**LAPORAN AKHIR PRAKTIKUM**

**ALGORITMA & STRUKTUR DATA**

****

**Oleh:**

**Indra Suryadilaga NIM. 2410817310014**

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS TEKNIK**

**UNIVERSITAS LAMBUNG MANGKURAT**

**2024**

# LEMBAR PENGESAHAN

**LAPORAN AKHIR PRAKTIKUM ALGORITMA & STRUKTUR DATA**

Laporan Akhir Praktikum Algoritma & Struktur Data

Modul 1: Struct & Pointer

Modul 2 : …

Modul 3: ….dst

ini disusun sebagai syarat lulus mata kuliah Praktikum Algoritma & Struktur Data. Laporan Prakitkum ini dikerjakan oleh:

Nama Praktikan : Indra Suryadilaga

NIM : 2410817310014

|  |  |
| --- | --- |
| Menyetujui,  Asisten Praktikum  Harry Pratama Yunus  NIM. 2310817210010 | Mengetahui,  Dosen Penanggung Jawab Praktikum  Andreyan Rizky Baskara, S.Kom., M.Kom  NIP. 199307032019031011 |

# DAFTAR ISI

[LEMBAR PENGESAHAN 2](#_Toc200546425)

[DAFTAR ISI 3](#_Toc200546426)

[DAFTAR GAMBAR 4](#_Toc200546427)

[DAFTAR TABEL 5](#_Toc200546428)

[MODUL 1 : STRUCT DAN POINTER 6](#_Toc200546429)

[SOAL 1 6](#_Toc200546430)

[A. Source code 7](#_Toc200546431)

[B. Output Program 8](#_Toc200546432)

[C. Pembahasan 8](#_Toc200546433)

[LINK GITHUB 8](#_Toc200546434)

# DAFTAR GAMBAR

[Gambar 1 Modul 1 Soal 1 6](#_Toc200546443)

[Gambar 2 Modul 1 soal 1 7](#_Toc200546444)

[Gambar 3 Modul 1 output saol 1 8](#_Toc200546445)

# DAFTAR TABEL

[Tabel 1 Modul 1 Source code soal 1 7](#_Toc200546459)

# MODUL 1 : STRUCT DAN POINTER

## SOAL 1

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

A picture containing text

Description automatically generated

Gambar 1 Modul 1 Soal 1

Table

Description automatically generated

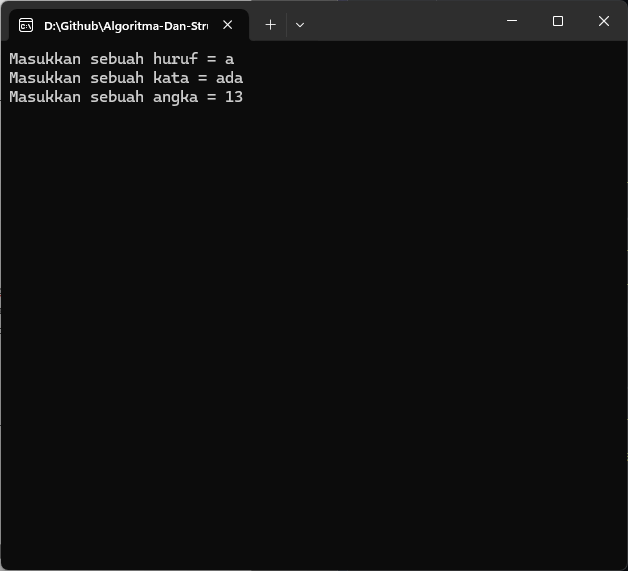
Gambar 2 Modul 1 soal 1

### A. Source code

Tabel 1 Modul 1 Source code soal 1

|  |  |
| --- | --- |
| 1.  2  3  4  5.  6.  7.  8.  9.  10.  11.  13.  14.  15.  16.  17.  18.  19.  20.  21.  22. | #include <iostream>  using *namespace* std;  *int* main() {  *char* huruf;      string kata;  *int* angka;  *// Input data*      cout << "Masukkan sebuah huruf = "; cin >> huruf;      cin.ignore(); *// Mengatasi masalah input buffer*      cout << "Masukkan sebuah kata = ";getline(cin, kata);      cout << "Masukkan sebuah angka = ";cin >> angka;  *// Output hasil*      cout << "\nHuruf yang Anda masukkan adalah " << huruf << endl;      cout << "Kata yang Anda masukkan adalah " << kata << endl;      cout << "Angka yang Anda masukkan adalah " << angka << endl;      return 0;  } |

### B. Output Program



Gambar 3 Modul 1 screenshot output saol 1

### C. Pembahasan

Program pada gambar tersebut digunakan untuk menghitung biaya SPP berdasarkan program studi D3 atau S1 dan jumlah SKS yang diambil.

# MODUL 2 : STACK DAN QUEUE

## SOAL 1

Apa perbedaan Stack dengan Queue?

### Pembahasan

Stack dan Queue merupakan dua jenis struktur data linear yang memiliki perbedaan utama dalam cara menyimpan dan mengambil data.

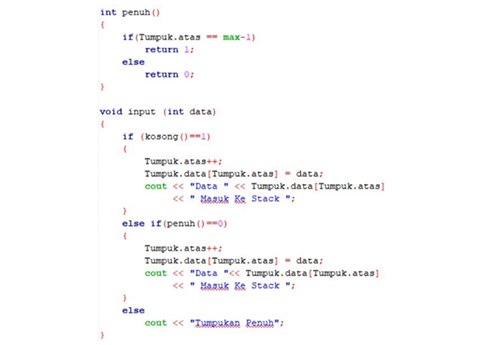
Stack bekerja dengan prinsip Last In, First Out (LIFO), artinya data yang terakhir dimasukkan akan menjadi data pertama yang dikeluarkan. Struktur ini dapat diibaratkan seperti tumpukan buku, di mana buku yang paling atas akan diambil terlebih dahulu. Operasi dasar pada Stack meliputi push (menambahkan data ke atas tumpukan) dan pop (menghapus data dari atas tumpukan).

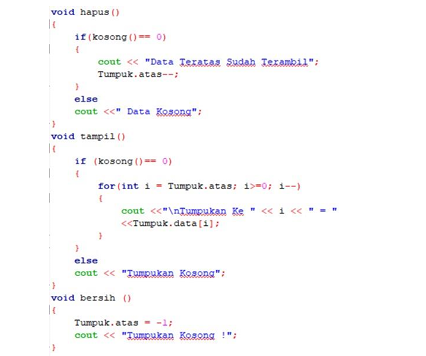
Sebaliknya, Queue menggunakan prinsip First In, First Out (FIFO), yang berarti data yang pertama kali dimasukkan akan menjadi data pertama yang dikeluarkan. Queue dapat dianalogikan seperti antrean di loket, di mana orang yang datang lebih dahulu akan dilayani lebih dahulu. Operasi dasar pada Queue terdiri dari enqueue (menambahkan data ke belakang antrean) dan dequeue (menghapus data dari depan antrean).

Kedua struktur data ini sangat penting dalam berbagai aplikasi komputer, seperti manajemen memori, pemrosesan data, dan pengendalian antrian.

## SOAL 2

Cobalah contoh program berikut, running dan analisis hasilnya!





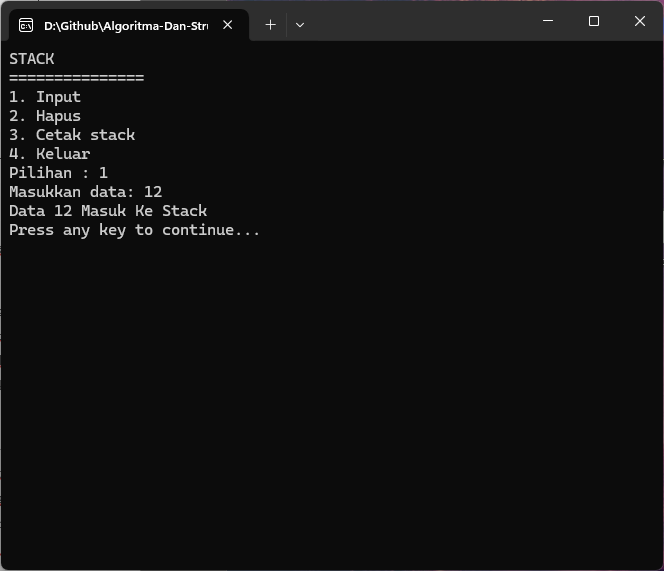
Gambar 4. Modul 2 soal 2

### Source Code

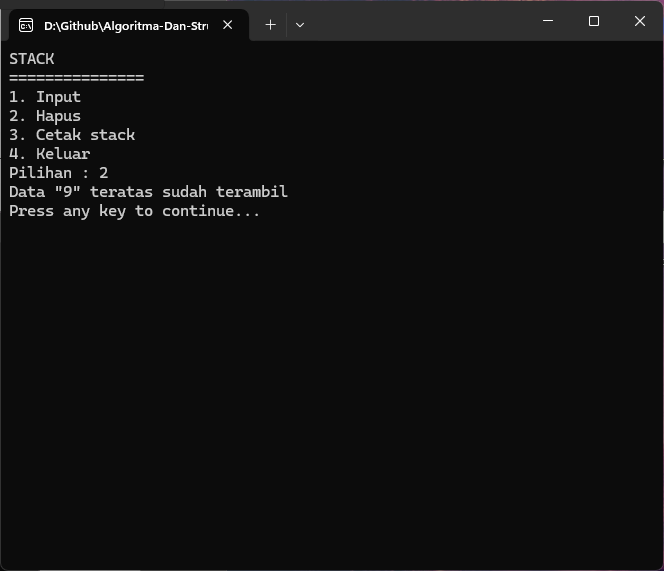
Table 1. Modul 2 Source code soal 2.

|  |  |
| --- | --- |
|  | #include <iostream> |
|  | #include <conio.h> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | const int max\_stack = 10; |
|  | string PILIHAN, data; |
|  | int PIL; |
|  |  |
|  | struct Stack { |
|  | int atas; |
|  | string data[max\_stack]; |
|  | } Tumpuk; |
|  |  |
|  | int penuh() { |
|  | if (Tumpuk.atas == max\_stack - 1) |
|  | return 1; |
|  | else |
|  | return 0; |
|  | } |
|  |  |
|  | int kosong() { |
|  | if (Tumpuk.atas == -1) |
|  | return 1; |
|  | else |
|  | return 0; |
|  | } |
|  |  |
|  | void input() |
|  | { |
|  | if (penuh() == 0){ |
|  | cout << "Masukkan data: "; |
|  | cin >> data; |
|  | Tumpuk.atas++; |
|  | Tumpuk.data[Tumpuk.atas] = data; |
|  | cout << "Data " << Tumpuk.data[Tumpuk.atas] << " Masuk Ke Stack" << endl; |
|  | }else if (penuh() == 0) { |
|  | cout << "Masukkan data: "; |
|  | cin >> data; |
|  | Tumpuk.atas++; |
|  | Tumpuk.data[Tumpuk.atas] = data; |
|  | cout << "Data " << Tumpuk.data[Tumpuk.atas] << " masuk ke stack\n"; |
|  | } |
|  | else |
|  | cout << "Tumpukan Penuh" << endl; |
|  | } |
|  |  |
|  |  |
|  | void hapus() { |
|  | if (kosong() == 0) { |
|  | cout << "Data \"" << Tumpuk.data[Tumpuk.atas] << "\" teratas sudah terambil\n"; |
|  | Tumpuk.atas--; |
|  | } |
|  | else { |
|  | cout << "Data kosong\n"; |
|  | } |
|  | } |
|  |  |
|  | void tampil() { |
|  | if (kosong() == 0) { |
|  | for (int i = Tumpuk.atas; i >= 0; i--) { |
|  | cout << "Tumpukan ke-" << i << " = \"" << Tumpuk.data[i] << "\"" << endl; |
|  | } |
|  | } |
|  | else { |
|  | cout << "Tumpukan kosong\n"; |
|  | } |
|  | } |
|  |  |
|  | void bersih() { |
|  | Tumpuk.atas = -1; |
|  | cout << "Tumpukan kosong!\n"; |
|  | } |
|  |  |
|  | int main() { |
|  | Tumpuk.atas = -1; |
|  | do { |
|  | cout << "STACK" << endl; |
|  | cout << "===============" << endl; |
|  | cout << "1. Input" << endl; |
|  | cout << "2. Hapus" << endl; |
|  | cout << "3. Cetak stack" << endl; |
|  | cout << "4. Keluar" << endl; |
|  | cout << "Pilihan : "; |
|  | cin >> PILIHAN; |
|  | PIL = stoi(PILIHAN); |
|  |  |
|  | switch (PIL) { |
|  | case 1: |
|  | input(); |
|  | break; |
|  | case 2: |
|  | hapus(); |
|  | break; |
|  | case 3: |
|  | tampil(); |
|  | break; |
|  | case 4: |
|  | cout << "TERIMA KASIH" << endl; |
|  | break; |
|  | default: |
|  | cout << "Pilihan tidak valid" << endl; |
|  | break; |
|  | } |
|  | cout << "Press any key to continue..." << endl; |
|  | getch(); |
|  | system("cls"); |
|  | } while (PIL != 4); |
|  | return 0; |
|  | } |
|  | #include <iostream> |
|  | #include <conio.h> |

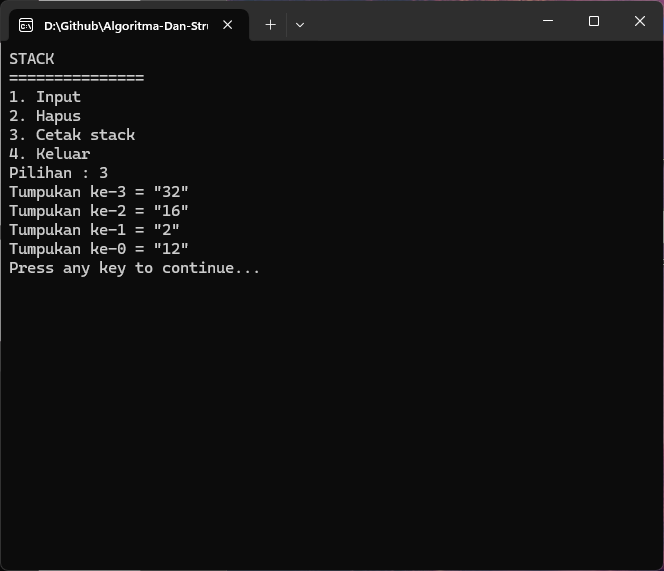
### Output Program



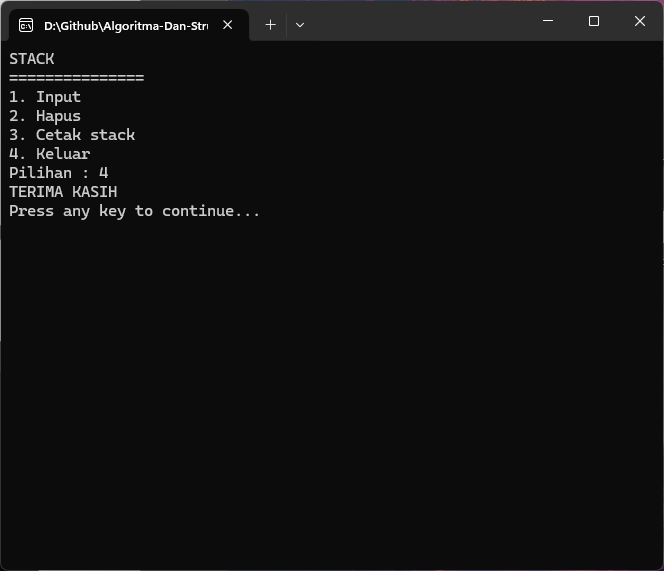
Gambar 5. Modul 2 screenshot output opsi satu saol 2



Gambar 6. Modul 2 screenshot output opsi dua saol 2



Gambar 7. Modul 2 screenshot output opsi 3 saol 2



Gambar 8. Modul 2 screenshot output opsi 4 saol 2

### Pembahasan

**Penambahan Library dan Namespace**

Bagian ini mendeklarasikan library yang dibutuhkan oleh program:

1. <iostream>: Diperlukan untuk operasi input/output standar seperti cin dan cout
2. <conio.h>: Library yang menyediakan fungsi getch() untuk menahan layar sampai pengguna menekan tombol
3. <string>: Library yang menambahkan dukungan untuk manipulasi string dalam C++.

Penggunaan using namespace std; memungkinkan program menggunakan fungsi-fungsi dari namespace standard (std) tanpa harus menuliskan prefiks "std::" setiap kali, seperti std::cout atau std::string.

**Deklarasi Variabel global**

Bagian ini mendefinisikan konstanta dan variabel global yang digunakan dalam program:

1. const int max\_stack = 10;: Mendefinisikan ukuran maksimum stack sebagai konstanta bernilai 10.
2. string PILIHAN;: Variabel untuk menyimpan input pilihan menu dari pengguna dalam bentuk string.
3. int PIL;: Variabel untuk menyimpan pilihan menu yang sudah dikonversi ke integer untuk digunakan dalam struktur kontrol program.
4. string data;: Variabel untuk menyimpan data yang akan dimasukkan ke dalam stack.

**Struktur Data Stack**

Bagian ini mendefinisikan struktur data stack dan variabel instance-nya:

1. struct Stack: Mendefinisikan tipe data struktur bernama Stack yang terdiri dari:
   1. int atas: Variabel yang berfungsi sebagai pointer/indeks untuk menunjukkan posisi elemen teratas pada stack.
   2. string data[max\_stack]: Array dengan ukuran max\_stack untuk menyimpan elemen-elemen stack dalam bentuk string.
2. Tumpuk: Variabel instance dari struktur Stack yang akan digunakan untuk operasi stack dalam program.

**Fungsi penuh()**

Fungsi penuh() bertugas untuk memeriksa apakah stack dalam keadaan penuh. Fungsi ini mengembalikan nilai:

1. 1 (true) jika stack penuh.
2. 0 (false) jika stack tidak penuh.

Logika yang digunakan adalah dengan memeriksa apakah indeks Tumpuk.atas sudah mencapai nilai maksimum stack dikurangi 1 (max - 1).

**Fungsi kosong()**

Fungsi kosong() bertugas untuk memeriksa apakah stack dalam keadaan kosong. Fungsi ini mengembalikan nilai:

1. 1 (true) jika stack kosong.
2. 0 (false) jika stack tidak kosong.

Logika yang digunakan adalah dengan memeriksa apakah indeks Tumpuk.atas sudah mencapai nilai maksimum stack dikurangi 1 (max - 1).

**Fungsi input()**

Fungsi input() digunakan untuk memasukkan data ke dalam stack. Fungsi ini memiliki beberapa kondisi:

1. Jika stack tidak penuh (penuh() == 0), maka data di input lalu nilai atas ditambah 1 dan data dimasukkan.
2. Jika stack tidak penuh (penuh() == 0), maka data juga di input lalu nilai atas ditambah 1 dan data dimasukkan.
3. Jika kondisi lainnya (yaitu stack penuh), maka ditampilkan pesan "Tumpukan Penuh".

Logika yang digunakan adalah dengan memeriksa apakah nilai dari fungsi penuh bernilai false.

**Fungsi hapus()**

Fungsi hapus() bertugas untuk menghapus elemen teratas dari stack. Fungsi ini:

1. Jika tidak kosong, maka indeks pointer Tumpuk.atas dikurangi 1, yang secara efektif menghapus elemen teratas.
2. Jika kosong, maka ditampilkan pesan "Data Kosong".

Fungsi ini tidak benar-benar menghapus data secara fisik dari array, melainkan hanya mengubah indeks teratas yang dapat diakses.

**Fungsi tampil ()**

Fungsi tampil() bertugas untuk menampilkan seluruh isi stack dari atas ke bawah. Fungsi ini:

1. Jika tidak kosong, menampilkan seluruh elemen stack mulai dari indeks teratas hingga indeks terbawah (0) .
2. Jika kosong, menampilkan pesan "Tumpukan Kosong".

Implementasi ini sesuai dengan prinsip LIFO (Last In First Out) dari stack, di mana elemen teratas ditampilkan terlebih dahulu.

**Fungsi bersih()**

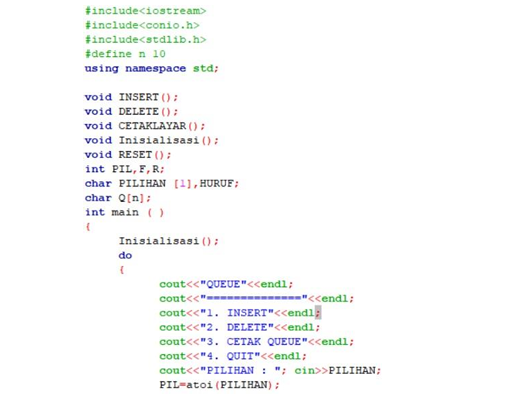
Fungsi bersih() bertugas untuk mengosongkan stack. Fungsi ini:

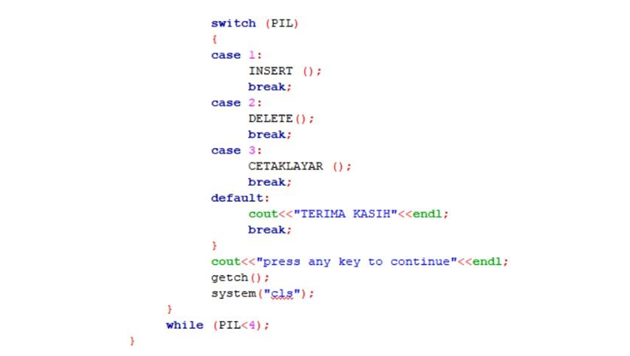
1. Menetapkan nilai pointer Tumpuk.atas menjadi -1, yang mengindikasikan stack kosong
2. Menampilkan pesan konfirmasi "Tumpukan Kosong!"

Sama seperti fungsi hapus, fungsi ini tidak benar-benar menghapus data secara fisik dari array, melainkan hanya mengatur ulang nilai atas menjadi -1 sehingga stack dianggap kosong dan siap diisi kembali.

## SOAL 3

Cobalah contoh program berikut, running dan analisis hasilnya!





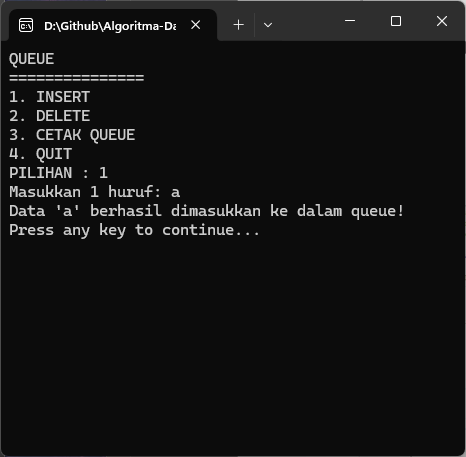
Gambar 9. Modul 2 soal 3

### Source Code

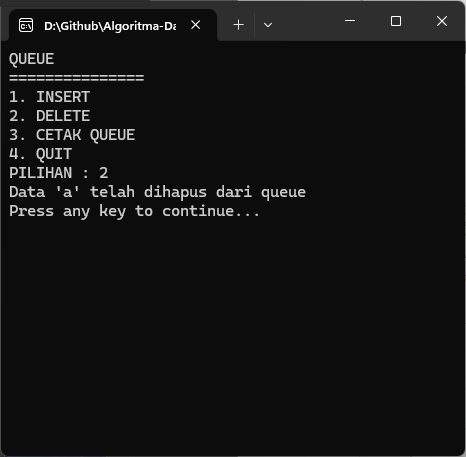
Table 2. Modul 2 Source code soal 3

|  |  |
| --- | --- |
|  | #include <iostream> |
|  | #include <conio.h> |
|  | #include <stdlib.h> |
|  | #define n 10 |
|  | using namespace std; |
|  |  |
|  | int PIL, F, R; |
|  | char PILIHAN[1], HURUF; |
|  | char Q[n]; |
|  |  |
|  | void Inisialisasi() { |
|  | F = 0; |
|  | R = -1; |
|  | } |
|  |  |
|  | bool isFull() { |
|  | return R == n - 1; |
|  | } |
|  |  |
|  | bool isEmpty() { |
|  | return F > R; |
|  | } |
|  |  |
|  | void INSERT() { |
|  | if (isFull()) { |
|  | cout << "Queue Penuh!" << endl; |
|  | } else { |
|  | cout << "Masukkan 1 huruf: "; |
|  | cin >> HURUF; |
|  | R++; |
|  | Q[R] = HURUF; |
|  | cout << "Data '" << HURUF << "' berhasil dimasukkan ke dalam queue!" << endl; |
|  | } |
|  | } |
|  |  |
|  | void DELETE() { |
|  | if (isEmpty()) { |
|  | cout << "Queue Kosong!" << endl; |
|  | } else { |
|  | cout << "Data '" << Q[F] << "' telah dihapus dari queue" << endl; |
|  | F++; |
|  | if (isEmpty()) { |
|  | Inisialisasi(); |
|  | cout << "Queue telah direset" << endl; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void CETAKLAYAR() { |
|  | if (isEmpty()) { |
|  | cout << "Queue Kosong!" << endl; |
|  | } else { |
|  | cout << "Isi Queue: "; |
|  | for (int i = F; i <= R; i++) { |
|  | cout << Q[i] << " "; |
|  | } |
|  | cout << endl; |
|  | cout << "Front: " << F << ", Rear: " << F << endl; |
|  | } |
|  | } |
|  |  |
|  | 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); |
|  | return 0; |

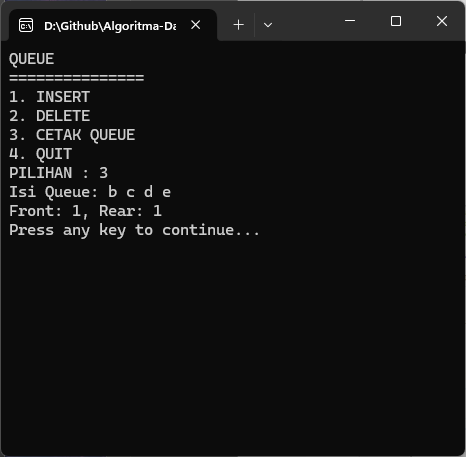
### Output Program



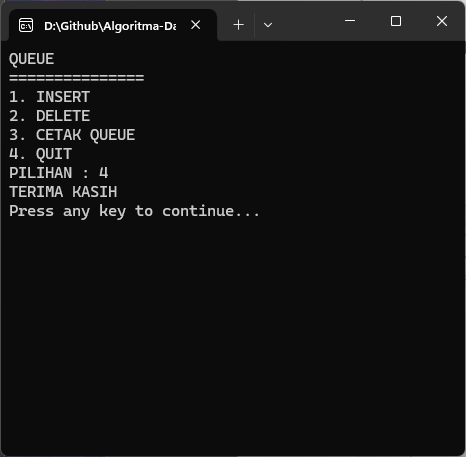
Gambar 10. Modul 2 screenshot output opsi satu saol 2



Gambar 11. Modul 2 screenshot output opsi dua saol 2



Gambar 12. Modul 2 screenshot output opsi 3 saol 2



Gambar 13. Modul 2 screenshot output opsi 4 saol 2

### Pembahasan

**Penambahan Library dan Namespace**

Bagian ini mendeklarasikan library yang dibutuhkan oleh program:

1. <iostream>: Untuk operasi input/output standar seperti cin dan cout
2. <conio.h>: Menyediakan fungsi getch() untuk menahan tampilan layar
3. <stdlib.h>: Menyediakan fungsi system() untuk membersihkan layar
4. #define n 10: Mendefinisikan konstanta n dengan nilai 10 yang digunakan sebagai ukuran maksimum queue
5. using namespace std: Memungkinkan penggunaan fungsi dari namespace standard tanpa prefiks std::

**Deklarasi Variabel global**

Bagian ini mendefinisikan konstanta dan variabel global yang digunakan dalam program:

1. PIL: Variabel integer untuk menyimpan pilihan menu setelah konversi
2. F (Front): Indeks untuk menunjukkan elemen terdepan dalam queue
3. R (Rear): Indeks untuk menunjukkan elemen terakhir dalam queue
4. PILIHAN[1]: Array karakter untuk menyimpan input pilihan menu (terbatas hanya 1 karakter)
5. HURUF: Variabel karakter untuk menyimpan data yang akan dimasukkan ke queue
6. Q[n]: Array dengan ukuran n untuk menyimpan elemen-elemen queue.

**Fungsi Inisialisasi**

Fungsi ini menginisialisasi queue dengan menetapkan nilai awal untuk indeks front (F) dan rear (R):

1. Front (F) diatur ke 0, menunjukkan posisi awal untuk pengambilan data
2. Rear (R) diatur ke -1, menandakan queue kosong (belum ada elemen yang dimasukkan).

inisialisasi standar untuk queue linear, di mana indeks rear awalnya diatur lebih rendah dari front untuk menandakan kekosongan.

**isFull() dan isEmpty()**

Dua fungsi boolean untuk memeriksa status queue:

1. isFull(): Mengembalikan true jika rear (R) telah mencapai indeks terakhir array (n-1)
2. isEmpty(): Mengembalikan true jika front (F) lebih besar dari rear (R).

**Fungsi INSERT()**

Fungsi INSERT() digunakan untuk memasukkan data ke dalam queue. Fungsi ini memiliki beberapa kondisi:

1. Jika stack penuh (isFull() == 1), maka ditampilkan pesan "Queue Penuh!”.
2. Jika kondisi lainnya (yaitu stack tidak penuh), "maka HURUF di input lalu nilai rear(R) ditambah 1 dan HURUF dimasukkan ke posisi rear(R), pesan konfirmasi akan ditampilkan dengan menunjukkan data yang dimasukkan.

**Fungsi DELETE ()**

Fungsi DELETE() bertugas untuk menghapus elemen teratas dari queue. Fungsi ini:

1. Jika tidak kosong, maka indeks pointer Tumpuk.atas dikurangi 1, yang secara efektif menghapus elemen teratas.
2. Jika kosong, maka ditampilkan pesan "Data Kosong".

Fungsi ini tidak benar-benar menghapus data secara fisik dari array, melainkan hanya mengubah indeks teratas yang dapat diakses.

**Fungsi CETAKLAYAR()**

Fungsi CETAKLAYAR() bertugas untuk menampilkan seluruh isi queue. Fungsi ini:

1. Jika kosong, menampilkan pesan "Tumpukan Kosong".
2. Jika tidak kosong, menampilkan seluruh elemen queue mulai dari front hingga rear.

Penambahan informasi tentang nilai indeks front dan rear juga ditambahkan, sangat berguna untuk memahami status internal queue.

**Fungsi main()**

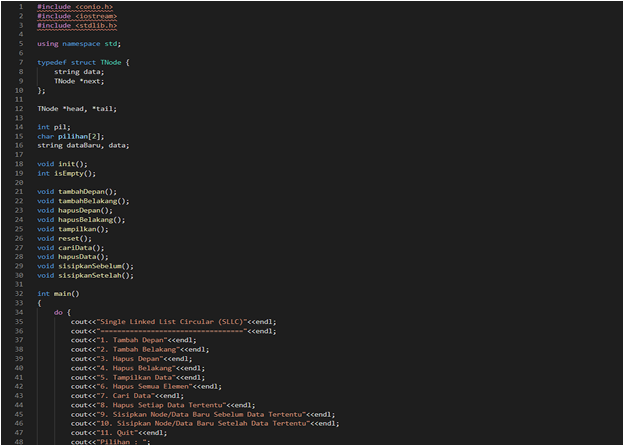
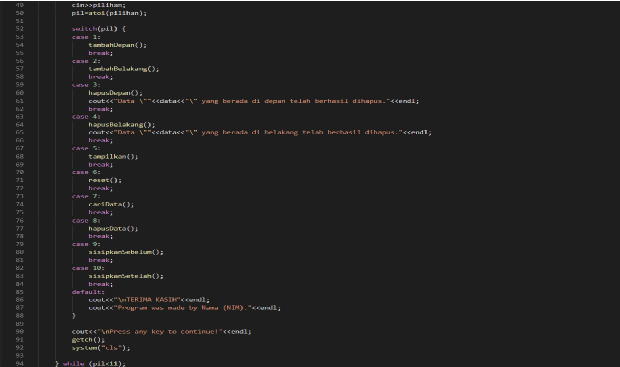
Fungsi utama yang mengontrol alur program:

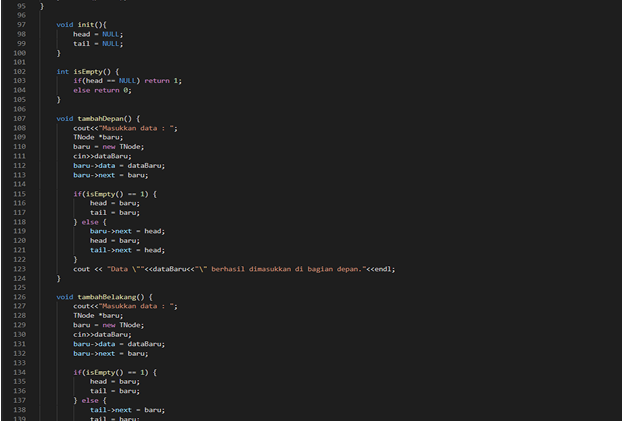
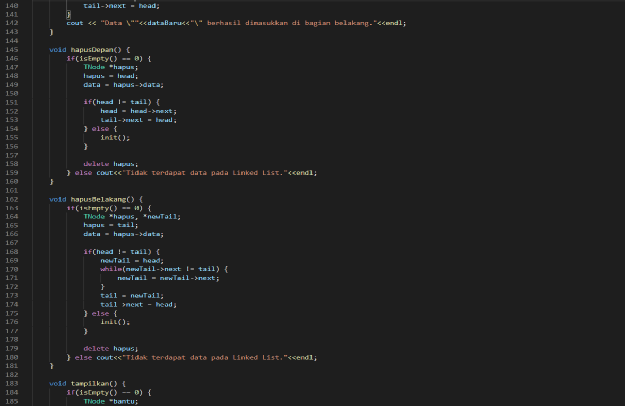
1. Memanggil Inisialisasi() untuk menyiapkan queue
2. Menampilkan menu interaktif dengan 4 pilihan
3. Membaca input pilihan pengguna dan mengkonversinya ke integer dengan atoi()
4. Menjalankan fungsi yang sesuai berdasarkan pilihan menggunakan struktur switch
5. Menunggu pengguna menekan tombol untuk melanjutkan
6. Membersihkan layar sebelum menampilkan menu kembali
7. Melanjutkan loop selama pilihan pengguna kurang dari 4

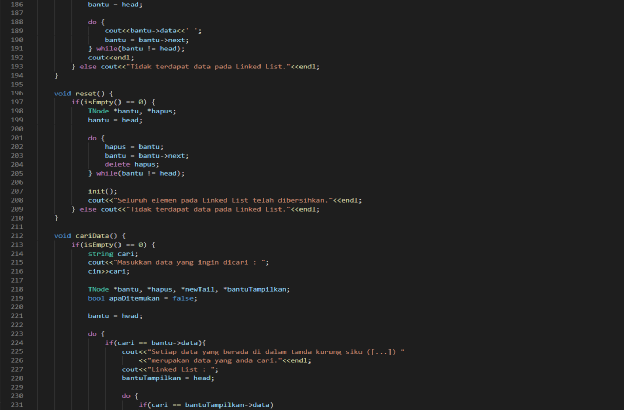
# MODUL 3 : Single Link List

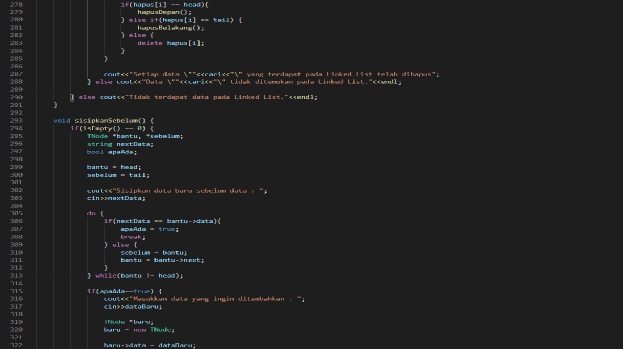
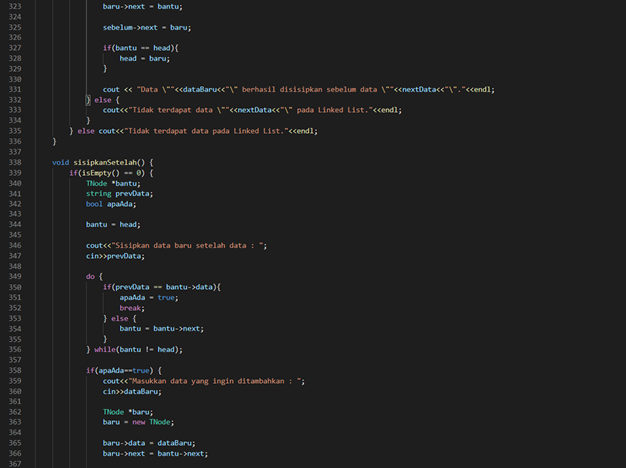
## SOAL 1

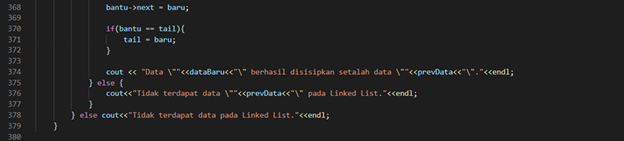
Cobalah program berikut, running, simpan program, dan screenshoot hasil running !











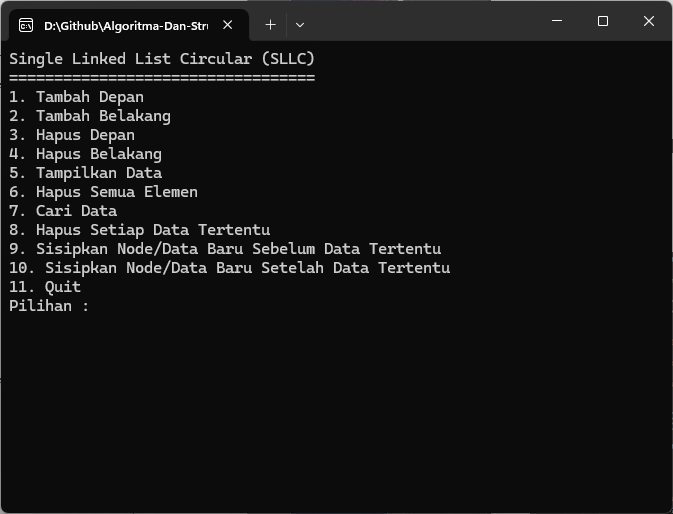
Gambar 14. Modul 3 soal 1

### Source Code

Table 3. Modul 3 Source code soal 1

|  |  |
| --- | --- |
|  | #include <conio.h> |
|  | #include <iostream> |
|  | #include <stdlib.h> |
|  |  |
|  | using namespace std; |
|  |  |
|  | typedef struct TNode { |
|  | string data; |
|  | TNode \*next; |
|  | }; |
|  |  |
|  | TNode \*head, \*tail; |
|  |  |
|  | int pil; |
|  | char pilihan[2]; |
|  | string dataBaru, data; |
|  |  |
|  | void init(); |
|  | int isEmpty(); |
|  |  |
|  | void tambahDepan(); |
|  | void tambahBelakang(); |
|  | void hapusDepan(); |
|  | void hapusBelakang(); |
|  | void tampilkan(); |
|  | void reset(); |
|  | void cariData(); |
|  | void hapusData(); |
|  | void sisipkanSebelum(); |
|  | void sisipkanSetelah(); |
|  |  |
|  | int main() |
|  | { |
|  | do { |
|  | cout<<"Single Linked List Circular (SLLC)"<<endl; |
|  | cout<<"=================================="<<endl; |
|  | cout<<"1. Tambah Depan"<<endl; |
|  | cout<<"2. Tambah Belakang"<<endl; |
|  | cout<<"3. Hapus Depan"<<endl; |
|  | cout<<"4. Hapus Belakang"<<endl; |
|  | cout<<"5. Tampilkan Data"<<endl; |
|  | cout<<"6. Hapus Semua Elemen"<<endl; |
|  | cout<<"7. Cari Data"<<endl; |
|  | cout<<"8. Hapus Setiap Data Tertentu"<<endl; |
|  | cout<<"9. Sisipkan Node/Data Baru Sebelum Data Tertentu"<<endl; |
|  | cout<<"10. Sisipkan Node/Data Baru Setelah Data Tertentu"<<endl; |
|  | cout<<"11. Quit"<<endl; |
|  | cout<<"Pilihan :"<<endl; |
|  | cin>>pilihan; |
|  | pil=atoi(pilihan); |
|  |  |
|  | switch(pil) { |
|  | case 1: |
|  | tambahDepan(); |
|  | break; |
|  | case 2: |
|  | tambahBelakang(); |
|  | break; |
|  | case 3: |
|  | hapusDepan(); |
|  | cout<<"data \""<<data<<"\" yang berada di depan telah berhasil dihapus."<<endl; |
|  | break; |
|  | case 4: |
|  | hapusBelakang(); |
|  | cout<<"data \""<<data<<"\" yang berada di belakang telah berhasil dihapus."<<endl; |
|  | break; |
|  | case 5: |
|  | tampilkan(); |
|  | break; |
|  | case 6: |
|  | reset(); |
|  | break; |
|  | case 7: |
|  | cariData(); |
|  | break; |
|  | case 8: |
|  | hapusData(); |
|  | break; |
|  | case 9: |
|  | sisipkanSebelum(); |
|  | break; |
|  | case 10: |
|  | sisipkanSetelah(); |
|  | break; |
|  | default: |
|  | cout<<"\nTERIMA KASIH"<<endl; |
|  | cout<<"Program was made by Indra Suryadilga (2410817310014)."<<endl; |
|  | } |
|  |  |
|  | cout<<"\nPress any key to continiue!"<<endl; |
|  | getch(); |
|  | system("cls"); |
|  |  |
|  | } while (pil<11); |
|  | } |
|  |  |
|  | void init() { |
|  | head = NULL; |
|  | tail = NULL; |
|  | } |
|  |  |
|  | int isEmpty() { |
|  | if(head==NULL) return 1; |
|  | else return 0; |
|  | } |
|  |  |
|  | void tambahDepan() { |
|  | cout<<"Masukkan data : "; |
|  | TNode \*baru; |
|  | baru = new TNode; |
|  | cin>>dataBaru; |
|  | baru->data = dataBaru; |
|  | baru->next = baru; |
|  |  |
|  | if(isEmpty()==1) { |
|  | head = baru; |
|  | tail = baru; |
|  | } else { |
|  | baru->next = head; |
|  | head = baru; |
|  | tail->next = head; |
|  | } |
|  | cout<<"Data \""<<dataBaru<<"\" berhasil dimasukan di bagian depan."<<endl; |
|  | } |
|  |  |
|  | void tambahBelakang() { |
|  | cout<<"Masukkan data : "; |
|  | TNode \*baru; |
|  | baru = new TNode; |
|  | cin>>dataBaru; |
|  | baru->data = dataBaru; |
|  | baru->next = baru; |
|  |  |
|  | if(isEmpty()==1) { |
|  | head = baru; |
|  | tail = baru; |
|  | } else { |
|  | tail->next = baru; |
|  | tail = baru; |
|  | tail->next = head; |
|  | } |
|  | cout<<"Data \""<<dataBaru<<"\" berhasil dimasukan di bagian belaknag."<<endl; |
|  | } |
|  |  |
|  | void hapusDepan() { |
|  | if(isEmpty()==0) { |
|  | TNode \*hapus; |
|  | hapus = head; |
|  | data = hapus->data; |
|  |  |
|  | if(head != tail) { |
|  | head = head->next; |
|  | tail->next = head; |
|  | } else { |
|  | init(); |
|  | } |
|  |  |
|  | delete hapus; |
|  | } else cout<<"Tidak terdapat data pada Linked List."<<endl; |
|  | } |
|  |  |
|  | void hapusBelakang() { |
|  | if(isEmpty()==0) { |
|  | TNode \* hapus, \*newTail; |
|  | hapus = tail; |
|  | data = hapus->data; |
|  |  |
|  | if(head != tail) { |
|  | newTail = head; |
|  | while(newTail->next != tail) { |
|  | newTail = newTail->next; |
|  | } |
|  | tail = newTail; |
|  | tail->next = head; |
|  | } else { |
|  | init(); |
|  | } |
|  |  |
|  | delete hapus; |
|  | } else cout<<"Tidak terdapat data pada Linked List."<<endl; |
|  | } |
|  |  |
|  | void tampilkan() { |
|  | if(isEmpty()==0) { |
|  | TNode \* bantu; |
|  | bantu = head; |
|  |  |
|  | do { |
|  | cout<<bantu->data<<' '; |
|  | bantu = bantu->next; |
|  | } while(bantu != head); |
|  | cout<<endl; |
|  | } else cout<<"Tidak terdapat data pada Linked List."<<endl; |
|  | } |
|  |  |
|  | void reset() { |
|  | if(isEmpty() == 0) { |
|  | TNode \*bantu, \*hapus; |
|  | bantu = head; |
|  |  |
|  | do { |
|  | hapus = bantu; |
|  | bantu = bantu->next; |
|  | delete hapus; |
|  | } while(bantu != head); |
|  |  |
|  | init(); |
|  | cout<<"Seluruh elemen pada Linked List telah dibersihkan."<<endl; |
|  | }else cout<<"Tidak terdapat data pada Linked List."<<endl; |
|  | } |
|  |  |
|  | void cariData() { |
|  | if(isEmpty() == 0) { |
|  | string cari; |
|  | cout<<"Masukan data yang ingin dicari : "; |
|  | cin>>cari; |
|  |  |
|  | TNode \*bantu, \*hapus, \*newTail, \*bantuTampilkan; |
|  | bool apaDitemukan = false; |
|  |  |
|  | bantu = head; |
|  |  |
|  | do { |
|  | if(cari == bantu->data){ |
|  | cout<<"Setiap data yang berada di dalam tanda kurung siku([..]) " |
|  | <<"merupakan data yang dicari."<<endl; |
|  | cout<<"Linked List : "; |
|  | bantuTampilkan = head; |
|  |  |
|  | do { |
|  | if(cari == bantuTampilkan->data) |
|  | cout<<"["<<bantuTampilkan->data<<"] "; |
|  | else |
|  | cout<<bantuTampilkan->data<<' '; |
|  | bantuTampilkan = bantuTampilkan->next; |
|  | } while(bantuTampilkan != head); |
|  |  |
|  | apaDitemukan = true; |
|  | cout<<endl; |
|  | break; |
|  | } |
|  | bantu = bantu->next; |
|  | } while(bantu != head); |
|  |  |
|  | if(apaDitemukan ==false) |
|  | cout<<"Data \""<<cari<<"\" tidak ditemukan pada Linked List."<<endl; |
|  | } else cout<<"Tidak terdapat data pada Linked List."<<endl; |
|  | } |
|  |  |
|  | void hapusData() { |
|  | if(isEmpty() == 0) { |
|  | string cari; |
|  | cout<<"Masukkan data yang ingin dihapus : "; |
|  | cin>>cari; |
|  |  |
|  | TNode \*bantu, \*sebelum, \*hapus[255], \*bantu2; |
|  | int hitung = 0; |
|  | bool apaDitemukan = false; |
|  |  |
|  | bantu = head; |
|  |  |
|  | do { |
|  | bantu2 = bantu; |
|  | if(cari == bantu->data){ |
|  | hapus[hitung++] = bantu; |
|  | apaDitemukan = true; |
|  | if(bantu != head && bantu != tail) { |
|  | sebelum->next = bantu->next; |
|  | bantu2 = sebelum; |
|  | } |
|  | } |
|  | sebelum = bantu2; |
|  | bantu = bantu->next; |
|  | } while(bantu != head); |
|  |  |
|  | if(apaDitemukan == true) { |
|  | for(int i = 0; i < hitung; i++) { |
|  | if(hapus[i] == head) { |
|  | hapusDepan(); |
|  | } else if(hapus[i] == tail) { |
|  | hapusBelakang(); |
|  | } else { |
|  | delete hapus[i]; |
|  | } |
|  | } |
|  |  |
|  | cout<<"Setipa data \""<<cari<<"\" yang terdapat pada Linked List telah dihapus"; |
|  | } else cout<<"Data \""<<cari<<"\" tidak ditemukan pada Linked List."<<endl; |
|  |  |
|  | } else cout<<"Tidak terdapat data pada Linked List."<<endl; |
|  | } |
|  |  |
|  | void sisipkanSebelum() { |
|  | if(isEmpty() == 0) { |
|  | TNode \*bantu, \*sebelum; |
|  | string nextData; |
|  | bool apaAda; |
|  |  |
|  | bantu = head; |
|  | sebelum = tail; |
|  |  |
|  | cout<<"Sisipkan data baru sebelum data : "; |
|  | cin>>nextData; |
|  |  |
|  | do { |
|  | if(nextData == bantu->data) { |
|  | apaAda = true; |
|  | break; |
|  | } else { |
|  | sebelum = bantu; |
|  | bantu = bantu->next; |
|  | } |
|  | } while(bantu != head); |
|  |  |
|  | if(apaAda == true) { |
|  | cout<<"Masukan data yang diinginkan : "; |
|  | cin>>dataBaru; |
|  |  |
|  | TNode \*baru; |
|  | baru = new TNode; |
|  |  |
|  | baru->data = dataBaru; |
|  | baru->next = bantu; |
|  |  |
|  | sebelum->next = baru; |
|  |  |
|  | if(bantu == head) { |
|  | head = baru; |
|  | } |
|  |  |
|  | cout<<"Data \""<<dataBaru<<"\" berhasil disisipkan sebelum data \""<<nextData<<"\"."<<endl; |
|  | } else { |
|  | cout<<"Tidak terdapat data \""<<nextData<<"\" pada Linked List."<<endl; |
|  | } |
|  | } else cout<<"Tidak terdapat data pada Linked List."<<endl; |
|  | } |
|  |  |
|  | void sisipkanSetelah() { |
|  | if(isEmpty() == 0) { |
|  | TNode \*bantu; |
|  | string prevData; |
|  | bool apaAda; |
|  |  |
|  | bantu = head; |
|  |  |
|  | cout<<"Sisipkan data baru setelah data : "; |
|  | cin>>prevData; |
|  |  |
|  | do { |
|  | if(prevData == bantu->data) { |
|  | apaAda = true; |
|  | break; |
|  | } else { |
|  | bantu = bantu->next; |
|  | } |
|  | } while(bantu != head); |
|  |  |
|  | if(apaAda == true) { |
|  | cout<<"Masukan data yang ditambahkan : "; |
|  | cin>>dataBaru; |
|  |  |
|  | TNode \*baru; |
|  | baru = new TNode; |
|  |  |
|  | baru->data = dataBaru; |
|  | baru->next = bantu->next; |
|  |  |
|  | bantu->next = baru; |
|  |  |
|  | if(bantu == tail) { |
|  | tail = baru; |
|  | } |
|  |  |
|  | cout<<"Data \""<<dataBaru<<"\" berhasil disisipkan setelah data \""<<prevData<<"\"."<<endl; |
|  | } else { |
|  | cout<<"Data \""<<prevData<<"\" tidak ditemukan pada Linked List."<<endl; |
|  | } |
|  | } else cout<<"Tidak terdapat data pada Linked List."<<endl; |

### Output Program



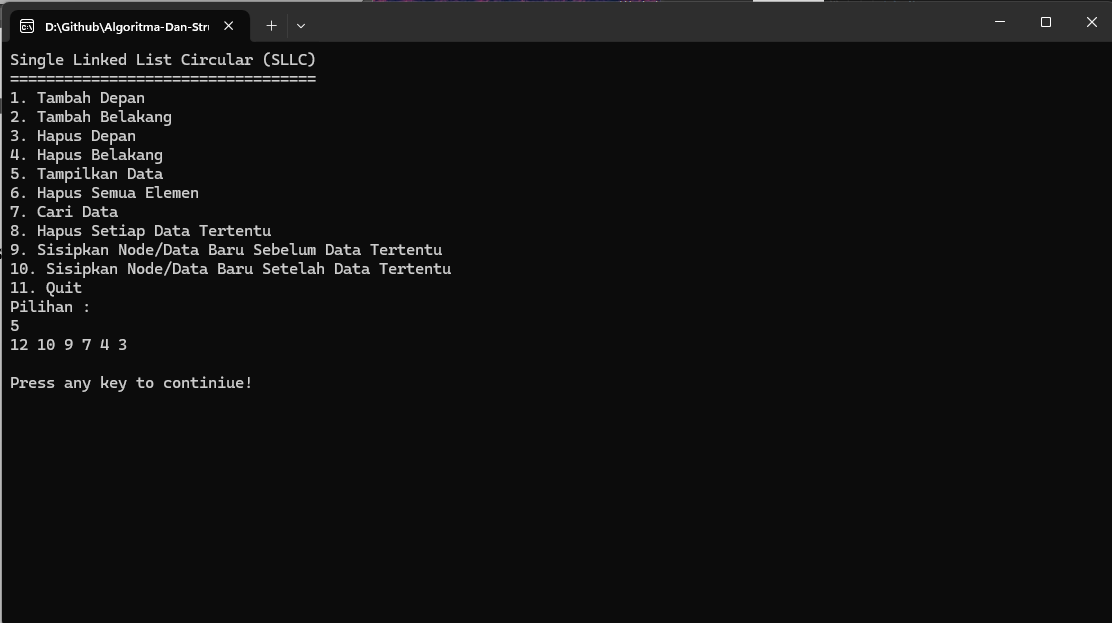
Gambar 15. Modul 3 screenshot output saol 1

### Pembahasan

## SOAL 2

Lakukan tambah data depan 3, 4, 7, 9, 10, 12 dan kemudian lakukan tampilkan data lalu screenshoot hasilnya !

### Output Program



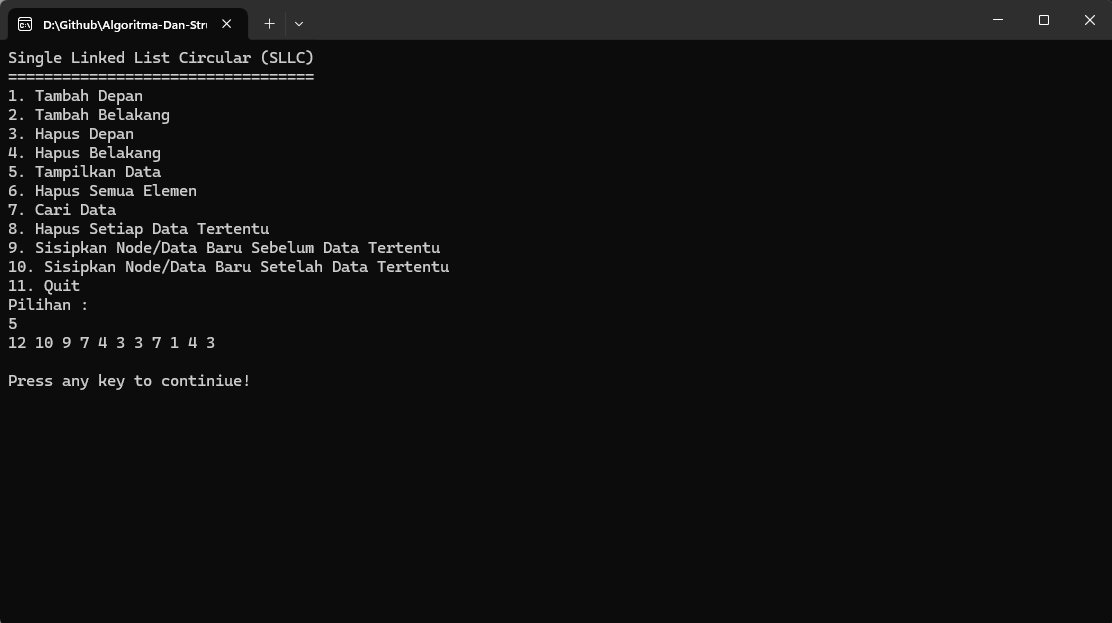
Gambar 16. Modul 3 screenshot output saol 2

### Pembahansan

## SOAL 3

Lakukan tambah data belakang 3, 7, 1, 4, 3 dan kemudian lakukan tampilkan data lalu screenshoot hasilnya !

### Output Program



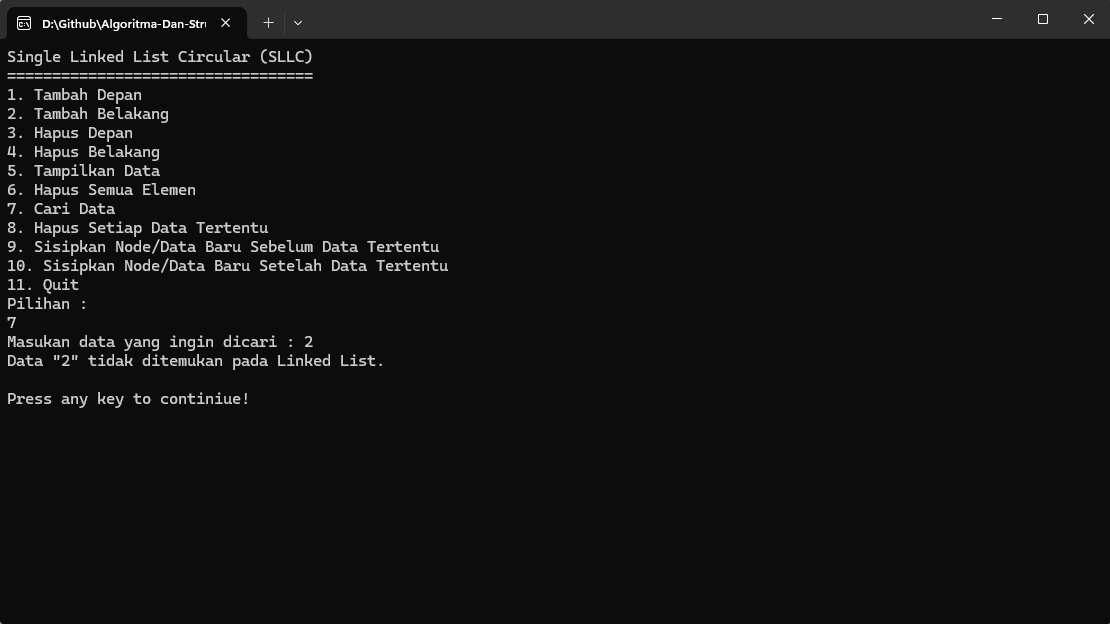
Gambar 17. Modul 3 screenshot output saol 3

### Pembahansan

## SOAL 4

Apa yang terjadi jika mencari angka 2 pada Single Linked List Circular (SLLC) pada data yang telah ditambahkan/dimasukkan sebelumnya dan screenshoot hasilnya !

### Output Program



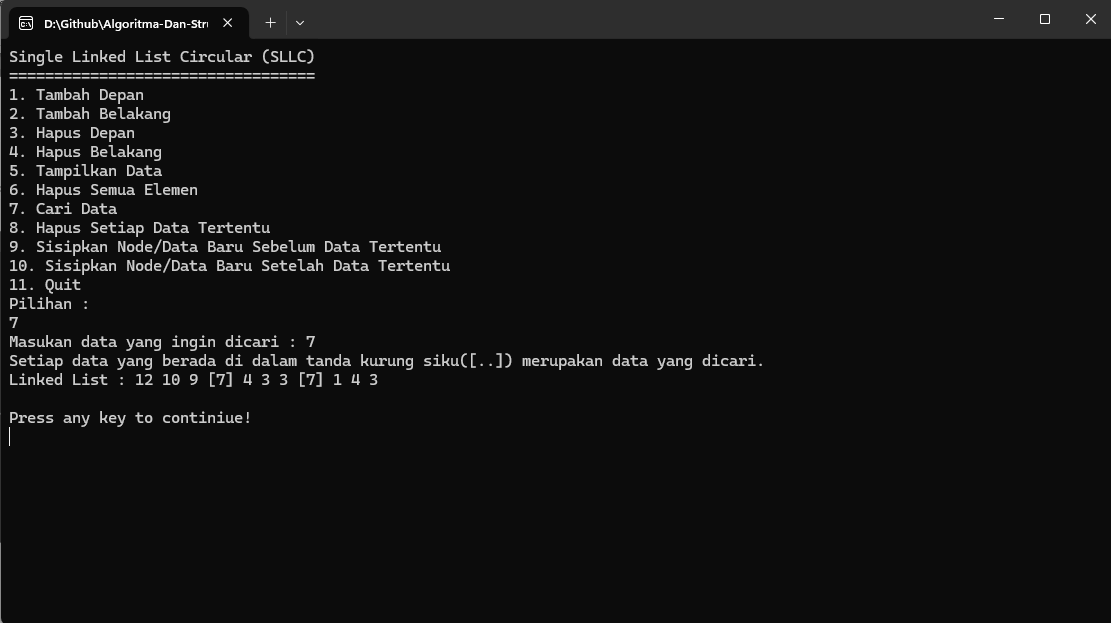
Gambar 18. Modul 3 screenshot output saol 4

### Pembahansan

## SOAL 5

Coba cari angka 7 dan screenshoot hasilnya !

### Output Program



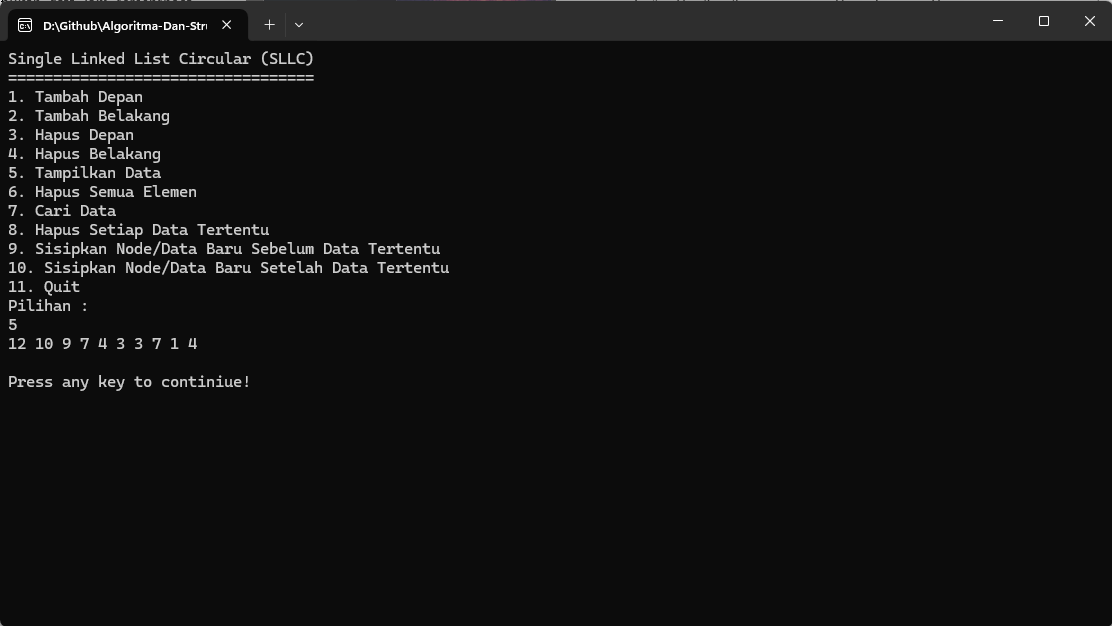
Gambar 19. Modul 3 screenshot output saol 5

### Pembahansan

## SOAL 6

Lakukan hapus belakang dan kemudian lakukan tampilkan data lalu screenshoot hasilnya !

### Output Program



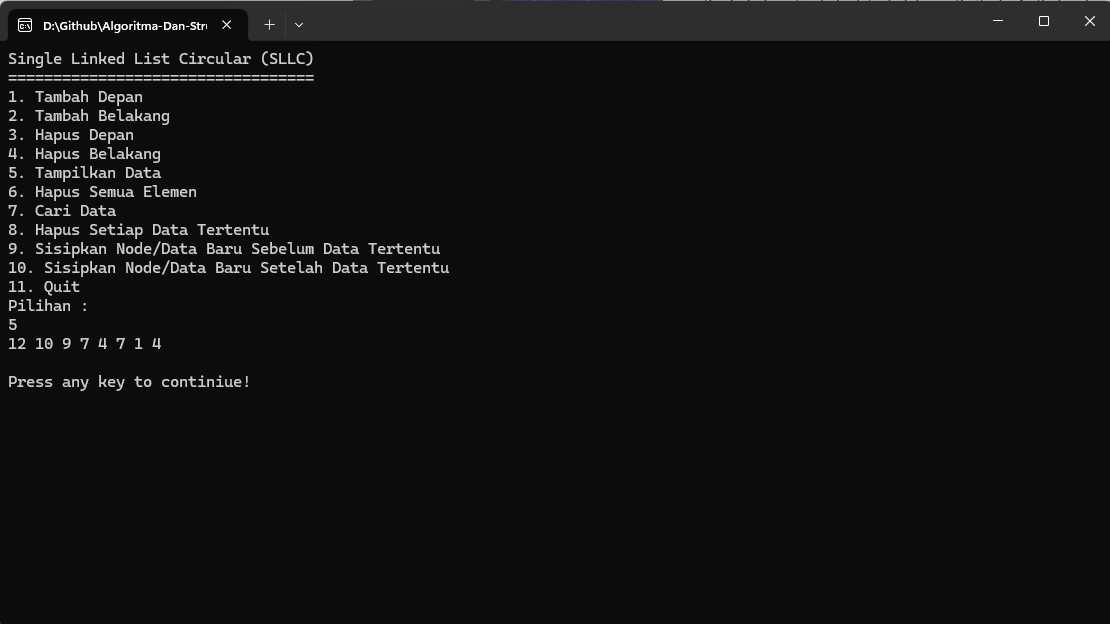
Gambar 19. Modul 3 screenshot output saol 6

### Pembahansan

## SOAL 7

Lakukan hapus setiap angka 3 dan kemudian lakukan tampilkan data lalu screenshoot hasilnya !

### Output Program



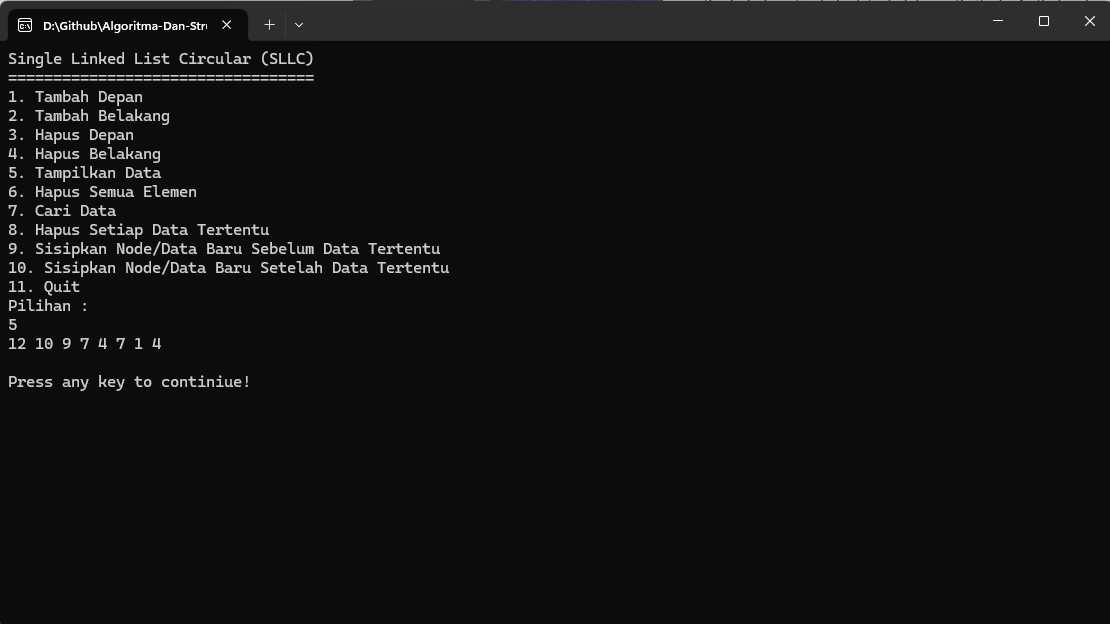
Gambar 21. Modul 3 screenshot output saol 7

### Pembahansan

## SOAL 8

Tampilkan data lalu jelaskan yang mana head dan yang mana tail.

### Output Program



Gambar 22. Modul 3 screenshot output saol 8

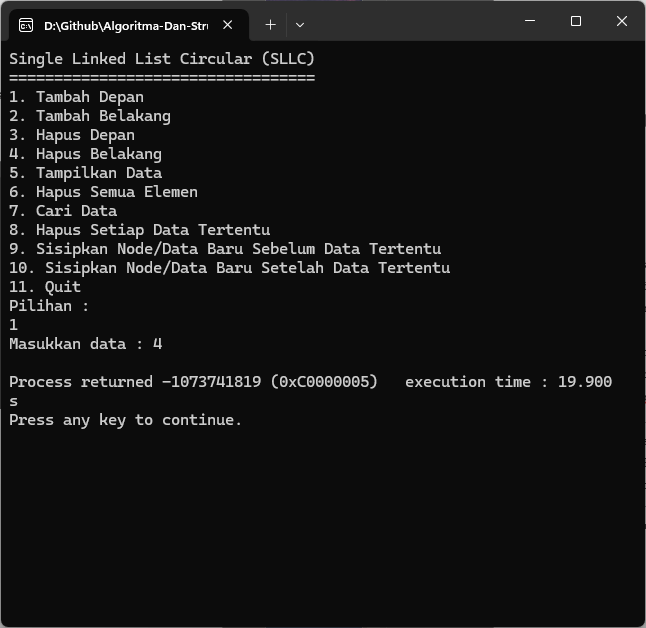
### Pembahansan

Berdasarkan dari gambar 8 urutan data yang ditampilkan secara berurutan adalah “12, 10, 9, 7, 4, 7, 1, 4”. Maka data yang menjadi head adalah data yang berada paling kiri yaitu 12, karena head adalah node yang menunjuk ke data paling pertama yang diakses dalam list. Sedangkan data yang menjadi tail adalah data yang berada paling kanan yaitu 4, karena tail adalah node terakhir dalam list, yang pointer next-nya menunjuk kembali ke head.

## SOAL 9

Jika baris ke 103 dan 104 dihapus maka apa yang akan terjadi pada saat memasukkan data, dan jelaskan mengapa?

### Output Program



Gambar 23. Modul 3 screenshot output saol 9

### Pembahansan

Di awal kita masih bisa menjalankan programnya, tetapi kita mencoba untuk menambahkan data menggunakan opsi menu satu dan dua maka program mungkin akan berhenti seketika. Ini disebabkan baris yang dihapus merupakan isi dari fungsi isEmpty(), yang mana fungsi ini pada normalnya akan mengembalikan value berupa 1 atau 0. Namun ketika pada kondisi di mana isi dari fungsi dari isEmpty() dihapus, nilai yang akan dikembalikan adalah garbage value atau nilai tidak pasti bisa saja 1 ataupun -11. Padahal nilai dari fungsi isEmpty() digunakan untuk logika pada fungsi penambahan data baru maupun fungsi lainya. Program bisa salah mengira bahwa list sudah berisi, dan memproses else membuat struktur pointer jadi salah dan berpotensi crash.

## SOAL 10

Jelaskan apa itu variabel head dan tail pada sllc!

### Pembahansan

Variabel head adalah pointer yang menunjuk ke simpul (node) pertama dalam struktur Single Linked List Circular (SLLC. Ketika pengguna ingin menampilkan seluruh isi list, maka penelusuran dimulai dari node yang ditunjuk oleh head. Karena sifat circular dari list ini, proses penelusuran akan berakhir kembali ke node head setelah semua node dilalui. Oleh karena itu, head memegang peran krusial sebagai acuan awal dan penanda batas dalam proses traversal list.

Variabel tail adalah pointer yang menunjuk ke simpul terakhir dalam SLLC. Tidak seperti linked list biasa yang node terakhirnya menunjuk ke NULL, pada SLLC node terakhir justru menunjuk kembali ke node pertama (yaitu head). Dengan demikian, tail->next akan selalu menunjuk ke head, yang menjadikan struktur data ini berbentuk melingkar atau circular. tail juga sering digunakan saat menambahkan node di bagian belakang list, karena node baru akan ditempatkan setelah node yang ditunjuk oleh tail, lalu tail diperbarui agar menunjuk ke node yang baru ditambahkan.

# LINK GITHUB

Berikut adalah link menuju repository github :