

**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

Mengetahui,  
Dosen Penanggung Jawab Praktikum

Muhammad Fauzan Ahsani  
NIM. 2310817310009

Muti'a Maulida, S.Kom., M.TI.  
NIP. 198810272019032013

## DAFTAR ISI

|                         |     |
|-------------------------|-----|
| LEMBAR PENGESAHAN ..... | i   |
| DAFTAR ISI.....         | ii  |
| DAFTAR TABEL.....       | iii |
| DAFTAR GAMBAR .....     | iv  |
| SOAL 1 .....            | 1   |
| A Pembahasan.....       | 1   |
| SOAL 2 .....            | 2   |
| A Source Code .....     | 3   |
| B Output Program .....  | 7   |
| C Pembahasan .....      | 10  |
| SOAL 3 .....            | 13  |
| A Source Code .....     | 14  |
| B Output Program .....  | 19  |
| C Pembahasan .....      | 22  |
| TAUTAN GIT HUB .....    | 25  |

## DAFTAR TABEL

|  |    |
|--|----|
| Tabel 1 Source Code Program Soal 2 ..... | 3  |
| Tabel 2 Source Code Program Soal 3 ..... | 14 |

## DAFTAR GAMBAR

|   |    |
|---|----|
| Gambar 1 Tampilan Awal Program Saat Dijalankan.....               | 7  |
| Gambar 2 Memasukkan Nilai Ke Dalam Stack .....                    | 8  |
| Gambar 3 Menampilkan Stack Yang Sudah Dimasukkan .....            | 8  |
| Gambar 4 Melakukan Pop Pada Nilai Di Dalam Stack .....            | 8  |
| Gambar 5 Tampilan Setelah Nilai Teratas Di Pop.....               | 9  |
| Gambar 6 Membersihkan Stack Dengan Pilihan Menu 4.....            | 9  |
| Gambar 7 Tampilan Setelah Dibersihkan .....                       | 9  |
| Gambar 8 Tampilan Jika Stack Sudah Penuh .....                    | 10 |
| Gambar 9 Tampilan Saat Memilih Menu 5 Quit .....                  | 10 |
| Gambar 10 Tampilan Awal Program Saat Dijalankan.....              | 19 |
| Gambar 11 Memasukkan Huruf ke Dalam Queue.....                    | 19 |
| Gambar 12 Menampilkan Queue Yang Sudah Dimasukkan .....           | 19 |
| Gambar 13 Melakukan Delete Pada Huruf yang Sudah Dimasukkan ..... | 20 |
| Gambar 14 Tampilan Queue Setelah Melakukan Delete .....           | 20 |
| Gambar 15 Mereset Queue.....                                      | 20 |
| Gambar 16 Tampilan Queue Setelah di Reset .....                   | 21 |
| Gambar 17 Tampilan Jika Queue Penuh.....                          | 21 |
| Gambar 18 Tampilan Saat Memilih Menu 5 Quit .....                 | 21 |

## SOAL 1

Apa perbedaan Stack dengan Queue?

### A 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 kasir ialah yang akan dilayani lebih dulu.

## SOAL 2

Cobalah program berikut, running dan analisis hasilnya. Buatlah algoritma untuk program tersebut.

```
int penuh()
{
    if(Tumpuk.atas == max-1)
        return 1;
    else
        return 0;
}

void input (int data)
{
    if (kosong()==1)
    {
        Tumpuk.atas++;
        Tumpuk.data[Tumpuk.atas] = data;
        cout << "Data " << Tumpuk.data[Tumpuk.atas]
            << " Masuk Ke Stack ";
    }
    else if(penuh()==0)
    {
        Tumpuk.atas++;
        Tumpuk.data[Tumpuk.atas] = data;
        cout << "Data " << Tumpuk.data[Tumpuk.atas]
            << " Masuk Ke Stack ";
    }
    else
        cout << "Tumpukan Penuh";
}

void hapus()
{
    if(kosong()== 0)
    {
        cout << "Data Teratas Sudah Terambil";
        Tumpuk.atas--;
    }
    else
        cout << " Data Kosong";
}

void tampil()
{
    if (kosong()== 0)
    {
        for(int i = Tumpuk.atas; i>=0; i--)
        {
            cout << "\nTumpukan Ke " << i << " = "
                << Tumpuk.data[i];
        }
    }
    else
        cout << "Tumpukan Kosong";
}

void bersih ()
{
    Tumpuk.atas = -1;
    cout << "Tumpukan Kosong !";
}
```

## A Source Code

*Tabel 1 Source Code Program Soal 2*

|    |                           |
|----|---------------------------|
| 1  | #include <iostream>       |
| 2  | #include <conio.h>        |
| 3  | #include <stdlib.h>       |
| 4  |                           |
| 5  | #define MAX 20            |
| 6  |                           |
| 7  | using namespace std;      |
| 8  |                           |
| 9  | struct Stack              |
| 10 | {                         |
| 11 | int Atas;                 |
| 12 | int Data[MAX];            |
| 13 | };                        |
| 14 |                           |
| 15 | Stack Tumpuk;             |
| 16 |                           |
| 17 | int Kosong()              |
| 18 | {                         |
| 19 | if (Tumpuk.Atas == -1)    |
| 20 | {                         |
| 21 | return 1;                 |
| 22 | }                         |
| 23 | else                      |
| 24 | {                         |
| 25 | return 0;                 |
| 26 | }                         |
| 27 | }                         |
| 28 |                           |
| 29 | int Penuh()               |
| 30 | {                         |
| 31 | if (Tumpuk.Atas == MAX-1) |



```

32     {
33         return 1;
34     }
35     else
36     {
37         return 0;
38     }
39 }
40
41 void Push(int Data)
42 {
43     if (Kosong()==1)
44     {
45         Tumpuk.Atas++;
46         Tumpuk.Data[Tumpuk.Atas] = Data;
47         cout << "Data " << Tumpuk.Data[Tumpuk.Atas]
48 << " dimasukkan ke dalam Stack." << endl;
49     }
50     else if (Penuh()==0)
51     {
52         Tumpuk.Atas++;
53         Tumpuk.Data[Tumpuk.Atas] = Data;
54         cout << "Data " << Tumpuk.Data[Tumpuk.Atas]
55 << " dimasukkan ke dalam Stack." << endl;
56     }
57     else
58     {
59         cout << "Stack telah Penuh!!!" << endl;
60     }
61 }
62 void Pop()
63 {

```

```

63     if (Kosong()==0)
64     {
65         cout << "Data " << Tumpuk.Data[Tumpuk.Atas]
66         << " diambil dari Stack."<< endl;
67         Tumpuk.Atas--;
68     }
69     else
70     {
71         cout << "Stack Kosong!!!" << endl;
72     }
73
74 void Cetak_Stack()
75 {
76     if (Kosong()==0)
77     {
78         for(int i = Tumpuk.Atas; i >= 0; i--)
79         {
80             cout << "\nTumpukan Ke " << i << " = " <<
81             Tumpuk.Data[i];
82             cout << endl;
83         }
84     }
85     else
86     {
87         cout << "Stack Kosong!!!" << endl;
88     }
89
90 void Bersihkan_Stack()
91 {
92     Tumpuk.Atas = -1;
93     cout << "Stack telah dibersihkan!!!" << endl;

```

```

94     }
95
96     void Inisialisasi()
97     {
98         Tumpuk.Atas = -1;
99     }
100
101     int main()
102     {
103         Inisialisasi();
104         int Pilihan, Data;
105         do{
106             cout << "\nSTACK" << endl;
107             cout << "======" << endl;
108             cout << "1. PUSH" << endl;
109             cout << "2. POP" << endl;
110             cout << "3. CETAK STACK" << endl;
111             cout << "4. BERSIHKAN STACK" << endl;
112             cout << "5. QUIT" << endl;
113             cout << "PILIHAN : "; cin >> Pilihan;
114             switch (Pilihan)
115             {
116                 case 1:
117                     cout << "Masukkan Nilai: "; cin >> Data;
118                     Push(Data);
119                     break;
120
121                 case 2:
122                     Pop();
123                     break;
124
125                 case 3:
126                     Cetak_Stack();

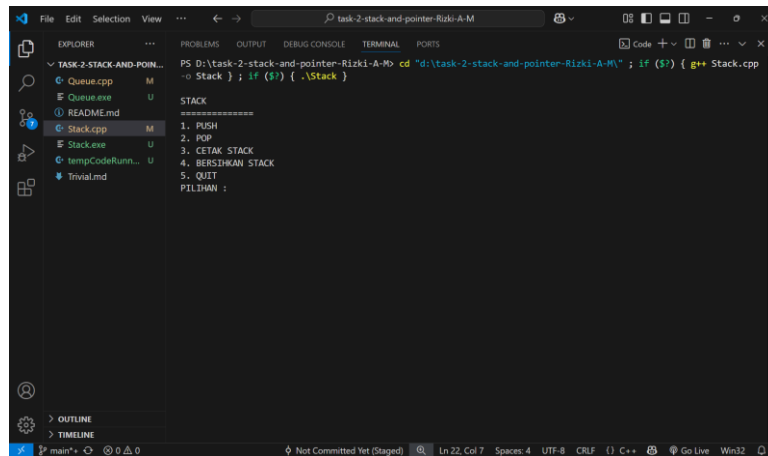
```

```

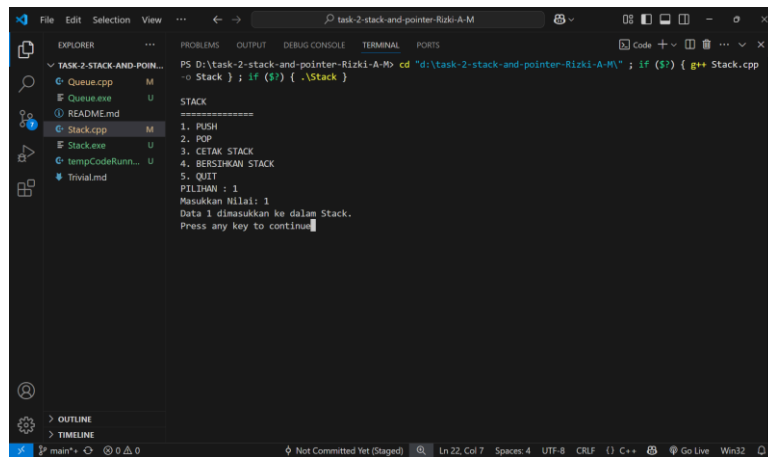
127         break;
128
129     case 4:
130         Bersihkan_Stack();
131         break;
132
133     default:
134         cout << "TERIMA KASIH" << endl;
135         break;
136     }
137     cout << "Press any key to continue";
138     getch();
139     system("cls");
140 }
141 while (Pilihan < 5);
142 return 0;
143 }
144

```

## B Output Program



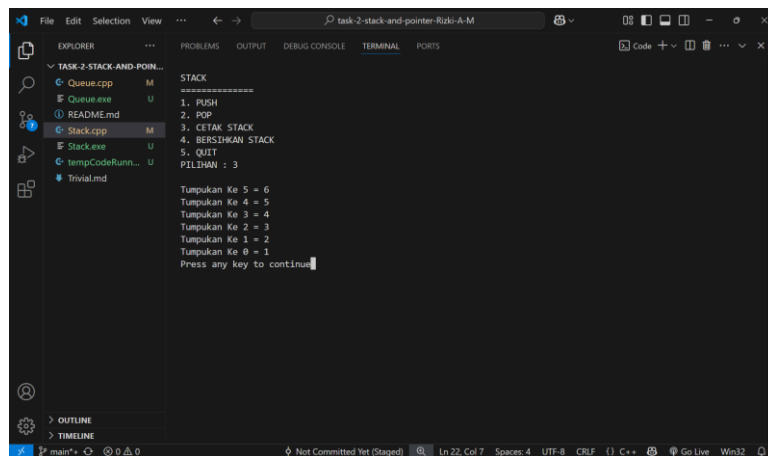
Gambar 1 Tampilan Awal Program Saat Dijalankan



```
PS D:\task-2-stack-and-pointer-Rizki-A-M> cd "d:\task-2-stack-and-pointer-Rizki-A-M\" ; if ($?) { g++ Stack.cpp
-> Stack } ; if ($?) { .\Stack }

STACK
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN : 1
Masukkan Nilai: 1
Data 1 dimasukkan ke dalam Stack.
Press any key to continue
```

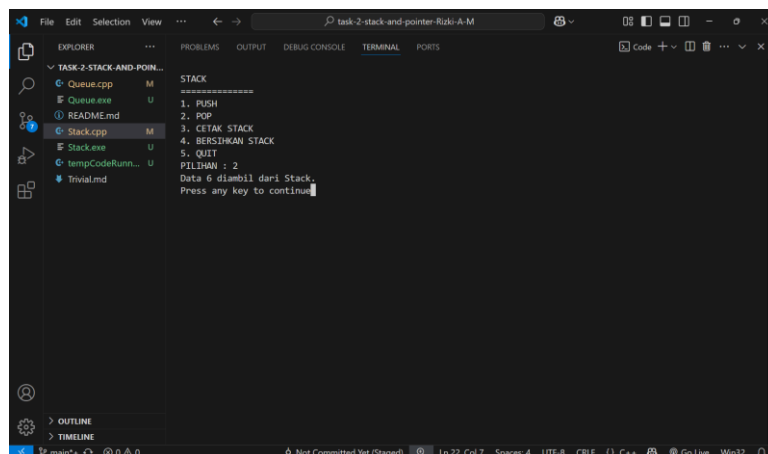
*Gambar 2 Memasukkan Nilai Ke Dalam Stack*



```
STACK
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN : 3

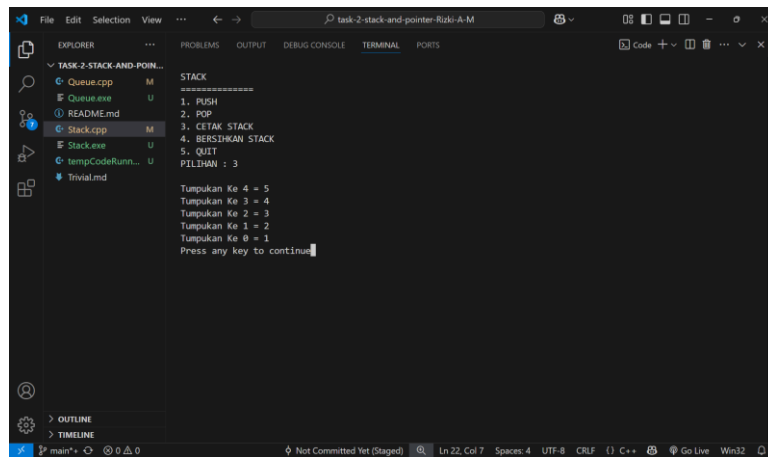
Tumpukan Ke 5 = 6
Tumpukan Ke 4 = 5
Tumpukan Ke 3 = 4
Tumpukan Ke 2 = 3
Tumpukan Ke 1 = 2
Tumpukan Ke 0 = 1
Press any key to continue
```

*Gambar 3 Menampilkan Stack Yang Sudah Dimasukkan*

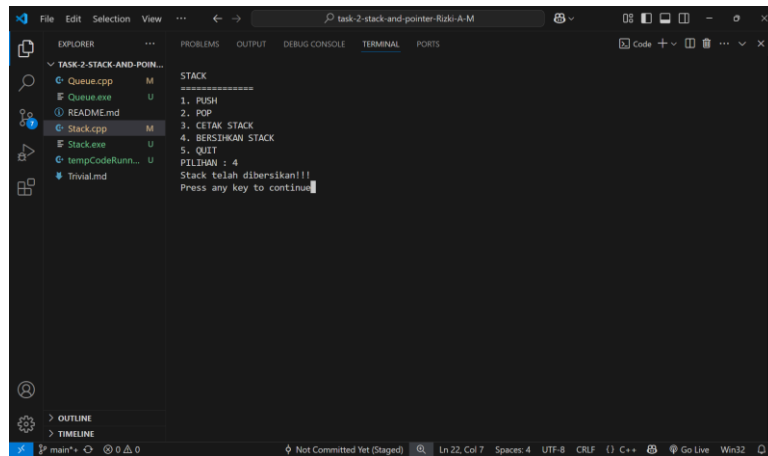


```
STACK
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN : 2
Data 6 diambil dari Stack.
Press any key to continue
```

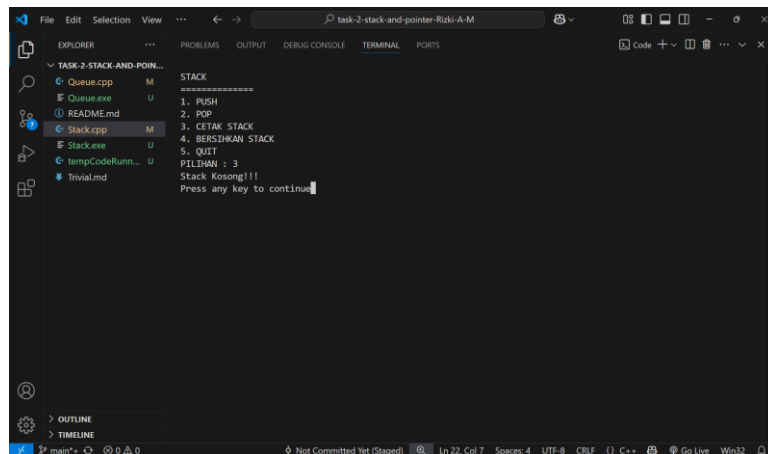
*Gambar 4 Melakukan Pop Pada Nilai Di Dalam Stack*



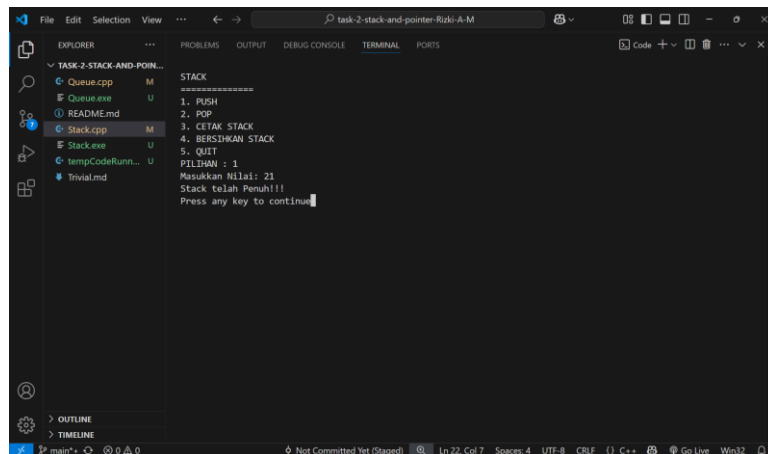
*Gambar 5 Tampilan Setelah Nilai Teratas Di Pop*



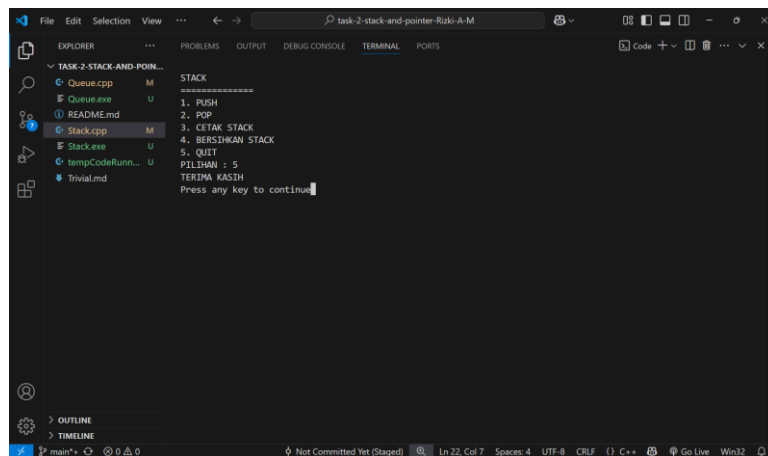
*Gambar 6 Membersihkan Stack Dengan Pilihan Menu 4*



*Gambar 7 Tampilan Setelah Dibersihkan*



Gambar 8 Tampilan Jika Stack Sudah Penuh



Gambar 9 Tampilan Saat Memilih Menu 5 Quit

## C 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:

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
#define n 10
using namespace std;

void INSERT();
void DELETE();
void CETAKLAYAR();
void Inisialisasi();
void RESET();
int PIL,F,R;
char PILIHAN [1],HURUF;
char Q[n];
int main ( )
{
    Inisialisasi();
    do
    {
        cout<<"QUEUE"<<endl;
        cout<<"======"<<endl;
        cout<<"1. INSERT"<<endl;
        cout<<"2. DELETE"<<endl;
        cout<<"3. CETAK QUEUE"<<endl;
        cout<<"4. QUIT"<<endl;
        cout<<"PILIHAN : "; cin>>PILIHAN;
        PIL=atoi(PILIHAN);
```

```

        switch (PIL)
        {
        case 1:
            INSERT ();
            break;
        case 2:
            DELETE ();
            break;
        case 3:
            CETAKLAYAR ();
            break;
        default:
            cout<<"TERIMA KASIH"<<endl;
            break;
        }
        cout<<"press any key to continue"<<endl;
        getch();
        system("cls");
    }
    while (PIL<4);
}

```

## A Source Code

*Tabel 2 Source Code Program Soal 3*

|    |                        |
|----|------------------------|
| 1  | #include <iostream>    |
| 2  | #include <conio.h>     |
| 3  | #include <stdlib.h>    |
| 4  |                        |
| 5  | #define MAX 20         |
| 6  |                        |
| 7  | using namespace std;   |
| 8  |                        |
| 9  | struct Queue           |
| 10 | {                      |
| 11 | int Front, Rear, Size; |
| 12 | char Q[MAX];           |
| 13 | };                     |
| 14 |                        |
| 15 | Queue Antrian;         |

```

16
17 int Kosong()
18 {
19     if (Antrian.Front == Antrian.Rear)
20     {
21         return 1;
22     }
23     else
24     {
25         return 0;
26     }
27 }
28
29 int Penuh()
30 {
31     if ((Antrian.Rear + 1) % Antrian.Size ==
Antrian.Front)
32     {
33         return 1;
34     }
35     else{
36         return 0;
37     }
38 }
39
40 void INSERT(char huruf)
41 {
42     if (Penuh() == 1)
43     {
44         cout << "Queue Penuh!!!" << endl;
45     }
46     else
47     {

```

|    |  |
|----|--|
| 48 | Antrian.Q[Antrian.Rear] = huruf;   |
| 49 | cout << "Data: " << Antrian.Q[Antrian.Rear] << "masuk ke dalam Queue" << endl; |
| 50 | Antrian.Rear = (Antrian.Rear + 1) % Antrian.Size;                              |
| 51 | }  |
| 52 | }  |
| 53 |  |
| 54 | void DELETE()  |
| 55 | {  |
| 56 | if (Kosong() == 1)   |
| 57 | {  |
| 58 | cout << "Queue kosong!!!" << endl;   |
| 59 | }  |
| 60 | else   |
| 61 | {  |
| 62 | cout << "Data yang dihapus: " << Antrian.Q[Antrian.Front] << endl;             |
| 63 | Antrian.Front = (Antrian.Front + 1) % Antrian.Size;                            |
| 64 | }  |
| 65 | }  |
| 66 |  |
| 67 | void CETAKLAYAR()  |
| 68 | {  |
| 69 | if(Kosong()==1)  |
| 70 | {  |
| 71 | cout << "Queue kosong" << endl;  |
| 72 | }  |
| 73 | else   |
| 74 | {  |
| 75 | int i = Antrian.Front;   |
| 76 | while(i != Antrian.Rear)   |

```

77         {
78             cout << "Queue ke- " << i << " = " <<
Antrian.Q[i] << endl;
79             i = (i + 1) % Antrian.Size;
80         }
81     }
82 }
83
84 void RESET()
85 {
86     Antrian.Front = 0;
87     Antrian.Rear = 0;
88     Antrian.Size = MAX;
89     cout << "Queue telah di-reset" << endl;
90 }
91
92 void Inisialisasi()
93 {
94     Antrian.Front = 0;
95     Antrian.Rear = 0;
96     Antrian.Size = MAX;
97 }
98
99 int main()
100 {
101     Inisialisasi();
102     int Pilihan;
103     char huruf;
104     do{
105         cout << "\nQUEUE" << endl;
106         cout << "======" << endl;
107         cout << "1. INSERT" << endl;
108         cout << "2. DELETE" << endl;

```

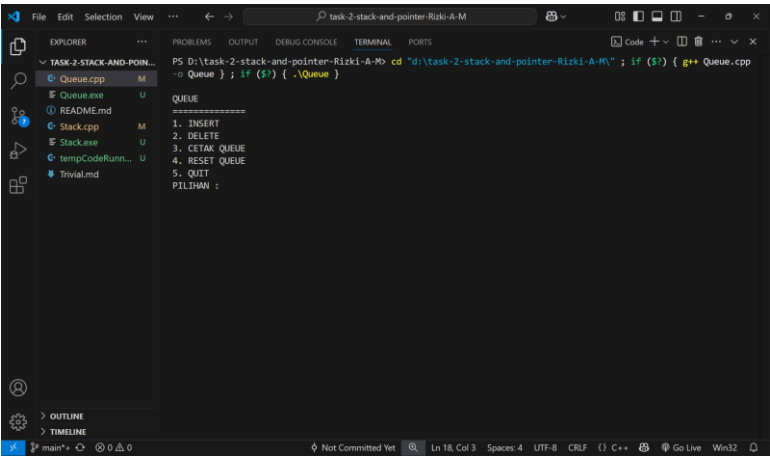
```

109     cout << "3. CETAK QUEUE" << endl;
110     cout << "4. RESET QUEUE" << endl;
111     cout << "5. QUIT" << endl;
112     cout << "PILIHAN : "; cin >> Pilihan;
113     switch (Pilihan)
114     {
115     case 1:
116         cout << "Masukkan Nilai: "; cin >> huruf;
117         INSERT(huruf);
118         break;
119
120     case 2:
121         DELETE();
122         break;
123
124     case 3:
125         CETAKLAYAR();
126         break;
127
128     case 4:
129         RESET();
130         break;
131
132     default:
133         cout << "TERIMA KASIH" << endl;
134         break;
135     }
136     cout << "Press any key to continue";
137     getch();
138     system("cls");
139 }
140 while (Pilihan < 5);
141 return 0;

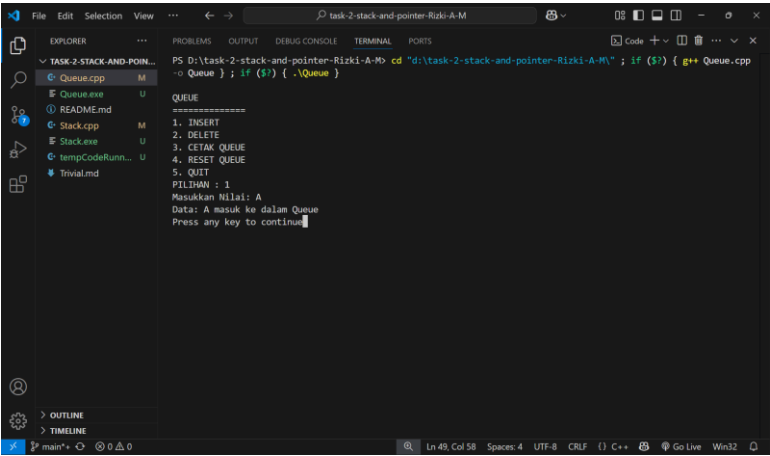
```

|     |   |
|-----|---|
| 142 | } |
|-----|---|

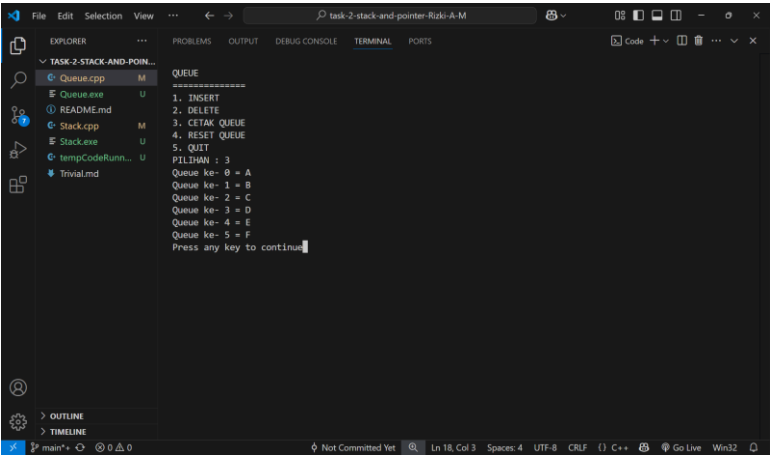
## B Output Program



Gambar 10 Tampilan Awal Program Saat Dijalankan



Gambar 11 Memasukkan Huruf ke Dalam Queue



Gambar 12 Menampilkan Queue Yang Sudah Dimasukkan



```
task-2-stack-and-pointer-Rizki-A-M
EXPLORER  PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
TASK-2-STACK-AND-POIN...
  Queue.cpp  M
  Queue.exe  U
  README.md  M
  Stack.cpp  M
  Stack.exe  U
  tempCodeRunn...  U
  Trivial.md

QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 2
Data yang dihapus: A
Press any key to continue
```

*Gambar 13 Melakukan Delete Pada Huruf yang Sudah Dimasukkan*

```
task-2-stack-and-pointer-Rizki-A-M
EXPLORER  PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
TASK-2-STACK-AND-POIN...
  Queue.cpp  M
  Queue.exe  U
  README.md  M
  Stack.cpp  M
  Stack.exe  U
  tempCodeRunn...  U
  Trivial.md

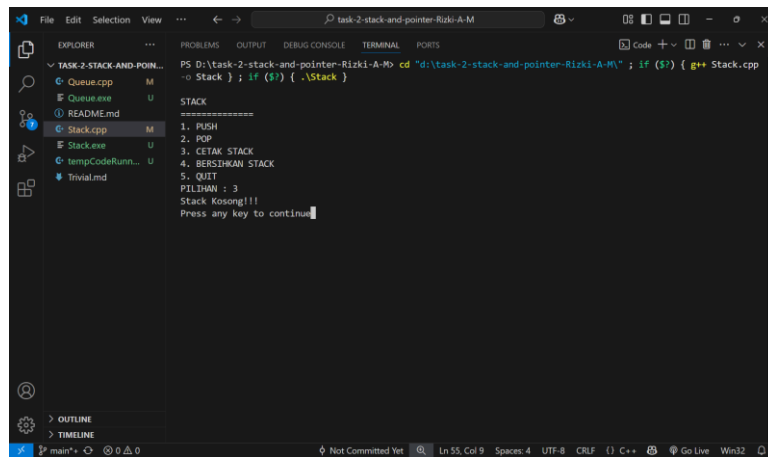
QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 3
Queue ke-1 = B
Queue ke-2 = C
Queue ke-3 = D
Queue ke-4 = E
Queue ke-5 = F
Press any key to continue
```

*Gambar 14 Tampilan Queue Setelah Melakukan Delete*

```
task-2-stack-and-pointer-Rizki-A-M
EXPLORER  PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
TASK-2-STACK-AND-POIN...
  Queue.cpp  M
  Queue.exe  U
  README.md  M
  Stack.cpp  M
  Stack.exe  U
  tempCodeRunn...  U
  Trivial.md

QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 4
Queue telah di-reset
Press any key to continue
```

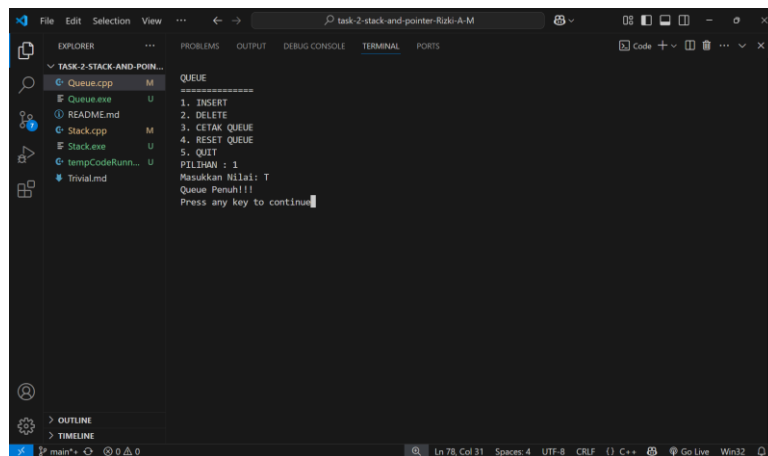
*Gambar 15 Mereset Queue*



```
PS D:\task-2-stack-and-pointer-Rizki-A-M> cd "d:\task-2-stack-and-pointer-Rizki-A-M\" ; if ($?) { g++ Stack.cpp
-> Stack } ; if ($?) { .\Stack }

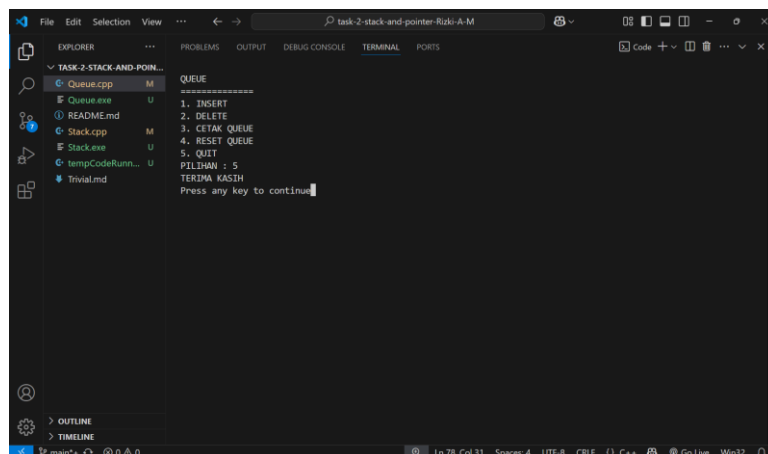
STACK
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHAN STACK
5. QUIT
PILIHAN : 3
Stack Kosong!!!
Press any key to continue
```

*Gambar 16 Tampilan Queue Setelah di Reset*



```
QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 1
Masukkan Nilai: T
Queue Penuh!!!
Press any key to continue
```

*Gambar 17 Tampilan Jika Queue Penuh*



```
QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 5
TERIMA KASIH
Press any key to continue
```

*Gambar 18 Tampilan Saat Memilih Menu 5 Quit*

## C 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 dimasukkan 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](https://github.com/Rizki-A-M/Rizki-A-M-PRAKTIKUM_ALGORITMA_DAN_STRUKTUR_DATA.git)