PRAKTIKUM ALGORITMA & STRUKTUR DATA

"STACK"

DOSEN :

Umi Sa'adah S.Kom, M.Kom (197404162000032003)

PENYUSUN :

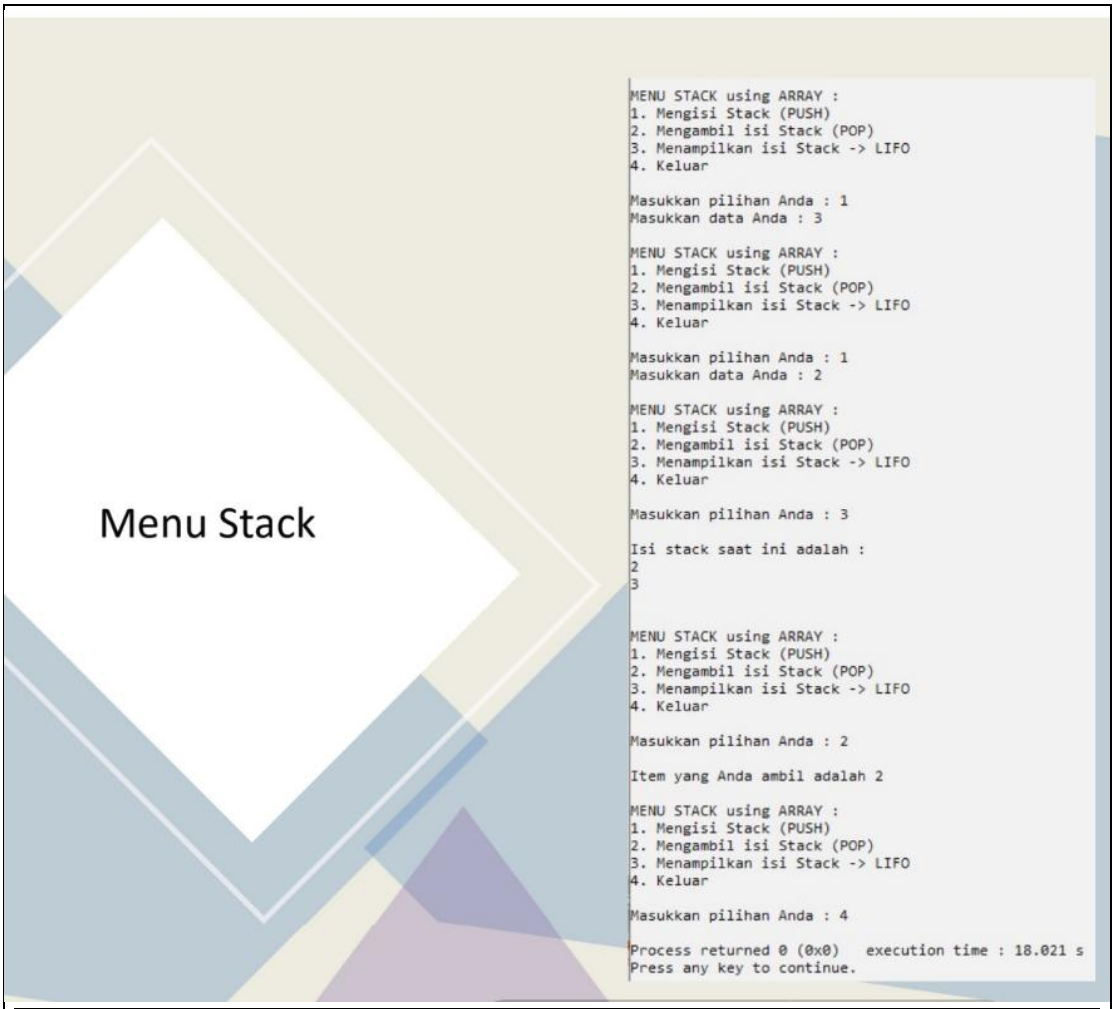
Mohammad Ihsan Septiano Firdaus (3125600041)

PROGRAM STUDI SARJANA TERAPAN

TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

Program




```c
#include <stdio.h>
#define MAX 5
typedef int itemType;
typedef struct {
    itemType data[MAX];
    int count; } stack;

//prototype
void initStack(stack *stk);
void push(itemType a, stack *stk);
void isi(stack *stk);
void ambil(stack *stk);
void tampil(stack stk);
itemType pop(stack *stk);
int isEmpty(stack *stk);
int isFull(stack *stk);

//fungsi utama
int main() {
    int choice; stack tumpukan;
```

```
    initStack(&tumpukan);
    do {
        printf("\nMenu STACK\n");
        printf("1. Mengisi STACK (POP)\n");
        printf("1. Mengambil STACK (POP)\n");
        printf("3. Menampilkan LIFO (Tanpa POP)\n");
        printf("4. Keluar\n");
        printf("Masukkan Operasi = ");
        scanf("%d", &choice);
    switch(choice) {
        case 1: isi(&tumpukan); break;
        case 2: ambil(&tumpukan); break;
        case 3: tampil(tumpukan); break;
        case 4: break;
        default: printf("Masukkan operasi yang tepat!\n"); break;    }
    } while (choice !=4 ); }

//fungsi inisialisasi
void initStack (stack *stk) {
    stk->count = 0; }

//kumpulan fugsi push
void push (itemType a, stack *stk) {
    stk->data[stk->count] = a;
    stk->count++; }

int isFull (stack *stk) {
    return (stk->count == MAX); }

void isi (stack *stk) {
    itemType a;
    if(isFull(stk)) {
        printf("\033[31mSTACK PENUH!\033[0m\n"];]
    else {printf("Operasi PUSH\n");
    printf("Angka yang mau dimasukkan = ");
    scanf("%d", &a);
    push(a, stk); })

//kumpulan fungsi pop
itemType pop (stack *stk) {
    stk->count--;
    return (stk->data[stk->count]); }

int isEmpty (stack *stk) {
    return (stk->count == 0); }

void ambil (stack *stk) {
    if(isEmpty(stk)) {
```

```
        printf("\033[31mSTACK KOSONG!\033[0m\n"); ]
    else {itemType temp = pop(stk);
    printf("\033[32mData yang diambil adalah %d\033[0m\n", temp]; ] )

//meanmpilkan
void tampil(stack stk) {
    if(stk.count == 0) printf("\033[31mKOSONG\033[0m\n"];
    else printf("\033[32mIsi Tumpukan\033[0m\n"];
    for (int i=stk.count; i>0;) {
        i--;
        printf("%d\033[0m ", stk.data[i]];
    )
}
```

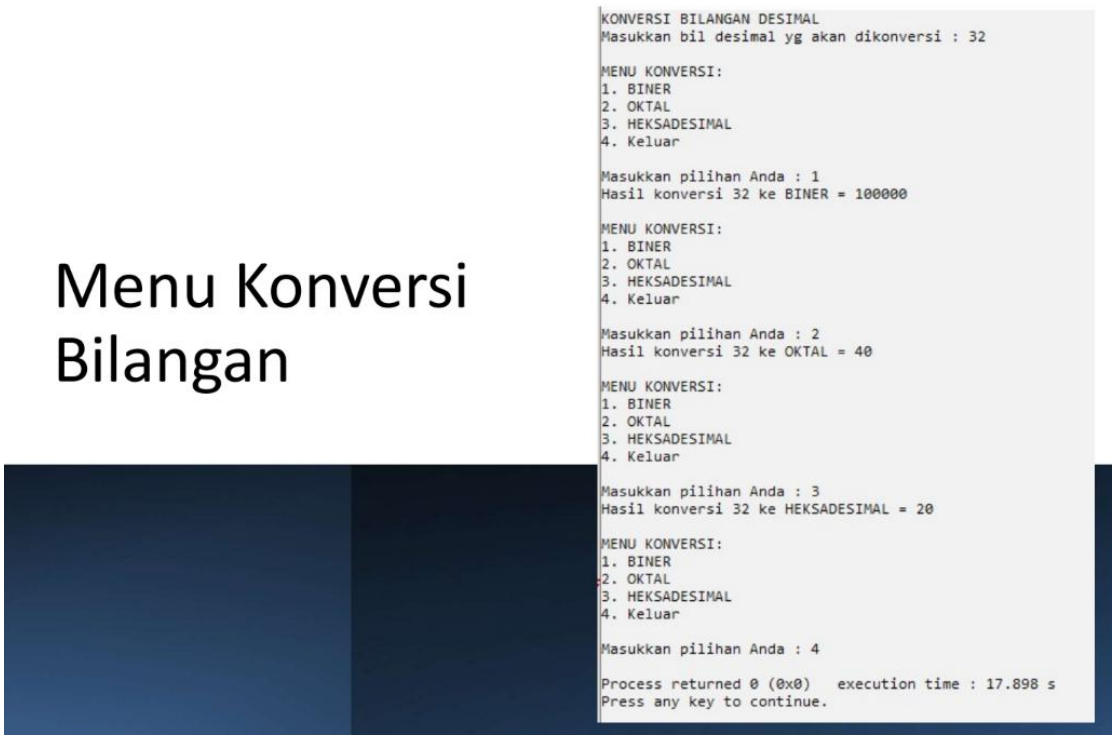

```
MENGUBAH NOTASI INFIX MENJADI PSOTFIX
DENGAN MEMANFAATKAN STRUKTUR STACK

Masukan ekspresi dalam notasi infix : 6-4
Ungkapan postfixnya = 6 4 -

Mau mencoba lagi (y/t) ? y

Masukan ekspresi dalam notasi infix : (3-2)*5
Ungkapan postfixnya = 3 2 - 5 *

Mau mencoba lagi (y/t) ? y

Masukan ekspresi dalam notasi infix : (1+2)/((3-4)*5^6)
Ungkapan postfixnya = 1 2 + 3 4 - 5 6 ^ * /

Mau mencoba lagi (y/t) ? t

Process returned 0 (0x0)   execution time : 52.942 s
Press any key to continue.
```

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAX 100
typedef char itemType;
typedef struct {
    itemType data[MAX];
    int count; } stack;

// prototype
void initStack(stack *stk);
void push(itemType a, stack *stk);
itemType pop(stack *stk);
itemType peek(stack *stk);
int isEmpty(stack *stk);
```

```
int isFull(stack *stk);
int derajat(char op);
void konversiInfix();
void infixToPostfix(char infix[], char postfix[]);

int main() {
    int choice;
    do {
        printf("\nMenu INFIX TO POSTFIX\n");
        printf("1. Konversi Ekspresi\n");
        printf("2. Keluar\n");
        printf("Masukkan Operasi = ");
        scanf("%d", &choice);
        while (getchar() != '\n');

        switch(choice) {
            case 1: konversiInfix(); break;
             case 2: printf("\033[32mKeluar dari program.\033[0m\n");
break;
                    default:  printf("\033[31mMasukkan  operasi  yang
tepat!\033[0m\n"); break;    }
    } while (choice != 2);
    return 0; }

void initStack (stack *stk) {
    stk->count = 0; }

int isEmpty (stack *stk) {
    return (stk->count == 0); }

int isFull (stack *stk) {
    return (stk->count == MAX); }

void push (itemType a, stack *stk) {
    if (!isFull(stk)) {
        stk->data[stk->count] = a;
        stk->count++; } }

itemType pop (stack *stk) {
    if (!isEmpty(stk)) {
        stk->count--;
        return (stk->data[stk->count]); }
    return '\0'; }

itemType peek (stack *stk) {
    if (!isEmpty(stk)) {
        return (stk->data[stk->count - 1]); }
    return '\0'; }
```

```c
int derajat (char op) {
    if (op == '^') return 3;
    if (op == '*' || op == '/') return 2;
    if (op == '+' || op == '-') return 1;
    return 0; }

void konversiInfix () {
    char infix[MAX], postfix[MAX];
    printf("Masukkan ekspresi Infix  = ");
    fgets(infix, sizeof(infix), stdin);
    infix[strcspn(infix, "\n")] = 0;

    infixToPostfix(infix, postfix);
    printf("\033[32mEkspresi Postfix = %s\033[0m\n", postfix); }

void infixToPostfix (char infix[], char postfix[]) {
    stack tumpukan;
    initStack(&tumpukan);
    int i = 0, j = 0;
    char x;

    while ((x = infix[i++]) != '\0') {
        if (isalnum(x)) {
            postfix[j++] = x; }
        else if (x == '(') {
            push(x, &tumpukan); }
        else if (x == ')') {
            while (!isEmpty(&tumpukan) && peek(&tumpukan) != '(') {
                postfix[j++] = pop(&tumpukan); }
            pop(&tumpukan);
        }
        else if (x == ' ') {
            continue;
        }
        else {
             while (!isEmpty(&tumpukan) && derajat(peek(&tumpukan)) >=
derajat(x)) {
                postfix[j++] = pop(&tumpukan); }
            push(x, &tumpukan); }
    }

    while (!isEmpty(&tumpukan)) {
        postfix[j++] = pop(&tumpukan); }

    postfix[j] = '\0';
}
```

Menu Konversi Bilangan




```c
#include <stdio.h>

#define MAX 100
typedef int itemType;

typedef struct {
    itemType data[MAX];
    int count;
} stack;

// prototype
void initStack(stack *stk);
void push(itemType a, stack *stk);
itemType pop(stack *stk);
int isEmpty(stack *stk);
int isFull(stack *stk);
void konversi(int bil, int basis);

int main() {
    int bil, pil;

    printf("KONVERSI BILANGAN DESIMAL\n");
    printf("Masukkan bil desimal yg akan dikonversi : ");
    scanf("%d", &bil);

    do {
        printf("\nMENU KONVERSI:\n");
        printf("1. BINER\n");
```

```
        printf("2. OKTAL\n");
        printf("3. HEKSADESIMAL\n");
        printf("4. Keluar\n");
        printf("\nMasukkan pilihan Anda : ");
        scanf("%d", &pil);

        switch (pil) {
            case 1:
                printf("Hasil konversi %d ke BINER = ", bil);
                konversi(bil, 2);
                printf("\n");
                break;
            case 2:
                printf("Hasil konversi %d ke OKTAL = ", bil);
                konversi(bil, 8);
                printf("\n");
                break;
            case 3:
                printf("Hasil konversi %d ke HEKSADESIMAL = ", bil);
                konversi(bil, 16);
                printf("\n");
                break;
            case 4:
                break;
            default:
                printf("Pilihan tidak valid.\n");
                break;
        }
    } while (pil != 4);

    return 0;
}

void initStack(stack *stk) {
    stk->count = 0;
}

void push(itemType a, stack *stk) {
    if (!isFull(stk)) {
        stk->data[stk->count] = a;
        stk->count++;
    }
}

itemType pop(stack *stk) {
    stk->count--;
    return (stk->data[stk->count]);
}
```

```c
int isEmpty(stack *stk) {
    return (stk->count == 0);
}

int isFull(stack *stk) {
    return (stk->count == MAX);
}

void konversi(int bil, int basis) {
    stack tumpukan;
    initStack(&tumpukan);
    char digit[] = "0123456789ABCDEF";
    int temp = bil;

    if (temp == 0) {
        printf("0");
        return;
    }

    while (temp > 0) {
        if (isFull(&tumpukan)) break;
        push(temp % basis, &tumpukan);
        temp /= basis;
    }

    while (!isEmpty(&tumpukan)) {
        int nilai = pop(&tumpukan);
        printf("%c", digit[nilai]);
    }
}
```

Pengecekan PALINDROM

```
MENGECEK PALINDROM
Masukkan kata yang dicek : MAKAN MALAM
MAKAN MALAM adalah BUKAN PALINDROM

Mau coba lagi (y/t) ? y
Masukkan kata yang dicek : KASUR RUSAK
KASUR RUSAK adalah PALINDROM

Mau coba lagi (y/t) ? T

Process returned 0 (0x0)   execution time : 31.588 s
Press any key to continue.
```

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

typedef struct {
    char data[50];
    int count;
} stack;

void cekPal();
void push(stack *, char[]);
int isPalindrome(stack *, char[]);
char pop(stack *);

int main () {
    char ch;
    do {
        cekPal();
        printf("Lagi (y)? ");
        scanf(" %c", &ch);
        while (getchar() != '\n');
    } while (ch == 'y' || ch == 'Y');
    return 0;
}

void cekPal () {
    stack tumpukan, tester;
    tumpukan.count = 0;
```

```
    char word[50];

    printf("Masukkan kata = ");
    fgets(word, sizeof(word), stdin);
    word[strcspn(word, "\n")] = 0;

    push(&tumpukan, word);

    for (int i=0; i < (strlen(word)); i++) {
        tester.data[i] = pop(&tumpukan);    }
    tester.data[(strlen(word))] = '\0';

    if (isPalindrome(&tester, word))
        printf("Kata tersebut \033[32mPalindrome\033[0m\n");
    else
        printf("Kata tersebut \033[31mtidak Palindrome\033[0m\n");
}

void push (stack *stk, char w[]) {
    for (int i = 0; i < strlen(w); i++) {
        stk->data[i] = w[i];
    }
    stk->count = strlen(w);
    stk->data[strlen(w)] = '\0';
}

int isPalindrome (stack *stk, char w[]) {
        return (strcmpi(stk->data, w ) ==0);
    }

char pop  (stack *stk) {
    stk->count--;
    return (stk->data[stk->count]);
}
```

Stack using SLL

```
MENU STACK using SINGLE LINKED LIST :
1. Mengisi Stack (PUSH)
2. Mengambil isi Stack (POP)
3. Menampilkan isi Stack -> LIFO
4. Keluar

Pilihan Anda : 1
Masukkan datanya : 3

MENU STACK using SINGLE LINKED LIST :
1. Mengisi Stack (PUSH)
2. Mengambil isi Stack (POP)
3. Menampilkan isi Stack -> LIFO
4. Keluar

Pilihan Anda : 1
Masukkan datanya : 6

MENU STACK using SINGLE LINKED LIST :
1. Mengisi Stack (PUSH)
2. Mengambil isi Stack (POP)
3. Menampilkan isi Stack -> LIFO
4. Keluar

Pilihan Anda : 3
Isi dari stack =
6
3

MENU STACK using SINGLE LINKED LIST :
1. Mengisi Stack (PUSH)
2. Mengambil isi Stack (POP)
3. Menampilkan isi Stack -> LIFO
4. Keluar

Pilihan Anda : 2
Data yg diambil dari Stack = 6

MENU STACK using SINGLE LINKED LIST :
1. Mengisi Stack (PUSH)
2. Mengambil isi Stack (POP)
3. Menampilkan isi Stack -> LIFO
4. Keluar

Pilihan Anda : 4

Process returned 0 (0x0)   execution time : 15.927 s
Press any key to continue.
```

```c
#include <stdio.h>
#include <stdlib.h>

typedef int itemType;

typedef struct node {
    itemType data;
    struct node *next;
} Node;

Node *head;

// prototype
void sllNull();
itemType pop();
void push(itemType x);
int isEmpty();
void tampil();

int main() {
    int choice;
    itemType d;
    head = NULL;
```

```c
    do {
        printf("\nMENU STACK using SINGLE LINKED LIST :\n");
        printf("1. Mengisi Stack (PUSH)\n");
        printf("2. Mengambil isi Stack (POP)\n");
        printf("3. Menampilkan isi Stack -> LIFO\n");
        printf("4. Keluar\n");
        printf("\nPilihan Anda : ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Masukkan datanya : ");
                scanf("%d", &d);
                push(d);
                break;
            case 2:
                if (isEmpty()) {
                    printf("Stack Kosong!\n");
                } else {
                   printf("Data yg diambil dari Stack = %d\n", pop());
                }
                break;
            case 3:
                if (isEmpty()) {
                    printf("Stack Kosong!\n");
                } else {
                    printf("Isi dari stack =\n");
                    tampil();
                }
                break;
            case 4:
                break;
            default:
                printf("Pilihan tidak valid!\n");
        }
    } while (choice != 4);

    return 0;
}

int isEmpty() {
    return (head == NULL); }

// push
void push(itemType x) {
    Node *p = (Node *)malloc(sizeof(Node));
    if (p == NULL) {
```

```
        printf("Memori penuh!\n");
    } else {
        p->data = x;
        p->next = head;
        head = p;
    }
}

// pop
itemType pop() {
    Node *temp = head;
    itemType nilai = temp->data;
    head = head->next;
    free(temp);
    return nilai;
}

// tampilk tanpa ubah data
void tampil() {
    Node *p = head;
    while (p != NULL) {
        printf("%d\n", p->data);
        p = p->next;
    }
}
```