MODUL IV

STACK

A. TUJUAN

1. Memahami terminologi yang terkait dengan struktur data stack.
2. Memahami operasi-operasi yang ada dalam stack.
3. Dapat mengidentifikasi permasalahan-permasalahan pemrograman yang harus diselesaikan dengan menggunakan stack, sekaligus menyelesaikannya.

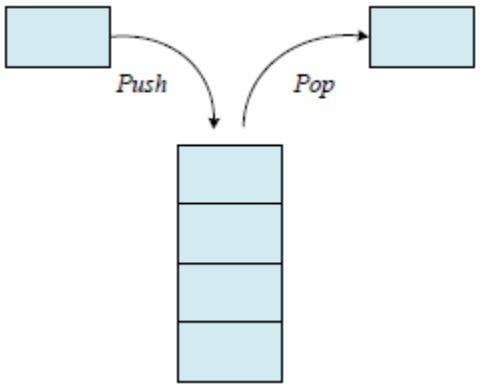
B. PETUNJUK

1. Awali setiap aktivitas anda dengan doa, agar anda lancar dalam belajar.
2. Pahami tujuan, dasar teori, dan latihan-latihan praktikum dengan baik.
3. Kerjakan tugas-tugas praktikum dengan baik, jujur, dan sabar.
4. Tanyakan kepada asisten apabila ada hal-hal yang kurang jelas.

C. DASAR TEORI

1. Pengertian Stack

Stack merupakan sebuah kumpulan data yang diletakkan di atas data lainya, seperti sebuah tumpukan. Dengan demikian, stack merupakan salah satu struktur data yang menerapkan prinsip LIFO (Last In First Out). Dimana elemen yang terakhir disimpan dalam stack, menjadi elemen yang pertama diambil. Untuk meletakkan sebuah elemen pada bagian atas dari stack, maka dilakukan operasi push. Sedangkan untuk memindahkan sebuah elemen dari tempat atas tersebut dalam sebuah stack, maka dilakukan operasi pop.



Gambar 4.1 Ilustrasi sebuah stack

2. Operasi Dasar Pada Stack

- **Create**
  Merupakan operator yang berfungsi untuk membuat sebuah stack kosong.

```
struct STACK {
      int top;
      float data[5];
};
float dta;
struct STACK stackbaru;
```

- **IsEmpty**
  Merupakan operator yang berfungsi untuk menentukan apakah suatu stack merupakan stack kosong. Tanda bahwa sebuah stack kosong adalah Top bernilai kurang dari nol (-1).

```
bool isempty() {
      if (stackbaru.top==1) return true;
      else return false;
}
```

- **IsFull**
  Merupakan operator yang digunakan untuk memeriksa apakah stack yang ada sudah penuh. Stack akan penuh jika puncak stack terletak tepat dibawah jumlah maksimum yang dapat ditampung stack (Top = MAX_STACK-1).

```
bool isfull() {
      if (stackbaru.top==maxstack) return
true;
      else return false;
}
```

- **Push**
  Merupakan operator yang berfungsi untuk menambahkan satu elemen ke dalam stack dan tidak dapat dilakukan jika stack dalam keadaan penuh.

```
void push(float dta) {
      if (isfull()==false) {
            puts("stack penuh");
} else {
      stackbaru.top++;
      stackbaru.data[top]=dta;
      }
}
```

- **Pop**
  Merupakan operator yang berfungsi untuk mengeluarkan satu elemen teratas dari dalam stack dengan syarat stack tidak dalam kondisi kosong.

```
void pop() {
      if (isempty()==false) {
            cout<<"data kosong";
} else {
      cout<<"data yang terambil :
"<<stackbaru.data[top]<<endl;
      stackbaru.top--;
      }
}
```

- **Clear**
  Fungsi yang digunakan untuk mengosongkan stack dengan cara mengeset Top dengan -1. Jika Top bernilai kurang dari nol maka stack dianggap kosong.

```
void clear () {
      top=-1
}
```

- **Retrieve**
  fungsi yang digunakan untuk melihat nilai yang berada pada posisi tumpukan teratas.

```
void print() {
      for (int i=0; i<=top; i++) {
            cout<<stackbaru.data[i]<<"
      ";
      }
}
```

3. Pointer Sebagai Penunjuk Stack

Selain menggunakan indeks, untuk menunjuk sebuah Top atau posisi teratas dari stack, dapat juga digunakan pointer sebagai berikut.

- **Membuat Stack Dengan Pointer**

```
int S[10], *Top, *BatasAtas
Top = &S[-1];
BatasAtas = &S[10];
```

- **Proses Push**

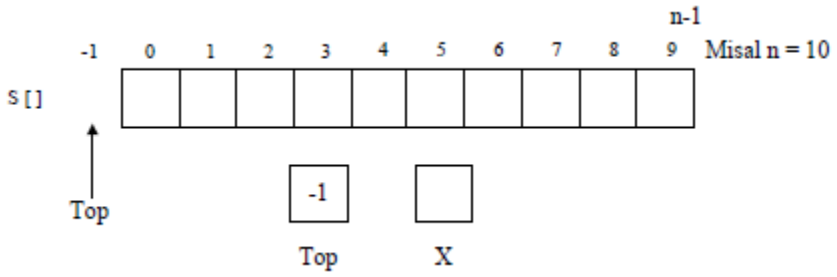
```
if (Top < BatasAtas)
Top++;
*Top = X;
else
printf("Stack Penuh");
```

- **Proses Pop**

```
if (Top > &A[-1])
X= *Top;
Top --;
else
printf("Stack Kosong");
```

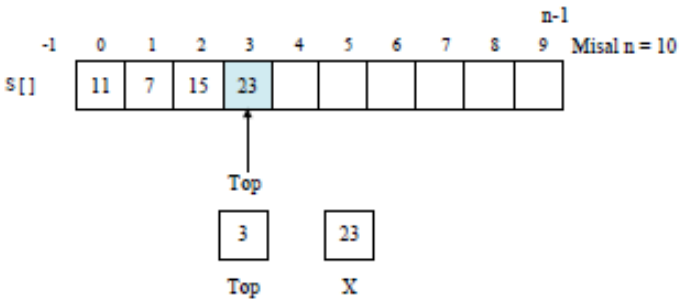
4. Representasi Proses Stack

Stack adalah salah satu dari contoh struktur data yang terdiri dari satu collection, yang juga menerapkan prinsip LIFO. Bila stack tersebut menggunakan array satu dimensi, maka stack tersebut dapat diilustrasikan sebagai berikut :



Gambar 4.2 Ilustrasi sebuah stack 1 Dimensi menggunakan indeks array

Pada gambar 4.2 diatas diilustrasikan ada sebuah indeks array yang masih kosong pada awal pembuatan stcak dimana n[10], variabel Top berada pada -1 yang menunjukkan indeks masih dalam keadaan kosong.



Gambar 4.3 Ilustrasi Stack ketika sudah diisi data

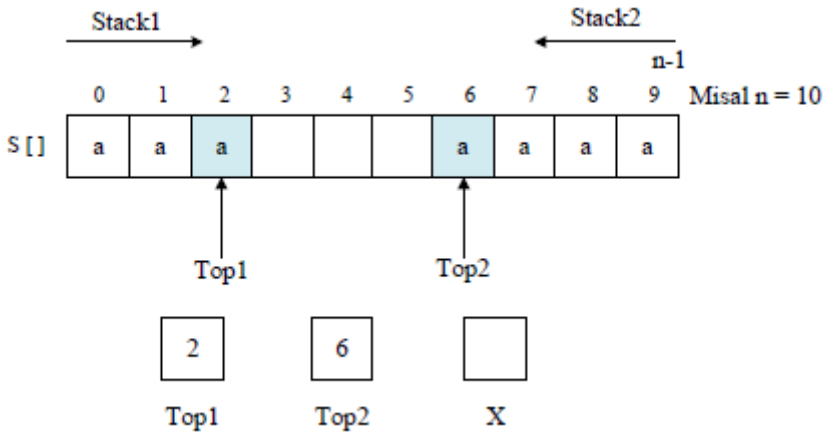
Pada gambar 4.3 adalah keadaan ketika stack sudah diisi data. Pada kondisi ini data pertama yang diinputkan adalah S[0]=11, data kedua S[1]=7, data ketiga S[2]=15, data keempat S[23]=23. Kondisi Top sudah berubah menjadi data yang terakhir diinputkan (data keempat) sehingga Top[3] X=23.

Variabel Top digunakan sebagai indeks untuk menunjuk nomor elemen array yang berisi nilai stack yang berada paling kanan atau Top, yang ditunjukan dengan angka 3. Variabel X bertipe integer digunakan sebagai perantara, dimana data yang akan disimpan kedalam stack harus berasal dari X. Demikian juga data yang baru diambil dari dalam stack harus diterima terlebih dahulu oleh variabel X, kemudian baru diberikan ke variabel lain untuk diolah.

Oleh karena itu, jika ada instruksi PUSH, maka data baru (yang diambil dari isi variabel X) akan disimpan dalam elemen S[4] sehingga indeks Top harus diarahkan ke posisi no.4. Artinya, Top maju terlebih dahulu satu langkah ke S[4], kemudian baru mengisi nilai pada S[4]. Sedangkan jika ada instruksi Pop, maka yang akan diambil adalah isi dari S[3] dan datanya akan disimpan terlebih dahulu dalam variabel X, kemudian indeks Top menjadi mundur satu langkah sehingga akan menunjuk S[2].

5. Double Stack

Double Stack atau Stack Ganda adalah dua stack yang berada dalam satu array. Satu array digunakan untuk dua stack dimana dasar Stack1 berada pada sisi indeks yang terkecil dan dasar Stack2 berada pada sisi indeks yang terbesar. Sama halnya dengan Single Stack, Double Stack juga menerapkan prinsip LIFO (Last in Firt Out).



Gambar 4.4 Ilustrasi double stack

Pada gambar 4.4 diatas adalah ilustrasi indeks array pada double stack. Padastack 1 kondisi data pertama yang diinputkan adalah S[0], data kedua S[1], data ketiga S[2] dan Top=data input terakhir S[2]. Sedangkan pada stack 2 data pertama adalah S[9], data kedua S[8], data ketiga S[7], data keempat S[6] dan Top=data input terakhir S[6].

6. Operasi Dasar Pada Double Stack

- **Inisialisasi**

  Proses awal adalah proses menyiapkan indeks penunjuk stack untuk pertama kali. Pada tahap ini ditetapkan Top1=-1 (sama seperti single stack) dan Top2=banyak jumlah data.

```
void AWAL (void)
{
Top1 = -1;
Top2 = n;
}
```

- **Is Empty**
  Sama dengan single stack, yaitu proses pengecekan stack dalam kondisi kosong

```
if(top1==-1) return true;
```

```
if(top2==n) return true;
```

- **Is Full**
  Sama dengan single stack, yaitu proses pengecekan stack dalam kondisi kosong

```
int full(void){
if(top1+1>=top2){
return true;
}
```

- **Push (Stack1 dan Stack2)**
  Proses mengisi data pada stack1 maupun stack2

```
void PUSH1 (void)
{
Top1 = Top1 + 1;
S[Top1] = X;
}
```

```
void PUSH2 (void)
{
Top2 = Top2 - 1;
S[Top2] = X;
}
```

- **Pop (Stack1 dan Stack2)**
  Proses mengambil data pada stack1 maupun stack2

```
void POP1 (void)
{
X = S[Top1];
Top1 = Top1 - 1;
}
```

```
void POP2 (void)
{
X = S[Top2];
Top2 = Top2 + 1;
}
```

D. LATIHAN

1. Program Single Stack

```
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <windows.h>
#define maxstack 4

using namespace std;
struct STACK {//Membuat jenis data abstrak 'STACK'
    int top;
    float data[4];
};
float dta;
struct STACK stackbaru;

void inisialisasi()
{
    stackbaru.top=-1;
}
bool isfull()//mengecek apakah stack kondisi penuh?
{
    if(stackbaru.top ==maxstack) return true;
    else return false;
}
bool isempty()//mengecek apakah stack kondisi kosong?
{
    if(stackbaru.top==-1) return true;
    else return false;
}

void push(float dta)//mengisi stack(menyimpan data di stack)
{
    if(isfull()==true)
    {
        puts("Maaf,stack penuh");
        getch();
    }
    else
    {
        stackbaru.top++;
        stackbaru.data[stackbaru.top]=dta;
    }
}

void pop()//mengambil isi satck
{
    if(isempty()==true)
    {
        cout<<"Data telah kosong!";
        getch();
    }
    else
    {
```

```
53          cout<<"data yang terambil adalah data ke : "<<stackbaru.data[stackbaru.top]<<endl;
54          stackbaru.top--;
55          getch();
56      }
57  }
58
59  void print()//mencetak stack
60  {
61      for(int i=0; i<=stackbaru.top; i++)
62      {cout<<stackbaru.data[i]<<"    ";}
63  }
64
65  void clear()//mengosongkan stack
66  {
67      stackbaru.top = -1;
68  }
69
70
71  int main()
72  {
73      inisialisasi();
74      char menu;
75      char uulang;
76      do{
77          system("cls");
78          printf("\t ------------------------------------------------\n");
79          printf("\t |                      STACK                   |\n");
80          printf("\t ------------------------------------------------\n");
81          printf("\n\ --> Menu STACK :\n\n");
82          puts("1. push stack");
83          puts("2. pop stack");
84          puts("3. cetak");
85          puts("4. bersihkan stack");
86          puts("5. exit");
87  cout<<"Menu pilihan anda : ";
88          cin>>menu;
89  if(menu =='2')
90          {
91
92
93
94  pop();
95              uulang ='y';
96          }
97          else if(menu =='1')
98          {
99              cout<<"data yang akan disimpan di stack: ";
100             cin>>dta;
101             push(dta);
102             uulang='y';
103         }
104         else if(menu =='3')
105         {
106             print();
107             cout<<"\nUlang?(y/t)";
108             cin>>uulang;
109         }
110         else if(menu=='4')
111         {
112             clear();
113             cout<<"\nUlang?(y/t)";
114             cin>>uulang;
115         }
116         else if(menu =='5')
117         {
118             exit(0);
119         }
120     }while(uulang =='Y' || uulang == 'y')  ;
121
122 }
123
```

2. Program Double Stack

```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define MAX 10
#define true 1
#define false 0
char stack[MAX];
int top1, top2;

void init(void);
void push(char data, int nomorstack);
char pop(int nomorstack);
void clear(int nomorstack);
int full(void);
int empty(int nomorstack);
void baca();

main(){
char data;
int pilih, nomorstack;
init();
do{
system("cls");
printf("Contoh program double stack");
printf("\n1. Push");
printf("\n2. Pop");
printf("\n3. Clear");
printf("\n4. Cetak Data");
printf("\n5. Selesai");
printf("\nPilihan anda  : \n");
scanf("%i",&pilih);
switch(pilih){

case 1: printf("Push\n");
printf("Masukkan datanya :\n"); scanf("%s",&data);
printf("Mau dimasukkan ke stack berapa ? 1 atau 2 ? :\n");
scanf("%i",&nomorstack);
push(data, nomorstack);
break;

case 2: printf("Pop\n");
printf("Masukkan nomor stack\n");
scanf("%i",&nomorstack);
data=pop(nomorstack);
printf("\nData yang dikeluarkan adalah %s", data);
break;

case 3: printf("Clear\n");
printf("Nomor Stack yang akan dikosongkan \n");
scanf("%s",&nomorstack);
clear(nomorstack);
break;

case 4: printf("Cetak Data :\n\n");
baca();
break;
```

```
57
58      case 5: printf("Exit");
59      break;
60      default: printf("Pilihan yang anda masukkan tidak ada");
61      break;
62      }
63
64      }while(pilih!=5);
65      getch();
66      }
67
68      //init
69      void init(){
70      top1=0;
71      top2=MAX+1;
72      }
73
74      //PUSH
75      void push(char data, int nomorstack){
76      if(full()!=true){
77      switch(nomorstack){
78
79      case 1: top1++;
80      stack[top1]=data+1;
81      break;
82
83      case 2: top2--;
84      stack[top2]=data-1;
85      break;
86      default: printf("\nNomor stack salah");
87      break;
88      }
89      }
90      else
91      printf("\nStack penuh");
92      getch();
93      }
94
95      //POP
96      char pop(int nomorstack){
97      char data;
98      if(empty(nomorstack)!=true){
99      switch(nomorstack){
100
101     case 1: data=stack[top1];
102     top1--;
103     return data;
104     break;
105
106     case 2: data=stack[top2];
107     top2++;
108     return data;
109     break;
110     default: printf("\nNomor stack salah");
111     break;
112     }
113     }
114     else printf("\nStack masih kosong");
115     getch();
```

```
116   return 0;
117   }
118   //cek full
119
120   int full(void){
121   if ((top1==0) && (top2==MAX+1)){
122   return true;
123   }
124   else return false;
125   }
126
127   //cek empty
128   int empty(int nomorstack){
129   switch(nomorstack){
130   case 1: if(top1==0) return true;
131   else return false;
132   break;
133
134   case 2: if(top2==MAX+1) return true;
135   else return false;
136   break;
137   default: printf("nomor stack salah");
138   break;
139   }
140   }
141
142   //clearing
143   void clear(int nomorstack){
144   switch(nomorstack){
145   case 1: top1=1;
146   break;
147
148   case 2: top2=MAX;
149   break;
150   default: printf("Nomor stack salah");
151   break;
152   }
153   }
```

```
154
155   void baca(){
156   int i;
157   printf("Cetak isi stack pertama : \n");
158   for(i=1; i<=top1; i++){
159   printf("  %c ",stack[i]);
160   printf("\n");
161   }
162   printf("Cetak Isi stack kedua :\n");
163   for(i=MAX; i>=top2; i--){
164   printf( " %c" ,stack[i]);
165   printf("\n");
166   }
167
168   getch();
169   }
170
```

3. Lengkapi sintaks program berikut ini agar bisa berjalan menjadi program pembalik kata dengan stack

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
#include<string.h>
#include<stdlib.h>
#define MAX 75
#define true 1
#define false 0
char stack[MAX];
int top;
void init(void);
int full(void);
int empty(void);
char pop(void);
void clear(void);
void push(char info);
main()
    {
        char pilih;
        char kal[75];
        int k,p;
        printf("-*- PROGRAM MEMBALIK KATA DENGAN STACK -*-\n\n");
        init();
        do
    {
        printf(" ======================\n");
        printf(" || MENU PILIHAN ||\n");
        printf(" ======================\n");
        printf(" ||[1] Masukan Kata ||\n");
        printf(" ||[2] Balik Kata ||\n");
        printf(" ||[3] Selesai ||\n");
        printf(" ======================\n\n");
        printf(" Pilihan : ");
        scanf("%s",&pilih);
        system("cls");
        switch(pilih)
        {
            case '1': printf("\n Masukkan kata : ");
                      scanf("%s",&kal);
                      p=strlen(kal);
                      for(..............)
                      push(kal[k]);
                      printf("\n Kata yang masuk : ");
                      for(..............)
                      printf("%c ",stack[k]);
                      printf("\n");
            break;

            case '2': if(empty()!=true)
        {
                      printf("\n Kata setelah dibalik : ");
                      ..................
                      ..................
                      ..................
        }
        else printf("\n Stack kosong !!\n");
            break;
```

```
58              
59                  case '3': break;
60                  default : printf("\n Pilihan anda salah !!\n");
61          }
62              printf("\n");
63              }while(pilih!='3');
64          }
65          void init(void)
66          {
67              top=0;
68          }
69          void push(char info )
70          {
71              if(full()!=true)
72          {
73          top++;
74          stack[top]=info;
75          }
76          else
77          printf("\n Stack overlow...\n");
78          }
79          char pop(void)
80          {
81              char info;
82              if(empty()!=true)
83          {
84          info=stack[top];
85          top--;
86          return(info);
87          }
88          else
89          printf("\n Stack underflow...\n");
90          }
91          int full(void)
92          {
93              if(top==MAX)
94              return(true);
95              else
96              return(false);
97          }
98          int empty(void)
99          {
100             if(top==0)
101             return(true);
102             else
103             return(false);
104         }
105
```

E. TUGAS RUMAH

1. Buatlah sebuah program dengan menggunakan konsep stack yang berfungsi untuk menyimpan data nilai mahasiswa yang terdiri dari id, nama, dan nilai mahasiswa. Tambahkan beberapa fasilitas seperti **sorting**(pengurutan data), **multi push** (menambahkan data lebih dari satu), dan **multi pop** (mengambil data lebih dari satu sekaligus).