IK141 Struktur Data

Struktur Data

STACK (TUMPUKAN)

Di Susun Oleh :

Wendi Kardian, 2100016

PROGRAM STUDI PENDIDIKAN ILMU KOMPUTER

FAKULTAS PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PENDIDIKAN INDONESIA

26 Maret 2022

1. Implementasi dan Hasil

Implementasi dan Hasil

A. Studi Kasus

1. Buatlah sebuah program untuk memeriksa apakah sebuah string adalah polindrom atau bukan. Diberikan sebuah string n dengan panjang string k dengan 0<k<1000.
Contoh input:
katak
output :
polindrom

input :
jumat
output :
bukan polindrom

Source Code :

a. Proses inisialisasi awal untuk melakukan import file header serta membuat structure yang nantinya akan di gunakan untuk membuat stack untuk mencek apakah polindrom atau bukan.

```
// Nama : Wendi Kardian
// NIM  : 2100016
// Kelas: Pendidikan Ilmu Komputer - A


#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 20

// deklarasi stack yang memiliki 2 data yaitu data stringnya dan index paling atasnya (top)
struct stack {
    char word[MAX];
    int top;
};

```

b. Prosedur untuk menambahkan data di akhir berupa char yang nantinya akan berupa tumpukan char yang nanti kelak akan di reverse dan di cek apakah kata tersebut polindrom atau tidak

```
// prosedur untuk menambahkan data di akhir tumpukan
void push(struct stack *stack, char val[]){
    int length;
    if(stack->top == MAX-1){ // pengecekan apabila data penuh
        printf("Full\n");
    }else{ // proses untuk menambahkan data kedalam stack
        length = strlen(val);
        stack->top+= length;
        strcpy(stack->word, val);
    }
}

```

c. Fungsi untuk melakukan proses pop atau menghapus data dari stack yang berada diujung stack tersebut, dan setelah di ambil datanya akan direturn char tersebut yang nantinya akan disimpen dalam sebuah char untuk di check apakah polindrom atau bukan.

```c

// Fungsi untuk menghapus data di akhir stack tersebut
char pop(struct stack *st){
    int val;
    if(st->top == -1){
        printf("\n STACK UNDERFLOW");
        return -1;
    }else{ //kondisi untuk mengembalikan char untuk mengembalikan nilai ujung char array of string tersebut
        val = st->word[st->top];
        st->top = st->top - 1;
        return val;
    }
}


```

d. Fungsi utama dimana terdapat beberapa proses deklarasi variabel, serta terdapat proses input data string yang nantinya akan di cek apakah polindrom atau tidak. Setelah itu, terdapat juga proses iterasi untuk melakukan push kedalam stack. Setelah data di push, langkah selanjutnya data di pop yang di simpan kedalam sebuah array of string. Untuk mengecek apakah kata tersebut string itu polindrom atau tidak menggunakan strcmp untuk melakukan pengecekan apakah array of string tersebut sama atau tidak.

```c
// fungsi utama
int main(){
    char polindrom[20]; //variabel untuk menampung data kata yang di reverse (untuk di cek apakah polindrom atau bukan)
    char polindrom1[20];
    int arrayLength,i;
    myStack.top = -1; //start stack
    scanf("%s", polindrom1); //proses input data kedalam variabel array polindrom
    push(&myStack, polindrom1);
    arrayLength = strlen(myStack.word); //menghitung panjang array of string
    for(i = 0; i<arrayLength; i++){ // looping untuk proses pop stack yang ditampung kedalam polindrom
        polindrom[i] = pop(&myStack);
    }

    // proses mengecek apakah dia polindrom atau bukan menggunakan strcmp
    if(strcmp(polindrom, myStack.word) == 0){
        printf("\nPolindrom");
    }else{
        printf("\nBukan Polindrom");
    }

}
```

Test Case

```
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum> cd "
tack.c -o PolindromStack } ; if ($?) { .\PolindromStack }
katak

Polindrom
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum> cd "
tack.c -o PolindromStack } ; if ($?) { .\PolindromStack }
jumat

Bukan Polindrom
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum> cd "
tack.c -o PolindromStack } ; if ($?) { .\PolindromStack }
madam

Polindrom
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum>
```

2. Diberikan sebuah bilngan biner a dengan n digit, 0<n<1000. Lakukan konversi bilangan biner tersebut menjadi bilangan desimal dengan memanfaatkan penggunaan penyimpanan data stack dengan ukuran n.

a. Import library yang diperlukan, selanjutnya buat structure untuk stack yang akan menampung tumpukan string berupa bilangan biner dan top untuk menentukan index teratas dari stack tersebut. Kemudian, deklarasi sebuah stack dengan tipe data struct stack yang diberi nama biner.

```c
// Nama  : Wendi kardian
// NIM   : 2100016
// Kekas : Pendidikan Ilmu Komputer - A


#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
#define MAX 32

// struct untuk menampung stack bilangan biner
struct stack{
    char basis[MAX];
    int top;
};

// deklarasi variabel dengan tipe struct yang sudah di buat
struct stack biner;

```

b. Prosedur untuk menambahkan data di akhir stack tersebut yang di isi dengan bilangan biner o dan 1.

```c
// function push untuk memasukan nilai biner kedalam struct tersebut
void push(struct stack *st, char val){
    if(st->top == MAX-1){
        printf("\n STACT OVERFLOW");
    }else{
        st->top = st->top +1;
        st->basis[st->top] = val;
    }
}
```

c. Fungsi pop untuk menghapus data yang berada paling akhir atau top. Dalam fungsi ini nanti akan dilakukan proses pengecekan apakah char tersebut 1 atau 0. Apabila 1 berarti basis tersebut akan mengembalikan sebuah integer yang bernilai 2 pangkat index dari basis tersebut. Apabila 0 berarti akan mengembalikan 0 juga (bilangan apapun yang dikalikan dengan 0 akan menghasilkan bilangan 0 juga)

```c
// function pop untuk menghapus 1 data di akhir tumpuk yang setiap basisnya akan mengembalikan nilai desimalnya yang akan dijumlahkan
int pop(struct stack *st){
    int val;
    if(st->top == -1){
        printf("\n STACK UNDERFLOW");
        return -1;
    }else{
        if(st->basis[st->top] == '1'){
            val = 1*pow(2, st->top);
            st->top = st->top - 1;
            return val;
        }else{
            st->top = st->top - 1;
            return 0;
        }

    }
}
```

d. Fungsi utama yang berisikan inisialisasi variable yang akan diperlukan, kemudian terdapat proses untuk memasukan data biner tersebut kedalam string yang nantinya tiap string tersebut akan dimasukan kedalam stack secara decrement. Setelah data dimasukan (di push) maka selanjutnya program akan melakukan proses pop (menghapus data) serta nilai returnnya akan disimpan dan terus ditambahkan kedalam variable yang diberi nama desimal, dimana variable tersebut memiliki fungsi untuk menampung nilai dari hasil convert biner ke desimal tersebut.

```c
// fungsi utama
int main(){
    biner.top = -1; //deklarasi awal nilai top stack tersebut
    int i, arrayLength;
    int desimal = 0;
    char word[MAX]; // variabel untuk menampung bilangan biner
    scanf("%s", word);
    arrayLength = strlen(word);
    for(i = arrayLength-1; i>=0; i--){   //iterasi untuk melakukan push nilai biner kedalam stack
        push(&biner, word[i]);
    };
    int j=biner.top;
    for(i=j; i>=0; i--){ // iterasi untuk melakukan proses pop serta menkalkulasi nilai desimal dari bilangan biner tersebut
        desimal = desimal + pop(&biner);
    }
    printf("\n%d", desimal);

    return 0;
}
```

TEST CASE

```
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum> cd "d:\U
ertoDecimal.c -o ConvertBinertoDecimal } ; if ($?) { .\ConvertBine
100

4
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum> cd "d:\U
ertoDecimal.c -o ConvertBinertoDecimal } ; if ($?) { .\ConvertBine
101101

45
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum> cd "d:\U
ertoDecimal.c -o ConvertBinertoDecimal } ; if ($?) { .\ConvertBine
100111100

316
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum>
```

3. Silahkan cari lagi dari berbagai referensi, minimal 2 operasi yang mungkin dapat diselesaikan dengan lebih mudah menggunakan penyimpanan data STACK.

A. Fibonnaci Sequence

Fibonnaci sequence adalah serangkaian deret angka sederhana yang susunan angkanya merupakan penjumlahan dari dua angka sebelumnya (0,1,1,2,3,5,8,13,21,...dst) rumus deret Fibonacci bisa ditulis sebagai berikut Un = Un-2 + Un-1. Adapun source code untuk membuat program fibonnaci menggunakan stack adalah sebagai berikut.

a. Import library yang diperlukan untuk program tersebut. Kemudian, terdapat pembuatan structure yang nantinya akan menampung stack untuk menghitung deret fibonnaci. Setelah itu, terdapat 2 variabel yang bertipe structure stack tersebut untuk menampung deret fibonnaci.

```c
// Wendi Kardian - 2100016
// Fibonnaci Sequence

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 100

// struktur untuk stack yang akan menampung deret fibonnaci
struct stack{
    int value[MAX];
    int top;
};

// deklarasi 2 penampung stack untuk menampung data deret fibonnaci
struct stack fibonnaci;
struct stack fibonnaci2;
```

b. Prosedur untuk menambahkan data kedalam deret fibonnaci tersebut, ada beberapa aturan diantaranya (1) untuk baris pertama dan kedua selalu bernilai 1. (2) untuk baris selanjutnya merupakan hasil penjumlahan dari 2 baris sebelumnya. Angka tersebut akan dimasukan kedalam stack tersebut.

```c
// prosedur untuk menambah deret fibonnaci di akhir
void push(struct stack* stack){
    if(stack->top == -1){ //inisialisasi apabila masih kosong maka defaultnya 1
        stack->top++;
        stack->value[stack->top] = 1; //inisialisasi apabila hanya terdiri 1 data saja maka defaultnya 1
    }else if(stack->top == 0){
        stack->top++;
        stack->value[stack->top] = 1;
    }else{ // kondisi fibonnaci di mana bilangan tersebut merupakan penjumlahan dari 2 bilangan sebelumnya
        stack->top++;
        stack->value[stack->top] = stack->value[stack->top-2] + stack->value[stack->top -1];
    }
}

```

c. Fungsi untuk menghapus data (pop) serta akan mengembalikan nilai baris fibonnaci paling ujung yang nantinya akan disimpan kedalam variable reverse fibonanci.

```
int pop(struct stack *stack){ //function untuk mengambil data di paling ujung stack tersebut
    int val;
    if(stack->top == -1){  //di cek apabila stack kosong atau tidak
        printf("Stack empty\n");
        return 0;
    }else{ //kondisi untuk mengembalikan nilai stack paling belakang
        val = stack->value[stack->top];
        stack->top--;
        return val;
    }
}
```

d. Fungsi utama program tersebut yang dimana didalamnya terdapat deklarasi variable untuk fibonnaci (normal fibonnaci) dan fibonnaci2 (reverse fibonnaci), dimana data akan di push berdasarkan jumlah angka yang di input oleh user (apabila 9 maka program akan membuatkan deret fibonnaci sepanjang 9 baris). Setelah data ditambahkan, program juga akan melakukan proses pop yang hasil return value nya akan ditampung kedalam variable fibonnaci2 (reverse fibonnaci). Lalu dari fibonnaci 2 akan di push dan di pop Kembali untuk mendapatkan nilai dari deret fibonnaci tersebut.s

```
int main(){
    int value, i;
    fibonnaci.top = -1;  //inisialisasi awal indexing stack
    fibonnaci2.top = -1;
    scanf("%d",&value);
    for(i=0; i<value; i++){ //proses menyisipkan deret fibonnaci dari data yang panjangnya ditentukan user input
        push(&fibonnaci);
    }

    int arr[MAX];
    int length = fibonnaci.top;
    for(i = 0; i<= length; i++){   //proses mengambil data dan di tampung ke variabel fibonnaci 2 untuk sementara
        fibonnaci2.top++;
        fibonnaci2.value[i] = pop(&fibonnaci);
    }

    printf("\nReverse Fibonnaci : ");   //menampilkan deret reverse fibonnaci
    for(i = 0;i<= length; i++){
        printf("%d  ", fibonnaci2.value[i]);
    }

    for(i = 0; i<= length; i++){ //proses mereverse kembali fibonnaci agar kembali tidak tereverese
        fibonnaci.value[i] = pop(&fibonnaci2);
    }

    printf("\nNormal Fibonnaci : "); //proses menampilkan deret fibonnaci ke dalam console
    for(i = 0;i<= length; i++){
        printf("%d  ", fibonnaci.value[i]);
    }


    return 0;
}
```

TEST CASE

```
PS D:\UPI\SEMESTER 2\STRUKTUR DATA> cd "d:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum\" ; if ($?) { gcc Fibonnaci
{ .\FibonnaciStack }
9

Reverse Fibonnaci : 34  21  13  8  5  3  2  1  1
Normal Fibonnaci : 1  1  2  3  5  8  13  21  34
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum> cd "d:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum\" ; i
naciStack } ; if ($?) { .\FibonnaciStack }
20

Reverse Fibonnaci : 6765  4181  2584  1597  987  610  377  233  144  89  55  34  21  13  8  5  3  2  1  1
Normal Fibonnaci : 1  1  2  3  5  8  13  21  34  55  89  144  233  377  610  987  1597  2584  4181  6765
```

B. Pharanthesis Checker

Pharanthesis checker merupakan sebuah program untuk melakukan proses pengecekan terhadap operasi aritmatika apakah penggunaan (// { // [sudah benar atau tidak. Ada beberapa rules yang diterapkan dalam program ini, (1) apabila pharanthesis sudah dibuka maka harus ditutup Kembali contoh, (A+(A*B) -> salah (2) apabila pharanthesis diawali oleh '(' maka harus di akhiri oleh ')', tidak boleh '}' maupun ']', contoh (A*B+C] -> salah. Adapun source code untuk program Pharanthesis Checker adalah sebagai berikut.

a. Import library, kemudian pembuatan structure untuk stack yang akan menampung expression yang akan di check pharenthesisnya. Kemudian, deklarasi sebuah variable berdasarkan structure yang sudah dibuat.

```c
// WENDI KARDIAN - 2100016
// PENDIDIKAN ILMU KOMPUTER - A

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 10


// STRUCTURE FOR STACK
struct stack{
    int stk[MAX];
    int top;
};

// DEKLARASI STACK
struct stack stk;
```

b. Prosedur untuk menambahkan char expression kedalam stack.

```c
// PROSEDUR UNTUK MENAMBAHKAN DATA DI AKHIR
void push(char c, struct stack *stk){
    if(stk->top == (MAX - 1)){
        printf("STACK OVERFLOW\n");
    }else{
        stk->top+=1;
        stk->stk[stk->top] = c;
    }
}
```

c. Fungsi untuk melakukan proses pop (menghapus data) dari stack yang nantinya akan mengembalikan sebuah char dari expression yang berada paling atas array of string tersebut.

```c
// PROSEDUR UNTUK MENGHAPUS DATA DAN RETURN DATA DALAM BENTUK CHARACTER
char pop(struct stack *stk){
    if(stk->top == -1){
        printf("STACK UNDER FLOW\n");
    }else{
        return stk->stk[stk->top--];
    }
}
```

d. Fungsi utama yang di dalamnya terdapat proses deklarasi variable yang diperlukan oleh program. Kemudian, terjadi proses input expression yang dilakukan oleh user, selanjutnya berdasarkan input yang user berikan akan dilakukan proses iterasi sebanyak panjangnya string expression tersebut dan akan di cek apakah expression tersebut valid atau tidak berdasarkan rules proses pembuatan expression.

```
void main(){
    // DEKLARASI STACK DAN VARIABEL LAINNYA
    stk.top = -1;
    char exp[MAX];
    char temp;
    int i, flag = 1; //DEFAULT FLAG BERNILAI 1 (TRUE) ATAU EKSPRESSION VALID
    printf("Print the Expression : ");
    scanf("%s", exp);
    for(i = 0; i<strlen(exp); i++){ //ITERASI UNTUK MELAKUKAN PENGECEKAN DARI EKSPRESSION TERSEBUT
        if(exp[i] == '(' || exp[i] == '{' || exp[i] == '['){ // APABILA SIMBOL PEMBUKA MAKA DATA AKAN DI MASUKAN KEDALAM STACK
            push(exp[i], &stk);     //CONTOH SIMBOL PEMBUKA '( // [ // {'
        }
        if(exp[i] == ')' || exp[i] == '}' || exp[i] == ']'){   // APABILA SIMBOL YANG DICEK ITU PENUTUP
            if(stk.top == -1){ //ARTINYA JIKA SIMBOL PENUTUP DILETAKAN DI INDEX PERTAMA (TIDAK ADA PEMBUKANYA)
                flag = 0; // FLAG = 0 ARTINYA EXSPRESSION INVALID
            }else{
                temp= pop(&stk); //MELAKUKAN PENGECEKAN EKSPRESSION
                if(exp[i] == ')' && (temp == '{' || temp == '[')){    // DILAKUKAN PENGECEKAN APAKAH ANTARA KURUNG PEMBUKA DAN PENUTUP
                    flag = 0;                                         // KONSISTEN ATAU TIDAK
                }                                                     // JIKA TIDAK KONSISTEM FLAG = 0 (EXPRESSION INVALID)
                if(exp[i] == '}' && (temp == '(' || temp == '[')){
                    flag = 0;
                }
                if(exp[i] == ']' && (temp == '(' || temp == '{')){
                    flag = 0;
                }
            }
        }

    }
```

e. Pada tahap sebelumnya, diinisialisasikan variable yang Bernama flag untuk menyimpan status benar atau tidak nya sebuah expression. Apabila flag bernilai 1 maka akan dicetakan valid expression, selain itu akan dicetakan invalid expression.

```
    if(stk.top >= 0){ // PROSES PENGECEKAN APAKAH MASIH ADA EXPRESSION YANG TERSISA, JIKA YA BERARTI EXPRESSION INVALID
        flag = 0;
    }
    if(flag ==1){ //PROSES PENYETAKAN EXPRESSION ITU VALID (APABILA FLAG = 1) JIKA FLAG = 0 MAKA EXPRESSION INVALID
        printf("\nValid expression");
    }else{
        printf("\nInvalid expression");
    }
}
```

TEST CASE

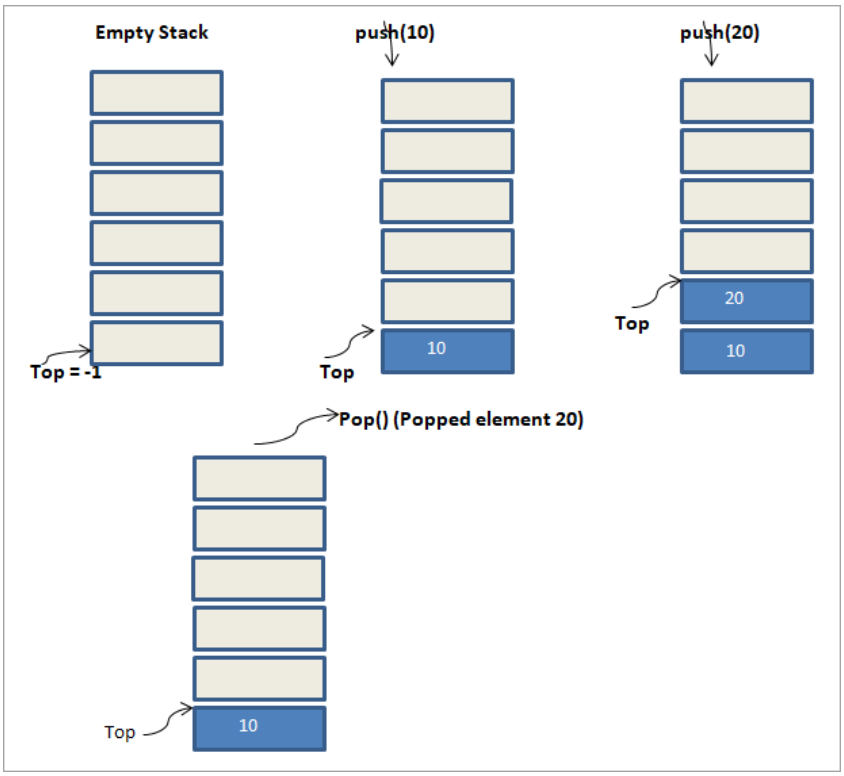
```
Print the Expression : (A*B+(C+2))

Valid expression
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum> cd "d:\UPI
PharanthesisChecker } ; if ($?) { .\PharanthesisChecker }
Print the Expression : (A+(B*C)

Invalid expression
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum> cd "d:\UPI
PharanthesisChecker } ; if ($?) { .\PharanthesisChecker }
Print the Expression : (C+(3*C)+(B*A)}

Invalid expression
PS D:\UPI\SEMESTER 2\STRUKTUR DATA\Pertemuan 7\Praktikum>
```

4. Kesimpulan

Kesimpulan
STACK atau tumpukan adalah sebuah bentuk penyimpanan data dengan cara ditumpuk, data disimpan diatas data lainnya. Struktur ini memungkinkan operasi akses data hanya dapat dilakukan pada data yang berada pada posisi puncak tumpukan saja yaitu data yang terakhir disimpan sehingga pada struktur data ini dikenal dengan istilah Last In First Out (LIFO). Proses dalam stack diantaranya adalah Push (untuk menambahkan data), Pop (untuk menghapus sebuah data), isEmpty (untuk mengecek apakah sebuah stack tersebut kosong atau tidak), dan isFull (untuk mengecek apakah sebuah stack tersebut sudah mencapai kapasitas maksimalnya atau belum).      Dalam pengimplementasiannya, stack dapat digunakan untuk berbagai jenis hal untuk melakukan berbagai jenis kalkulasi, seperti mengecek kata polindrom atau bukan, melakukan conversi dari biner ke desimal, membuat deret fibonnaci, serta mengecek apakah sebuah expression sudah benar atau tidak.