**LAPORAN PRAKTIKUM**

**STRUKTUR DATA**

**MODUL 5**

**ADT Stack Representasi Kontigu**

**DISUSUN OLEH :**

**AJI KARTIKO HARTANTO - 2350081062**

Logo

Description automatically generated

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS SAINS DAN INFORMATIKA**

**UNIVERSITAS JENDERAL ACHMAD YANI**

**TAHUN 2024**

DAFTAR ISI

[DAFTAR GAMBAR ii](#_Toc167173183)

[BAB I. HASIL PRAKTIKUM 1](#_Toc167173184)

[I.1 Program x 1](#_Toc167173185)

[I.1.A. Source Code Boolean.h 1](#_Toc167173186)

[I.1.B. Source Code Stack.h 2](#_Toc167173187)

[I.1.C. Source Code Stack.c 4](#_Toc167173188)

[I.1.D. Source Code mstack.c 7](#_Toc167173189)

[I.1.E. Hasil 9](#_Toc167173190)

[I.1.F. Analisa 10](#_Toc167173191)

[BAB II. TUGAS PRAKTIKUM 11](#_Toc167173192)

[II.1 Tugas 1 (Berkelompok) 11](#_Toc167173193)

[II.1.A. Source Code Boolean.h 11](#_Toc167173194)

[II.1.B. Source Code stack2.h 11](#_Toc167173195)

[II.1.C. Source Code stack2.c 13](#_Toc167173196)

[II.1.D. Source Code mstack2.c 16](#_Toc167173197)

[II.1.E. Hasil 17](#_Toc167173198)

[II.1.F. Analisa 18](#_Toc167173199)

[BAB III. KESIMPULAN 19](#_Toc167173200)

# DAFTAR GAMBAR

[Gambar I.1 Output Program mstack.c 9](#_Toc167173175)

[Gambar II.1 Output Program Tugas Kelompok Stack2 18](#_Toc167173176)

# HASIL PRAKTIKUM

## Program x

### Source Code Boolean.h

|  |
| --- |
| /\*  Program : boolean.h  Author : 2350081062, Aji Kartiko Hartanto  Kelas : C  Deskripsi : Header file dari boolean  Tanggal : 15 Mei 2024  \*/  #ifndef boolean\_H  #define boolean\_H  #define true 1  #define false 0  #define ya 1  #define tidak 0  #define boolean unsigned char  #endif |

### Source Code Stack.h

|  |
| --- |
| /\*  Program : stack.h  Author : 2350081062, Aji Kartiko Hartanto  Kelas : C  Deskripsi : Header file dari prototype stack  Tanggal : 15 Mei 2024  \*/  #ifndef ADT\_SRK\_STACK\_H  #define ADT\_SRK\_STACK\_H  #include <stdio.h>  #include <conio.h>  #include "boolean.h"  #define Nil 0  #define MaxEl 10  #define IdxUndef (-999)  // Definisi akses dengan selektor: Set dan Get  #define Top(S) (S).TOP  #define InfoTop(S) (S).T[(S).TOP]  #define PopTop(S) (S).T[(S).TOP - 1]  #define ElemenTop(S) (S).T[i]  // Definisi ADT tipe stack  typedef int infoType;  typedef int address;  typedef struct {  infoType T[MaxEl + 1];  address TOP;  } Stack;  // konstruktor  void CreateStack(Stack \*S);  // {Kelompok operasi pada stack}  // Predikat untuk test keadaan Koleksi  boolean IsStackEmpty(Stack S);  boolean IsStackFull(Stack S);  // Menambahkan sebuah elemen ke stack  void Push(Stack \*S, infoType X);  // Menghapus sebuah elemen stack  void Pop(Stack \*S, infoType \*X);  // Kelompok interaksi dengan I/O device, Baca/tulis  void PrintStackInfo(Stack S);  // Kelompok operasi lain terhadap type  boolean IsInfoKetemu(Stack S, infoType X);  address CariElemenStack(Stack S, infoType X);  #endif |

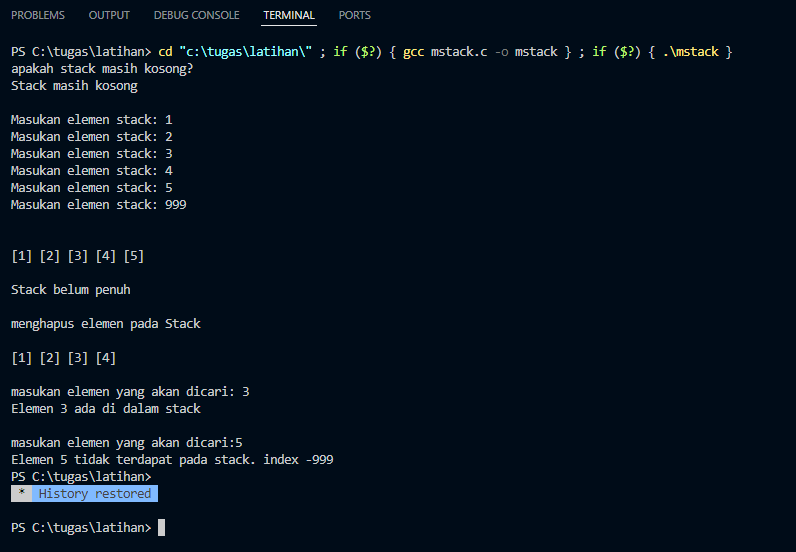
### Source Code Stack.c

|  |
| --- |
| /\*  Program : stack.c  Author : 2350081062, Aji Kartiko Hartanto  Kelas : C  Deskripsi : Header file dari stack  Tanggal : 16 Mei 2024  \*/  #include "stack.h"  // konstruktor  void CreateStack(Stack \*S) {  Top(\*S) = Nil;  }  // {Kelompok operasi pada stack}  // Predikat untuk test keadaan Koleksi  boolean IsStackEmpty(Stack S) {  if (Top(S) == Nil) {  return true;  } else {  return false;  }  }  boolean IsStackFull(Stack S) {  if (Top(S) == MaxEl) {  return true;  } else {  return false;  }  }  // Menambahkan sebuah elemen ke stack  void Push(Stack \*S, infoType X) {  if (!IsStackFull(\*S)) {  Top(\*S)++;  InfoTop(\*S) = X;  } else {  printf("Stack penuh");  }  }  // Menghapus sebuah elemen stack  void Pop(Stack \*S, infoType \*X) {  if (!IsStackEmpty(\*S)) {  \*X = InfoTop(\*S);  Top(\*S) = PopTop(\*S);  } else {  printf("Stack kosong");  }  }  // Kelompok interaksi dengan I/O device, Baca/tulis  void PrintStackInfo(Stack S) {  // kamus  int i;  // algoritma  for (i = 1; i <= Top(S); i++) {  printf("[%d] ", ElemenTop(S));  }  }  // Kelompok operasi lain terhadap type  boolean IsInfoKetemu(Stack S, infoType X) {  // kamus  int i;  boolean ketemu;  // algoritma  i = 1;  ketemu = tidak;  while (i <= Top(S) && !ketemu) {  if (ElemenTop(S) == X) {  ketemu = ya;  } else {  i++;  }  }  return ketemu;  }  address CariElemenStack(Stack S, infoType X) {  // kamus  int i;  // algoritma  if (!IsStackEmpty(S)) {  for (i = 1; i <= Top(S); i++) {  if (ElemenTop(S) == X) {  return i;  }  }  return IdxUndef;  } else {  return IdxUndef;  }  } |

### Source Code mstack.c

|  |
| --- |
| /\*  Program : mstack.c  Author : 2350081062, Aji Kartiko Hartanto  Kelas : C  Deskripsi : Main driver dari stack  Tanggal : 17 Mei 2024  \*/  #include "stack.c"  int main() {  // kamus  Stack stack;  infoType tmp, x;  address Elm;  //algoritma  // membuat stack  CreateStack(&stack);  // mengecek apakah stack kosong atau tidak  printf("apakah stack masih kosong?\n");  if (IsStackEmpty(stack)) {  printf("Stack masih kosong");  } else {  printf("Stack tidak kosong");  }  // menambah elemen pada stack  printf("\n\n");  printf("Masukan elemen stack: ");  scanf("%d", &x);  while (x != 999) {  Push(&stack, x);  printf("Masukan elemen stack: ");  scanf("%d", &x);  }  printf("\n\n");  PrintStackInfo(stack);  // mengecek apakah stack penuh atau tidak  printf("\n\n");  if (IsStackFull(stack)) {  printf("Stack sudah penuh");  } else {  printf("Stack belum penuh");  }  // penghapusan pada stack  printf("\n\n");  printf("menghapus elemen pada Stack");  Pop(&stack, &tmp);  printf("\n\n");  PrintStackInfo(stack);  // mecari elemen stack dan mengembalikan true atau false  printf("\n\n");  printf("masukan elemen yang akan dicari: ");  scanf("%d", &x);  if (IsInfoKetemu(stack, x)) {  printf("Elemen %d ada di dalam stack", x);  } else {  printf("Elemen %d tidak ada di dalam stack", x);  }  // mencari elemen pada stack dan mengembalikan nilai address nya  printf("\n\n");  printf("masukan elemen yang akan dicari:");  scanf("%d", &x);  Elm = CariElemenStack(stack, x);  if (Elm != IdxUndef) {  printf("Elemen %d terdapat pada stack dengan index %d", x, Elm);  } else {  printf("Elemen %d tidak terdapat pada stack. index %d", x, IdxUndef);  }  return 0;  } |

### Hasil



Gambar I.1 Output Program mstack.c

### Analisa

Program ini adalah program Bahasa C untuk mencetak stack dengan menggunakan operasi bernama push, dalam program tersebut kita memasukkan data ke dalam stack lalu kita hapus satu, dua, atau lebih elemen tertentu dengan menggunakan operasi yang dinamakan pop, dengan syarat aturan *“last in first out”.*

Dalam hasil output diatas, saya mencetak 5 elemen dan mengambil elemen yang paling terakhir di masukkan, kemudian mengecek apakah salah satu elemen ada di dalam stack atau tidak, jika ada maka pencarian elemen akan lanjut sampai elemen yang di cari tidak ada dalam stack.

# TUGAS PRAKTIKUM

## Tugas 1 (Berkelompok)

### Source Code Boolean.h

|  |
| --- |
| /\* Program : boolean.h  Deskripsi : Header file boolean  \*/    #ifndef boolean\_H  #define boolean\_H  #define true 1  #define false 0  #define boolean unsigned char  #endif |

### Source Code stack2.h

|  |
| --- |
| /\* Program : stack2.h  Author : 2350081062 Aji Kartiko Hartanto  2350081064 Jarwo Eddy Wicaksono  2350081079 Rifqi Fauzi Anwar  Kelas : C  Deskripsi : Header File ADT Stack List  Tanggal : 15 Mei 2024  \*/  #ifndef \_STACK2\_H  #define \_STACK2\_H  #include "boolean.h"  #include <stdio.h>  #include <conio.h>  //Pendefinisian Pointer  #define nil NULL  #define next(P) (P)->next  #define info(P) (P)->info  #define TOP(S) (S).TOP  //Pendefinisian tipe Stack  typedef int infotype; /\* ELemen Stack bertipe integer \*/  typedef struct tElmStack \*address;  typedef struct tElmStack{  infotype info;  address next;  } ElmStack;  typedef struct{  address BOTTOM;  address TOP;  } Stack;  // Prototype Stack/primitif Stack Pointer  // Konstruktor  void CreateStack(Stack \*S);  /\* I.S. : S terdefinisi tidak diketahui isinya  F.S. : S diinisialisasi dengan TOP(S) = nil  \*/  address Alokasi(infotype x);  /\* Mengirim sebuah elemen Stack dalam bentuk address \*/  //Destruktor  void Dealokasi(address P);  /\* Me-release memori dari P sebuah elemen Stack\*/  //Operasi Stack  void Push(Stack \*S, infotype X);  /\* Menambahkan X sebagai elemen Stack S \*/  /\* I.S. : S terdefinisi, mungkin kosong, dan  S penampung elemen stack tidak penuh  F.S. : X menjadi TOP yang baru, TOP bertambah 1  \*/  void Pop(Stack \*S, infotype \*X);  // Menghapus X dari Stack S  /\* I.S. : S terdefinisi, tidak kosong  F.S. : Mengambil elemen pada Top sehingga X berisi nilai elemen TOP  yang lama, kemudian TOP berkurang 1  \*/  void CetakStack(Stack S);  /\* I.S. : S terdefinisi sembarang, tidak kosong  F.S. : Menampilkan semua elemen S ke layar  \*/  boolean IsStackEmpty(Stack S);  // Mengecek apakah Stack tidak memiliki elemen  // Mengirim true jika S kosong, false sebaliknya  boolean CariElemenStack(Stack S, infotype x);  /\* Mengirim true jika x ditemukan pada S, false jika tidak ditemukan  skema pencarian dengan search dengan boolean  \*/  #endif |

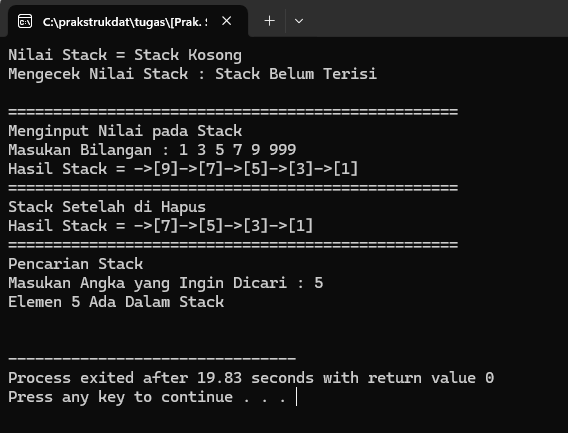
### Source Code stack2.c

|  |
| --- |
| /\* Program : stack2.c  Author : 2350081062 Aji Kartiko Hartanto  2350081064 Jarwo Eddy Wicaksono  2350081079 Rifqi Fauzi Anwar  Kelas : C  Deskripsi : Body File ADT Stack List  Tanggal : 15 Mei 2024  \*/  #include "stack2.h"  #include "boolean.h"  #include <stdio.h>  #include <conio.h>  #include <stdlib.h>  // Konstruktor  void CreateStack(Stack \*S){  TOP(\*S) = nil;  }  address Alokasi(infotype x){  address P;  P = (address) malloc(sizeof(ElmStack));  if(P != nil){  info(P) = x;  next(P) = nil;  return P;  }  else{  printf("Alokasi Gagal");  return(nil);  }  }  //Destruktor  void Dealokasi(address P){  free(P);  }  //Operasi Stack  void Push(Stack \*S, infotype X){  address PX;    PX = Alokasi(X);  next(PX) = TOP(\*S);  TOP(\*S) = PX;  }  void Pop(Stack \*S, infotype \*X){  address atas;    atas = TOP(\*S);  \*X = info(atas);  TOP(\*S) = next(atas);  Dealokasi(atas);  }  void CetakStack(Stack S){  address atas;    atas = TOP(S);  if(atas != nil){  while(atas != nil){  printf("->[%d]",info(atas));  atas = next(atas);  }  }  else{  printf("Stack Kosong");  }  }  boolean IsStackEmpty(Stack S){  if(TOP(S) == nil){  return true;  }else  return false;  }  boolean CariElemenStack(Stack S, infotype x){  address atas;    atas = TOP(S);  while (atas != nil) {  if (info(atas) == x) {  return true;  }  atas = next(atas);  }  return false;  } |

### Source Code mstack2.c

|  |
| --- |
| /\* Program : mstack2.c  Author : 2350081062 Aji Kartiko Hartanto  2350081064 Jarwo Eddy Wicaksono  2350081079 Rifqi Fauzi Anwar  Kelas : C  Deskripsi : Main Driver File ADT Stack List  Tanggal : 15 Mei 2024  \*/  #include "stack2.h"  #include "boolean.h"  #include <stdio.h>  #include <conio.h>  #include <stdlib.h>  int main(){  //Kamus  Stack MyStack;  infotype x, y, cari;  boolean ketemu;  //Algoritma  CreateStack(&MyStack);  printf("Nilai Stack = "); CetakStack(MyStack);  printf("\nMengecek Nilai Stack : ");  if(IsStackEmpty(MyStack)){  printf("Stack Belum Terisi\n");  }  printf("\n==================================================\n");  printf("Menginput Nilai pada Stack\n");  printf("Masukan Bilangan : ");  scanf("%d", &x);  while(x != 999){  Push(&MyStack, x);  scanf("%d", &x);  }  printf("Hasil Stack = "); CetakStack(MyStack);  printf("\n==================================================\n");  printf("Stack Setelah di Hapus\n");  Pop(&MyStack, &y);  printf("Hasil Stack = "); CetakStack(MyStack);  printf("\n==================================================");  printf("\nPencarian Stack\n");  printf("Masukan Angka yang Ingin Dicari : ");  scanf("%d", &cari);  ketemu = CariElemenStack(MyStack, cari);  if(!IsStackEmpty(MyStack)){  if(ketemu != false){  printf("Elemen %d Ada Dalam Stack\n\n", cari);  }  else{  printf("Elemen %d Tidak Ada Dalam Stack\n\n", cari);  }  }  } |

### Hasil



Gambar II.1 Output Program Tugas Kelompok Stack2

### Analisa

Program ini mirip seperti pada program latihan di atas yang operasi nya juga sama yaitu mencetak nilai stack, menhapus elemen pada stack, kemudian mencari elemen pada stack, dan memastikan apakah elemen pada stack itu ada atau tidak.

# KESIMPULAN

Dalam praktikum ini, kita mempelajari konsep dasar dari tipe data abstrak tumpukan (stack) dan bagaimana mengimplementasikannya dalam program. Stack merupakan struktur data linier yang mengikuti prinsip Last-In-First-Out, di mana elemen yang terakhir dimasukkan akan menjadi elemen pertama yang dikeluarkan.

Melalui praktikum ini, kita dapat memahami dengan lebih baik bagaimana stack bekerja dan mengaplikasikan konsepnya dalam pemrograman. Selain itu, kita juga dapat mengembangkan keterampilan dalam mengimplementasikan struktur data dasar dan memahami pentingnya memilih struktur data yang tepat untuk menyelesaikan masalah tertentu.