

LAPORAN UTS PRATIKUM STRUKTUR DATA DAN ALGORITMA

KONVERSI EKSPRESI ARITMATIKA

Nama:

MAULANA SHIDQI ALBADWIE (2408107010047)

MUHAMMAD FARHAN ALAFIF (2408107010075)

A. Penggunaan Metode

Dalam pembuatan kode konversi ekspresi aritmatika (infix, postfix, prefix) ini kami menerapkan metode struktur data stack karena sifatnya yaitu LIFO (Last In, First Out), yang sangat cocok untuk menangani operasi dengan prioritas tertentu seperti konversi ekspresi matematika. Ada 14 fungsi yang terdapat dalam kode program selain fungsi main.

B. Kode Program

1. *init(Stack *s)*


Fungsi "init" adalah singkatan dari initialize, yang berarti menginisialisasi stack agar kosong.

A screenshot of a code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. The code is written in C and defines the 'init' function for a stack. It consists of two lines: a comment and a function definition.

```
1 // Menginisialisasi stack
2 void init(Stack *s) { s->top = -1; }
```

2. *isEmpty(Stack *s)*

berfungsi untuk mengecek apakah stack kosong.

A screenshot of a code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. The code is written in C and defines the 'isEmpty' function for a stack. It consists of two lines: a comment and a function definition.

```
1 // Mengecek apakah stack kosong
2 int isEmpty(Stack *s) { return s->top == -1; }
```

3. *push(Stack *s, char item)*

berfungsi untuk menambahkan elemen ke dalam stack.

```
1 // Menambahkan elemen ke stack
2 void push(Stack *s, const char *str) {
3     if (s->top < MAX - 1) {
4         s->top++;
5         strcpy(s->arr[s->top], str);
6     }
7 }
```

4. *pop(Stack *s)*

berfungsi untuk menghapus elemen dari stack.

```
1 // Menghapus elemen dari stack
2 char *pop(Stack *s) {
3     if (!isEmpty(s)) return s->arr[s->top--];
4     return NULL;
5 }
```

5. *peek(Stack *s)*

berfungsi untuk melihat elemen teratas stack tanpa menghapusnya.

```
1 // Melihat elemen teratas stack
2 char *peek(Stack *s) {
3     if (!isEmpty(s)) return s->arr[s->top];
4     return NULL;
5 }
6
```

6. *precedence(char op)*

Fungsi "precedence" berarti prioritas, sesuai dengan fungsinya untuk mengembalikan prioritas operator.

```
1 // Mengembalikan prioritas operator
2 int precedence(char ch) {
3     switch (ch) {
4         case '+': case '-': return 1;
5         case '*': case '/': return 2;
6         case '^': return 3;
7     }
8     return -1;
9 }
```

7. *isOperator(char ch)*

Fungsi "isOperator" menunjukkan fungsi ini mengecek apakah suatu karakter adalah operator.

```
1 // Mengecek apakah karakter adalah operator
2 int isOperator(char ch) { return ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '^'; }
```

8. *reverseString(char *exp)*

berfungsi untuk membalik urutan karakter dalam string.

```
1 // Membalik string untuk konversi infix ke prefix
2 void reverseString(char *str) {
3     int len = strlen(str);
4     for (int i = 0; i < len / 2; i++) {
5         char temp = str[i];
6         str[i] = str[len - i - 1];
7         str[len - i - 1] = temp;
8     }
9 }
```

9. *infixToPostfix(char *infix, char *postfix)*

berfungsi untuk mengubah ekspresi infix menjadi postfix.

```
1 // Mengonversi infix ke postfix
2 void infixToPostfix(const char *infix, char *postfix) {
3     Stack operators;
4     Init(&operators);
5     int i = 0, k = 0;
6     char token, buffer[MAX];
7
8     while ((token = infix[i++]) != '\0') {
9         if (isalnum(token)) {
10             int j = 0;
11             while (isalnum(token)) {
12                 buffer[j++] = token;
13                 token = infix[i++];
14             }
15             buffer[j] = '\0'; i--;
16             sprintf(postfix + k, "%s ", buffer);
17             k += strlen(buffer) + 1;
18         } else if (token == '(') {
19             push(&operators, "(");
20         } else if (token == ')') {
21             while (!isEmpty(&operators) && strcmp(peek(&operators), "(") != 0) {
22                 sprintf(postfix + k, "%s ", pop(&operators));
23                 k += 2;
24             }
25             pop(&operators);
26         } else if (isOperator(token)) {
27             while (!isEmpty(&operators) && precedence(peek(&operators)[0]) >= precedence(token)) {
28                 sprintf(postfix + k, "%s ", pop(&operators));
29                 k += 2;
30             }
31             char opStr[2] = {token, '\0'};
32             push(&operators, opStr);
33         }
34     }
35     while (!isEmpty(&operators)) {
36         sprintf(postfix + k, "%s ", pop(&operators));
37         k += 2;
38     }
39     postfix[k - 1] = '\0';
40 }
41
```

10. *infixToPrefix(char *infix, char *prefix)*

berfungsi untuk mengubah ekspresi infix menjadi prefix.

```
1 // Mengonversi infix ke prefix
2 void infixToPrefix(const char *infix, char *prefix) {
3     char reversed[MAX], postfix[MAX] = "";
4     strcpy(reversed, infix);
5     reverseString(reversed);
6
7     for (int i = 0; i < strlen(reversed); i++) {
8         if (reversed[i] == '(') reversed[i] = ')';
9         else if (reversed[i] == ')') reversed[i] = '(';
10    }
11    infixToPostfix(reversed, postfix);
12    reverseString(postfix);
13    strcpy(prefix, postfix);
14 }
```

11. *postfixToInfix(char *postfix, char *infix)*

berfungsi untuk mengubah ekspresi postfix menjadi infix.

```
1 // Mengonversi postfix ke infix
2 void postfixToInfix(const char *postfix, char *infix) {
3     Stack operands;
4     init(&operands);
5     int i = 0;
6     char token, buffer[MAX];
7
8     while ((token = postfix[i++]) != '\0') {
9         if (isalnum(token)) {
10             int j = 0;
11             while (isalnum(token)) {
12                 buffer[j++] = token;
13                 token = postfix[i++];
14             }
15             buffer[j] = '\0'; i--;
16             push(&operands, buffer);
17         } else if (isOperator(token)) {
18             char op2[MAX], op1[MAX], expr[MAX];
19             strcpy(op2, pop(&operands));
20             strcpy(op1, pop(&operands));
21             sprintf(expr, "(%s %c %s)", op1, token, op2);
22             push(&operands, expr);
23         }
24     }
25     strcpy(infix, pop(&operands));
26 }
27
```

12. *prefixToInfix(char *prefix, char *infix)*

berfungsi untuk mengubah ekspresi prefix menjadi infix.

```

1 // Mengonversi prefix ke infix
2 void prefixToInfix(const char *prefix, char *infix) {
3     Stack operands;
4     init(&operands);
5     int len = strlen(prefix);
6
7     for (int i = len - 1; i >= 0; i--) {
8         if (isdigit(prefix[i])) {
9             char temp[2] = {prefix[i], '\0'};
10            push(&operands, temp);
11        } else if (isOperator(prefix[i])) {
12            char op1[MAX], op2[MAX], expr[MAX];
13            strcpy(op1, pop(&operands));
14            strcpy(op2, pop(&operands));
15            sprintf(expr, "(%s %c %s)", op1, prefix[i], op2);
16            push(&operands, expr);
17        }
18    }
19    strcpy(infix, pop(&operands));
20 }
21

```

13. *prefixToPostfix(char *prefix, char *postfix)*

berfungsi untuk mengubah ekspresi prefix menjadi postfix.

```

1 // Mengonversi prefix ke postfix
2 void prefixToPostfix(const char *prefix, char *postfix) {
3     char infix[MAX];
4     prefixToInfix(prefix, infix);
5     infixToPostfix(infix, postfix);
6 }

```

14. *postfixToPrefix(char *postfix, char *prefix)*

berfungsi untuk mengubah ekspresi postfix menjadi prefix.

```

1 // Mengonversi postfix ke prefix
2 void postfixToPrefix(const char *postfix, char *prefix) {
3     char infix[MAX];
4     postfixToInfix(postfix, infix);
5     infixToPrefix(infix, prefix);
6 }

```

Untuk menjalankan semua fungsi yang ada, tentunya kita memerlukan fungsi main yang mana berfungsi untuk memanggil fungsi yang ingin dijalankan.

```
1  int main() {
2      char input[MAX], output[MAX] = "";
3      int pilihan;
4
5      do {
6          printf("\nPilih Konversi:\n");
7          printf("1. Infix ke Postfix\n");
8          printf("2. Postfix ke Infix\n");
9          printf("3. Infix ke Prefix\n");
10         printf("4. Prefix ke Infix\n");
11         printf("5. Prefix ke Postfix\n");
12         printf("6. Postfix ke Prefix\n");
13         printf("0. Keluar\n");
14         printf("Masukkan pilihan (0-6): ");
15         scanf("%d", &pilihan);
16         getchar();
17
18         if (pilihan == 0) {
19             printf("Terima kasih! Program selesai.\n");
20             break;
21         }
22
23         printf("Masukkan ekspresi: ");
24         fgets(input, MAX, stdin);
25         input[strcspn(input, "\n")] = 0;
26
27         if (pilihan == 1) infixToPostfix(input, output);
28         else if (pilihan == 2) postfixToInfix(input, output);
29         else if (pilihan == 3) infixToPrefix(input, output);
30         else if (pilihan == 4) prefixToInfix(input, output);
31         else if (pilihan == 5) prefixToPostfix(input, output);
32         else if (pilihan == 6) postfixToPrefix(input, output);
33         else {
34             printf("Pilihan tidak valid.\n");
35             continue;
36         }
37
38         printf("Hasil: %s\n", output);
39     } while (1);
```