LEMBAR ASISTENSI

PRAKTIKUM STRUKTUR DATA

LABORATORIUM TEKNIK KOMPUTER

JURUSAN TEKNIK ELEKTRO FAKULTAS TEKNIK

UNIVERSITAS LAMPUNG

Judul Praktikum : STACK

Praktikan : Alvin Reihansyah Makarim (2115061083)

Asisten : Charles Gunawan (2015061044)

Michel (2015061018)

No.	Catatan	Tanggal	Paraf

Bandar Lampung, 2022

.....

NPM.

I. JUDUL PERCOBAAN

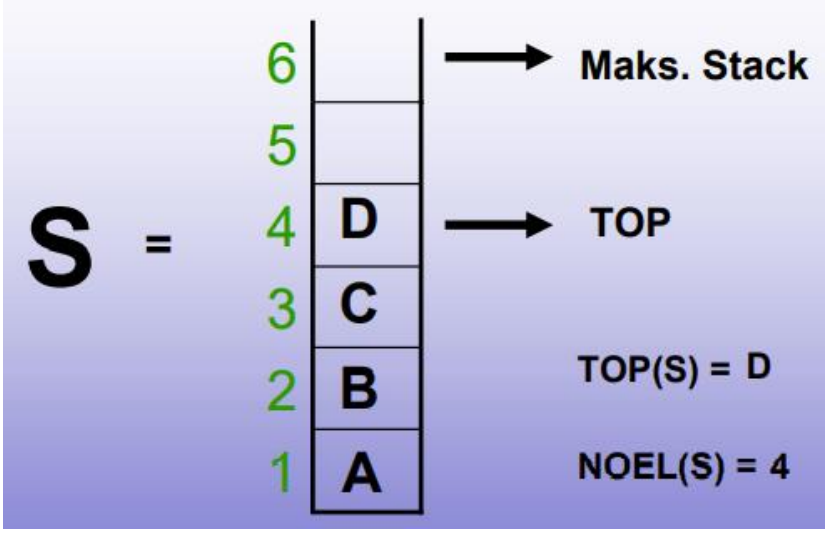
STACK

II. TUJUAN PERCOBAAN

Adapun tujuan dari percobaan ini adalah sebagai berikut :

1. Memahami bagaimana Stack Bekerja
2. Dapat membuat Stack

III. TEORI DASAR



Gambar 3.1 Representasi stack

Pengertian

Stack pada struktur data adalah sebagai tumpukan dari benda, sekumpulan data yang seolah-olah diletakkan di atas data yang lain, koleksi dari objek-objek homogen, atau suatu urutan elemen yang elemennya dapat diambil dan ditambah hanya pada posisi akhir (top) saja. Dalam struktur data stack ada dua kondisi yang

perlu dihindari, yaitu underflow dan overflow.

- Stack underflow, yaitu keadaan dimana kita mencoba mengakses atau menghapus elemen data pada stack yang kosong
- Stack overflow, yaitu keadaan di mana ruang memori yang dialokasikan untuk struktur data stack sudah penuh namun masih dilakukan operasi penyisipan elemen

Karakteristik

Struktur data stack memiliki ciri sebagai berikut:

- Stack digunakan pada banyak algoritma yang berbeda seperti Tower of Hanoi, Tree traversal, rekursi dll.
- Stack diimplementasikan dengan struktur data array atau linked list.
- Mengikuti prinsip operasi Last In First Out, yaitu elemen yang dimasukkan pertama akan muncul terakhir dan sebaliknya.
- Penyisipan dan penghapusan terjadi di satu ujung yaitu dari atas tumpukan.
- Apabila ruang memori yang dialokasikan untuk struktur data stack sudah penuh namun masih dilakukan operasi penyisipan elemen maka akan terjadi stack overflow.
- Apabila struktur data tidak memiliki elemen data atau kosong, namun tetap dilakukan operasi penghapusan maka akan terjadi stack underflow

Jenis-jenis Stack

Berdasarkan kemampuan menyimpan data, struktur data stack dapat dibagi menjadi dua jenis, yaitu: register stack dan memory stack.

1. Register stack

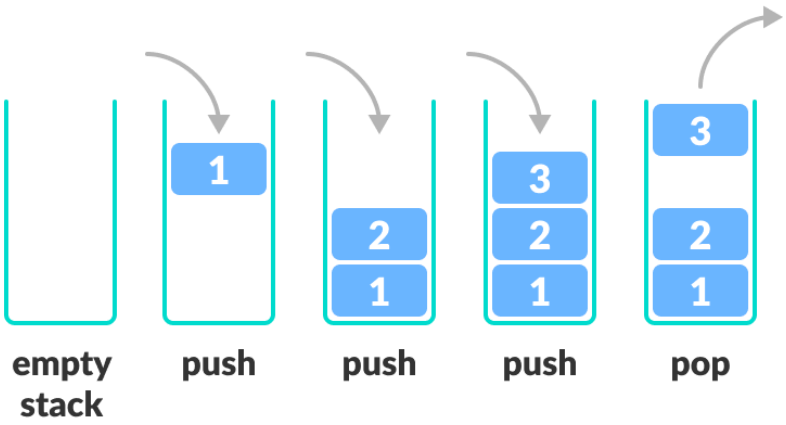
Register stack merupakan stack yang hanya mampu menampung data dalam jumlah yang kecil. Kedalaman maksimum pada register stack cenderung dibatasi karena ukuran unit memorinya sangat kecil dibandingkan dengan memory stack.

2. Memory stack

Pada stack jenis ini, kedalaman dari stack cukup fleksibel dan mampu menangani data dalam skala yang lebih besar dibandingkan jenis sebelumnya.

Cara Kerja

Cara struktur data stack dalam menyimpan sebuah nilai dapat kita bayangkan seperti piring yang disusun rapi secara bertumpuk ke atas.



Gambar 3.2 Ilustrasi cara kerja stack

Apabila kita ingin mengambil piring bagian bawah, kita harus terlebih dahulu menyisihkan semua piring yang ada di atas. Dalam istilah pemrograman, upaya menambahkan elemen pada struktur data stack disebut dengan push. Sedangkan proses menghapus atau menghilangkan elemen data dari stack disebut pop. Dari gambar di atas, dapat terlihat bahwa meskipun elemen ke-3 adalah yang paling terakhir ditambahkan, namun elemen tersebut justru yang pertama dihapus. Operasi inilah yang kemudian disebut sebagai prinsip operasi LIFO (Last In First Out).

Kelebihan Menggunakan Stack

Adapun kelebihan menggunakan struktur data stack di antaranya:

- Manajemen data yang efisien: Stack membantu mengelola data berdasarkan prinsip operasi LIFO yang tidak bisa dilakukan dengan linked list dan array.
- Manajemen fungsi yang efisien: Ketika suatu fungsi dipanggil, variabel lokal disimpan dalam stack, dan secara otomatis dihancurkan setelah dikembalikan.
- Kontrol atas memori: Stack memungkinkan kita untuk mengontrol bagaimana memori dialokasikan dan tidak dialokasikan.

- Manajemen memori cerdas: Stack secara otomatis membersihkan objek.
- Tidak mudah rusak: Stack tidak mudah rusak, oleh karena itu stack cenderung lebih aman dan dapat diandalkan.
- Tidak mengizinkan pengubahan ukuran variabel: Variabel pada stack tidak dapat diubah ukurannya.

Kekurangan Menggunakan Stack

Adapun kekurangan menggunakan struktur data stack di antaranya

- Ukuran memori terbatas: Memori pada stack cukup terbatas.
- Kemungkinan stack overflow: Terlalu banyak membuat objek di stack dapat meningkatkan risiko stack overflow.
- Akses acak tidak dimungkinkan: Dalam stack, akses data secara acak tidak bisa dilakukan. Data yang dapat diakses adalah data yang berada pada elemen atas.
- Dapat menyebabkan fungsi tidak terdefinisi: Ketika penyimpanan variabel akan ditimpa, kadang-kadang akan menyebabkan perilaku fungsi atau program yang tidak terdefinisi.
- Penghentian yang tidak diinginkan: Jika stack berada di luar memori maka dapat menyebabkan penghentian yang tidak normal.

Operasi yang ada pada stack

- Pendeklarasian Stack

  Proses pembuatan struktur stack dalam memori. Karena stack dapat direpresentasikan dalam 2 cara, maka pendeklarasian stack pun ada 2 yaitu pendeklarasian stack menggunakan array dan single linked list, hanya pada bagian ini hanya dibahas stack menggunakan array.
- Inisialisasi

  Proses pembuatan suatu stack kosong. Proses inisialisasi untuk stack yang menggunakan array adalah dengan mengisi nilai field top dengan 0 (nol) jika

elemen pertama diawali dengan nomor 1.

- Pop

  Operasi Pop pada stack adalah operasi yang berfokus pada penghapusan elemen. Dikarenakan dalam stack programmer hanya memiliki akses pada bagian atas, hanya ada satu elemen yang dapat dihapus.

- Push

  Kebalikan dari pop, operasi push justru lebih berfokus pada memasukkan elemen ke dalam stack atau tumpukan.

- isFull

  Operasi stack yang satu ini adalah untuk mengetahui apakah tumpukan sudah penuh atau belum.

- isEmpty

  Kebalikan dari isFull, isEmpty merupakan operasi yang digunakan untuk memeriksa apakah tumpukan kosong atau tidak.

- Clear

  Berfungsi digunakan untuk mengosongkan stack dengan cara melakukan tes Top dengan kondisi jika Top bernilai kurang dari nol maka stack dianggap kosong.

- Print / display

  Berfungsi untuk mencetak semua data dalam tumpukan

IV. PROSEDUR PERCOBAAN

Adapun source code untuk percobaan ini adalah sebagai berikut :

4.1. Percobaan 5-1: Single Stack dengan Struct

```cpp
#include<iostream>
#include <cstring>
using namespace std;
#define MAX_STACK 10

struct STACK
{
    int top;
    char data[10][10];
};
STACK tumpuk;

void inisialisasi();
int isFull();
int isEmpty();
void push(char d[10]);
void pop();
void clear();
void tampilStack();

int main()
{
    int pil;
    inisialisasi();
    char dt[10];
    do{
        cout << "1. push" << endl;
        cout << "2. pop" << endl;
        cout << "3. print" << endl;
        cout << "4. clear" << endl;
        cout << "5. exit" << endl;
        cout << "Pilihan : ";
        cin >> pil;
        switch(pil){
            case 1:
                if(isFull() != 1){
                    cout << "Data = ";
                    cin >> dt;
                    push(dt);
                }
                else
                    cout << "Sudah penuh!" << endl;
                break;
                case 2:
                    if(isEmpty() != 1)
                        pop();
                    else
                        cout << "Masih kosong!" << endl;
                break;
                case 3:
                    if(isEmpty() != 1)
                        tampilStack();
                else
                    cout << "Masih kosong!" << endl;
                break;
                case 4:
                    clear();
                    cout << "Sudah kosong!" << endl;
                break;
        }
```

```
61
62      }while(pil != 5);
63
64      return 0;
65  }
66
67  void inisialisasi()
68  {
69      tumpuk.top = -1;
70  }
71  int isFull()
72  {
73      if(tumpuk.top == MAX_STACK-1)
74          return 1;
75      else
76          return 0;
77  }
78  int isEmpty()
79  {
80      if(tumpuk.top == -1)
81          return 1;
82      else
83          return 0;
84  }
85  void push(char d[10])
86  {
87      tumpuk.top++;
88      strcpy(tumpuk.data[tumpuk.top],d);
89  }
90  void pop()
91  {
92      cout << "Data yang terambil = " << tumpuk.data[tumpuk.top] << endl;
93      tumpuk.top--;
94  }
95  void clear()
96  {
97      tumpuk.top=-1;
98  }
99  void tampilStack()
100 {
101     for(int i=tumpuk.top;i>=0;i--)
102     {
103         cout << "Data : " << tumpuk.data[i] << endl;
104     }
105 }
```

Gambar 4.1 stackStruct

4.2 Percobaan 5-2: Stack dengan Array Push

```cpp
#include<iostream>
#include <cstring>
using namespace std;
#define MAX_STACK 10

int push(int [], int &, int);
void display(int [], int);
const int SIZE = 50;

int main()
{
    int stack[SIZE], item, top=-1, res;
    char ch='y';
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan elemen: ";
        cin>>item;
        res = push(stack, top, item);
        if(res == -1)
        {
            cout<<"Overflow..!!..Keluar program..!!";
            return 0;
        }
        cout<<"Elemen berhasil dimasukkan" << endl;
        cout<<"Stack: ";
        display(stack, top);
        cout<<"Mau menambahkan elemen ? (y/n) ";
        cin>>ch;
    }
    return 0;
}

int push(int stack[], int &top, int elem)
{
    if(top == SIZE-1)
    {
        return -1;
    }
    else
    {
        top++;
        stack[top] = elem;
    }
    return 0;
}
void display(int stack[], int top)
{
    cout<<stack[top]<<" <-- "<<"\n";
    for(int i=top-1; i>=0; i--)
    {
        cout<<stack[i]<<"\n";
    }
}
```

Gambar 4.2 stackArrayPush

4.3 Percobaan 5-3: Stack dengan Array Pop

```cpp
#include<iostream>
#include <cstring>
using namespace std;
#define MAX_STACK 10

int pop(int [], int &);
int push(int [], int &, int);
void display(int [], int);
const int SIZE = 50;

int main()
{
    int stack[SIZE], item, top=-1, res;
    char ch='y';
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan elemen: ";
        cin>>item;
        res = push(stack, top, item);
        if(res == -1)
        {
            cout<<"Overflow..!!..Keluar program..!!";
            return 0;
        }
        cout<<"Elemen berhasil dimasukkan" << endl;
        cout<<"Stack: ";
        display(stack, top);
        cout<<"Mau menambahkan elemen? (y/n) ";
        cin>>ch;
    }

    cout << "Penghapusan elemen dimulai..." << endl;
    ch='y';
    while(ch=='y' || ch=='Y')
    {
        res = pop(stack, top);
        if(res==-1)
        {
            cout<<"Underflow..!!..Aborting..!!..Keluar program...";
            return 0;
    }
    else
    {
            cout<<"Elemen yang dihapus adalah: "<<res<<endl;
            cout<<"Stack: ";
            display(stack, top);
        }
        cout<<"Mau menghapus lagi? (y/n).. ";
        cin>>ch;
    }
```

```
51      return 0;
52  }
53
54  int push(int stack[], int &top, int elem)
55  {
56      if(top == SIZE-1)
57      {
58          return -1;
59      }
60      else
61      {
62          top++;
63          stack[top] = elem;
64      }
65      return 0;
66  }
67
68  int pop(int stack[], int &top)
69  {
70      int ret;
71      if(top==-1)
72      {
73          return -1;
74      }
75      else
76      {
77          ret=stack[top];
78          top--;
79      }
80      return ret;
81  }
82
83  void display(int stack[], int top)
84  {
85      cout<<stack[top]<<" <-- "<<"\n";
86      for(int i=top-1; i>=0; i--)
87      {
88          cout<<stack[i]<<"\n";
89      }
90  }
```

Gambar 4.3 stackArrayPop

4.4 Percobaan 4-4: Stack dengan Single Linked List

```
#include<iostream>
#include <cstring>
using namespace std;

struct node
{
    int info;
    node *next;
} *top, *newptr, *save, *ptr;

node *create_new_node(int);
void push(node *);
void display(node *);

int main()
{
    int inf;
    char ch='y';
    top=NULL;
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan elemen: ";
        cin>>inf;
        newptr = create_new_node(inf);
        if(newptr == NULL)
        {
            cout<<"Maaf, tidak bisa membuat node..Keluar program..!!";
            return 0;
        }
        cout<<"Elemen dimasukkan..." << endl;
        push(newptr);
        cout<<"Elemen berhasil dimasukkan..." << endl;
        cout<<"Stack: ";
        display(top);
        cout<<"Mau menambahkan elemen ? (y/n) ";
        cin>>ch;
    }
    return 0;
}

nod e *create_new_node(int x)
{
    ptr = new node;
    ptr->info = x;
    ptr->next = NULL;
    return ptr;
}

void push(node *n)
{
```

```
51      if(top==NULL)
52      {
53          top=n;
54      }
55      else
56      {
57          save = top;
58          top = n;
59          n->next = save;
60      }
61  }
62
63  void display(node *n)
64  {
65      while(n != NULL)
66      {
67          cout<<n->info<<" -> ";
68          n = n->next;
69      }
70      cout<<"!!\n";
71  }
```

Gambar 4.4 stackLinkedListPush

4.5 Percobaan 5-5: Stack dengan Linked List Pop

```cpp
#include<iostream>
#include <cstring>
using namespace std;

struct node
{
    int info;
    node *next;
} *top, *newptr, *save, *ptr;

node *create_new_node(int);
void push(node *);
void pop();
void display(node *);

int main()
{
    int inf;
    char ch='y';
    top=NULL;
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan elemen: ";
        cin>>inf;
        newptr = create_new_node(inf);
        if(newptr == NULL)
        {
            cout<<"Maaf, tidak bisa membuat node..Keluar program..!!";
            return 0;
        }
        cout<<"Elemen dimasukkan..." << endl;
        push(newptr);
        cout<<"Elemen berhasil dimasukkan..." << endl;
        cout<<"Stack: ";
        display(top);
        cout<<"Mau menambahkan elemen ? (y/n) ";
        cin>>ch;
    }

    do
    {
        cout<<"Stack: \n";
        display(top);
        cout<<"Mau mengeluarkan elemen? (y/n) ";
        cin>>ch;
        if(ch=='y' || ch=='Y')
        {
            cout << "Elemen yang dikeluarkan: " << top->info << endl;
            pop();
        }
        cout<< endl;
    }while(ch=='y' || ch=='Y');
```

```
53
54      return 0;
55  }
56
57  node *create_new_node(int x)
58  {
59      ptr = new node;
60      ptr->info = x;
61      ptr->next = NULL;
62      return ptr;
63  }
64
65  void push(node *n)
66  {
67      if(top==NULL)
68      {
69          top=n;
70      }
71      else
72      {
73          save = top;
74          top = n;
75          n->next = save;
76      }
77  }
78
79  void pop()
80  {
81      if(top==NULL)
82      {
83          cout<<"Underflow..!!..Keluar dari program...";
84          exit(1);
85      }
86      else
87      {
88          ptr = top;
89          top = top->next;
90      }
91  }
92
93  void display(node *n)
94  {
95      while(n != NULL)
96      {
97          cout<<n->info<<" -> ";
98          n = n->next;
99      }
100     cout<<"!!\n";
101 }
```

Gambar 4.5 stackLinkedListPop

V. PEMBAHASAN

Adapun source code percobaan ini adalah sebagai berikut :

5.1. Percobaan 5-1: Single Stack dengan Struct

5.1.a Source code

```cpp
#include<iostream>
#include <cstring>
using namespace std;
#define MAX_STACK 10

struct STACK
{
    int top;
    char data[10][10];
};
STACK tumpuk;

void inisialisasi();
int isFull();
int isEmpty();
void push(char d[10]);
void pop();
void clear();
void tampilStack();

int main()
{
    int pil;
    inisialisasi();
    char dt[10];
    do{
        cout << "1. push" << endl;
        cout << "2. pop" << endl;
        cout << "3. print" << endl;
        cout << "4. clear" << endl;
        cout << "5. exit" << endl;
        cout << "Pilihan : ";
        cin >> pil;
        switch(pil){
            case 1:
                if(isFull() != 1){
                    cout << "Data = ";
                    cin >> dt;
                    push(dt);
                }
                else
                    cout << "Sudah penuh!" << endl;
                break;
                case 2:
                    if(isEmpty() != 1)
                        pop();
                    else
                        cout << "Masih kosong!" << endl;
                break;
                case 3:
                    if(isEmpty() != 1)
                        tampilStack();
                else
                    cout << "Masih kosong!" << endl;
                break;
                case 4:
                    clear();
                    cout << "Sudah kosong!" << endl;
                break;
        }
```

```
61
62      }while(pil != 5);
63
64      return 0;
65  }
66
67  void inisialisasi()
68  {
69      tumpuk.top = -1;
70  }
71  int isFull()
72  {
73      if(tumpuk.top == MAX_STACK-1)
74          return 1;
75      else
76          return 0;
77  }
78  int isEmpty()
79  {
80      if(tumpuk.top == -1)
81          return 1;
82      else
83          return 0;
84  }
85  void push(char d[10])
86  {
87      tumpuk.top++;
88      strcpy(tumpuk.data[tumpuk.top],d);
89  }
90  void pop()
91  {
92      cout << "Data yang terambil = " << tumpuk.data[tumpuk.top] << endl;
93      tumpuk.top--;
94  }
95  void clear()
96  {
97      tumpuk.top=-1;
98  }
99  void tampilStack()
100 {
101     for(int i=tumpuk.top;i>=0;i--)
102     {
103         cout << "Data : " << tumpuk.data[i] << endl;
104     }
105 }
```

Gambar 5.1.a Source Code Single Stack dengan Struct

Berdasarkan gambar 5.1.a yang merupakan source code dari program single stack dengan struct, dapat dilihat pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang berfungsi untuk menambahkan perintah input dan output. Pada baris ke 2 terdapat penggunaan libarary cstring yang berfungsi untuk memanggil fungsi-fungsi yang berkaitan dengan manipulasi string. Kemudian pada baris ke 3

terdapat instruksi penggunaan penamaan standar untuk compiler. Pada baris ke 4 terdapat pndefinisian variabel global MAX_STACK dengan nilai 10. Variabel ini nantinya akan menjadi penanda jumlah maksimum dari banyak data yang dapat ditumpuk. Kemudian pada baris ke 6 terdapat pendeklarasian structure STACK dengan atribut integer top dan array dua dimensi bertipe data char dengan panjang array 10 × 10. Pada baris ke 11 terdapat inisialisasi nama STACK dengan nama tumpuk. Pada baris ke 13 terdapat deklarasi void function dengan nama inisialisasi. Pada baris ke 14 terdapat deklarasi fungsi bertipe data integer dengan nama isFull. Pada baris ke 15 terdapat deklarasi fungsi bertipe data integer dengan nama isEmpty. Pada baris ke 16 terdapat inisialisasi coid function dengan nama push dan parameter array char d[10]. Pada baris ke 17 terdapat void function dengan nama pop. Pada baris ke 18 terdapat deklarasi void function dengan nama clear. Pada baris ke 19 terdapat deklarsi void function dengan nama tampilStack. Pada baris ke 21 terdapat inisialisasi fungsi main. Di dalam fungsi main, pada baris ke 23 terdapat deklarasi variabel integer pil. Pada baris ke 24 terdapat perintah untuk menjalankan fungsi inisialisasi. Pada baris ke 25 terdapat inisialisasi array char dt dengan panjang array 10. Pada baris ke 26 terdapat do dari perulangan while-do . Di dalamnya, pada baris ke 27 hingga 32, terdapat perintah untuk menampilkan pilihan menu push, pop, print, celar, dan exit. Kemudian pada baris ke 33 terdapat perintah agar user memasukkan nilai kepada variabel pil. Nilai dari pil nantinya akan menjadi penanda bagi user untuk menjalankan operasi yang dipilih. Kemudian pada baris ke 34 terdapat perulangan switch dengan kondisi dari pil. Pada perulangan ini, program memproses input dari user untuk menentukan operasi apa yang akan dijalankan. Pada baris ke 35, terdapat case 1, yaitu program melakukan operasi push. Pada baris ke 36 terdapat percabangan if dengan kondisi jika stack belum penuh, maka meminta user untuk memasukkan data dan lakukan fungsi push. Jika stack sudah penuh maka tampilkan pesan “Sudah penuh!”. Pada baris ke 44 terdapat case ke 2 yaitu program melakukan operasi pop. Pada baris ke 45 terdapat percabangan if dengan kondisi jika stack belum kosong maka lakukan fungsi pop. Jika stack kosong maka tampilkan pesan “Masih kosong!”. Pada baris ke 50 terdapat case 3 yaitu menjalankan operasi print. Pada baris ke 51 terdapat percabangan if dengan kondisi jika stack tidak kosong, maka jalankan fungsi

tampilStack. Jika stack kosong, maka tampilkan pesan “Masih ksong!”. Pada baris ke 56 terdapat case 4 yaitu operasi clear. Di dalamnya, pada baris ke 57 terdapat perintah untuk menjalankan fungsi clear. Kemudian pada baris ke 58 terdapat perintah untuk menampilkan pesan “Sudah kosong!”. Kemudian pada baris ke 62 terdapat kondisi dari perulangan do-while, yaitu jika pil != 5. Pada baris ke 64 terdapat return 0 yang berarti bahwa program berjalan dengan seharusnya. Pada baris ke 67 terdapat inisialisasi dari void function inisialisasi. Di dalamnya terdapat inisialisasi nilai tumpuk.top = -1. Kemudian pada baris ke 71 terdapat inisialisasi dari fungsi isFull. Di dalamnya terdapat percabangan if dengan kondisi jika tumpuk.top = MAX_STACK -1 (stack penuh), maka kembalikan nilai 1. Jika tidak, maka kembalikan nilai 0. Kemudian pada baris ke 78 terdapat inisialisasi dari fungsi isEmpty. Di dalamnya terdapat percabangan dengan kondisi jika tumpuk.top = -1 (stack kosong), maka kembalikan nilai 1. Jika tidak maka kembalikan nilai 0. Pada baris ke 85 terdapat inisialisasi fungsi push dengan parameter char. Di dalamnya terdapat perintah increment nilai dari tumpuk.top dan memanggil fungsi strcpy dengan tujuan tumpuk.data[tumpuk.top] dan asal yaitu d. pada baris ke 90 terdapat inisialisasi voif function pop. Di dalamnya terdapat perintah untuk menampilkan data yang dihapus dan perintah untuk melakukan decrement pada nilai tumpuk.top. Pada baris ke 95 terdapat inisialisasi void function clear dan di dalmnya terdapat inisialisasi nilai tumpuk.top = -1. Kemudian pada baris ke 99 terdapat inisialisasi void function tampilStack. Di dalamnya terdapat perulangan dengan kondisi int i = tumpuk.top dan jika i >= 0 maka lakukan decrement nilai i dan tampilkan nilai dari tumpuk data indeks ke i.

5.1.b Output

```
+ 5-1stackStruct.cpp -o 5-1stackStruct } ; if ($?) { .\5-1stackStruct }
1. push
2. pop
3. print
4. clear
5. exit
Pilihan : 1
Data = 12
1. push
2. pop
3. print
4. clear
5. exit
Pilihan : 1
Data = 34
1. push
2. pop
3. print
4. clear
5. exit
Pilihan : 2
Data yang terambil = 34
1. push
2. pop
3. print
4. clear
5. exit
Pilihan : 3
Data : 12
1. push
2. pop
3. print
4. clear
5. exit
Pilihan : 4
Sudah kosong!
1. push
2. pop
3. print
4. clear
5. exit
Pilihan : 5
```

Gambar 5.1.b Output Single Stack dengan Struct

Berdasarkan gambar 5.1.b yang merupakan output dari single stack dengan struct, dapat dilihat bahwa pertama kali dijalankan, program menampilkan 5 menut yang terdiri dari push, pop, print, clear, dan exit, sesuai dengan source ode pada baris ke 28 – 33. Kemudian user diminta untuk memilih salah satu diantara proses tersebut.

Pada percobaan kali ini, pertama user memasukkan nilai 1 untuk menjalankan fungsi push. User kemudian diminta untuk memasukkan data yang akan dipush. Kemudian user memasukkan nilai 12 sebagai data yang akan di psuh. Selanjutnya, karena kita tidak memilih menu 5, maka sesuai dengan kondisi perulangan pada baris ke 62, maka program kembali menampilkan menu dan meminta user untuk memilih salah satu dari menu tersebut. Kemudian user memilih menu 1 kembali untuk melakukan push. Kemudian program kembali meminta data yang akan di push, dan user memasukkan nilai 34 sebagai data yang akan di push. Kemudian karena kita tidak memilih menu 5, maka sesuai dengan kondisi perulangan pada baris ke 62, maka program kembali menampilkan menu. Kemudian user memilih menu ke 2 untuk melakukan operasi pop. Kemudian program mengecek apakah stack sudah terisi atau masih kosong. Karena stack sudah terisi dua nilai, maka program mengeluarkan stack yang berada di paling atas, yaitu 34. Kemudian karena kita tidak memilih menu 5, maka sesuai dengan kondisi perulangan pada baris ke 62, program kembali menampilkan menu dan meminta user untuk memasukkan pilihan. Kemudian user memasukkan nilai 3 untuk memilih operasi print. Kemudian program menampilkan stack yang telah dibuat. Karena stack hanya berisi satu nilai, maka program hanya menampilkan nilai 12. Kemudian karena kita tidak memilih menu 5, maka sesuai dengan kondisi perulangan pada baris ke 62, program kembali menampilkan menu dan meminta user untuk memasukkan pilihan. Kemudian user memasukkan nilai 4 untuk memilih menu clear dan menghapus seluruh isi stack. Kemudian program menjalankan fungsi clear yang mana menginisialisasi nilai tumpuk.top menjadi -1, sehingga stack seperti diinisialisasikan ulang dan kemudian menampilkan pesan “Sudah kosong!”. Kemudian karena kita tidak memilih menu 5, maka sesuai dengan kondisi perulangan pada baris ke 62, program kembali menampilkan menu dan meminta user untuk memasukkan pilihan. Kemudian user memasukkan nilai 5 untuk memilih menu exit. Karena user memilih menu 5, maka perulangan pada baris ke 62 menjadi bernilai false sehingga perulangan berhenti dan program selesai.

5.2 Percobaan 5-2: Stack dengan Array Push

5.2.a Source code

```cpp
#include<iostream>
#include <cstring>
using namespace std;
#define MAX_STACK 10

int push(int [], int &, int);
void display(int [], int);
const int SIZE = 50;

int main()
{
    int stack[SIZE], item, top=-1, res;
    char ch='y';
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan elemen: ";
        cin>>item;
        res = push(stack, top, item);
        if(res == -1)
        {
            cout<<"Overflow..!!..Keluar program..!!";
            return 0;
        }
        cout<<"Elemen berhasil dimasukkan" << endl;
        cout<<"Stack: ";
        display(stack, top);
        cout<<"Mau menambahkan elemen ? (y/n) ";
        cin>>ch;
    }
    return 0;
}

int push(int stack[], int &top, int elem)
{
    if(top == SIZE-1)
    {
        return -1;
    }
    else
    {
        top++;
        stack[top] = elem;
    }
    return 0;
}
void display(int stack[], int top)
{
    cout<<stack[top]<<" <-- "<<"\n";
    for(int i=top-1; i>=0; i--)
    {
        cout<<stack[i]<<"\n";
    }
}
```

Gambar 5.2.a Source Code Stack dengan Array Push

Berdasarkan gambar 5.2.a yang merupakan source code dari program stack dengan array push, dapat dilihat pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang berfungsi untuk menambahkan perintah input dan output. Pada baris ke 2 terdapat penggunaan libarary cstring yang berfungsi untuk memanggil fungsi-fungsi yang berkaitan dengan manipulasi string. Kemudian pada baris ke 3 terdapat instruksi penggunaan penamaan standar untuk compiler. Pada baris ke 4 terdapat pndefinisian variabel global MAX_STACK dengan nilai 10. Variabel ini nantinya akan menjadi penanda jumlah maksimum dari banyak data yang dapat ditumpuk. Pada baris ke 6 terdapat ini deklarasi fungsi integer push dengan parameter array integer, address variabel integer, dan variabel integer. Pada baris ke 7 terdapat deklarasi fungsi void display dengan parameter array integer dan variabel integer. Pada baris ke 8 terdapat inisialisasi variabel integer konstan SIZE dengan nilai 50. Pada baris ke 10 terdpat inisialisasi fungsi main. Pada baris ke 12 terdapat deklarasi array integer stack dengan panjang array sebanyak nilai dari variabel SIZE, variabel integer item, variabel integer top dengan nilai -1, dan variabel integer res. Pada baris ke 13 terdapat inisialisasi variabel char ch dengan nilai y. Pada baris ke 14 terdapat perulangan yang akan berkondisi true jika ch = y atau Y. Kemudian di dalam perulangan tersebut terdapat perintah bagi user untuk memasukkan data. Kemudian program akan menginisialisasi nilai res dari hasil menjalankan fungsi push. Pada baris ke 19 terdapat percabangan if yang jika nilai res = -1, maka program akan menampilkan pesan “Overflow..!!..Keluar program..!!” dan mengakhiri fungsi main dengan return 0 yang berarti program berjalan secara normal. Kemudian pada baris ke 24 terdapat perintah untuk menampilkan pesan “Elemen berhasil dimasukkan” dan kemudian menjalankan fungsi display dengan argument stack dan top sehingga menampilkan stack yang telah dibuat. Pada baris ke 27 dan 28 program menanyakan apakah user ingin menambahkan elemen lagi atau kelauar dari program. Pada baris ke 33 terdapat inisialisasi fungsi integer push dengan parameter array integer stack, addres variabel integer top, dan variabel integer elemen. Di dalamnya, pada aris ke 35 terdapat percabangan yang berfungsi untuk mengecek apakah stack sudah penuh atau belum. Jika sudah penuh, maka program akan mengembalikan nilai -1. Jika belum penuh maka program akan melakukan increment terhadap nilai top dan

inisialisasi array stack [top] dengan nilai dari variabel elem. Kemudian program mengembalikan nilai return 0 yang berarti program berjalan dengan normal hingga akhir. Pada baris ke 46 terdapat inisialisasi fungsi display dengan parameter array integer stack dan nilai dari variabel top. Pada baris ke 48 terdapat perintah untuk menampilkan stack[top] dan pada baris berikutnya terdapatperulangan for dengan kondisi int i = top 1 dan jika i >= 0 maka lakukan increment nilai i dan tampilkan nilai dari stack[i].

5.2.b Output

```
($?) { g++ 5-2stackArrayPush.cpp -o 5-2stackArrayPush
 } ; if ($?) { .\5-2stackArrayPush }
Masukkan elemen: 12
Elemen berhasil dimasukkan
Stack: 12 <--
Mau menambahkan elemen ? (y/n) y
Masukkan elemen: 90
Elemen berhasil dimasukkan
Stack: 90 <--
12
Mau menambahkan elemen ? (y/n) y
Masukkan elemen: 34
Elemen berhasil dimasukkan
Stack: 34 <--
90
12
Mau menambahkan elemen ? (y/n) n
```

Gambar 5.2.b Output Stack dengan Array Push

Berdasarkan percobaan 5.2.b yang merupakan output dari stack dengan array push, dapat dilihat bahwa pertama program meminta user untuk memasukkan nilai untuk elemen stack yang pertama. Kemudian user memasukkan nilai 12. Input tersebut kemudian menjadi argument untuk memanggil fungsi push pada baris ke 33 bersama dengan array stack dan top dan hasil dari pemanggilan fungsi akan disimpan ke dalam variabel res. Kemudian karena nilai res bukan -1, maka program menyatakan bahwa elemen berhasil dimasukkan dan menampilkan stack yang dibuat dengan memanggil fungsi display, yaitu 12. Kemudian program menanyakan pada user apakah ingin menambahkan elemen lagi. Kemudian user memasukkan nilai y yang berarti user akan menambahkan elemen baru pada stack. Nilai tersebut disimpan ke dalam variabel ch dan kemudian dijadikan acuan bagi program untuk menjalankan perulangan kembali atau tidak. Karena ch bernilai y, maka program kembali meminta user untuk memasukkan nilai untuk elemen stack yang kedua. Kemudian user memasukkan nilai 90. Input tersebut kemudian menjadi argument untuk memanggil fungsi push pada baris ke 33 bersama dengan array stack dan top dan hasil dari pemanggilan fungsi akan disimpan ke dalam variabel res. Kemudian karena nilai res bukan -1, maka program menyatakan bahwa elemen berhasil dimasukkan dan menampilkan stack yang dibuat dengan memanggil fungsi display, yaitu 90 dan 12. Kemudian program menanyakan pada user apakah ingin

menambahkan elemen lagi. Kemudian user memasukkan nilai y yang berarti user akan menambahkan elemen baru pada stack. Nilai tersebut disimpan ke dalam variabel ch dan kemudian dijadikan acuan bagi program untuk menjalankan perulangan kembali atau tidak. Karena ch bernilai y, maka program kembali meminta user untuk memasukkan nilai untuk elemen stack yang ketiga. Kemudian user memasukkan nilai 34. Input tersebut kemudian menjadi argument untuk memanggil fungsi push pada baris ke 33 bersama dengan array stack dan top dan hasil dari pemanggilan fungsi akan disimpan ke dalam variabel res. Kemudian karena nilai res bukan -1, maka program menyatakan bahwa elemen berhasil dimasukkan dan menampilkan stack yang dibuat dengan memanggil fungsi display, yaitu 34, 90, dan 12. Kemudian program menanyakan pada user apakah ingin menambahkan elemen lagi. Kemudian user memasukkan nilai n yang berarti user tidak ingin menambahkan elemen baru pada stack. Nilai tersebut disimpan ke dalam variabel ch dan kemudian dijadikan acuan bagi program untuk menjalankan perulangan kembali atau tidak. Karena ch bernilai n, maka program keluar dari perulangan dan kemudian program selesai.

5.3 Percobaan 5-3: Stack dengan Array Pop

5.3.a Source Code Stack dengan Array Pop

```cpp
#include<iostream>
#include <cstring>
using namespace std;
#define MAX_STACK 10

int pop(int [], int &);
int push(int [], int &, int);
void display(int [], int);
const int SIZE = 50;

int main()
{
    int stack[SIZE], item, top=-1, res;
    char ch='y';
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan elemen: ";
        cin>>item;
        res = push(stack, top, item);
        if(res == -1)
        {
            cout<<"Overflow..!!..Keluar program..!!";
            return 0;
        }
        cout<<"Elemen berhasil dimasukkan" << endl;
        cout<<"Stack: ";
        display(stack, top);
        cout<<"Mau menambahkan elemen? (y/n) ";
        cin>>ch;
    }

    cout << "Penghapusan elemen dimulai..." << endl;
    ch='y';
    while(ch=='y' || ch=='Y')
    {
        res = pop(stack, top);
        if(res==-1)
        {
            cout<<"Underflow..!!..Aborting..!!..Keluar program...";
            return 0;
    }
    else
    {
            cout<<"Elemen yang dihapus adalah: "<<res<<endl;
            cout<<"Stack: ";
            display(stack, top);
        }
        cout<<"Mau menghapus lagi? (y/n).. ";
        cin>>ch;
    }
```

```
51      return 0;
52  }
53
54  int push(int stack[], int &top, int elem)
55  {
56      if(top == SIZE-1)
57      {
58          return -1;
59      }
60      else
61      {
62          top++;
63          stack[top] = elem;
64      }
65      return 0;
66  }
67
68  int pop(int stack[], int &top)
69  {
70      int ret;
71      if(top==-1)
72      {
73          return -1;
74      }
75      else
76      {
77          ret=stack[top];
78          top--;
79      }
80      return ret;
81  }
82
83  void display(int stack[], int top)
84  {
85      cout<<stack[top]<<" <-- "<<"\n";
86      for(int i=top-1; i>=0; i--)
87      {
88          cout<<stack[i]<<"\n";
89      }
90  }
```

Gambar 5.3.a Source Code Stack dengan Array Pop

Berdasarkan gambar 5.3.a yang merupakan source code dari program stack dengan array pop, dapat dilihat pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang berfungsi untuk menambahkan perintah input dan output. Pada baris ke 2 terdapat penggunaan libarary cstring yang berfungsi untuk memanggil fungsi-fungsi yang berkaitan dengan manipulasi string. Kemudian pada baris ke 3 terdapat instruksi penggunaan penamaan standar untuk compiler. Pada baris ke 4 terdapat pndefinisian variabel global MAX_STACK dengan nilai 10. Variabel ini nantinya akan menjadi penanda jumlah maksimum dari banyak data yang dapat ditumpuk. Pada baris ke 6 terdapat deklarasi fungsi pop dengan parameter array integer dan address variabel integer. Pada baris ke 7 terdapat ini deklarasi fungsi integer push dengan parameter array integer, address variabel integer, dan variabel integer. Pada baris ke 8 terdapat deklarasi fungsi void display dengan parameter array integer dan variabel integer. Pada baris ke 9 terdapat inisialisasi variabel integer konstan SIZE dengan nilai 50. Pada baris ke 11 terdpat inisialisasi fungsi main. Pada baris ke 13 terdapat deklarasi array integer stack dengan panjang array sebanyak nilai dari variabel SIZE, variabel integer item, variabel integer top dengan nilai -1, dan variabel integer res. Pada baris ke 14 terdapat inisialisasi variabel char ch dengan nilai y. Pada baris ke 15 terdapat perulangan yang akan berkondisi true jika ch = y atau Y. Kemudian di dalam perulangan tersebut terdapat perintah bagi user untuk memasukkan data. Kemudian program akan menginisialisasi nilai res dari hasil menjalankan fungsi push. Pada baris ke 20 terdapat percabangan if yang jika nilai res = -1, maka program akan menampilkan pesan “Overflow..!!..Keluar program..!!” dan mengakhiri fungsi main dengan return 0 yang berarti program berjalan secara normal. Kemudian pada baris ke 25 terdapat perintah untuk menampilkan pesan “Elemen berhasil dimasukkan” dan kemudian menjalankan fungsi display dengan argument stack dan top sehingga menampilkan stack yang telah dibuat. Pada baris ke 28 dan 29 program menanyakan apakah user ingin menambahkan elemen lagi atau tidak. Jika tidak, maka perulangan awal akan berhenti dan kemudian program menampilkan pesan “Penghapusan elemen dimulai”. Kemudian terdapat inisialisasi nilai ch = y dan pada baris ke 34 terdapat perulangan yang bernilai true jika ch = y atau Y. Di dalamnya terdapat inisialisasi nilai res = hasil dari menjalankan fungsi pop dengan argument stack dan top.

Kemudian pada baris ke 37 terdapat percabangan yang jika res = -1, maka tampilkan pesan "Underflow..!!..Aborting..!!..Keluar program..." dan return 0 yang berarti program berjalan dengan normal. Jika res ≠ 1 maka program akan menampilkan elemen yang dihapus dan kemudian menjalankan fungsi stack sehingga menampilkan stack yang terbentuk. Kemudian program menanyakan pada user apakah ingin melakukan penghapusan elemen kembali dan meminta input y atau n. Pada baris ke 54 terdapat inisialisasi fungsi integer push dengan parameter array integer stack, addres variabel integer top, dan variabel integer elemen. Di dalamnya, pada aris ke 56 terdapat percabangan yang berfungsi untuk mengecek apakah stack sudah penuh atau belum. Jika sudah penuh, maka program akan mengembalikan nilai -1. Jika belum penuh maka program akan melakukan increment terhadap nilai top dan inisialisasi array stack [top] dengan nilai dari variabel elem. Kemudian program mengembalikan nilai return 0 yang berarti program berjalan dengan normal hingga akhir. Pada baris ke 68 terdapat inisialisasi fungsi integer pop dengan parameter array integer stack dan address dari variabel top. Pada baris ke 70 terdapat deklarasi variabel integer ret. Pada baris ke 71 terdapat percabangan if dengan kondisi jika top = -1 maka program akan mengembalikan nilai -1. Dan jika tidak maka program akan menginisialisasi nilai ret = stack[top] dan melakukan decrement kepada nilai top. Kemudian program akan mengembalikan nilai ret. Pada baris ke 83 terdapat inisialisasi fungsi display dengan parameter array integer stack dan nilai dari variabel top. Pada baris ke 85 terdapat perintah untuk menampilkan stack[top] dan pada baris berikutnya terdapat perulangan for dengan kondisi int i = top 1 dan jika i >= 0 maka lakukan increment nilai i dan tampilkan nilai dari stcak[i].

5.3.b Output Stack dengan Array Pop

```
 ; if ($?) { g++ 5-3stackArrayPop.cpp -o 5-3stackAr
rayPop } ; if ($?) { .\5-3stackArrayPop }
Masukkan elemen: 12
Elemen berhasil dimasukkan
Stack: 12 <--
Mau menambahkan elemen? (y/n) y
Masukkan elemen: 90
Elemen berhasil dimasukkan
Stack: 90 <--
12
Mau menambahkan elemen? (y/n) y
Masukkan elemen: 34
Elemen berhasil dimasukkan
Stack: 34 <--
90
12
Mau menambahkan elemen? (y/n) n
Penghapusan elemen dimulai...
Elemen yang dihapus adalah: 34
Stack: 90 <--
12
Mau menghapus lagi? (y/n).. y
Elemen yang dihapus adalah: 90
Stack: 12 <--
Mau menghapus lagi? (y/n).. y
Elemen yang dihapus adalah: 12
Stack: 34 <--
Mau menghapus lagi? (y/n).. y
Underflow..!!..Aborting..!!..Keluar program...
```

Gambar 5.3.b Output Stack dengan Array Pop

Berdasarkan percobaan 5.3.b yang merupakan output dari stack dengan array pop, dapat dilihat bahwa pertama program meminta user untuk memasukkan nilai untuk elemen stack yang pertama. Kemudian user memasukkan nilai 12. Input tersebut kemudian menjadi argument untuk memanggil fungsi push pada baris ke 33 bersama dengan array stack dan top dan hasil dari pemanggilan fungsi akan disimpan ke dalam variabel res. Kemudian karena nilai res bukan -1, maka program menyatakan bahwa elemen berhasil dimasukkan dan menampilkan stack yang dibuat dengan memanggil fungsi display, yaitu 12. Kemudian program menanyakan pada user apakah ingin menambahkan elemen lagi. Kemudian user memasukkan nilai y yang berarti user akan menambahkan elemen baru pada stack. Nilai tersebut

disimpan ke dalam variabel ch dan kemudian dijadikan acuan bagi program untuk menjalankan perulangan kembali atau tidak. Karena ch bernilai y, maka program kembali meminta user untuk memasukkan nilai untuk elemen stack yang kedua. Kemudian user memasukkan nilai 90. Input tersebut kemudian menjadi argument untuk memanggil fungsi push pada baris ke 33 bersama dengan array stack dan top dan hasil dari pemanggilan fungsi akan disimpan ke dalam variabel res. Kemudian karena nilai res bukan -1, maka program menyatakan bahwa elemen berhasil dimasukkan dan menampilkan stack yang dibuat dengan memanggil fungsi display, yaitu 90 dan 12. Kemudian program menanyakan pada user apakah ingin menambahkan elemen lagi. Kemudian user memasukkan nilai y yang berarti user akan menambahkan elemen baru pada stack. Nilai tersebut disimpan ke dalam variabel ch dan kemudian dijadikan acuan bagi program untuk menjalankan perulangan kembali atau tidak. Karena ch bernilai y, maka program kembali meminta user untuk memasukkan nilai untuk elemen stack yang ketiga. Kemudian user memasukkan nilai 34. Input tersebut kemudian menjadi argument untuk memanggil fungsi push pada baris ke 33 bersama dengan array stack dan top dan hasil dari pemanggilan fungsi akan disimpan ke dalam variabel res. Kemudian karena nilai res bukan -1, maka program menyatakan bahwa elemen berhasil dimasukkan dan menampilkan stack yang dibuat dengan memanggil fungsi display, yaitu 34, 90, dan 12. Kemudian program menanyakan pada user apakah ingin menambahkan elemen lagi. Kemudian user memasukkan nilai n yang berarti user tidak ingin menambahkan elemen baru pada stack. Nilai tersebut disimpan ke dalam variabel ch dan kemudian dijadikan acuan bagi program untuk menjalankan perulangan kembali atau tidak. Karena ch bernilai n, maka program keluar dari perulangan dan kemudian program menampilkan pesan “Penghapusan elemen dimulai”. Kemudian berdasarkan source code pada baris ke 34, program masuk ke perulangan while yang berfungsi untuk menghapus elemen. Pada perulangan tersebut, dijalankan fungsi pop dengan argument stack dan top yang kemudian returnnya disimpan dalam variabel res. Kemudian karena res tidak bernilai -1, maka program menampilkan elemen yang dihapus, yaitu 34 dan menampilkan stack yang dibuat dengan memanggil fungsi display, yaitu 90 dan 12. Kemudian program menanyakan apakah user ingin menghapus elemen lagi. Kemudian user

memasukkan nilai y yang berarti user akan menghapus satu elemen lagi. Input tersebut dimasukkan ke dalam variabel ch dan menjadi acuan bagi perulangan untuk menghapus elemen. Karena ch = y, maka program kembali menghapus elemen paling atas pada stack, yaitu 90 dan kembali menampilkan stack yang tersisa yaitu 12. Kemudian program menanyakan apakah user ingin menghapus elemen lagi. Kemudian user memasukkan nilai y yang berarti user akan menghapus satu elemen lagi. Karena ch = y, maka program kembali menghapus elemen paling atas pada stack, yaitu 12. Pada saat ini, seharusnya stack sudah habis, namun pada pemanggilan fungsi display tetap menampilkan nilai 34, yaitu stack terakhir yang dimasukkan. Ini dikarenakan pada variabel top bernilai -1 sehingga program melakukan kesalahan dan seakan-akan masih bernilai 34, sehingga terjadi bug. Kemudian program menanyakan apakah user ingin menghapus elemen lagi. Kemudian user memasukkan nilai y yang berarti user akan menghapus satu elemen lagi. Karena ch = y, maka program kembali mecoba menghapus satu elemen paling atas pada stack. Namun karena top = -1, maka fungsi pop mengembalikan nilai -1 yang kemudian membuat percabangan pada baris ke 37 menjadi true dan menampilkan pesan "Underflow..!!..Aborting..!!..Keluar program...". Kemudian progam selesai.

5.4 Percobaan 5-4: Stack dengan Single Linked List

5.4.a Source code Stack dengan Single Linked List

```
#include<iostream>
#include <cstring>
using namespace std;

struct node
{
    int info;
    node *next;
} *top, *newptr, *save, *ptr;

node *create_new_node(int);
void push(node *);
void display(node *);

int main()
{
    int inf;
    char ch='y';
    top=NULL;
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan elemen: ";
        cin>>inf;
        newptr = create_new_node(inf);
        if(newptr == NULL)
        {
            cout<<"Maaf, tidak bisa membuat node..Keluar program..!!";
            return 0;
        }
        cout<<"Elemen dimasukkan..." << endl;
        push(newptr);
        cout<<"Elemen berhasil dimasukkan..." << endl;
        cout<<"Stack: ";
        display(top);
        cout<<"Mau menambahkan elemen ? (y/n) ";
        cin>>ch;
    }
    return 0;
}

nod e *create_new_node(int x)
{
    ptr = new node;
    ptr->info = x;
    ptr->next = NULL;
    return ptr;
}

void push(node *n)
{
```

```
51      if(top==NULL)
52      {
53          top=n;
54      }
55      else
56      {
57          save = top;
58          top = n;
59          n->next = save;
60      }
61  }
62
63  void display(node *n)
64  {
65      while(n != NULL)
66      {
67          cout<<n->info<<" -> ";
68          n = n->next;
69      }
70      cout<<"!!\n";
71  }
```

Gambar 5.4.a Source Code Stack dengan Single Linked List

Berdasarkan gambar 5.4.a yang merupakan source code dari program stack dengan single linked list, dapat dilihat bahwa pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang berfungsi untuk menambahkan perintah input dan output. Kemudian pada baris ke 2 terdapat penggunaan libarary cstring yang berfungsi untuk memanggil fungsi-fungsi yang berkaitan dengan manipulasi string. Pada baris ke 3 terdapat instruksi penggunaan penamaan standar untuk compiler. Kemudian pada baris ke 5 terdapat deklarasi structure dengan nama node. Di dalamnya, terdapat dua field, yaitu integer info pada baris ke 7 dan node *next pada baris ke 8. Selain itu terdapat penggunaan start, newptr, save, dan ptr di dalam structure node. Kemudian terdapat inisialisasi create_new_node dengan tipe data integer pada baris ke 11. Kemudian pada baris ke 12 terdapat inisialisasi void function push dengan parameter node address. Pada baris ke 13 terdapat deklarasi void function display dengan parameter node address. Kemudian pada baris ke 15

terdapat deklarasi fungsi utama. Pada baris ke 17 terdapat deklarasi variabel integer dengan nama inf. Pada baris ke 18 terdapat inisialisasi variabel char ch dengan nilai y. Pada baeis ke 19 terdapat inisialisasi nilai top = NULL. Pada baris ke 20 terdapat perulangan while dengan kondisi ch == y atau ch == Y. Di dalam perulangan tersebut, pada baris ke 22 terdapat perintah untuk menampilkan pesan "Masukkan elemen: ". Pada baris ke 23 terdapat perintah agar user memasukkan nilai ke variabel inf. Pada baris ke 24 terdapat inisialisasi nilai variabel newptr milik node dengan nilai dari menjalankan fungsi create_new_node dengan parameter inf. Pada baris ke 25 terdapat percabangan yang jika newptr = NULL maka tampilkan pesan "Maaf, tidak bisa membuat node..Keluar program..!!" dan kembalikan program dengan nilai 0 yang berarti program berjalan dengan normal. Pada baris ke 30 terdapat perintah untuk menampilkan pesan "Elemen dimasukkan…" dan kemudian menjalankan fungsi push dengan argmument newptr. Kemudian pada baris ke 32 terdapat perintah untuk menampilkan pesan "Elemen berhasil dimasukkan…" dan kemudian menjalankan fungsi display sehingga menampilkan stack yang dibuat. Kemudian pada baris ke 35 dan 36 program menanyakan apakah user ingin menambah elemen dan meminta input y atau n yang nantinya akan dimasukkan ke variabel ch. Pada baris ke 38 terdapat return 0 yang menyatakan hasil keluaran dari fungsi main() bahwa program berakhir dengan normal. Pada baris ke 41 terdapat inisialisasi dari node create_new_node dengan parameter int x. Pada baris ke 42 terdapat inisialisasi nilai ptr dengan nilai dari new node. Pada baris ke 43 terdapat assignment nilai ptr info dengan x. Pada baris ke 44 terdapat assignment nilai ptr nextde dengan NULL. Fungsi kemudian mengembalikan nilai ptr. Pada baris ke 49 terdapat inisialisasi fungsi push dengan parameter node n. Pada baris ke 51, terdapat percabangan if dengan kondisi jika top = Null maka inisialisasi nilai top = n, jika tidak terpenuhi, maka inisialisasi nilai save = top, top = n dan n->next = save. Kemudian pada baris ke 63 terdapat inisialisasi fungsi display dengan parameter node n. Pada baris ke 65 terdapat perulangan while dengan kondisi n != NULL), maka program akan menampilkan nilai dari n->info dan kemudian menginisialisasi nilai dari n dengan n->next. Kemudian pada baris ke 70 terdapat perintah untuk menampilkan pesan !! dan berpindah baris.

5.4.b Output Stack dengan Single Linked List

```
Push.cpp -o 5-4stackLinkedListPush } ; if ($?)
{ .\5-4stackLinkedListPush }
Masukkan elemen: 12
Elemen dimasukkan...
Elemen berhasil dimasukkan...
Stack: 12 -> !!
Mau menambahkan elemen ? (y/n) y
Masukkan elemen: 90
Elemen dimasukkan...
Elemen berhasil dimasukkan...
Stack: 90 -> 12 -> !!
Mau menambahkan elemen ? (y/n) y
Masukkan elemen: 34
Elemen dimasukkan...
Elemen berhasil dimasukkan...
Stack: 34 -> 90 -> 12 -> !!
Mau menambahkan elemen ? (y/n) n
```

Gambar 5.4.b Output Stack dengan Single Linked List

Berdasarkan gambar 5.4.b yang merupakan gambar dari output stack dengan single linked list, dapat dilihat bahwa pertama program meminta user untuk memasukkan nilai untuk elemen pertama dari stack. Ini sesuai dengan perintah pada baris ke 23. Pada percobaan ini user memasukkan nilai 12. Kemudian program menampilkan pesan "Membuat node baru…". Pada saat ini program sedang menjalankan fungsi create_new_node dan kemudian akan memasukkan returnnya ke variabel newptr. Kemudian, berdasarkan percabangan pada baris ke 25, karena newptr tidak bernilai NULL, maka program menampilkan pesan "Elemen dimasukkan…" dan kemudian menjalankan fungsi push dengan argument newptr. Kemudian program menampilkan pesan "Elemen berhasil dimasukkan…" dan menampilkan stack yang telah dibuat, yaitu 12 ->. Kemudian program kembali menanyakan apakah user ingin menambahkan elemen lagi dan meminta input kepada user. User kemudian memasukkan nilai y yang berarti user ingin membuat elemen baru. Input tersebut kemudian dimasukkan ke dalam variabel ch dan dijadikan ajuan untuk menjalankan perulangan while pada baris ke 20 yang berfungsi untuk menambahkan elemen pada stack. Karena ch bernilai y, maka program kembali meminta user untuk memasukkan nilai untuk elemen kedua pada stack. Ini sesuai dengan perintah pada

baris ke 23. Pada percobaan ini user memasukkan nilai 90. Kemudian program menampilkan pesan “Membuat node baru…”. Pada saat ini program sedang menjalankan fungsi create_new_node dan kemudian akan memasukkan returnnya ke variabel newptr. Kemudian, berdasarkan percabangan pada baris ke 25, karena newptr tidak bernilai NULL, maka program menampilkan pesan “Elemen dimasukkan…” dan kemudian menjalankan fungsi push dengan argument newptr. Kemudian program menampilkan pesan “Elemen berhasil dimasukkan…” dan menampilkan stack yang telah dibuat, yaitu 90 -> 12 ->. Kemudian program kembali menanyakan apakah user ingin menambahkan elemen lagi dan meminta input kepada user. User kemudian memasukkan nilai y yang berarti user ingin membuat elemen baru. Karena ch bernilai y, maka program kembali meminta user untuk memasukkan nilai untuk elemen kedua pada stack. Ini sesuai dengan perintah pada baris ke 23. Pada percobaan ini user memasukkan nilai 34. Kemudian program menampilkan pesan “Membuat node baru…”. Pada saat ini program sedang menjalankan fungsi create_new_node dan kemudian akan memasukkan returnnya ke variabel newptr. Kemudian, berdasarkan percabangan pada baris ke 25, karena newptr tidak bernilai NULL, maka program menampilkan pesan “Elemen dimasukkan…” dan kemudian menjalankan fungsi push dengan argument newptr. Kemudian program menampilkan pesan “Elemen berhasil dimasukkan…” dan menampilkan stack yang telah dibuat, yaitu 34 -> 90 -> 12 ->. Kemudian program kembali menanyakan apakah user ingin menambahkan elemen lagi dan meminta input kepada user. User kemudian memasukkan nilai n yang berarti user tidak lagi membuat elemen baru. Karena ch bernilai n, maka program keluar dari perulangan while pada baris ke 20 tersebut dan kemudian program selesai.

5.5 Percobaan 5-5: Stack dengan Single Linked List Pop

5.5.a Source Code Stack dengan Single Linked List Pop

```
#include<iostream>
#include <cstring>
using namespace std;

struct node
{
    int info;
    node *next;
} *top, *newptr, *save, *ptr;

node *create_new_node(int);
void push(node *);
void pop();
void display(node *);

int main()
{
    int inf;
    char ch='y';
    top=NULL;
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan elemen: ";
        cin>>inf;
        newptr = create_new_node(inf);
        if(newptr == NULL)
        {
            cout<<"Maaf, tidak bisa membuat node..Keluar program..!!";
            return 0;
        }
        cout<<"Elemen dimasukkan..." << endl;
        push(newptr);
        cout<<"Elemen berhasil dimasukkan..." << endl;
        cout<<"Stack: ";
        display(top);
        cout<<"Mau menambahkan elemen ? (y/n) ";
        cin>>ch;
    }

    do
    {
        cout<<"Stack: \n";
        display(top);
        cout<<"Mau mengeluarkan elemen? (y/n) ";
        cin>>ch;
        if(ch=='y' || ch=='Y')
        {
            cout << "Elemen yang dikeluarkan: " << top->info << endl;
            pop();
        }
        cout<< endl;
    }while(ch=='y' || ch=='Y');
```

```
53
54      return 0;
55  }
56
57  node *create_new_node(int x)
58  {
59      ptr = new node;
60      ptr->info = x;
61      ptr->next = NULL;
62      return ptr;
63  }
64
65  void push(node *n)
66  {
67      if(top==NULL)
68      {
69          top=n;
70      }
71      else
72      {
73          save = top;
74          top = n;
75          n->next = save;
76      }
77  }
78
79  void pop()
80  {
81      if(top==NULL)
82      {
83          cout<<"Underflow..!!..Keluar dari program...";
84          exit(1);
85      }
86      else
87      {
88          ptr = top;
89          top = top->next;
90      }
91  }
92
93  void display(node *n)
94  {
95      while(n != NULL)
96      {
97          cout<<n->info<<" -> ";
98          n = n->next;
99      }
100     cout<<"!!\n";
101 }
```

Gambar 5.5.a Source Code Stack dengan Single Linked List Pop

Berdasarkan gambar 5.5.a yang merupakan source code dari single linked list pop, dapat dapat dilihat bahwa pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang berfungsi untuk menambahkan perintah input dan output. Kemudian pada baris ke 2 terdapat penggunaan libarary cstring yang berfungsi untuk memanggil fungsi-fungsi yang berkaitan dengan manipulasi string. Pada baris ke 3 terdapat instruksi penggunaan penamaan standar untuk compiler. Kemudian pada baris ke 5 terdapat deklarasi structure dengan nama node. Di dalamnya, terdapat dua field, yaitu integer info pada baris ke 7 dan node *next pada baris ke 8. Selain itu terdapat penggunaan start, newptr, save, dan ptr di dalam structure node. Kemudian terdapat inisialisasi create_new_node dengan tipe data integer pada baris ke 11. Kemudian pada baris ke 12 terdapat inisialisasi void function push dengan parameter node address. Pada baris ke 13 terdapat deklarasi void function pop. Pada baris ke 14 terdapat deklarasi void function display dengan parameter node address. Kemudian pada baris ke 16 terdapat deklarasi fungsi utama. Pada baris ke 18 terdapat deklarasi variabel integer dengan nama inf. Pada baris ke 19 terdapat inisialisasi variabel char ch dengan nilai y. Pada baeis ke 20 terdapat inisialisasi nilai top = NULL. Pada baris ke 21 terdapat perulangan while dengan kondisi ch == y atau ch == Y. Di dalam perulangan tersebut, pada baris ke 23 terdapat perintah untuk menampilkan pesan “Masukkan elemen: “. Pada baris ke 24 terdapat perintah agar user memasukkan nilai ke variabel inf. Pada baris ke 25 terdapat inisialisasi nilai variabel newptr milik node dengan nilai dari menjalankan fungsi create_new_node dengan parameter inf. Pada baris ke 26 terdapat percabangan yang jika newptr= NULL maka tampilkan pesan “Maaf, tidak bisa membuat node..Keluar program..!!” dan kembalikan program dengan nilai 0 yang berarti program berjalan dengan normal. Pada baris ke 31 terdapat perintah untuk menampilkan pesan “Elemen dimasukkan…” dan kemudian menjalankan fungsi push dengan argmument newptr. Kemudian pada baris ke 33 terdapat perintah untuk menampilkan pesan “Elemen berhasil dimasukkan…” dan kemudian menjalankan fungsi display sehingga menampilkan stack yang dibuat. Kemudian pada baris ke 36 dan 37 program menanyakan apakah user ingin menambah elemen dan meminta input y atau n yang nantinya akan dimasukkan ke variabel ch. Pada baris ke 40, terdapat do dari perulangan while-do. Pada baris ke

42, terdapat printah untuk menampilkan pesan “Stack “. Kemudian pada baris ke 43, terdapat perintah untuk menjalankan perintah display dengan argument top. Pada baris ke 44 terdapat perintah untuk menampilkan pesan “Mau mengeluarkan elemen? (y/n)” Kemudian terdapat perintah agar user memasukkan nilai y atau n yang nantinya nilai tersebut akan dimasukkan ke dalam variabel ch. Kemudian pada baris ke 46 terdapat percabangan dengan kondisi jika ch = y atau Y, maka tampilkan elemen yang dikeluarkan dan jalankan fungsi pop. Kemudian pada baris ke 52 terdapat pendefinisian kondisi while dari perulangan while-do sebelumnya, yaitu jika nilai ch = y atau Y. Pada baris ke 54 terdapat return 0 yang menyatakan hasil keluaran dari fungsi main() bahwa program berakhir dengan normal. Pada baris ke 57 terdapat inisialisasi dari node create_new_node dengan parameter int x. Pada baris ke 59 terdapat inisialisasi nilai ptr dengan nilai dari new node. Pada baris ke 60 terdapat assignment nilai ptr info dengan x. Pada baris ke 61 terdapat assignment nilai ptr nextde dengan NULL. Fungsi kemudian mengembalikan nilai ptr. Pada baris ke 65 terdapat inisialisasi fungsi push dengan parameter node n. Pada baris ke 67, terdapat percabangan if dengan kondisi jika top = Null maka inisialisasi nilai top = n, jika tidak terpenuhi, maka inisialisasi nilai save = top, top = n dan n->next = save. Pada baris ke 79 terdapat inisialisasi dari fungsi pop. Pada baris ke 81, di dalam fungsi tersebut, terdapat percabangan dengan kondisi jika top = NULL maka tampilkan pesan “Underflow…!!..Keluar dari program…” dan exit(1); jika kondisi tidak terpenuhi, maka inisialisasi nilai ptr dengan top dan nilai top dengan nilai dari top->next. Kemudian pada baris ke 93 terdapat inisialisasi fungsi display dengan parameter node n. Pada baris ke 95 terdapat perulangan while dengan kondisi n != NULL), maka program akan menampilkan nilai dari n->info dan kemudian menginisialisasi nilai dari n dengan n->next. Kemudian pada baris ke 98 terdapat perintah untuk menampilkan pesan !! dan berpindah baris.

5.5.b Output Stack dengan Single Linked List Pop

```
Pop.cpp -o 5-5stackLinkedListPop } ; if ($?) {
.\5-5stackLinkedListPop }
Masukkan elemen: 12
Elemen dimasukkan...
Elemen berhasil dimasukkan...
Stack: 12 -> !!
Mau menambahkan elemen ? (y/n) y
Masukkan elemen: 90
Elemen dimasukkan...
Elemen berhasil dimasukkan...
Stack: 90 -> 12 -> !!
Mau menambahkan elemen ? (y/n) y
Masukkan elemen: 34
Elemen dimasukkan...
Elemen berhasil dimasukkan...
Stack: 34 -> 90 -> 12 -> !!
Mau menambahkan elemen ? (y/n) n
Stack:
34 -> 90 -> 12 -> !!
Mau mengeluarkan elemen? (y/n) y
Elemen yang dikeluarkan: 34

Stack:
90 -> 12 -> !!
Mau mengeluarkan elemen? (y/n) y
Elemen yang dikeluarkan: 90

Stack:
12 -> !!
Mau mengeluarkan elemen? (y/n) y
Elemen yang dikeluarkan: 12

Stack:
!!
Mau mengeluarkan elemen? (y/n) y
Elemen yang dikeluarkan:
```

Gambar 5.5.b Output Stack dengan Single Linked List Pop

Berdasarkan gambar 5.5.b yang merupakan output dari single linked list pop, dapat dapat dilihat bahwa pertama program meminta user untuk memasukkan nilai untuk elemen pertama dari stack. Ini sesuai dengan perintah pada baris ke 23. Pada percobaan ini user memasukkan nilai 12. Kemudian program menampilkan pesan "Membuat node baru…". Pada saat ini program sedang menjalankan fungsi create_new_node dan kemudian akan memasukkan returnnya ke variabel newptr. Kemudian, berdasarkan percabangan pada baris ke 25, karena newptr tidak bernilai

NULL, maka program menampilkan pesan “Elemen dimasukkan…” dan kemudian menjalankan fungsi push dengan argument newptr. Kemudian program menampilkan pesan “Elemen berhasil dimasukkan…” dan menampilkan stack yang telah dibuat, yaitu 12 -> !!. Kemudian program kembali menanyakan apakah user ingin menambahkan elemen lagi dan meminta input kepada user. User kemudian memasukkan nilai y yang berarti user ingin membuat elemen baru. Input tersebut kemudian dimasukkan ke dalam variabel ch dan dijadikan ajuan untuk menjalankan perulangan while pada baris ke 20 yang berfungsi untuk menambahkan elemen pada stack. Karena ch bernilai y, maka program kembali meminta user untuk memasukkan nilai untuk elemen kedua pada stack. Ini sesuai dengan perintah pada baris ke 23. Pada percobaan ini user memasukkan nilai 90. Kemudian program menampilkan pesan “Membuat node baru…”. Pada saat ini program sedang menjalankan fungsi create_new_node dan kemudian akan memasukkan returnnya ke variabel newptr. Kemudian, berdasarkan percabangan pada baris ke 25, karena newptr tidak bernilai NULL, maka program menampilkan pesan “Elemen dimasukkan…” dan kemudian menjalankan fungsi push dengan argument newptr. Kemudian program menampilkan pesan “Elemen berhasil dimasukkan…” dan menampilkan stack yang telah dibuat, yaitu 90 -> 12 -> !!. Kemudian program kembali menanyakan apakah user ingin menambahkan elemen lagi dan meminta input kepada user. User kemudian memasukkan nilai y yang berarti user ingin membuat elemen baru. Karena ch bernilai y, maka program kembali meminta user untuk memasukkan nilai untuk elemen kedua pada stack. Ini sesuai dengan perintah pada baris ke 23. Pada percobaan ini user memasukkan nilai 34. Kemudian program menampilkan pesan “Membuat node baru…”. Pada saat ini program sedang menjalankan fungsi create_new_node dan kemudian akan memasukkan returnnya ke variabel newptr. Kemudian, berdasarkan percabangan pada baris ke 25, karena newptr tidak bernilai NULL, maka program menampilkan pesan “Elemen dimasukkan…” dan kemudian menjalankan fungsi push dengan argument newptr. Kemudian program menampilkan pesan “Elemen berhasil dimasukkan…” dan menampilkan stack yang telah dibuat, yaitu 34 -> 90 -> 12 -> !!. Kemudian program kembali menanyakan apakah user ingin menambahkan elemen lagi dan meminta input kepada user. User kemudian memasukkan nilai n yang berarti user tidak lagi

membuat elemen baru. Karena ch bernilai n, maka program keluar dari perulangan while pada baris ke 20 tersebut dan kemudian program berpindah ke perulangan while-do pada baris ke 40 yang berfungsi untuk menghapus elemen pada stack. Kemudian program menampilkan stack yang telah dibuat dengan memanggil fungsi display, yaitu 34 -> 90 -> 12 -> !!. Berikutnya, program menanyakan kepada user apakah ingin mengeluarkan elemen. Kemudian user memasukkan nilai y yang berarti user ingin mengeluarkan satu elemen paling atas pada stack dan kemudian input tersebut dimasukkan ke dalam variabel ch. Kemudian pada percabangan if di 46, karena ch bernilai y, maka program menampilkan elemen yang dikeluarkan, yaitu top-> info yang bernilai 34 dan kemudian menjalankan fungsi pop. Kemudian setelah itu program menampilkan stack yang telah dibuat dengan memanggil fungsi display, yaitu 90 -> 12 -> !!. Berikutnya, program menanyakan kepada user apakah ingin mengeluarkan elemen. Kemudian user memasukkan nilai y yang berarti user ingin mengeluarkan satu elemen paling atas pada stack dan kemudian input tersebut dimasukkan ke dalam variabel ch. Kemudian pada percabangan if di 46, karena ch bernilai y, maka program menampilkan elemen yang dikeluarkan, yaitu top-> info yang bernilai 90 dan kemudian menjalankan fungsi pop. Kemudian setelah itu program menampilkan stack yang telah dibuat dengan memanggil fungsi display, yaitu 12 -> !!. Berikutnya, program menanyakan kepada user apakah ingin mengeluarkan elemen. Kemudian user memasukkan nilai y yang berarti user ingin mengeluarkan satu elemen paling atas pada stack dan kemudian input tersebut dimasukkan ke dalam variabel ch. Kemudian pada percabangan if di 46, karena ch bernilai y, maka program menampilkan elemen yang dikeluarkan, yaitu top-> info yang bernilai 12 dan kemudian menjalankan fungsi pop. Kemudian setelah itu program menampilkan stack yang telah dibuat dengan memanggil fungsi display. Karena stack kosong, maka program hanya menampilkan pesan "!!". Berikutnya, program menanyakan kepada user apakah ingin mengeluarkan elemen. Pada titik ini, seharusnya program tidak dapat mengeluarkan stack lagi karena stack sudah kosong. Kemudian user memasukkan nilai y yang berarti user ingin mengeluarkan satu elemen paling atas pada stack dan kemudian input tersebut dimasukkan ke dalam variabel ch. Kemudian pada percabangan if di 46, karena ch bernilai y, maka program menampilkan elemen yang dikeluarkan, yaitu top-> info. Karena stack

sudah kosong, maka program tidak menampilkan apapun dan mengalami error dan akhirnya terjadi timeout sehingga program selesai dan fungsi pop tidak dijalankan.

VI. KESIMPULAN

Adapun kesimpulan dari percobaan ini adalah sebagai berikut :

1. Berdasarkan percobaan 5.1 Single Stack dengan Struct, dapat disimpulkan bahwa stack menganut prinsip Last In First Out, yaitu elemen yang terakhir ditambahkan adalah elemen pertama yang dikeluarkan.
2. Berdasarkan percobaan 5.1 Single Stack dengan Struct, dapat disimpulkan bahwa pada program tersebut digunakan library <cstring> agar dapat menggunakan fungsi strcpy pada baris ke 88 yang berfungsi untuk menyalin suatu string dari variabel asal ke variabel lain. Sintak dari fungsi strcpy adalah

   Strcpy(char *tujuan, const char *asal)
3. Berdasarkan percobaan 5.2 Stack dengan Array Push dan 5.3 Stack dengan Array Pop, dapat disimpulkan bahwa stack dapat direpresentasikan menggunakan array, meskipun jumlah elemen maksimal pada array bersifat statis. Jika data yang ditambahkan melebihi panjang array, maka akan terjadi overflow, yaitu keadaan dimana stack sudah tidak mampu menampung data.
4. Berdasarkan percobaan 5.4 Stack dengan Single Linked List Push dan 5.5 Stack dengan Single Linked List Pop, dapat disimpulkan bahwa stack dapat direpresentasikan menggunakan linked list, karena prinsip stack mampu diinisialisasikan menggunakan bentuk struktur linked list.
5. Berdasarkan percobaan 5.4 Stack dengan Single Linked List Pop, dapat disimpulkan bahwa untuk membuat elemen baru, program akan memanggil fungsi create_new_node dengan argument data yang user masukkan. Kemudian elemen yang dibuat tersebut akan ditempatkan pada bagian atas stack dan menjadi elemen top baru.

DAFTAR PUSTAKA

Erliansyah Nasution dan Indra Yatini B.2005. Algoritma dan Struktur Data. Graha Ilmu. Yogyakarta.

Sjukani, Moh. 2010. (Algoritma Dan Struktur Data I) Dengan C, C++ dan Java. Jakarta: Mitra Wacana Media

Muhammad, M.A. (018. Struktur Data. Modul Struktur Data Unila PSTI. 2 (2) : 1 - 2.

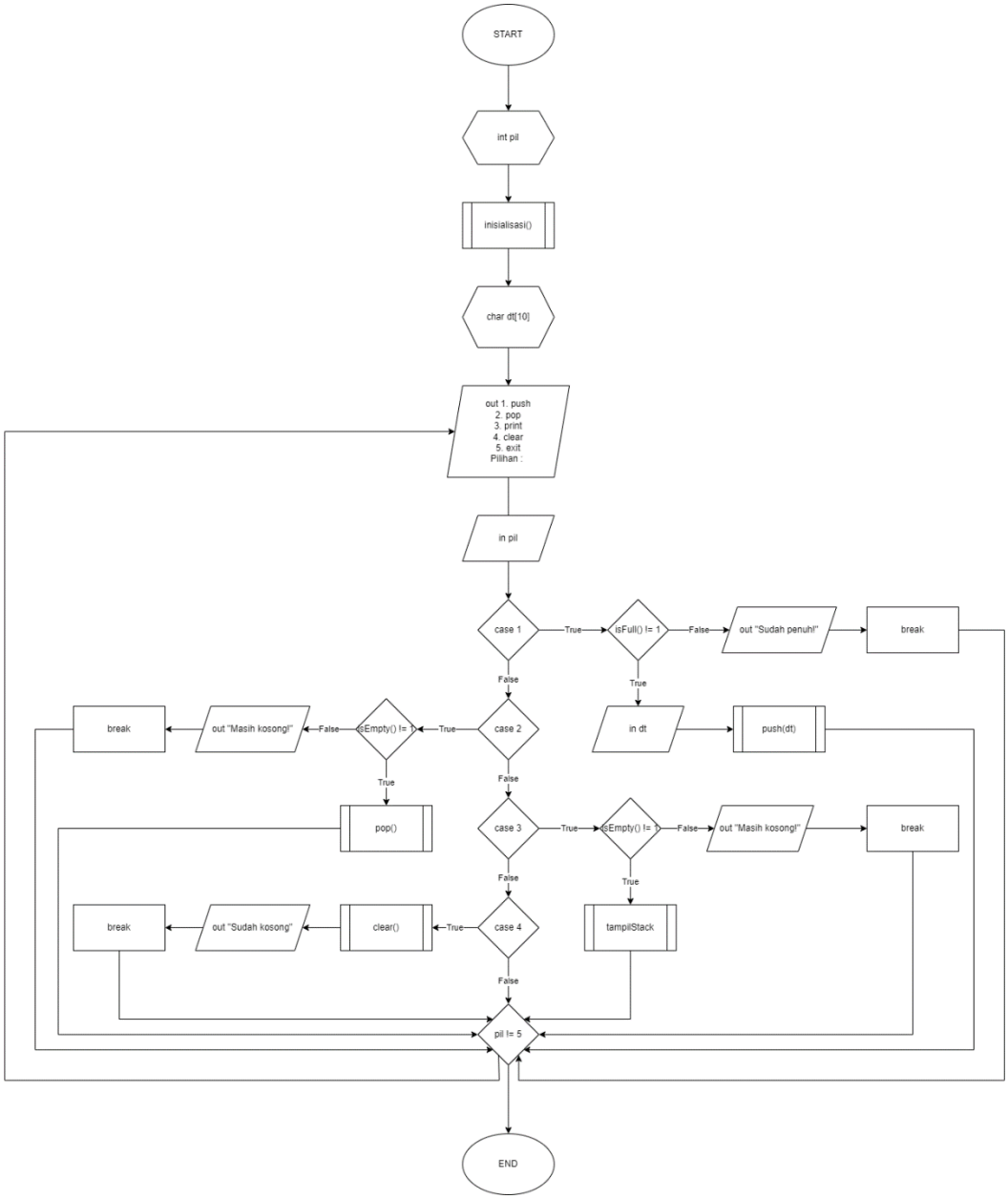
Muhammad, Meizano Ardhi. 2018. Modul Praktikum Struktur Data. Lampung Jurusan Teknik Elektro Fakultas Teknik Universitas Lampung.

Ulum, M. Bahrul Ulum.2018. Modul Praktikum Struktur Data. Jakarta Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Esa Unggul.

TUGAS AKHIR

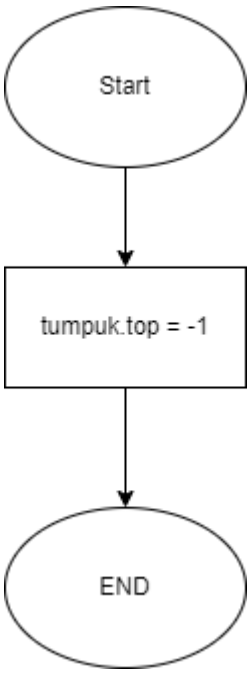
1. Flowchart percobaan 5-1 Single Stack dengan Array

1.1 Flowchart Fungsi main Percobaan 5.1 Single Stack dengan Array



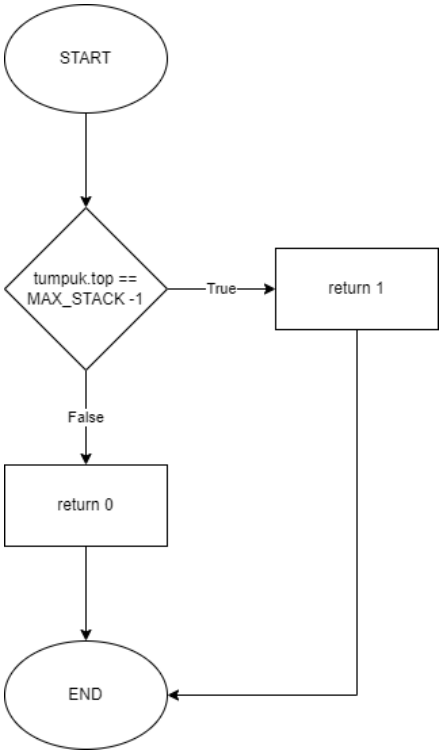
Gambar 1 Flowchart Fungsi main Percobaan 5.1 Single Stack dengan Array

1.2 Flowchart Fungsi inisialisasi Percobaan 5.1 Single Stack dengan Array



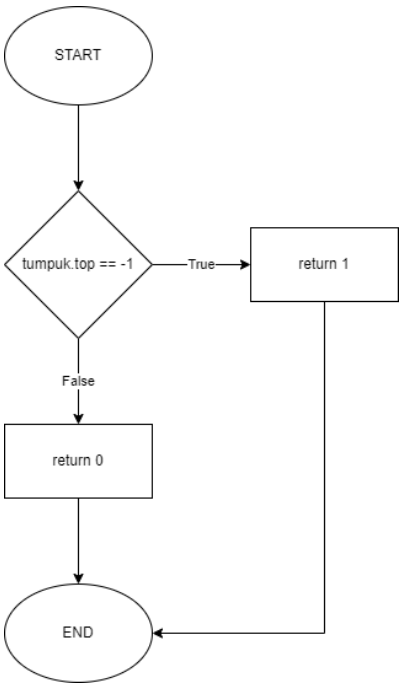
Gambar 2 Flowchart Fungsi inisialisasi Percobaan 5.1 Single Stack dengan Array

1.3 Flowchart Fungsi isFull Percobaan 5.1 Single Stack dengan Array



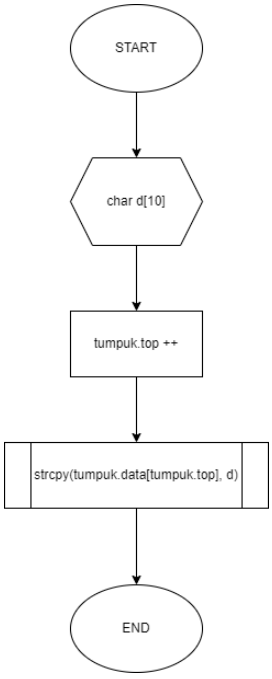
Gambar 3 Flowchart Fungsi isFull Percobaan 5.1 Single Stack dengan Array

1.4 Flowchart Fungsi isEmpty Percobaan 5.1 Single Stack dengan Array



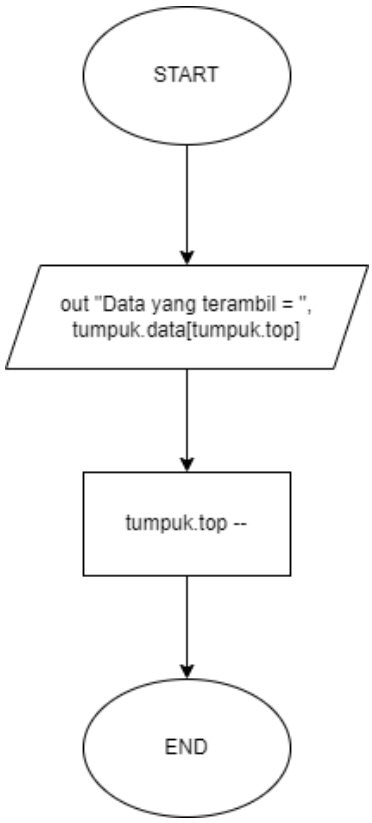
Gambar 4 Flowchart Fungsi isEmpty Percobaan 5.1 Single Stack dengan Array

1.5 Flowchart Fungsi push Percobaan 5.1 Single Stack dengan Array



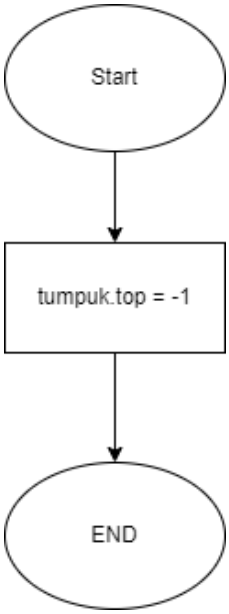
Gambar 5 Flowchart Fungsi push Percobaan 5.1 Single Stack dengan Array

1.6 Flowchart Fungsi pop Percobaan 5.1 Single Stack dengan Array



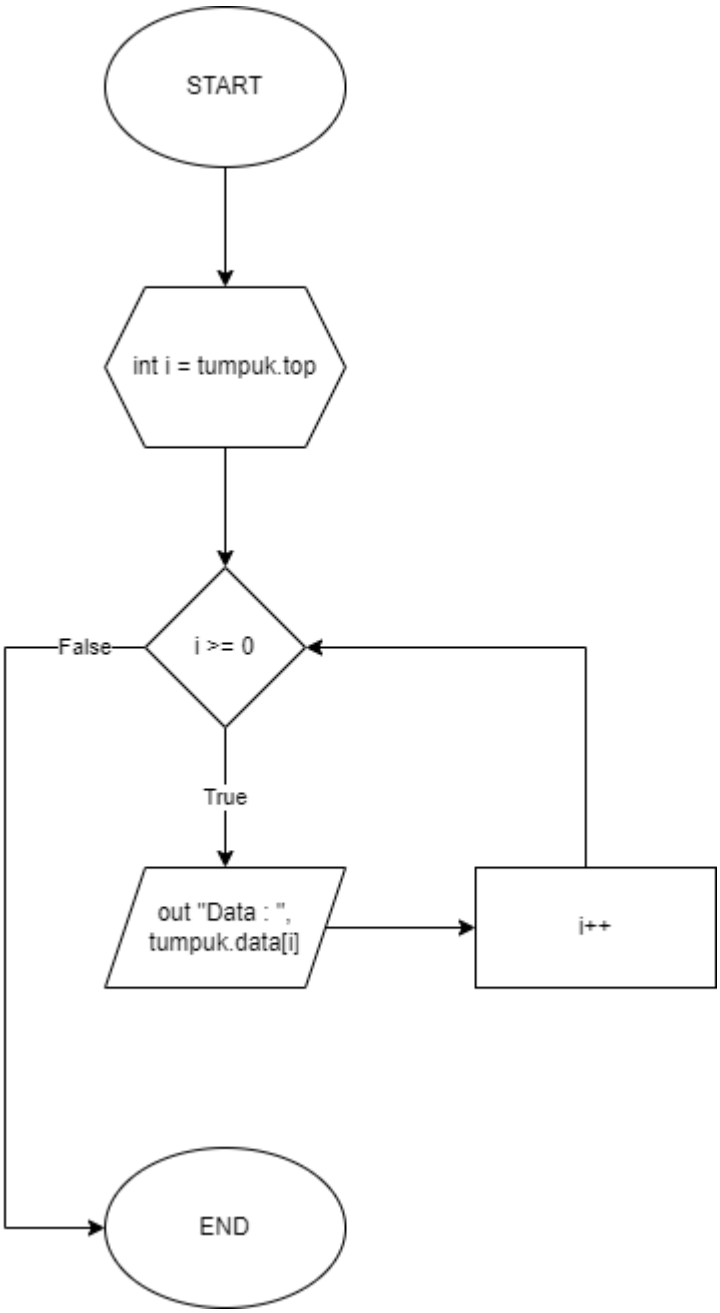
Gambar 6 Flowchart Fungsi pop Percobaan 5.1 Single Stack dengan Array

1.7 Flowchart Fungsi clear Percobaan 5.1 Single Stack dengan Array



Gambar 7 Flowchart Fungsi clear Percobaan 5.1 Single Stack dengan Array

1.8 Flowchart Fungsi tampilStack Percobaan 5.1 Single Stack dengan Array



Gambar 8 Flowchart Fungsi tampilStack Percobaan 5.1 Single Stack dengan Array

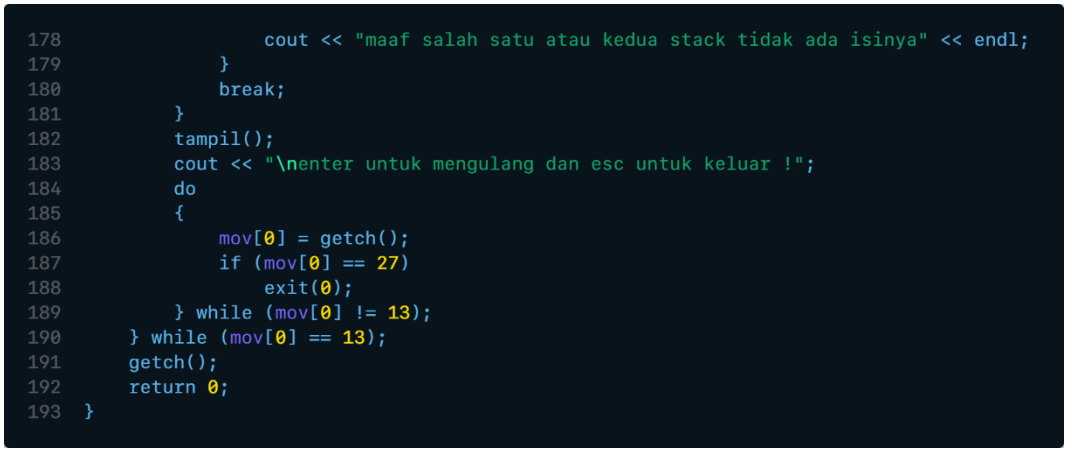
2. Double Stack Array

2.1 Source Code Double Stack Array

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

char P[] = {'>', ' ', ' ', ' ', ' ', ' '};
int S[n], mov[2], X, Y, pil = 0;
int *top1, *top2, *dasar1, *dasar2, *helpI;

// utama
void awal()
{
    top1 = &S[-1];
    top2 = &S[n];
    dasar1 = &S[-1];
    dasar2 = &S[n];
    helpI = &S[-1];
}

void push1(int x)
{
    top1 = top1 + 1;
    *top1 = x;
}

void push2(int y)
{
    top2 = top2 - 1;
    *top2 = y;
}

void pop1()
{
    X = *top1;
    *top1 = 0;
    top1 = top1 - 1;
}

void pop2()
{
    Y = *top2;
    *top2 = 0;
    top2 = top2 + 1;
}

int BisaDiisi(int k)
{
    if (top2 - top1 > k)
        return 1;
    else
        return 0;
}

int BisaDiambil1()
{
    if (top1 > dasar1)
        return 1;
    else
```

```
        return 0;
}

int BisaDiambil2()
{
    if (top2 < dasar2)
        return 1;
    else
        return 0;
}

void tampil()
{
    cout << "\n================ data menjadi ==================" << endl;
    while (helpI != (dasar2 - 1))
    {
        helpI++;
        cout << *helpI << " ";
    }
    cout << "\n=======================================================" << endl;
    helpI = &S[-1];
}

// tambahan
void tampilMenu()
{
    cout << "=======================================================" << endl;
    cout << "1. isi stack 1 (kiri)" << endl;
    cout << "2. isi stack 2 (kanan)" << endl;
    cout << "3. isi kedua stack" << endl;
    cout << "4. pop stack 1" << endl;
    cout << "5. pop stack 2" << endl;
    cout << "6. pop kedua stack" << endl;
    cout << "=======================================================" << endl;
    cout << "pilihan anda : ";
    cin >> pil;
}

int main()
{
    awal();
    do
    {
        tampilMenu();
        cout << endl;
        // cout<<"nilai pil = "<<pil<<endl;
        switch (pil)
        {
        case 1:
            if (BisaDiisi(1))
            {
                cout << "masukkan bilangan = ";
                cin >> X;
                push1(X);
            }
            else
            {
                cout << "maaf tidak ada tempat untuk push";
            }
```

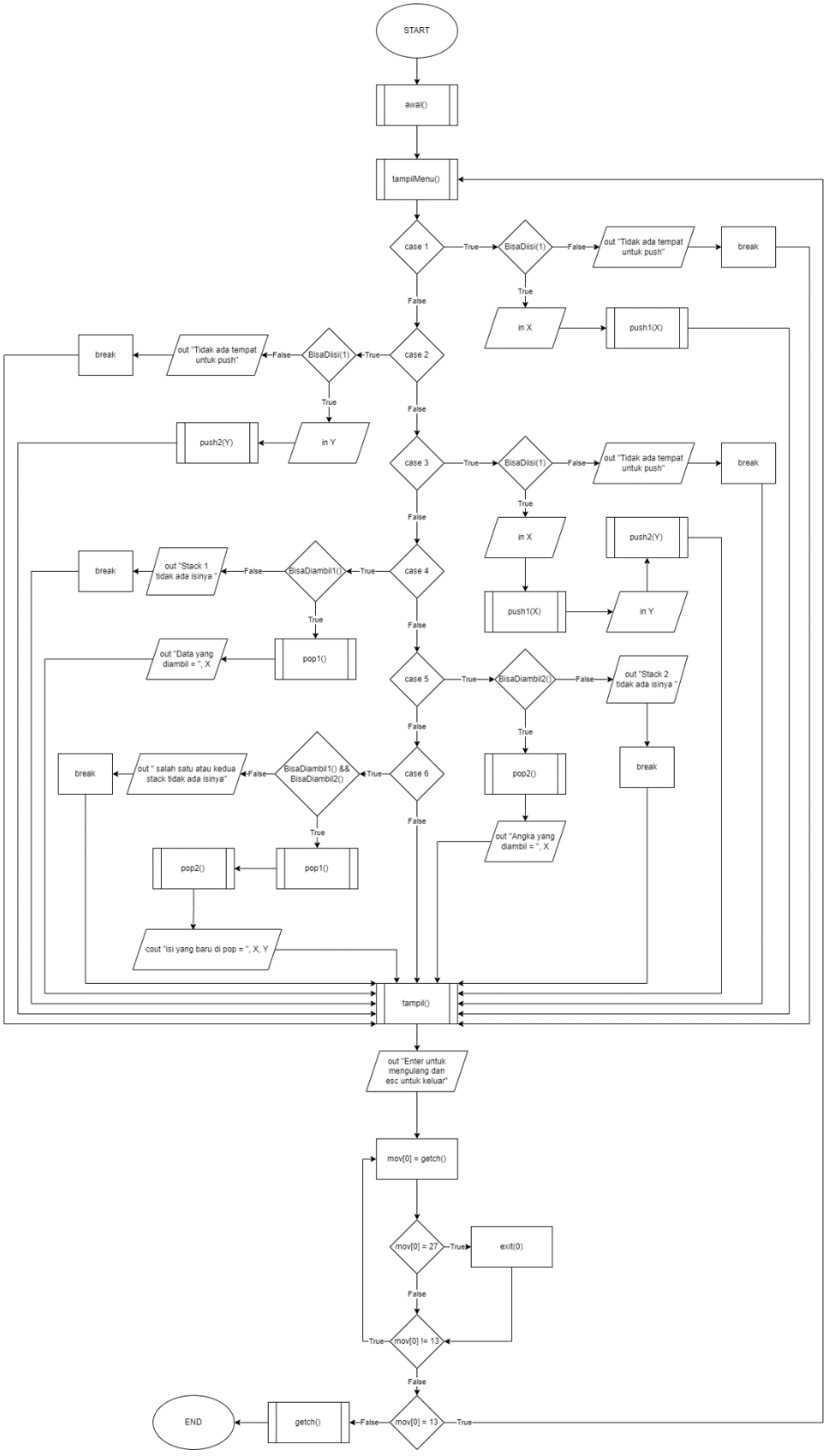
```
    break;
case 2:
    if (BisaDiisi(1))
    {
        cout << "masukkan bilangan = ";
        cin >> Y;
        push2(Y);
    }
    else
    {
        cout << "maaf tidak ada tempat untuk push";
    }
    break;
case 3:
    if (BisaDiisi(2))
    {
        cout << "masukkan bilangan 1= ";
        cin >> X;
        push1(X);
        cout << "masukkan bilangan 2= ";
        cin >> Y;
        push2(Y);
    }
    else
    {
        cout << "maaf ruang tidak cukup";
    }
    break;
case 4:
    if (BisaDiambil1())
    {
        pop1();
        cout << "data yang diambil = " << X << endl;
    }
    else
    {
        cout << "maaf stack 1 tidak ada isinya" << endl;
    }
    break;
case 5:
    if (BisaDiambil2())
    {
        pop2();
        cout << "angka yang diambil = " << Y << endl;
    }
    else
    {
        cout << "maaf stack 2 tidak ada isinya" << endl;
    }
    break;
case 6:
    if (BisaDiambil1() && BisaDiambil2())
    {
        pop1();
        pop2();
        cout << "isi yang baru di pop = " << X << " dan " << Y << endl;
    }
    else
    {
```



```
178                 cout << "maaf salah satu atau kedua stack tidak ada isinya" << endl;
179             }
180             break;
181         }
182         tampil();
183         cout << "\nenter untuk mengulang dan esc untuk keluar !";
184         do
185         {
186             mov[0] = getch();
187             if (mov[0] == 27)
188                 exit(0);
189         } while (mov[0] != 13);
190     } while (mov[0] == 13);
191     getch();
192     return 0;
193 }
```

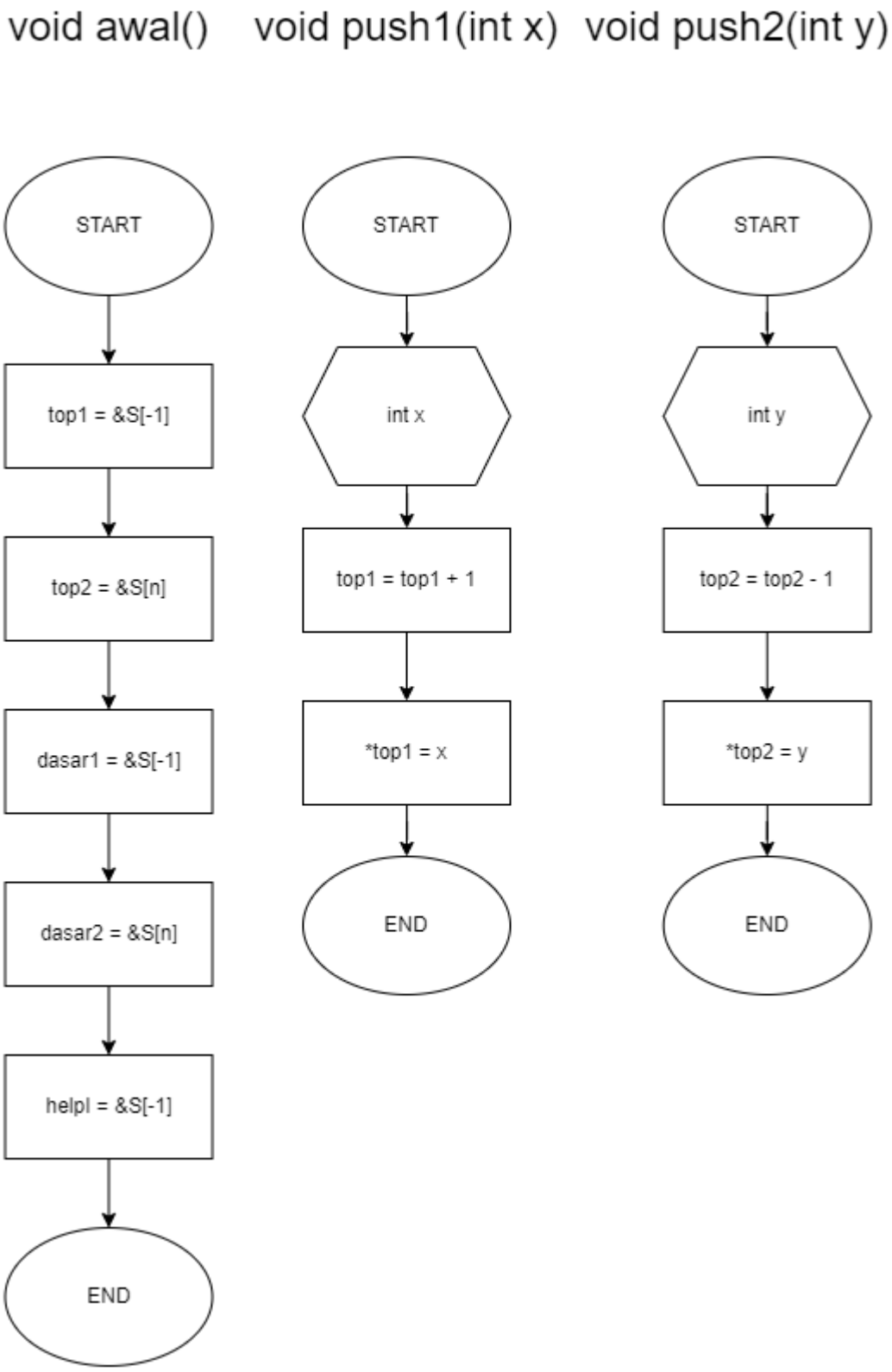
Gambar 9 Source Code Program Double Stack Array

2.2 Flowchart Fungsi main Double Stack Array



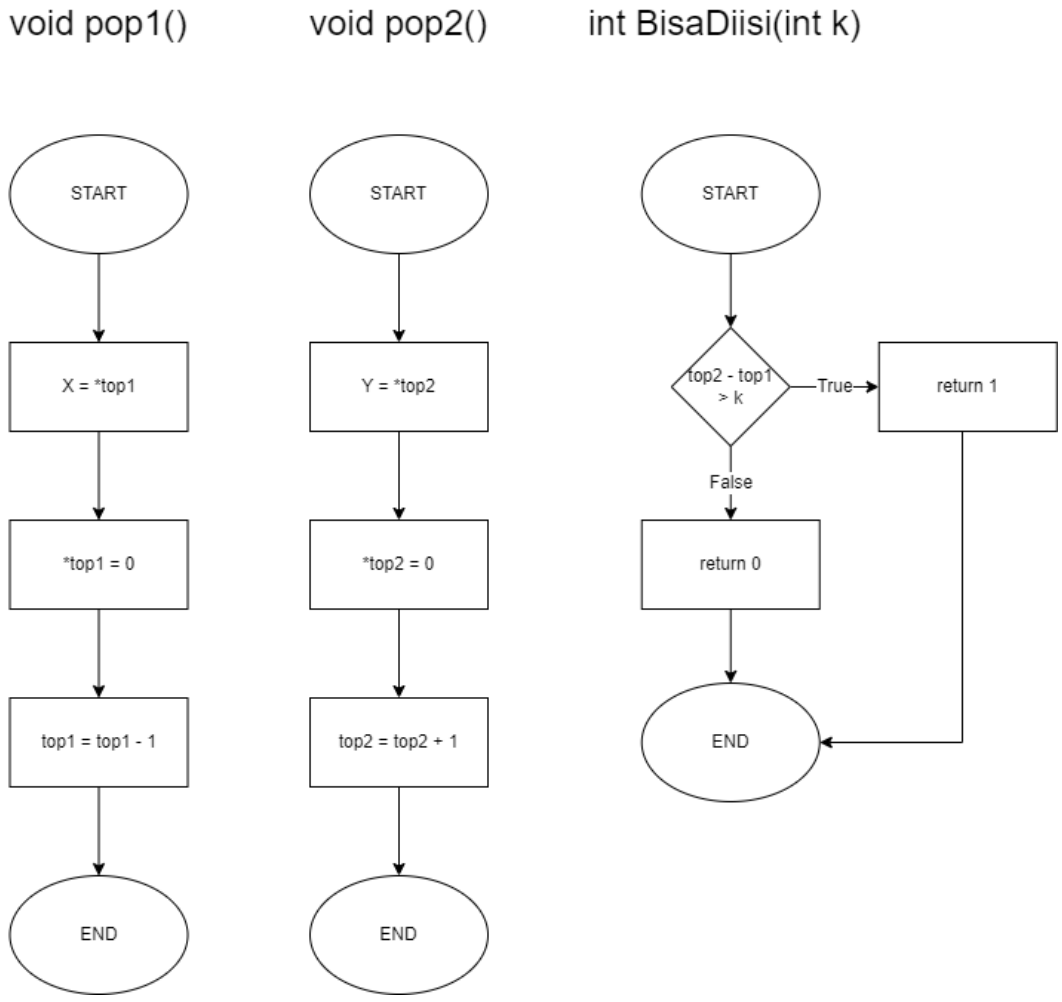
Gambar 10 Flowchart Fungsi main Double Stack Array

2.3 Flowchart Fungsi awal, push1, dan push2 Double Stack Array



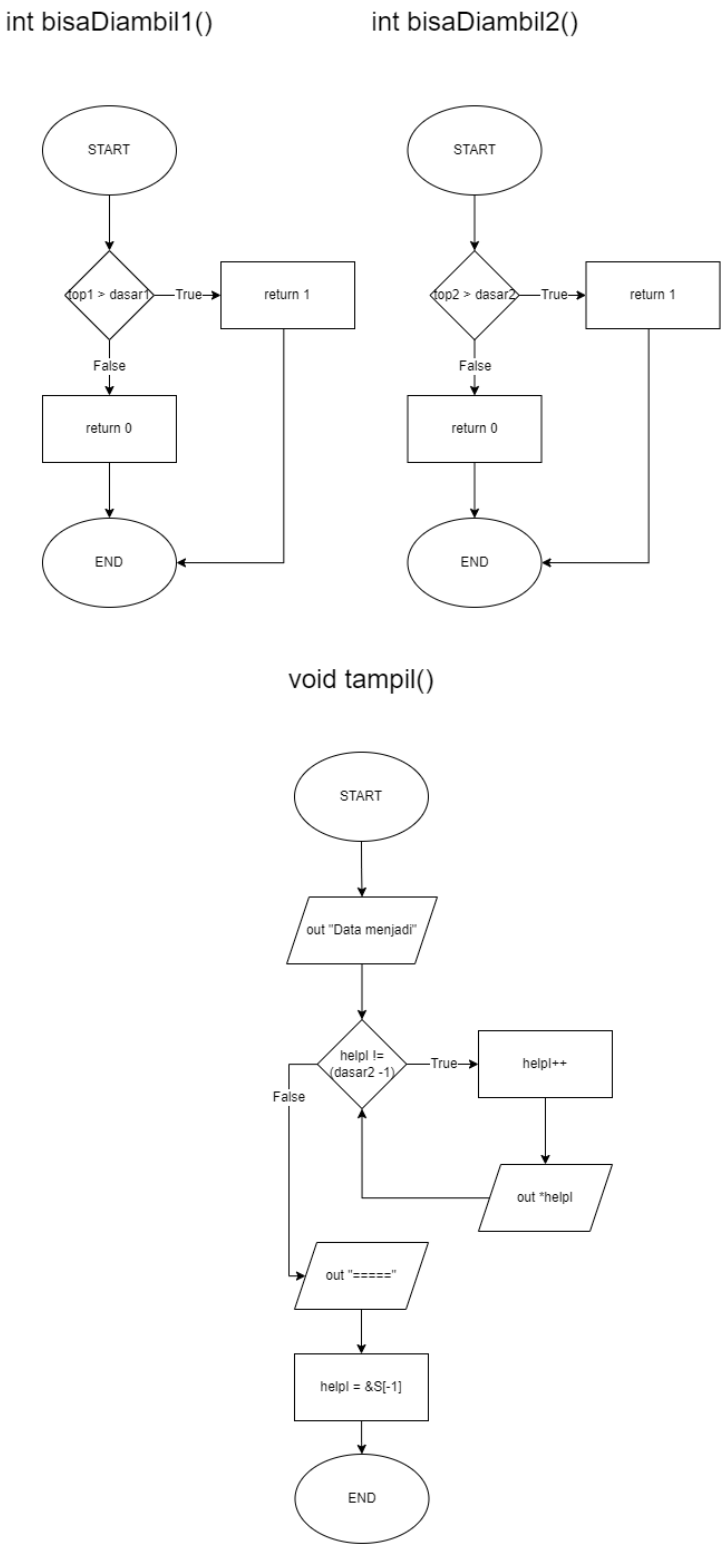
Gambar 11 Flowchart Fungsi awal, push1, dan push 2 Double Stack Array

2.4 Flowchart Fungsi pop1, pop2, dan bisaDiisi Double Stack Array



Gambar 12 Flowchart Fungsi pop1, pop2, dan bisaDiisi Double Stack Array

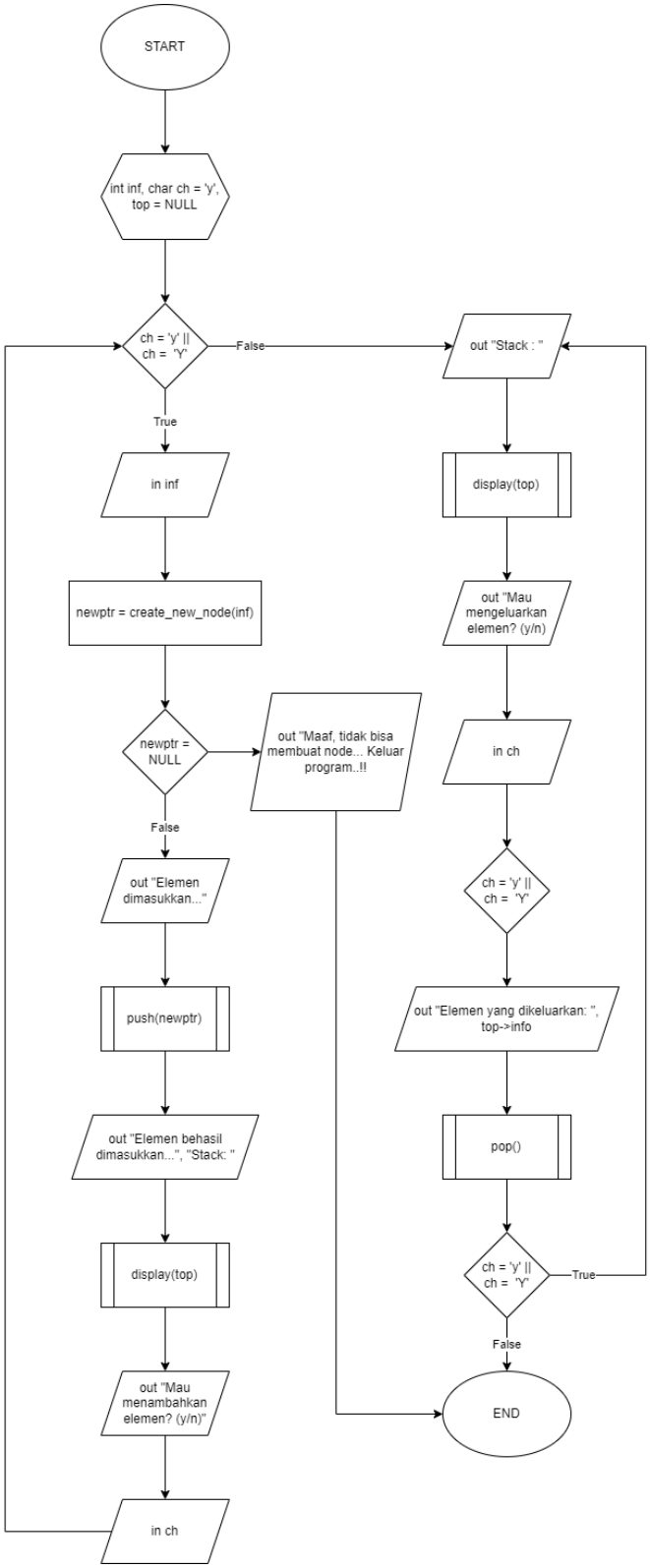
2.5 Flowchart Fungsi BisaDiambil1, BisaDiambil2, dan tampil Double Stack Array



Gambar 13 Flowchart Fungsi BisaDiambil1, BisaDiambil2, dan tampil Double Stack Array

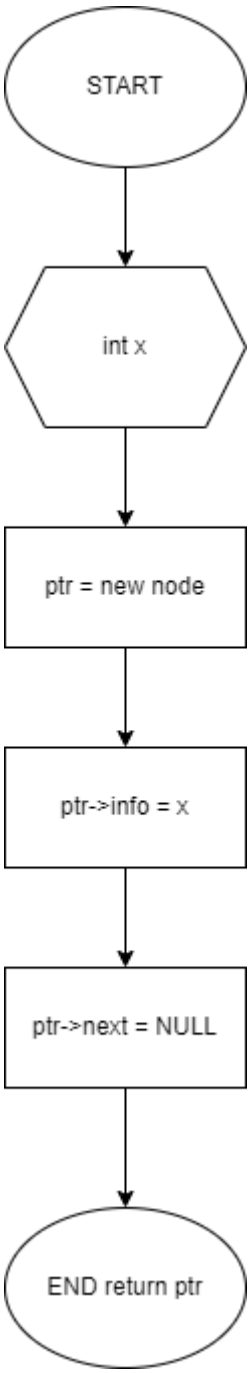
3. Flowchart percobaan 5-5 Single Stack Linked List

3.1 Flowchart Fungsi main Percobaan 5.5 Single Stack Linked List



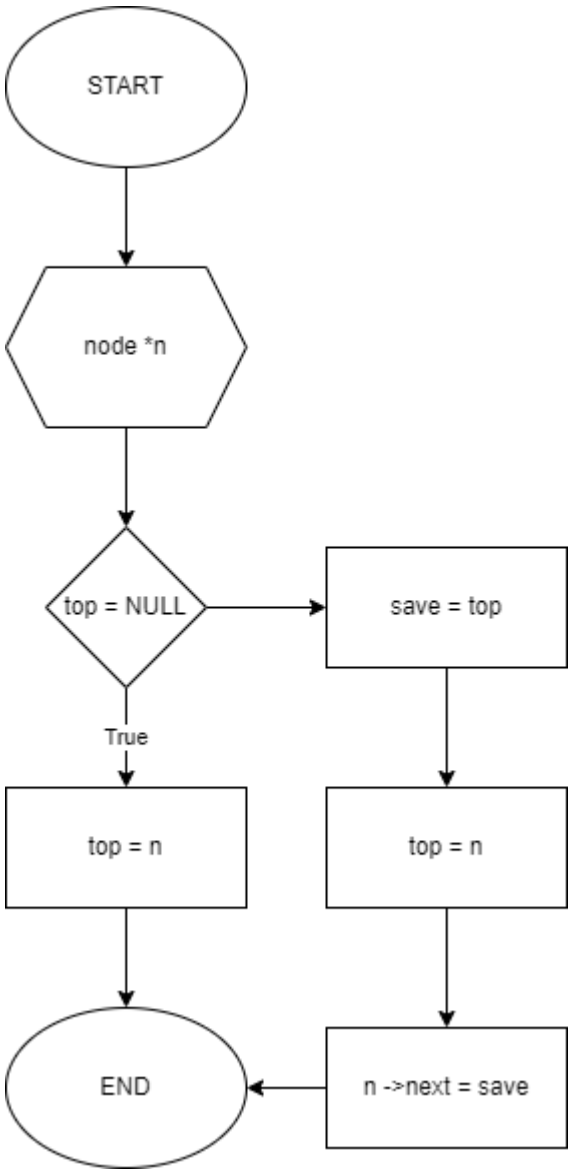
Gambar 14 Flowchart Fungsi main Percobaan 5.5 Single Stack Linked List

3.2 Flowchart Fungsi create_new_node Percobaan 5.5 Single Stack Linked List



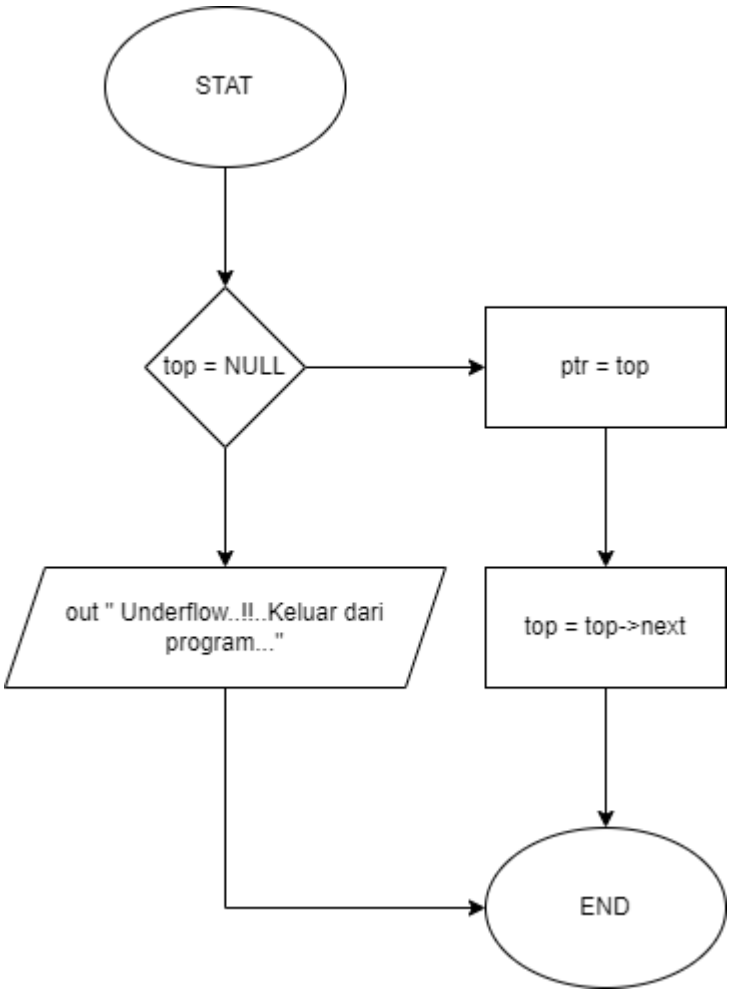
Gambar 15 Flowchart Fungsi create_new_node Percobaan 5.5 Single Stack Linked List

3.3 Flowchart Fungsi push Percobaan 5.5 Single Stack Linked List

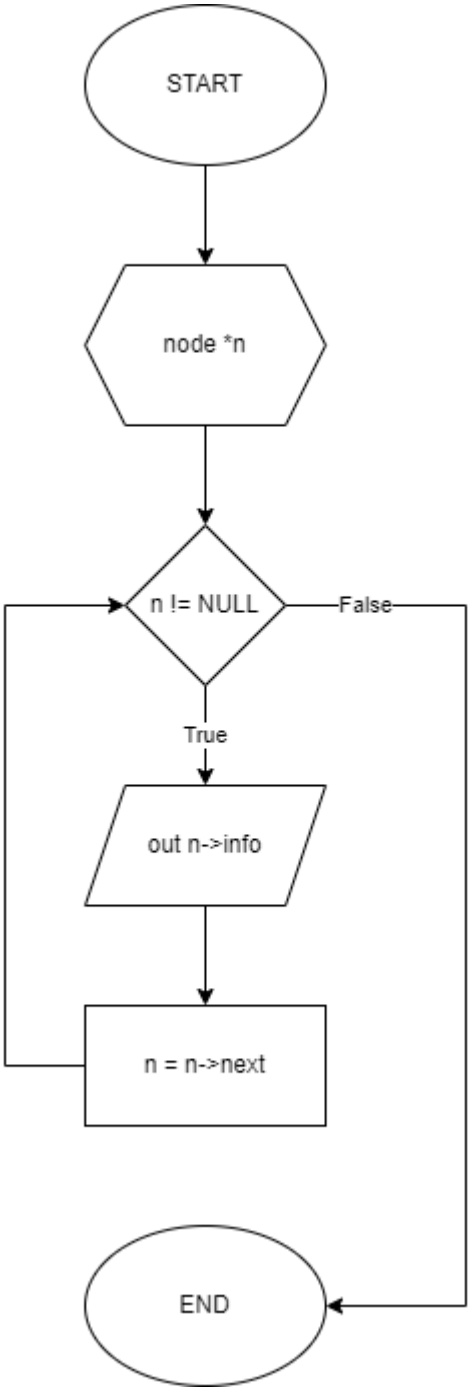


Gambar 16 Flowchart Fungsi push Percobaan 5.5 Single Stack Linked List

3.4 Flowchart Fungsi pop Percobaan 5.5 Stack Linked List



Gambar 17 Flowchart Fungsi pop Percobaan 5.5 Stack Linked List



Gambar 18 Flowchart Fungsi display Percobaan 5.5