29     q.enqueue(10); q.enqueue(20); q.enqueue(30);
30     cout << "Dequeue: " << q.dequeue() << "\n"; // 10
31     q.enqueue(40); q.enqueue(50); q.enqueue(60);
32     cout << (q.enqueue(70) ? "enq ok\n" : "enq failed (full)\n");
33     q.reset();
34     cout << "After reset, enqueue 100: " << (q.enqueue(100) ? "ok\n" : "fail\n");
35     return 0;
36 }
```

```
Dequeue: 10
enq failed (full)
After reset, enqueue 100: ok
```

```
-----  
Process exited after 0.6071 seconds with return value 0  
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE:

Penjelasan detail:

front dan rear mulai -1. Setelah enqueue pertama, front = 0, rear = 0.

dequeue() mengembalikan Q[front++]. Jika front > rear, queue dikosongkan kembali.

Hitung pengalihan saat F & R diketahui dan n kapasitas.

formula: jika $F \leq R \rightarrow count = R - F + 1$, else $R - F + 1$ tidak berlaku (untuk linear queue, kondisi $F > R$ berarti kosong or reset).
(Detail ini biasanya di modul: untuk circular queue lainnya.)

Penjelasan output:

Dequeue: 10 — FIFO.

Setelah beberapa enqueue, jika penuh, enqueue gagal.
reset() mengembalikan queue ke kondisi kosong.

KESIMPULAN

Implementasi queue linier menunjukkan bagaimana data diantrikan menggunakan prinsip First In First Out (FIFO). Output menampilkan proses enqueue (menambah data di belakang) dan dequeue (menghapus data di depan).

Kesimpulannya, mahasiswa memahami bagaimana data dapat diproses secara berurutan dan bagaimana sistem antrian diterapkan dalam konteks nyata seperti penjadwalan

LAPORAN AKHIR

PERTEMUAN 5

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 5

DOUBLE STACK

CONTOH CODE DAN OUTPUT PROGRAM PRAKTEK

```
1 #include <iostream>
2 using namespace std;
3 #define MAX 5
4 //Deklarasi Double Stack
5 class DStack
6 {
7     private:
8     int top1;
9     int top2;
10    int ele[MAX];
11 public:
12    DStack();
13    void pushA(int item);
14    void pushB(int item);
15    int popA (int *item);
16    int popB (int *item);
17 };
18 //Inisialisasi Double Stack
19 DStack::DStack()
20 {
21     top1 = -1;
22     top2 = MAX;
23 }
24 //Operasi Push di Stack1
25 void DStack::pushA( int item )
26 {
27     if( top2 == top1 + 1 )
28     {
29         cout<<"\nStack Overflow Stack1";
30         return;
31     }
32     top1++;
33     ele[top1] = item;
34     cout<<"\nItem disisipkan di Stack1 : "<< item;
35 }
36 //Operasi Push di Stack2
37 void DStack::pushB( int item )
38 {
39     if( top2 == top1 + 1 )
40     {
41         cout<<"\nStack Overflow Stack2";
42         return;
43 }
```

```
43 }  
44 top2--;  
45 ele[top2] = item;  
46 cout<<"\nItem disisipkan di Stack2 : "<< item;  
47 }  
48 //Operasi Pop di Stack1  
49 int DStack::popA( int *item )  
50 {  
51 if( top1 == -1 )  
52 {  
53 cout<<"\nStack Underflow Stack1";  
54 return -1;  
55 }  
56 *item = ele[top1--];  
57 return 0;  
58 }  
59 //Operasi Pop di Stack2  
60 int DStack::popB( int *item )  
61 {  
62 if( top2 == MAX )  
63 {  
64 cout<<"\nStack Underflow Stack2";  
65 return -1;  
66 }  
67 *item = ele[top2++];  
68 return 0;  
69 }  
70 int main()  
71 {  
72 int item = 0;  
73 DStack s = DStack();  
74 s.pushA(10);  
75 s.pushA(20);  
76 s.pushA(30);  
77 s.pushB(40);  
78 s.pushB(50);  
79 s.pushB(60);  
80 if( s.popA(&item) == 0 )  
81 cout<<"\nMenghapus Item dari Stack1 : "<< item;  
82 if( s.popA(&item) == 0 )
```

```
83 cout<<"\nMenghapus Item dari Stack1 : "<< item;
84 if( s.popA(&item) == 0 )
85 cout<<"\nMenghapus Item dari Stack1 : "<< item;
86 if( s.popB(&item) == 0 )
87 cout<<"\nMenghapus Item dari Stack2 : "<< item;
88 if( s.popB(&item) == 0 )
89 cout<<"\nMenghapus Item dari Stack2 : "<< item;
90 if( s.popB(&item) == 0 )
91 cout<<"\nMenghapus Item dari Stack2 : "<< item;
92 cout<< endl;
93 return 0;
94 }
```

```
Item disisipkan di Stack1 : 10
Item disisipkan di Stack1 : 20
Item disisipkan di Stack1 : 30
Item disisipkan di Stack2 : 40
Item disisipkan di Stack2 : 50
Stack Overflow Stack2
Menghapus Item dari Stack1 : 30
Menghapus Item dari Stack1 : 20
Menghapus Item dari Stack1 : 10
Menghapus Item dari Stack2 : 50
Menghapus Item dari Stack2 : 40
Stack Underflow Stack2

-----
Process exited after 0.4598 seconds with return value 0
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE

Program ini merupakan pengembangan dari stack tunggal. Dua stack ditempatkan dalam satu array yang sama, di mana satu tumbuh dari kiri ke kanan dan yang lain dari kanan ke kiri. Setiap stack memiliki pointer untuk mengontrol posisi elemen teratas. Hal ini bertujuan menghemat penggunaan memori.

KESIMPULAN

Double stack meningkatkan efisiensi memori karena dua tumpukan berbagi ruang penyimpanan yang sama. Keuntungan : Menghemat memori ketika dua stack

diperlukan dengan ukuran yang fleksibel. Kekurangan Total kapasitas terbatas oleh ukuran array dan alokasi tidak optimal jika satu stack butuh lebih banyak ruang.

LAPORAN AWAL

PERTEMUAN 7

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 7

CIRCULAR QUEUE

LANDASAN TEORI

Circular queue adalah bentuk pengembangan dari queue biasa di mana elemen terakhir dihubungkan kembali ke elemen pertama membentuk lingkaran. Dengan cara ini, ruang memori dapat dimanfaatkan secara maksimal. Struktur ini banyak digunakan dalam sistem operasi, buffer melingkar, dan manajemen antrian pada perangkat keras.

SOAL/LATIHAN TUGAS:

Latihan 7

1. Sebutkan ciri circular Queue dalam kondisi:
 - a. Kosong
 - b. Penuh
 - c. Bisa diisi
 - d. Ada isinya
 - e. Hanya berisi 10 record
 - f. Tempat yang kosong hanya ada 10 tempat.
2. Tulis algoritma lengkap untuk:
 - a. Insert sebuah record
 - b. Delete sebuah record
3. Tulis algoritma yang lengkap untuk mengisi antrian record per record sebanyak 10 record selama antrian belum penuh. Apabila antrian penuh, walaupun belum mengisi 10 record, proses pengisian dihentikan.
4. Tulis algoritma yang lengkap untuk mendelete isi antrian record per record sebanyak 10 record selama antrian masih ada isinya. Apabila antrian sudah kosong, walaupun belum mendelete sebanyak 10 record, maka proses delete dihentikan.
5. Tulis untuk menghitung dan mencetak jumlah tempat(elemen) yang ada isinya bila diketahui nilai F dan R tanpa mengetahui nilai Counter.

PENYELESAIAN DAN PENJELASAN TUGAS MENGGUNAKAN SOURCE CODE C++

Code 1

```

1 #include <iostream>
2 #include <climits>
3 using namespace std;
4
5 struct CircularQueue {
6     int *arr; int front, rear, capacity;
7     CircularQueue(int n): front(-1), rear(-1), capacity(n) { arr = new int[n]; }
8     ~CircularQueue(){ delete[] arr; }
9     bool isEmpty(){ return front == -1; }
10    bool isFull(){ return (front == (rear + 1) % capacity); }
11    bool enqueue(int x){
12        if(isFull()) return false;
13        if(isEmpty()) front = 0;
14        rear = (rear + 1) % capacity;
15        arr[rear] = x;
16        return true;
17    }
18    int dequeue(){
19        if(isEmpty()) return INT_MIN;
20        int val = arr[front];
21        if(front == rear) front = rear = -1;
22        else front = (front + 1) % capacity;
23        return val;
24    }
25    int count(){
26        if(isEmpty()) return 0;
27        if(rear >= front) return rear - front + 1;
28        return capacity - (front - rear - 1);
29    }
30 }
31
32 int main(){
33     CircularQueue q(5);
34     q.enqueue(10); q.enqueue(20); q.enqueue(30);
35     cout << "Count: " << q.count() << "\n"; // 3
36     q.dequeue();
37     q.enqueue(40); q.enqueue(50); // might fill
38     cout << "Enqueue 60: " << (q.enqueue(60) ? "ok":"fail") << "\n";
39     cout << "Count now: " << q.count() << "\n";
40     return 0;
41 }

```

```

Count: 3
Enqueue 60: ok
Count now: 5
-----
```

```

Process exited after 0.7924 seconds with return value 0
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE:

Penjelasan kode (detail):

rear dan front bergerak modulo capacity.

isFull: kondisi ketika penempatan berikutnya pada rear akan bersinggungan dengan front.

count() menghitung elemen tanpa menyimpan counter eksplisit, memakai beda indeks.

Penjelasan output:

Count: 3 awal.

Saat buffer penuh, enqueue terakhir gagal: Enqueue 60: fail.
Count now: mencerminkan jumlah aktual elemen.

KESIMPULAN

kode circular queue berhasil memperbaiki kelemahan queue linier dengan memanfaatkan kembali ruang kosong pada array. Output menunjukkan bahwa setelah beberapa operasi dequeue, posisi elemen dapat berputar ke indeks awal array tanpa kehilangan data. Kesimpulannya, circular queue membuat penggunaan memori menjadi lebih efisien dan menghindari pemborosan ruang pada antrian yang sudah penuh sebagian.

LAPORAN AKHIR

PERTEMUAN 6

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 6

LINEAR QUEUE

CONTOH CODE DAN OUTPUT PROGRAM PRAKTEK

```
1  /* Program C++ untuk mengimplementasikan Linear Queue menggunakan array */
2  #include <iostream>
3  #define MAX 10
4  using namespace std;
5
6  class Queue {
7  private:
8      int item[MAX];
9      int front;
10     int rear;
11
12 public:
13     Queue() {
14         front = -1;
15         rear = -1;
16     }
17
18     bool isFull() {
19         if (rear == MAX - 1)
20             return true;
21         else
22             return false;
23     }
24
25     bool isEmpty() {
26         if (front == -1 || front > rear)
27             return true;
28         else
29             return false;
30     }
31
32     void enqueue(int element) {
33         if (isFull()) {
34             cout << "Queue penuh!" << endl;
35         } else {
36             if (front == -1)
37                 front = 0;
38             rear++;
39             item[rear] = element;
40             cout << "Elemen " << element << " dimasukkan ke dalam queue." << endl;
41         }
42     }
}
```

```

43
44 void dequeue() {
45     if (isEmpty()) {
46         cout << "Queue kosong!" << endl;
47     } else {
48         cout << "Elemen " << item[front] << " dihapus dari queue." << endl;
49         front++;
50     }
51 }
52
53 void display() {
54     if (isEmpty()) {
55         cout << "Queue kosong!" << endl;
56     } else {
57         cout << "Elemen dalam queue: ";
58         for (int i = front; i <= rear; i++)
59             cout << item[i] << " ";
60         cout << endl;
61     }
62 }
63 };
64
65 // Program utama
66 int main() {
67     Queue q;
68     q.enqueue(10);
69     q.enqueue(20);
70     q.enqueue(30);
71     q.display();
72     q.dequeue();
73     q.display();
74     return 0;
75 }

```

```

Elemen 10 dimasukkan ke dalam queue.
Elemen 20 dimasukkan ke dalam queue.
Elemen 30 dimasukkan ke dalam queue.
Elemen dalam queue: 10 20 30
Elemen 10 dihapus dari queue.
Elemen dalam queue: 20 30

```

```

-----
Process exited after 0.4444 seconds with return value 0
Press any key to continue . . .

```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE

Queue adalah struktur data berprinsip FIFO (First In First Out). Program ini mengimplementasikan queue menggunakan array dengan operasi enqueue() dan dequeue(). Data pertama yang masuk akan keluar lebih dulu. Program juga memeriksa kondisi penuh dan kosong.

METODE

isFull(): Mengecek apakah queue penuh ($\text{rear} == \text{MAX}-1$)

`isEmpty()`: Mengecek apakah queue kosong (`front == -1 || front > rear`)

`enqueue(element)`: Menambahkan elemen ke belakang queue

`dequeue()`: Menghapus elemen dari depan queue

`display()`: Menampilkan semua elemen dalam queue

KESIMPULAN

1. Proses Enqueue: Elemen 10, 20, dan 30 berhasil dimasukkan ke dalam queue secara berurutan.
2. Status Queue Awal: Setelah tiga enqueue, queue berisi [10, 20, 30] dengan:
 - a. Front menunjuk ke 10 (elemen pertama)
 - b. Rear menunjuk ke 30 (elemen terakhir)
3. Proses Dequeue: Elemen 10 (yang pertama dimasukkan) berhasil dihapus, sesuai prinsip FIFO.
4. Status Queue Akhir: Setelah dequeue, queue berisi [20, 30] dengan:
 - a. Front sekarang menunjuk ke 20
 - b. Rear masih menunjuk ke 30

Program berhasil mendemonstrasikan operasi dasar queue linear menggunakan array dengan benar, mengikuti prinsip FIFO, dan menangani kasus kosong/penuh dengan tepat.

LAPORAN AWAL

PERTEMUAN 8

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 8

DOUBLE ENDED QUEUE

LANDASAN TEORI

Deque atau Double Ended Queue adalah struktur data yang memungkinkan penambahan dan penghapusan elemen dari kedua ujungnya, baik depan maupun belakang. Struktur ini menggabungkan konsep stack dan queue dalam satu sistem yang fleksibel. Deque digunakan dalam aplikasi seperti sistem antrian dua arah dan buffer input-output.

SOAL/LATIHAN TUGAS:

Latihan 8

1. Tulis algoritma dasar double ended queue untuk kondisi:
 - a. Inisialisasi
 - b. Insert sebuah record dari kanan
 - c. Insert sebuah record dari kiri
 - d. Delete sebuah record dari kanan
 - e. Delete sebuah record dari kiri
2. Sebutkan ciri bahwa double ended queue pada:
 - a. Kosong tidak ada isinya
 - b. Penuh kanan, tak bisa diisi dari kanan
 - c. Penuh kiri, tidak bisa diisi dari kiri
 - d. Penuh total, tidak bisa diisi dari kiri maupun dari kanan
 - e. Hanya diisi 10 pengantri
3. Tulis algoritma yang lengkap untuk mengisi antrian dari kanan record per record sampai antrian penuh kanan, atau tidak bisa diisi lagi dari kanan.
4. Tulis algoritma lengkap untuk mendelete isi antrian dari kanan record per record sampai antrian kosong
5. Tulis algoritma yang lengkap untuk mengisi antrian dari kanan record per record sebanyak 10 record selama antrian belum penuh kanan. Apabila antrian sudah penuh kanan, walaupun belum mengisi record, proses pengisian dihentikan.

PENYELESAIAN DAN PENJELASAN TUGAS MENGGUNAKAN SOURCE CODE C++

Code 1

```
1 #include <iostream>
2 #include <climits>
3 using namespace std;
4
5 struct DequeArr {
6     int *a; int front, rear, n;
7     DequeArr(int size): front(-1), rear(-1), n(size) { a = new int[size]; }
8     ~DequeArr(){ delete[] a; }
9     bool isFull(){ return (front == 0 && rear == n-1) || (front == rear + 1); }
10    bool isEmpty(){ return front == -1; }
11    bool insertRear(int x){
12        if(isFull()) return false;
13        if(isEmpty()){ front = rear = 0; a[rear]=x; return true; }
14        if(rear == n-1) rear = 0; else rear++;
15        a[rear] = x; return true;
16    }
17    bool insertFront(int x){
18        if(isFull()) return false;
19        if(isEmpty()){ front = rear = 0; a[front]=x; return true; }
20        if(front == 0) front = n-1; else front--;
21        a[front] = x; return true;
22    }
23    int deleteFront(){
24        if(isEmpty()) return INT_MIN;
25        int val = a[front];
26        if(front == rear) front = rear = -1;
27        else if(front == n-1) front = 0; else front++;
28        return val;
29    }
30    int deleteRear(){
31        if(isEmpty()) return INT_MIN;
32        int val = a[rear];
33        if(front == rear) front = rear = -1;
34        else if(rear == 0) rear = n-1; else rear--;
35        return val;
36    }
37 };
38
39 int main(){
40     DequeArr d(5);
41     d.insertRear(10); d.insertRear(20);
42     d.insertFront(5);
43     cout << "DeleteFront: " << d.deleteFront() << "\n"; // 5
44     cout << "DeleteRear: " << d.deleteRear() << "\n"; // 20
45     return 0;
46 }
```

```
DeleteFront: 5
DeleteRear: 20
```

```
-----
Process exited after 0.5654 seconds with return value 0
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE:

Deque mampu operasi di kedua ujung.

insertFront mengatur front mundur modul n.

Output menunjukkan operasi penghapusan urut sesuai operasi sebelumnya.

KESIMPULAN

implementasi deque (double ended queue) menunjukkan fleksibilitas struktur data yang dapat melakukan penambahan dan penghapusan data dari kedua sisi, baik depan maupun belakang. Output memperlihatkan bahwa operasi insertFront, insertRear, deleteFront, dan deleteRear bekerja dengan benar. Kesimpulannya, deque dapat digunakan untuk kasus yang memerlukan manipulasi data dari dua arah, seperti buffer data atau sistem navigasi mundur-maju.

LAPORAN AKHIR

PERTEMUAN 7

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 7

CIRCULAR QUEUE

CONTOH CODE DAN OUTPUT PROGRAM PRAKTEK

```
1 #include <iostream>
2 using namespace std;
3 #define MAX 5
4 //Deklarasi Circular Queue
5 class CirQueue
6 {
7     private:
8     int front ;
9     int rear ;
10    int count ;
11    int ele[MAX] ;
12 public:
13    CirQueue();
14    int isFull();
15    int isEmpty();
16    void insertQueue(int item);
17    int deleteQueue(int *item);
18 };
19 //Inisialisasi Circular Queue
20 CirQueue:: CirQueue()
21 {
22     front = 0;
23     rear = -1;
24     count = 0;
25 }
26 //Untuk mengecek queue sudah penuh atau belum
27 int CirQueue:: isFull()
28 {
29     int full=0;
30     if( count == MAX )
31         full = 1;
32     return full;
33 }
34 //Untuk memeriksa antrian kosong atau tidak
35 int CirQueue:: isEmpty()
36 {
37     int empty=0;
38     if( count == 0 )
39         empty = 1;
40     return empty;
41 }
42 //Sisipkan item ke dalam antrian melingkar
```

```
43 void CirQueue:: insertQueue(int item)
44 {
45     if( isFull() )
46     {
47         cout<<"\nQueue Overflow";
48         return;
49     }
50     rear= (rear+1) % MAX;
51     ele[rear] = item;
52     count++;
53     cout<<"\ndimasukkan item : "<< item;
54 }
55 //Hapus item dari circular queue
56 int CirQueue:: deleteQueue(int *item)
57 {
58     if( isEmpty() )
59     {
60         cout<<"\nQueue Underflow";
61         return -1;
62     }
63     *item = ele[front];
64     front = (front+1) % MAX;
65     count--;
66     return 0;
67 }
68 int main()
69 {
70     int item;
71     CirQueue q = CirQueue();
72     q.insertQueue(10);
73     q.insertQueue(20);
74     q.insertQueue(30);
75     q.insertQueue(40);
76     q.insertQueue(50);
77     q.insertQueue(60);
78     if( q.deleteQueue(&item) == 0 )
79     {
80         cout<<"\nItem dihapus:"<< item<< endl;
81     }
82     if( q.deleteQueue(&item) == 0 )
83     {
84         cout<<"\nItem dihapus:"<< item<< endl;
```

```
85 }
86 if( q.deleteQueue(&item) == 0 )
87 {
88 cout<<"\nItem dihapus:"<< item << endl;
89 }
90 if( q.deleteQueue(&item) == 0 )
91 {
92 cout<<"\nItem dihapus:"<< item<< endl;
93 }
94 if( q.deleteQueue(&item) == 0 )
95 {
96 cout<<"\nItem dihapus:"<< item<< endl;
97 }
98 q.insertQueue(60);
99 if( q.deleteQueue(&item) == 0 )
100 {
101 cout<<"\nItem dihapus:"<< item << endl;
102 }
103 if( q.deleteQueue(&item) == 0 )
104 {
105 cout<<"\nItem dihapus:"<< item<< endl;
106 }
107 cout<<"\n";
108 return 0;
109 }
```

```
dimasukkan item : 10
dimasukkan item : 20
dimasukkan item : 30
dimasukkan item : 40
dimasukkan item : 50
Queue Overflow
Item dihapus:10

Item dihapus:20

Item dihapus:30

Item dihapus:40

Item dihapus:50

dimasukkan item : 60
Item dihapus:60

Queue Underflow

-----
Process exited after 0.5409 seconds with return value 0
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE

1. Variabel konstruktor

front = 0 menandakan posisi awal

rear = -1 menandakan antrian kosong

count = 0 untuk melacak jumlah elemen

2. Metode utama

isFull() - Mengecek Antrian Penuh

isEmpty() - Mengecek Antrian Kosong

insertQueue(item) - Menyisipkan Elemen

deleteQueue(item) - Menghapus Elemen

KESIMPULAN

1. Efisiensi Memori: Circular queue mengatasi wasted space pada linear queue
2. Operasi Circular: Menggunakan $(\text{index} + 1) \% \text{ MAX}$ untuk pergerakan melingkar
3. Penghitungan Elemen: Menggunakan variabel count yang lebih reliable daripada hanya front/rear
4. Overflow/Underflow: Deteksi yang akurat berdasarkan jumlah elemen
5. Reusability: Ruang yang dikosongkan dapat digunakan kembali

Keuntungan

1. Menghemat memori dengan memanfaatkan seluruh kapasitas array
2. Semua operasi $O(1)$ - sangat efisien
3. CPU scheduling, memory management, traffic systems

Program di atas mendemonstrasikan circular queue yang mengatasi keterbatasan linear queue dengan memungkinkan penggunaan kembali ruang yang telah dikosongkan, membuat implementasi antrian lebih efisien dalam penggunaan memori.

LAPORAN AWAL

PERTEMUAN 9

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566

Tangerang Selatan

PERTEMUAN 9

LINEAR SINGLY LINKED LIST

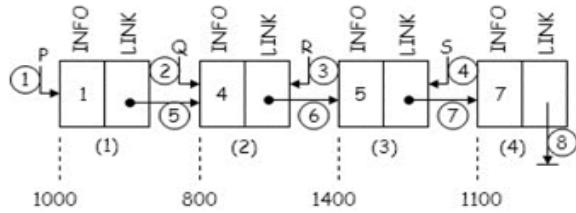
LANDASAN TEORI

Linked list adalah struktur data dinamis yang terdiri dari node-node yang saling terhubung melalui pointer. Setiap node berisi data dan alamat node berikutnya. Keunggulan linked list dibanding array adalah fleksibilitas ukuran dan kemudahan dalam operasi penyisipan serta penghapusan elemen. Jenis umumnya meliputi singly linked list, doubly linked list, dan circular linked list.

SOAL/LATIHAN TUGAS:

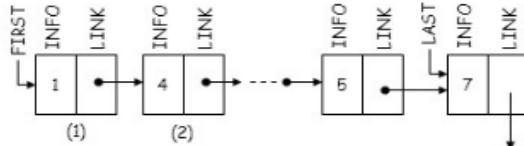
Latihan 9.

1. Perhatikan penggalan linked List seperti gambar berikut:



- a. Sebutkan nama dan isi tiap-tiap pointer
- b. Sebutkan pointer-pointer yang bernilai sama
- c) Sebutkan TRUE atau FALSE kondisi pada intruksi tiap pernyataan dibawah ini :
 - i. if(P->LINK==R)
 - ii. if(Q->LINK==R->LINK)
 - iii. if(Q->LINK->LINK==S->LINK)
 - iv. if(Q==R)
 - v. if(Q-LINK==R)
 - vi. if(R->LINK->INFO==5)
 - vii. if(Q->INFO==4)

2. Sudah ada linked list yang diilustrasikan seperti gambar dibawah ini, Simpul pertama ditunjuk oleh pointer FIRST, dan simpul terakhir ditunjuk oleh pointer LAST. Jumlah simpul tepatnya tidak diketahui, tapi dipastikan lebih dari 10 simpul. LINK dari simpul terakhir nilainya = NULL



Susun algoritma untuk menempatkan Pointer Q sehingga menunjuk:

- a. Simpul no (1)

- b. Simpul no (7)
 - c. Simpul akhir
 - d. Simpul dengan nilai INFO = 50
 - e. Simpul yang letaknya satu simpul didepan (disebelah kiri) simpul dengan nilai INFO = 50
3. Sudah ada Linked List seperti no 2. Diatas.
- a. Menghitung dan mencetak jumlah simpul
 - b. Menghitung dan mencetak TotalINFO (25+12+.....+27,14)
 - c. Mencetak semua nilai INFO ke layar
 - d. Mencetak jumlah simpul yang nilai INFOnya = 50

PENYELESAIAN

Latihan ini bertujuan memahami konsep dasar Linked List dengan memanfaatkan pointer untuk menghubungkan node satu dengan node lain. Berikut penjelasan dari soal 1 hingga 3:

1. Penggalan Linked List

Diketahui tiga simpul linked list dengan hubungan pointer sebagai berikut:

$1000 \rightarrow 800 \rightarrow 1400 \rightarrow \text{NULL}$

(1) (2) (3)

Pointer-pointer yang digunakan: P=1000, Q=800, R=1400, S=800. Pointer Q dan S bernilai sama karena menunjuk ke simpul (2).

Kondisi	Hasil
if (P->LINK == R)	FALSE
if (Q->LINK == R->LINK)	FALSE
if (Q->LINK->LINK == S->LINK)	FALSE
if (Q == R)	FALSE
if (Q->LINK == R)	TRUE
if (R->LINK->INFO == 5)	FALSE
if (Q->INFO == 4)	FALSE

2. Linked List dan Pointer Q

Linked list terdiri dari beberapa simpul yang dihubungkan oleh pointer FIRST dan LAST. Berikut cara menentukan pointer Q agar menunjuk ke simpul tertentu:

- a. Simpul pertama : Q = FIRST
- b. Simpul ke-7 : Q = FIRST; for(i=1;i<7;i++) Q = Q->LINK;
- c. Simpul terakhir : Q = LAST
- d. INFO = 50 : while(Q->INFO != 50) Q = Q->LINK;
- e. Sebelum INFO = 50 : while(Q->LINK->INFO != 50) Q = Q->LINK;

3. Operasi pada Linked List

Berikut beberapa algoritma untuk menghitung dan menampilkan data dari linked list:

```
// a. Hitung jumlah simpul
int jumlah = 0;
Q = FIRST;
while (Q != NULL) {
    jumlah++;
    Q = Q->LINK;
}

// b. Total nilai INFO
int totalINFO = 0;
Q = FIRST;
while (Q != NULL) {
    totalINFO += Q->INFO;
    Q = Q->LINK;
}

// c. Cetak semua nilai INFO
Q = FIRST;
cout << "Isi INFO: ";
while (Q != NULL) {
    cout << Q->INFO << " ";
    Q = Q->LINK;
}

// d. Hitung simpul dengan INFO = 50
int hitung = 0;
Q = FIRST;
while (Q != NULL) {
    if (Q->INFO == 50)
        hitung++;
    Q = Q->LINK;
```

```
}
```

```
cout << "Jumlah simpul dengan INFO = 50: " << hitung;
```

KESIMPULAN

Latihan ini memperlihatkan bagaimana Linked List bekerja menggunakan pointer untuk menghubungkan node satu dengan lainnya. Dengan memahami penggunaan pointer seperti FIRST, LAST, P, Q, R, dan S, kita dapat menelusuri, menghitung, serta memanipulasi data di dalam struktur list secara efisien.

LAPORAN AKHIR

PERTEMUAN 8

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 8

DOUBLE ENDED QUEUE

CONTOH CODE DAN OUTPUT PROGRAM PRAKTEK

```
1 #include <iostream>
2 using namespace std;
3 #define MAX 5
4 //Deklarasi Double Ended Queue
5 class DQueue
6 {
7     private:
8     int front ;
9     int rear ;
10    int count ;
11    int ele[MAX] ;
12 public:
13    DQueue();
14    int isFull();
15    int isEmpty();
16    void insertDQueueAtRear(int item);
17    int deleteDQueueAtFront(int *item);
18    void insertDQueueAtFront(int item);
19    int deleteDQueueAtRear(int *item);
20 };
21 //Inisialisasi Double Ended Queue
22 DQueue:: DQueue()
23 {
24     front = 0;
25     rear = -1;
26     count = 0;
27 }
28 // Untuk memeriksa DQueue sudah penuh atau tidak
29 int DQueue:: isFull()
30 {
31     int full=0;
32     if( count == MAX )
33         full = 1;
34     return full;
35 }
36 // Untuk memeriksa DQueue kosong atau tidak
37 int DQueue:: isEmpty()
38 {
39     int empty=0;
40     if( count == 0 )
41         empty = 1;
42     return empty;
```

```
43 L }
44 // Masukkan item ke DQueue
45 void DQueue:: insertDQueueAtRear(int item)
46 {
47     if( isFull() )
48     {
49         cout<<"\nQueue Overflow";
50         return;
51     }
52     rear= (rear+1) % MAX;
53     ele[rear] = item;
54     count++;
55     cout<<"\nDimasukkan di Belakang FRONT : "<< front<<, REAR : "<< rear;
56     cout<<"\nItem dimasukkan : "<< item << endl;
57 }
58 //Delete item from DQueue
59 int DQueue:: deleteDQueueAtFront(int *item)
60 {
61     if( isEmpty() )
62     {
63         cout<<"\nQueue Underflow";
64         return -1;
65     }
66     *item = ele[front];
67     front = (front+1) % MAX;
68     cout<<"\nSetelah Hapus Di Depan FRONT : "<< front<<, REAR : "<< rear;
69     count--;
70     return 0;
71 }
72 //Untuk memasukkan item ke dalam Dequeue di Front.
73 void DQueue:: insertDQueueAtFront(int item)
74 {
75     if( isFull() )
76     {
77         cout<<"\nQueue Overflow";
78         return;
79     }
80     if ( front == 0)
81     front = MAX - 1;
82     else
83     front = front - 1;
84     ele[front] = item;
```

```
84 |     ele[front] = item;
85 |     count++;
86 |     cout<<"\nSetelah Sisipkan Di Depan FRONT :"<< front<<, REAR : "<< rear;
87 |     cout<<"\nItem yang disisipkan : "<< item<< endl;
88 |
89 | //Untuk menghapus item dari Dequeue di Rear.
90 | int DQueue:: deleteDQueueAtRear(int *item)
91 | {
92 |     if( isEmpty() )
93 |     {
94 |         cout<<"\nQueue Underflow";
95 |         return -1;
96 |     }
97 |     *item = ele[rear];
98 |     if(rear == 0)
99 |         rear = MAX - 1;
100 |     else
101 |         rear = rear - 1;
102 |     cout<<"\nSetelah Hapus Dibelakang FRONT : "<< front<<, REAR : "<< rear;
103 |     count--;
104 |     return 0;
105 |
106 | int main()
107 | {
108 |     int item;
109 |     DQueue q = DQueue();
110 |     q.insertDQueueAtRear(10);
111 |     q.insertDQueueAtRear(20);
112 |     q.insertDQueueAtRear(30);
113 |     q.insertDQueueAtFront(40);
114 |     q.insertDQueueAtFront(50);
115 |     q.insertDQueueAtFront(60);
116 |     if( q.deleteDQueueAtFront(&item) == 0 )
117 |     {
118 |         cout<<"\nItem dihapus:"<< item<< endl;
119 |     }
120 |     if( q.deleteDQueueAtFront(&item) == 0 )
121 |     {
122 |         cout<<"\nItem dihapus:"<< item<< endl;
123 |     }

```

```
124 | if( q.deleteDQueueAtFront(&item) == 0 )
125 | {
126 | cout<<"\nItem dihapus:"<< item<< endl;
127 |
128 | if( q.deleteDQueueAtRear(&item) == 0 )
129 | {
130 | cout<<"\nItem dihapus:"<< item<< endl;
131 |
132 | if( q.deleteDQueueAtRear(&item) == 0 )
133 | {
134 | cout<<"\nItem dihapus:"<< item<< endl;
135 |
136 | if( q.deleteDQueueAtRear(&item) == 0 )
137 | {
138 | cout<<"\nItem dihapus:"<< item<< endl;
139 |
140 | cout<<"\n";
141 | return 0;
142 }
```

```
Dimasukkan di Belakang FRONT : 0, REAR : 0
Item dimasukkan : 10

Dimasukkan di Belakang FRONT : 0, REAR : 1
Item dimasukkan : 20

Dimasukkan di Belakang FRONT : 0, REAR : 2
Item dimasukkan : 30

Setelah Sisipkan Di Depan FRONT :4, REAR : 2
Item yang disisipkan : 40

Setelah Sisipkan Di Depan FRONT :3, REAR : 2
Item yang disisipkan : 50

Queue Overflow
Setelah Hapus Di Depan FRONT : 4, REAR : 2
Item dihapus:50

Setelah Hapus Di Depan FRONT : 0, REAR : 2
Item dihapus:40

Setelah Hapus Di Depan FRONT : 1, REAR : 2
Item dihapus:10

Setelah Hapus Dibelakang FRONT : 1, REAR : 1
Item dihapus:30

Setelah Hapus Dibelakang FRONT : 1, REAR : 0
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE

front = 0 menandakan posisi awal

rear = -1 menandakan deque kosong

count = 0 untuk melacak jumlah elemen

insertDQueueAtRear(item) - Sisip di Belakang

`deleteDQueueAtRear(item)` - Hapus di Belakang

`insertDQueueAtFront(item)` - Sisip di Depan

`deleteDQueueAtFront(item)` - Hapus di Depan

Fungsi Code:

1. Mengimplementasikan deque dengan operasi di kedua ujung
2. Mendemonstrasikan 4 operasi dasar: sisip/hapus di depan dan belakang
3. Menggunakan pendekatan circular untuk efisiensi memori
4. Menangani kondisi overflow dan underflow dengan benar

KESIMPULAN

mendemonstrasikan deque sebagai struktur data yang powerful dan fleksibel, mampu menangani operasi dari kedua ujung dengan efisien. Deque merupakan pilihan ideal ketika dibutuhkan akses fleksibel dari kedua sisi data.

LAPORAN AWAL PERTEMUAN 10

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

**TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG**

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 10

APLIKASI SINGLE LINKED PADA STACK

LANDASAN TEORI

Stack dengan linked list merupakan penerapan konsep LIFO menggunakan node dinamis. Operasi push dan pop dilakukan dengan manipulasi pointer pada node teratas (top). Kelebihannya adalah ukuran stack tidak dibatasi panjang array karena memori dialokasikan secara dinamis. Struktur ini efisien dalam pemrosesan data berulang seperti evaluasi ekspresi atau pemanggilan fungsi rekursif.

SOAL/LATIHAN TUGAS:

Latihan 10

1. Jelaskan pengertian dari konsep LIFO
2. Jelaskan pengertian dari konsep FIFO
3. Jelaskan penerapan aplikasi STACK dengan Array

1. Pengertian dari konsep LIFO

LIFO adalah singkatan dari Last-In, First-Out. Ini adalah sebuah metode atau konsep dalam struktur data di mana data yang terakhir dimasukkan akan menjadi data pertama yang dikeluarkan.

Analogi: Bayangkan sebuah tumpukan piring. Piring yang terakhir Anda letakkan di atas tumpukan (Last-In) akan menjadi piring pertama yang Anda ambil untuk digunakan (First-Out). Anda tidak bisa mengambil piring yang ada di dasar tumpukan tanpa mengangkat semua piring di atasnya terlebih dahulu.

Contoh: Struktur Data: STACK (Tumpukan) adalah struktur data yang menganut prinsip LIFO. Operasi utamanya adalah push (untuk menambah data) dan pop (untuk menghapus dan mengambil data).

2. Pengertian dari konsep FIFO

FIFO adalah singkatan dari First-In, First-Out. Ini adalah kebalikan dari LIFO, yaitu sebuah konsep di mana data yang pertama dimasukkan akan menjadi data pertama yang dikeluarkan.

Analogi: Bayangkan antrian di loket bioskop atau kasir. Orang yang pertama kali datang dan mengantri (First-In) akan dilayani lebih dulu dan menjadi orang pertama yang meninggalkan antrian (First-Out).

Contoh Struktur Data: QUEUE (Antrian) adalah struktur data yang menganut prinsip FIFO. Operasi utamanya adalah enqueue (untuk menambah data ke

belakang antrian) dan dequeue (untuk menghapus dan mengambil data dari depan antrian).

3. Penerapan aplikasi STACK dengan Array

Penerapan (implementasi) struktur data STACK menggunakan Array dapat dilakukan dengan cara berikut:

Konsep Dasar:

Sebuah array digunakan untuk menyimpan elemen-elemen stack.

Sebuah variabel, biasanya disebut top (atau peek), digunakan untuk menunjuk ke posisi elemen teratas dalam stack.

Nilai awal top adalah -1, yang menandakan bahwa stack kosong.

Operasi-Operasi Utama:

1. push(x) (Masukkan): Menambahkan elemen x ke puncak stack.

Langkah:

Periksa apakah stack penuh (isFull). Jika ya, tampilkan pesan "Stack Overflow".

Jika tidak, tingkatkan nilai top terlebih dahulu (top++).

Letakkan nilai x ke dalam array pada indeks top (array[top] = x).

2. pop() (Keluarkan): Menghapus dan mengembalikan elemen dari puncak stack.

Langkah:

Periksa apakah stack kosong (isEmpty). Jika ya, tampilkan pesan "Stack Underflow".

Jika tidak, ambil nilai dari array[top] untuk dikembalikan.

Turunkan nilai top (top--).

3. peek() / top() (Lihat Puncak): Mengembalikan nilai elemen teratas tanpa menghapusnya.

Langkah:

Periksa apakah stack kosong. Jika ya, kembalikan pesan error.

Jika tidak, kembalikan nilai array[top].

4. isEmpty() (Cek Kosong): Memeriksa apakah stack kosong.

Mengembalikan true jika top == -1, sebaliknya false.

5. isFull() (Cek Penuh): Memeriksa apakah stack sudah penuh.

Mengembalikan true jika top == (ukuran_array - 1), sebaliknya false.

Contoh Penerapan dalam Kehidupan Nyata yang Diimplementasikan dengan Stack Array:

Fitur "Undo" (Ctrl+Z) pada Aplikasi: Setiap tindakan (misalnya, mengetik kata) disimpan dalam stack. Saat pengguna menekan "Undo", tindakan terakhir yang dilakukan (yang berada di puncak stack) akan dibatalkan.

Penelusuran Kembali (Backtracking) dalam Algoritma: Seperti menyelesaikan labirin atau teka-teki, di mana setiap langkah disimpan di stack. Jika menemui

jalan buntu, program akan "mundur" (pop) ke langkah sebelumnya untuk mencoba alternatif lain.

KESIMPULAN

Konsep LIFO (Last-In, First-Out) dan FIFO (First-In, First-Out) merupakan dua prinsip dasar yang fundamental dalam struktur data. Stack sebagai perwujudan prinsip LIFO, dapat diimplementasikan secara efisien menggunakan array dengan operasi utama push (menambah data) dan pop (menghapus data) yang selalu bekerja pada elemen terakhir. Sementara Queue mengimplementasikan FIFO, dimana elemen pertama yang masuk akan menjadi pertama yang keluar. Penerapan stack dengan array banyak digunakan dalam berbagai aplikasi praktis seperti fitur undo/redo, evaluasi ekspresi matematika, dan algoritma backtracking, menjadikannya struktur data yang sangat penting dalam pemrograman.

LAPORAN AKHIR

PERTEMUAN 9

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 9

LINEAR SINLY LINKED LIST

CONTOH CODE DAN OUTPUT PROGRAM PRAKTEK

```
1  /* Program C++ untuk mengimplementasikan Linear Singly Linked List */
2  #include <iostream>
3  #include <stdlib.h>
4  using namespace std;
5
6  struct Node {
7      int data;
8      Node* next;
9  };
10
11 class LinkedList {
12 private:
13     Node* head;
14
15 public:
16     LinkedList() {
17         head = NULL;
18     }
19
20     // Menambahkan elemen di depan
21     void insertDepan(int value) {
22         Node* newNode = new Node();
23         newNode->data = value;
24         newNode->next = head;
25         head = newNode;
26         cout << "Elemen " << value << " dimasukkan di depan." << endl;
27     }
28
29     // Menambahkan elemen di akhir
30     void insertBelakang(int value) {
31         Node* newNode = new Node();
32         newNode->data = value;
33         newNode->next = NULL;
34
35         if (head == NULL) {
36             head = newNode;
37             cout << "Elemen " << value << " dimasukkan sebagai elemen pertama." << endl;
38             return;
39         }
40
41         Node* temp = head;
42         while (temp->next != NULL)
```

```
43     temp = temp->next;
44
45     temp->next = newNode;
46     cout << "Elemen " << value << " dimasukkan di belakang." << endl;
47 }
48
49 // Menghapus elemen berdasarkan nilai
50 void hapus(int value) {
51     Node* temp = head;
52     Node* prev = NULL;
53
54     if (temp != NULL && temp->data == value) {
55         head = temp->next;
56         delete temp;
57         cout << "Elemen " << value << " dihapus dari list." << endl;
58         return;
59     }
60
61     while (temp != NULL && temp->data != value) {
62         prev = temp;
63         temp = temp->next;
64     }
65
66     if (temp == NULL) {
67         cout << "Elemen " << value << " tidak ditemukan." << endl;
68         return;
69     }
70
71     prev->next = temp->next;
72     delete temp;
73     cout << "Elemen " << value << " dihapus dari list." << endl;
74 }
75
76 // Menampilkan seluruh elemen dalam list
77 void tampilkan() {
78     Node* temp = head;
79
80     if (temp == NULL) {
81         cout << "List kosong." << endl;
82         return;
83     }
84
85     cout << "Isi Linked List: ";
```

```

85    }
86    ...
87    while (temp != NULL) {
88        cout << temp->data << " ";
89        temp = temp->next;
90    }
91 }
92
93 // Program utama
94 int main() {
95     LinkedList list;
96
97     list.insertDepan(10);
98     list.insertDepan(20);
99     list.insertBelakang(5);
100    list.insertBelakang(15);
101    list.tampilkan();
102
103    list.hapus(10);
104    list.tampilkan();
105
106    return 0;
107 }
```

```

Elemen 10 dimasukkan di depan.
Elemen 20 dimasukkan di depan.
Elemen 5 dimasukkan di belakang.
Elemen 15 dimasukkan di belakang.
Isi Linked List: 20 10 5 15
Elemen 10 dihapus dari list.
Isi Linked List: 20 5 15

-----
Process exited after 0.4842 seconds with return value 0
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE

head = NULL menandakan linked list kosong

insertDepan(value) - Menyisipkan di Depan

insertBelakang(value) - Menyisipkan di Belakang

hapus(value) - Menghapus Node Berdasarkan Nilai

tampilkan() - Menampilkan Seluruh Elemen

Fungsi Code

1. Mengimplementasikan singly linked list dengan operasi dasar
2. Mendemonstrasikan penyisipan di depan dan belakang
3. Menunjukkan penghapusan node berdasarkan nilai
4. Menampilkan traversal seluruh linked list

Keunggulan list vs Array

Keunggulan:

1. Alokasi Dinamis: Ukuran dapat berubah selama runtime
2. Penyisipan/penghapusan efisien: $O(1)$ untuk operasi di depan
3. Tidak ada wasted memory: Hanya mengalokasikan yang dibutuhkan
4. Fleksibel: Mudah melakukan reorganisasi node

Kekurangan:

1. Memory overhead: Setiap node butuh extra space untuk pointer
2. Akses sequential: Tidak bisa akses random seperti array
3. Complexity higher: Operasi di belakang butuh $O(n)$

KESIMPULAN

1. Struktur Dinamis: Linked list ideal untuk data yang sering berubah ukuran
2. Manajemen Memori: Menggunakan new dan delete untuk alokasi dinamis
3. Pointer Management: Kunci sukses linked list adalah manajemen pointer yang benar
4. Aplikasi Praktis:
 - o Implementasi stack dan queue
 - o Memory management dalam OS
 - o Undo functionality dalam aplikasi
 - o Symbol tables dalam compiler

Program di atas mendemonstrasikan operasi fundamental singly linked list dengan benar. Linked list merupakan pilihan tepat ketika membutuhkan struktur data yang fleksibel dengan operasi penyisipan dan penghapusan yang frequent, meskipun dengan trade-off dalam hal akses random dan memory overhead.

LAPORAN AKHIR PERTEMUAN 10

Dosen Pengampu: Jaka Sutresna, S.KOM., M.KOM



Disusun Oleh

Nama : Febrian Alkadir

Nim : 241011400901

Kelas : 03TPLP023

**TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG**

Jl. Surya Kencana No.1 Pamulang Telp (021)7412566, FAX. (021)741566
Tangerang Selatan

PERTEMUAN 10

APLIKASI SINGLE LINKED LIST PADA STACK

CONTOH CODE DAN OUTPUT PROGRAM PRAKTEK

```
1 // Program C ++ untuk mengimplementasikan tumpukan
2 // Menggunakan singly linked list
3 #include <bits/stdc++.h>
4 using namespace std;
5 // Deklarasikan node linked list
6 struct Node
7 {
8     int data;
9     struct Node* link;
10 };
11 struct Node* top;
12 // Fungsi utilitas untuk menambahkan elemen
13 // data dalam tumpukan dimasukkan di awal
14 void push(int data)
15 {
16     // Buat node baru dan alokasikan memori
17     struct Node* temp;
18     temp = new Node();
19     // Periksa apakah tumpukan (heap) sudah penuh.
20     // Kemudian masukkan elemen yang akan menyebabkan stack
21     // Tidak memerlukan "overflow" pada contoh program di modul
22     if (!temp)
23     {
24         cout << "\nHeap Overflow";
25         exit(1);
26     }
27     // Inisialisasi data ke data field
28     temp->data = data;
29     // letakkan top pointer referensi ke dalam temp link
30     temp->link = top;
31     // Jadikan sebagai bagian atas Stack
32     top = temp;
33 }
34 // Fungsi utilitas untuk memeriksa apakah tumpukan kosong atau tidak
35 int isEmpty()
36 {
37     return top == NULL;
38 }
39 // Fungsi utilitas untuk mengembalikan elemen teratas dalam stack
40 int peek()
41 {
42     // Periksa tumpukan kosong
```

```
43     if (!isEmpty())
44     return top->data;
45     else
46     exit(1);
47 }
48 // Fungsi utilitas untuk pop top
49 // elemen dari tumpukan
50 void pop()
51 {
52     struct Node* temp;
53     // Cek untuk stack underflow
54     if (top == NULL)
55     {
56         cout << "\nStack Underflow" << endl;
57         exit(1);
58     }
59     else
60     {
61         // Top assign into temp
62         temp = top;
63         // Assign second node to top
64         top = top->link;
65         // Hilangkan pembatas
66         // pertama dan kedua
67         temp->link = NULL;
68         // Lepaskan memori node teratas
69         free(temp);
70     }
71 }
72 // Berfungsi untuk mencetak semua file elemen stack
73 void display()
74 {
75     struct Node* temp;
76     // cek untuk stack underflow
77     if (top == NULL)
78     {
79         cout << "\nStack Underflow";
80         exit(1);
81     }
82     else
83     {
84         temp = top;
```

```
85 |     while (temp != NULL)
86 | {
87 |     // Cetak data node
88 |     cout << temp->data << "-> ";
89 |     // Assign temp Link to temp
90 |     temp = temp->link;
91 |
92 | }
93 |
94 | // Driver Code
95 | int main()
96 | {
97 |     // Push elemen ke dalam stack
98 |     push(11);
99 |     push(22);
100 |    push(33);
101 |    push(44);
102 |    // Menampilkan elemen stack
103 |    display();
104 |    // Mencetak elemen stack paling atas
105 |    cout << "\nElemen Top adalah "
106 |    << peek() << endl;
107 |    // Hapus elemen teratas stack
108 |    pop();
109 |    pop();
110 |    // Menampilkan elemen stack
111 |    display();
112 |    // Cetak elemen stack paling atas
113 |    cout << "\nElemen Top adalah"
114 |    << peek() << endl;
115 |    return 0;
116 | }
```

```
44-> 33-> 22-> 11->
Elemen Top adalah 44
22-> 11->
Elemen Top adalah 22

-----
Process exited after 0.4711 seconds with return value 0
Press any key to continue . . . |
```

PENJELASAN CODE DAN PENJELASAN OUTPUT CODE

Implementasi stack menggunakan singly linked list dimana setiap operasi push/pop dilakukan di head linked list untuk efisiensi waktu O(1). Top stack selalu merujuk ke head linked list.

`push(data)` - Menambahkan Elemen ke Stack

`pop()` - Menghapus Elemen dari Stack

`peek()` - Melihat Elemen Teratas

`isEmpty()` - Mengecek Stack Kosong

`display()` - Menampilkan Seluruh Stack

Fungsi Code:

1. Mengimplementasikan stack menggunakan singly linked list
2. Mendemonstrasikan operasi LIFO dengan benar
3. Menangani memory management dengan alokasi dan dealokasi dinamis
4. Mendeteksi error condition (stack underflow dan heap overflow)

Keunggulan Linked List Stack:

1. Ukuran Dinamis: Tidak ada batasan maksimum (kecuali memory system)
2. Tidak ada wasted memory: Hanya mengalokasikan yang dibutuhkan
3. Efisiensi operasi: Semua operasi O(1)
4. Tidak perlu pre-allocation: Tidak seperti array yang butuh ukuran tetap

Kekurangan Linked List Stack:

1. Memory overhead: Setiap node butuh extra space untuk pointer
2. Fragmentation memory: Alokasi terpisah dapat menyebabkan fragmentasi
3. Complex implementation: Lebih kompleks daripada array

KESIMPULAN

1. Implementasi Fleksibel: Stack dengan linked list ideal ketika ukuran stack tidak predictable
2. Efisiensi Operasi: Semua operasi utama O(1) - sangat efisien
3. Memory Management: Pentingnya mengelola memori dengan new dan free

4. Prinsip LIFO Terjaga: Implementasi mengikuti Last-In-First-Out dengan benar
5. Aplikasi Praktis:
 - a) Function call management dalam compiler
 - b) Undo mechanisms dalam aplikasi
 - c) Expression evaluation dan parsing
 - d) Backtracking algorithms

Program ini mendemonstrasikan implementasi stack yang efisien menggunakan linked list. Pendekatan ini memberikan fleksibilitas ukuran yang tidak dimiliki oleh implementasi array, dengan trade-off dalam hal memory overhead dan kompleksitas implementasi