**LAPORAN PRAKTIKUM   
ALGORITMA & STRUKTUR DATA  
MODUL 2**



**STACK & QUEUE**

|  |  |
| --- | --- |
| **Oleh:** | |
| **Rizki Adhitiya Maulana** | **NIM. 2410817110014** |

**PROGRAM STUDI TEKNOLOGI INFORMASI   
FAKULTAS TEKNIK   
UNIVERSITAS LAMBUNG MANGKURAT   
APRIL 2025**

# LEMBAR PENGESAHAN LAPORAN PRAKTIKUM ALGORITMA & STRUKTUR DATA MODUL 2

Laporan Praktikum Algoritma & Struktur Data Modul 2 : Stack & Queue ini disusun sebagai syarat lulus mata kuliah Praktikum Algoritma & Struktur Data. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Rizki Adhitiya Maulana

NIM : 2410817110014

|  |  |
| --- | --- |
| Menyetujui,  Asisten Praktikum  Muhammad Fauzan Ahsani  NIM. 2310817310009 | Mengetahui,  Dosen Penanggung Jawab Praktikum  Muti’a Maulida, S.Kom., M.TI.  NIP. 198810272019032013 |

# DAFTAR ISI

[LEMBAR PENGESAHAN i](#_Toc196276804)

[DAFTAR ISI ii](#_Toc196276805)

[DAFTAR TABEL iii](#_Toc196276806)

[DAFTAR GAMBAR iv](#_Toc196276807)

[SOAL 1 1](#_Toc196276808)

[A Pembahasan 1](#_Toc196276809)

[SOAL 2 2](#_Toc196276810)

[A Source Code 3](#_Toc196276811)

[B Output Program 7](#_Toc196276812)

[C Pembahasan 10](#_Toc196276813)

[SOAL 3 13](#_Toc196276814)

[A Source Code 14](#_Toc196276815)

[B Output Program 19](#_Toc196276816)

[C Pembahasan 22](#_Toc196276817)

[TAUTAN GIT HUB 25](#_Toc196276818)

# DAFTAR TABEL

[Tabel 1 Source Code Program Soal 2 3](#_Toc196276857)

[Tabel 2 Source Code Program Soal 3 14](#_Toc196276867)

# DAFTAR GAMBAR

[Gambar 1 Tampilan Awal Program Saat Dijalankan 7](#_Toc196276918)

[Gambar 2 Memasukkan Nilai Ke Dalam Stack 8](#_Toc196276919)

[Gambar 3 Menampilkan Stack Yang Sudah Dimasukkan 8](#_Toc196276920)

[Gambar 4 Melakukan Pop Pada Nilai Di Dalam Stack 8](#_Toc196276921)

[Gambar 5 Tampilan Setelah Nilai Teratas Di Pop 9](#_Toc196276922)

[Gambar 6 Membersihkan Stack Dengan Pilihan Menu 4 9](#_Toc196276923)

[Gambar 7 Tampilan Setelah Dibersihkan 9](#_Toc196276924)

[Gambar 8 Tampilan Jika Stack Sudah Penuh 10](#_Toc196276925)

[Gambar 9 Tampilan Saat Memilih Menu 5 Quit 10](#_Toc196276926)

[Gambar 10 Tampilan Awal Program Saat Dijalankan 19](#_Toc196276928)

[Gambar 11 Memasukkan Huruf ke Dalam Queue 19](#_Toc196276929)

[Gambar 12 Menampilkan Queue Yang Sudah Dimasukkan 19](#_Toc196276930)

[Gambar 13 Melakukan Delete Pada Huruf yang Sudah Dimasukkan 20](#_Toc196276931)

[Gambar 14 Tampilan Queue Setelah Melakukan Delete 20](#_Toc196276932)

[Gambar 15 Mereset Queue 20](#_Toc196276933)

[Gambar 16 Tampilan Queue Setelah di Reset 21](#_Toc196276934)

[Gambar 17 Tampilan Jika Queue Penuh 21](#_Toc196276935)

[Gambar 18 Tampilan Saat Memilih Menu 5 Quit 21](#_Toc196276936)

# SOAL 1

Apa perbedaan Stack dengan Queue?

## Pembahasan

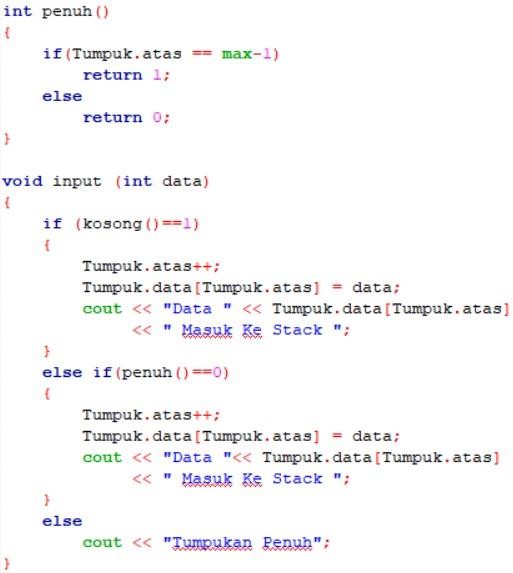
Stack dan Queue adalah struktur data yang mempunyai perbedaan dalam cara memasukkan/input (*push/enqueue*) dan mengeluarkan/output (*pop/dequeue*) data atau elemen yang ada.

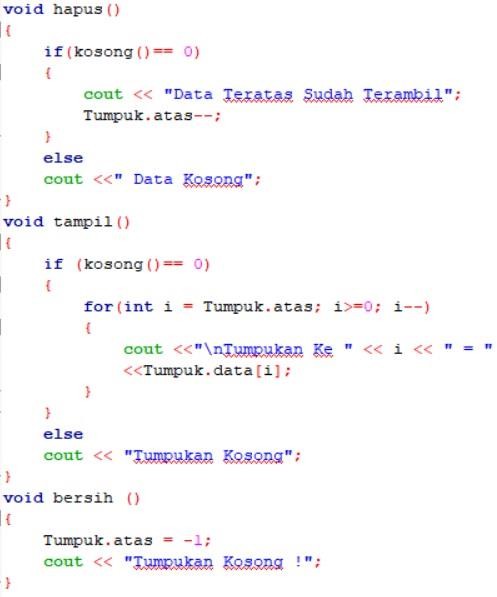
Pada struktur data Stack, prinsip yang digunakan adalah **LIFO** (*Last in First Out*), yang mana data atau elemen yang paling terakhir dimasukkan (*push*) akan menjadi data atau elemen yang paling awal dikeluarkan (*pop*). Sebagai contoh, kita ambil analogi seperti ketika kita menumpuk piring kotor yang ada, piring kotor yang berada di tumpukkan paling bawah akan selesai dibersihkan paling akhir sedangkan, piring kotor yang terakhir ditumpuk akan menjadi yang paling awal untuk dibersihkan.

Sedangkan, pada struktur data Queue, prinsip yang digunakan adalah **FIFO** (*First in First Out*), yang mana data atau elemen yang pertama dimasukkan (enqueue) akan menjadi data atau elemen yang pertama untuk dikeluarkan (*dequeue*). Sebagai contoh kita ambil analogi seperti ketika kita mengantri untuk memesan gacoan, orang yang datang atau mendaftar pertama kali ke meja kasih ialah yang akan dilayani lebih dulu.

# SOAL 2

Cobalah prog*r*am berikut, running dan analisis hasilnya. Buatlah algoritma untuk program tersebut.





## Source Code

Tabel 1 Source Code Program Soal 2

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144 | #include <iostream>  #include <conio.h>  #include <stdlib.h>  #define MAX 20    using namespace std;    struct Stack  {      int Atas;      int Data[MAX];  };    Stack Tumpuk;    int Kosong()  {      if (Tumpuk.Atas == -1)      {          return 1;      }      else      {          return 0;      }  }    int Penuh()  {      if (Tumpuk.Atas == MAX-1)      {          return 1;      }      else      {          return 0;      }  }    void Push(int Data)  {      if (Kosong()==1)      {          Tumpuk.Atas++;          Tumpuk.Data[Tumpuk.Atas] = Data;          cout << "Data " << Tumpuk.Data[Tumpuk.Atas] << " dimasukkan ke dalam Stack." << endl;      }      else if (Penuh()==0)      {          Tumpuk.Atas++;          Tumpuk.Data[Tumpuk.Atas] = Data;          cout << "Data " << Tumpuk.Data[Tumpuk.Atas] << " dimasukkan ke dalam Stack." << endl;      }      else      {          cout << "Stack telah Penuh!!!" << endl;      }  }    void Pop()  {      if (Kosong()==0)      {          cout << "Data " << Tumpuk.Data[Tumpuk.Atas] << " diambil dari Stack."<< endl;          Tumpuk.Atas--;      }      else      {          cout << "Stack Kosong!!!" << endl;      }  }    void Cetak\_Stack()  {      if (Kosong()==0)      {          for(int i = Tumpuk.Atas; i >= 0; i--)          {              cout << "\nTumpukan Ke " << i << " = " << Tumpuk.Data[i];          }          cout << endl;      }      else      {          cout << "Stack Kosong!!!" << endl;      }  }    void Bersihkan\_Stack()  {      Tumpuk.Atas = -1;      cout << "Stack telah dibersikan!!!" << endl;  }    void Inisialisasi()  {      Tumpuk.Atas = -1;  }    int main()  {      Inisialisasi();      int Pilihan, Data;      do{          cout << "\nSTACK" << endl;          cout << "==============" << endl;          cout << "1. PUSH" << endl;          cout << "2. POP" << endl;          cout << "3. CETAK STACK" << endl;          cout << "4. BERSIHKAN STACK" << endl;          cout << "5. QUIT" << endl;          cout << "PILIHAN : "; cin >> Pilihan;          switch (Pilihan)          {          case 1:              cout << "Masukkan Nilai: "; cin >> Data;              Push(Data);              break;            case 2:              Pop();              break;            case 3:              Cetak\_Stack();              break;            case 4:              Bersihkan\_Stack();              break;            default:              cout << "TERIMA KASIH" << endl;              break;          }          cout << "Press any key to continue";          getch();          system("cls");      }      while (Pilihan < 5);      return 0;  } |

## Output Program

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 1 Tampilan Awal Program Saat Dijalankan

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 2 Memasukkan Nilai Ke Dalam Stack

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 3 Menampilkan Stack Yang Sudah Dimasukkan

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 4 Melakukan Pop Pada Nilai Di Dalam Stack

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 5 Tampilan Setelah Nilai Teratas Di Pop

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 6 Membersihkan Stack Dengan Pilihan Menu 4

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 7 Tampilan Setelah Dibersihkan

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 8 Tampilan Jika Stack Sudah Penuh

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 9 Tampilan Saat Memilih Menu 5 Quit

## Pembahasan

Pada baris [1] sampai [3] terdapat *#include* yang mana digunakan untuk mengakses sebuah file yang diinginkan. *<iostream>* yang ada digunakan untuk input dan output. Kemudian *<conio.h>* digunakan agar menyediakan fungsi-fungsi yang berguna ketika ada interaksi langsung dengan keyboard, tanpa perlu menekan Enter.Terus *<stdlib.h>* digunakan untuk fungsi fungsi manajemen memori, konversi angka, kontrol proses, dan lingkungan program.

Pada baris [5] terdapat *#define* yang mana digunakan untuk membuat sebuah konstanta, *MAX 20* yang ada menjadi penjelas kalau kapasitas dari stack maksimal adalah 20 elemen.

Pada baris [7] terdapat *using namespace std;* yang mana digunakan untuk menghindari penulisan std

Pada baris [9] sampai [13] terdapat *struct stack* yang mana digunakan untuk menyimpan data dan info dari stack yang ada, dimana *int Atas* berguna untuk menyimpan posisi dari elemen paling atas pada stack dan *int Data[MAX]* berguna untuk menyimpan elemen pada stack sesuai dengan besar array yang ada.

Pada baris [15] terdapat *Stack Tumpuk* yang mana digunakan untuk membuat variabel yang bernama Tumpuk dari tipe data stack.

Pada baris [17] sampai [27] terdapat *int Kosong()* yang mana digunakan sebagai fungsi untuk mengecek apakah stack yang ada kosong atau tidak. Fungsi ini akan mengembalikan nilai 1 apabila nilai dari *Tumpuk.Atas* sama dengan *-1,* namun apabila nilai yang ada tidak sama dengan *-1* fungsi akan mengembalikan nilai 0.

Pada baris [29] sampai [39] terdapat *int Penuh()* yang mana digunakan sebagai fungsi untuk mengecek apakah stack yang ada sudah penuh atau belum. Fungsi ini akan mengembalikan nilai 1 apabila nilai dari *Tumpuk.Atas* sama dengan *MAX – 1 atau 19*, namun apabila nilai yang ada tidak sama dengan *MAX – 1 atau 19* fungsi akan mengembalikan 0.

Pada baris [41] sampai [59] terdapat *void Push()* yang mana digunakan sebagai fungsi untuk menambahkan nilai yang diinginkan ke dalam stack. Hal pertama yang dilakukan fungsi ini adalah mengecek apakah stack yang ada kosong, apabila stack yang ada kosong maka nilai yang ingin ditambahkan akan langsung dimasukkan ke dalam stack. Kemudian, apabila di dalam stack masih ada ruang untuk menambahkan nilai, maka nilai yang ingin ditambahkan akan langsung dimasukkan lagi ke dalam stack. Terus apabila nilai tidak bisa untuk ditambahkan lagi, menandakan kalau stack yang ada penuh dan akan muncul pesan “stack telah penuh!!!”.

Pada baris [61] sampai [72] terdapat *void Pop()* yang mana digunakan sebagai fungsi untuk menghapus data yang ada di paling atas dari stack. Hal pertama yang dilakukan fungsi ini adalah mengecek apakah stack yang ada kosong, apabila stack yang ada tidak kosong maka nilai yang ada di tumpukan paling atas akan dihapus. Kemudian apabila stack yang ada dalam keadaan kosong, akan muncul pesan “Stack Kosong!!!”.

Pada baris [74] sampai [88] terdapat *void Cetak\_Stack()* yang mana digunakan sebagai fungsi untuk menampilkan isi dari stack mulai dari atas ke bawah. Hal pertama yang dilakukan fungsi ini adalah mengecek apakah stack yang ada kosong, apabila stack yang ada tidak kosong maka *loop for* yang ada di dalam fungsi akan melakukan perulangan untuk menampilkan semua nilai yang telah diinput sebelumnya atau yang telah mengalami penghapusan mulai dari atas ke bawah. Kemudian apabila stack yang ada dalam keadaan kosong, akan muncul pesan “Stack Kosong!!!”.

Pada baris [90] sampai [94] terdapat *void Bersihkan\_Stack()* yang mana digunakan sebagai fungsi untuk mengosongkan atau membersihkan semua nilai yang telah diinput ke dalam stack. *Tumpuk.Atas = -1* pada fungsi mengatur nilai indeks yang ada menjadi -1 yang mana tidak ada nilai pada indeks tersebut, apabila di isi kembali nilai yang ada akan dimasukkan ke dalam indeks 0 yang menjadi stack atas.

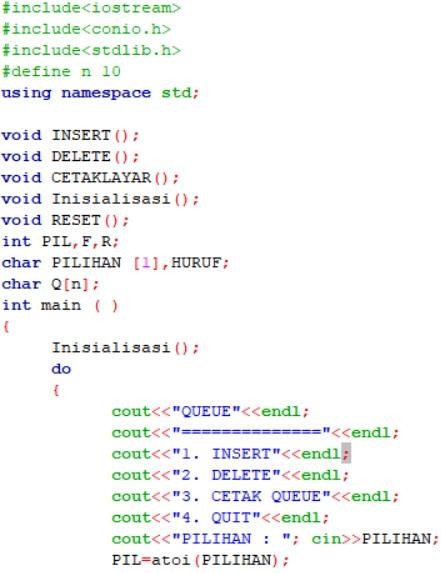
Pada baris [96] sampai [99] terdapat *void Inisialisasi()* yang mana digunakan sebagai fungsi untuk mengatur nilai awal dari stack agar berada di dalam keadaan kosong. Hal ini dilakukan dengan menetapkan nilai *Tumpuk.Atas = -1*. Nilai -1 menandakan bahwa tidak ada nilai yang tersimpan di dalam stack.

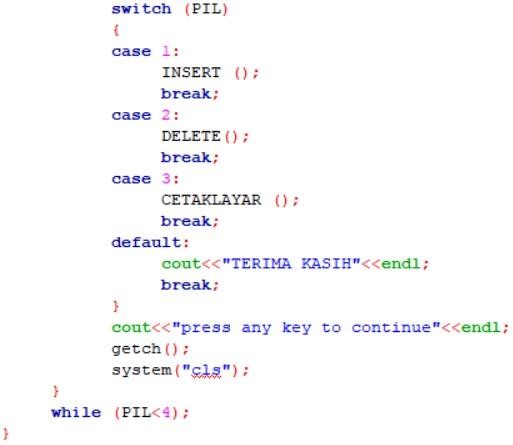
Pada baris [101] sampai [143] terdapat *int main()* yang mana digunakan untuk menjalankan dan menampilkan menu CLI. Fungsi *inisialisasi()* dipanggil di awal untuk mengatur stack agar berada di dalam kondisi kosong, kemudian ada beberapa pilihan seperti *Push, Pop, Cetak, Bersihkan, dan Quit* yang dapat dipilih sesuai dengan *switch-case* yang diinput user. Terdapat *getch()* untuk menunggu tombol yang ditekan oleh pengguna dan membersihkan layar menggunakan *system("cls")*. Terus program akan terus berjalan selama user tidak memilih pilihan lima (5) untuk keluar atau menghentikan program yang ada.

# 

# SOAL 3

Buatlah program dengan tampilan sebagai berikut:





## Source Code

Tabel 2 Source Code Program Soal 3

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142 | #include <iostream>  #include <conio.h>  #include <stdlib.h>    #define MAX 20    using namespace std;    struct Queue  {      int Front, Rear, Size;      char Q[MAX];  };    Queue Antrian;    int Kosong()  {      if (Antrian.Front == Antrian.Rear)      {          return 1;      }      else      {          return 0;      }  }    int Penuh()  {      if ((Antrian.Rear + 1) % Antrian.Size == Antrian.Front)      {          return 1;      }      else{          return 0;      }  }    void INSERT(char huruf)  {      if (Penuh() == 1)      {          cout << "Queue Penuh!!!" << endl;      }      else      {          Antrian.Q[Antrian.Rear] = huruf;          cout << "Data: " << Antrian.Q[Antrian.Rear] << "masuk ke dalam Queue" << endl;          Antrian.Rear = (Antrian.Rear + 1) % Antrian.Size;      }  }    void DELETE()  {      if (Kosong() == 1)      {          cout << "Queue kosong!!!" << endl;      }      else      {          cout << "Data yang dihapus: " << Antrian.Q[Antrian.Front] << endl;          Antrian.Front = (Antrian.Front + 1) % Antrian.Size;      }  }    void CETAKLAYAR()  {      if(Kosong()==1)      {          cout << "Queue kosong" << endl;      }      else      {          int i = Antrian.Front;          while(i != Antrian.Rear)          {              cout << "Queue ke- " << i << " = " << Antrian.Q[i] << endl;              i = (i + 1) % Antrian.Size;          }      }  }    void RESET()  {      Antrian.Front = 0;      Antrian.Rear = 0;      Antrian.Size = MAX;      cout << "Queue telah di-reset" << endl;  }    void Inisialisasi()  {      Antrian.Front = 0;      Antrian.Rear = 0;      Antrian.Size = MAX;  }    int main()  {      Inisialisasi();      int Pilihan;      char huruf;      do{          cout << "\nQUEUE" << endl;          cout << "==============" << endl;          cout << "1. INSERT" << endl;          cout << "2. DELETE" << endl;          cout << "3. CETAK QUEUE" << endl;          cout << "4. RESET QUEUE" << endl;          cout << "5. QUIT" << endl;          cout << "PILIHAN : "; cin >> Pilihan;          switch (Pilihan)          {          case 1:              cout << "Masukkan Nilai: "; cin >> huruf;              INSERT(huruf);              break;            case 2:              DELETE();              break;            case 3:              CETAKLAYAR();              break;            case 4:              RESET();              break;            default:              cout << "TERIMA KASIH" << endl;              break;          }          cout << "Press any key to continue";          getch();          system("cls");      }      while (Pilihan < 5);      return 0;  } |

## Output Program

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 10 Tampilan Awal Program Saat Dijalankan

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 11 Memasukkan Huruf ke Dalam Queue

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 12 Menampilkan Queue Yang Sudah Dimasukkan

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 13 Melakukan Delete Pada Huruf yang Sudah Dimasukkan

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 14 Tampilan Queue Setelah Melakukan Delete

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 15 Mereset Queue

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 16 Tampilan Queue Setelah di Reset

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 17 Tampilan Jika Queue Penuh

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

Gambar 18 Tampilan Saat Memilih Menu 5 Quit

## Pembahasan

Pada baris [1] sampai [3] terdapat *#include* yang mana digunakan untuk mengakses sebuah file yang diinginkan. *<iostream>* yang ada digunakan untuk input dan output. Kemudian *<conio.h>* digunakan agar menyediakan fungsi-fungsi yang berguna ketika ada interaksi langsung dengan keyboard, tanpa perlu menekan Enter.Terus *<stdlib.h>* digunakan untuk fungsi fungsi manajemen memori, konversi angka, kontrol proses, dan lingkungan program.

Pada baris [5] terdapat *#define* yang mana digunakan untuk membuat sebuah konstanta, *MAX 20* yang ada menjadi penjelas kalau kapasitas dari stack maksimal adalah 20 elemen.

Pada baris [7] terdapat *using namespace std;* yang mana digunakan untuk menghindari penulisan std

Pada baris [9] sampai [13] terdapat *struct Queue* yang mana digunakan untuk menyimpan data dan info antrian, dimana *Front* berguna untuk menunjukkan posisi data terdepan di dalam queue. Kemudian *Rear* berguna untuk menunjukkan posisi data terakhir di dalam queue, *Size* berguna untuk menerangkan seberapa banyak data yang dapat ditampung pada queue, dan *Q[MAX]* berguna untuk menyimpan elemen pada stack sesuai dengan besar array yang ada.

Pada baris [15] terdapat *Queue Antrian* yang mana digunakan untuk membuat variabel yang bernama Tumpuk dari tipe data stack.

Pada baris [17] sampai [27] terdapat *int Kosong()* yang mana digunakan sebagai fungsi untuk mengecek apakah queue yang ada kosong atau tidak. Fungsi ini akan mengembalikan nilai 1 apabila nilai dari *antrean.Front* sama dengan *antrean.Rear.* Begitu juga sebaliknya apabila nilai yang ada tidak sama dengan,fungsi akan mengembalikan nilai 0.

Pada baris [29] sampai [38] terdapat *int Penuh()* yang mana digunakan sebagai fungsi untuk mengecek apakah queue yang ada sudah penuh atau belum. Fungsi ini akan mengembalikan nilai 1 apabila kondisi dari *(Antrian.Rear + 1) % Antrian.Size == Antrian.Front* terpenuhi, apabila tidak terpenuhi maka fungsi akan mengembalikan nilai 0.

Pada baris [40] sampai [52] terdapat *void INSERT(char huruf)* yang mana digunakan sebagai fungsi untuk menambahkan huruf yang diinginkan ke dalam queue. Hal pertama yang dilakukan fungsi ini adalah mengecek apakah queue yang ada kosong, apabila queue yang ada kosong maka huruf yang ingin ditambahkan ke dalam queue. Terus apabila huruf yang diinginkan tidak bisa lagi untuk ditambahkan, itu menandakan kalau queue yang ada penuh dan akan muncul pesan “Queue Penuh!!!”.

Pada baris [54] sampai [65] terdapat *void DELETE()* yang mana digunakan sebagai fungsi untuk menghapus data yang posisinya berada di paling depan pada queue. Hal pertama yang dilakukan fungsi ini adalah mengecek apakah queue yang ada kosong, apabila queue yang ada tidak kosong maka huruf yang ada di paling depan akan dihapus. Kemudian apabila queue yang ada dalam keadaan kosong, akan muncul pesan “Queue kosong!!!”.

Pada baris [67] sampai [82] terdapat *void CETAKLAYAR()* yang mana digunakan sebagai fungsi untuk menampilkan isi dari queue mulai dari yang pertama dimasukkan sampai yang terakhir dimasukkan (sesuai dengan urutan diinput). Hal pertama yang dilakukan fungsi ini adalah mengecek apakah queue yang ada kosong, apabila queue yang ada tidak kosong maka *loop* yang ada akan melakukan perulangan dalam menampilkan semua nilai yang telah diinput sebelumnya atau yang telah mengalami penghapusan mulai dari atas ke bawah. Kemudian apabila queue yang ada dalam keadaan kosong, akan muncul pesan “Queue kosong!!!”.

Pada baris [84] sampai [90] terdapat *void RESET()* yang mana digunakan sebagai fungsi untuk mengosongkan atau membersihkan semua yang telah diinput ke dalam queue. *antrean.Front = 0* dan *antrean.Rear = 0* pada fungsi untuk mengatur ulang posisi *Front* dan *Rear* kembali ke indeks 0, yang mana sama seperti kondisi awal sebelum diinput.

Pada baris [92] sampai [97] terdapat *void Inisialisasi()* yang mana digunakan sebagai fungsi untuk mengatur ulang posisi *Front* dan *Rear* kembali ke indeks 0.

Pada baris [99] sampai [142] terdapat *int main()* yang mana digunakan untuk menjalankan dan menampilkan menu CLI. Fungsi *inisialisasi()* dipanggil di awal untuk mengatur queue agar berada di dalam kondisi kosong, kemudian ada beberapa pilihan seperti *Insert, Delete, Cetaklayar, Reset, dan Quit* yang dapat dipilih sesuai dengan *switch-case* yang diinput user. Pada bagian *switch-case 1* user diminta untuk memasukkan sebuah string dan hanya karakter pertama yang di masukkan pada program yang akan diproses untuk dimasukkan ke dalam queue. Kemudian terdapat *getch()* untuk menunggu tombol yang ditekan oleh pengguna dan membersihkan layar menggunakan *system("cls")*. Terus program akan terus berjalan selama user tidak memilih pilihan lima (5) untuk keluar atau menghentikan program yang ada.

# TAUTAN GIT HUB

https://github.com/Rizki-A-M/Rizki-A-M-PRAKTIKUM\_ALGORITMA\_DAN\_STRUKTUR\_DATA.git