Gabriel Chrisoberryl Guevara Kaawoan
1203230041
IF 03-03

Source Code :

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

// Struktur data untuk node dalam stack
struct Node {
    char data;
    struct Node* next;
};

// Fungsi untuk membuat node baru
struct Node* createNode(char data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Fungsi untuk mengecek apakah stack kosong
int isEmpty(struct Node* top) {
    return top == NULL;
}

// Fungsi untuk menambahkan elemen ke dalam stack
void push(struct Node** top, char data) {
    struct Node* newNode = createNode(data);
    newNode->next = *top;
    *top = newNode;
}

// Fungsi untuk menghapus elemen dari stack
char pop(struct Node** top) {
    if (isEmpty(*top)) {
        printf("Stack is empty\n");
        exit(EXIT_FAILURE);
    }
    struct Node* temp = *top;
    char popped = temp->data;
    *top = (*top)->next;
```

```
    free(temp);
    return popped;
}

// Fungsi untuk mengecek apakah tanda kurung berpasangan
int isBalanced(char expr[]) {
    struct Node* stack = NULL;
    for (int i = 0; expr[i] != '\0'; i++) {
        if (expr[i] == '(' || expr[i] == '[' || expr[i] == '{') {
            push(&stack, expr[i]);
        } else if (expr[i] == ')' || expr[i] == ']' || expr[i] == '}')
{
            if (isEmpty(stack)) {
                return 0;
            }
            char popped = pop(&stack);
            if ((expr[i] == ')' && popped != '(') ||
                (expr[i] == ']' && popped != '[') ||
                (expr[i] == '}' && popped != '{')) {
                return 0;
            }
        }
    }
    return isEmpty(stack);
}

int main() {
    char expr[100];
    printf("Masukkan urutan tanda kurung: ");
    scanf("%s", expr);

    if (isBalanced(expr)) {
        printf("YES\n");
    } else {
        printf("NO\n");
    }

    return 0;
}
```

Penjelasan Source Code :

Libraries

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

- Baris ini mendeklarasikan dua library standar yang digunakan dalam program:
  - **<stdio.h>** : Ini adalah library standar untuk input-output, yang menyediakan fungsi seperti **printf()** dan **scanf()**.
  - **<stdlib.h>** : Ini adalah library standar yang menyediakan fungsi-fungsi umum seperti alokasi memori dinamis (**malloc()**) dan fungsi **exit()**.

Struktur Data Stack

```
struct Node {
    char data;
    struct Node* next;
};
```

- Ini mendefinisikan struktur data untuk node dalam stack. Setiap node memiliki dua bagian:
  - **data** : Menyimpan karakter.
  - **next** : Pointer ke node berikutnya dalam stack.

Fungsi createNode()

```
struct Node* createNode(char data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
```

- Ini adalah fungsi untuk membuat node baru dalam stack.
- Pertama, itu mengalokasikan memori untuk node baru menggunakan **malloc()**.
- Kemudian, menetapkan karakter **data** ke node baru dan mengatur **next** menjadi **NULL**.
- Akhirnya, mengembalikan pointer ke node baru yang telah dibuat.

Fungsi isEmpty()

```
int isEmpty(struct Node* top) {
    return top == NULL;
}
```

- Fungsi ini memeriksa apakah stack kosong.
- Jika **top** (puncak stack) adalah **NULL**, itu berarti stack kosong, sehingga mengembalikan 1 (true).
- Jika **top** tidak **NULL**, itu berarti stack tidak kosong, sehingga mengembalikan 0 (false).

Fungsi push()

```
void push(struct Node** top, char data) {
    struct Node* newNode = createNode(data);
    newNode->next = *top;
    *top = newNode;
}
```

- Ini adalah fungsi untuk menambahkan elemen ke dalam stack.
- Membuat node baru dengan menggunakan fungsi **createNode()**.
- Mengatur **next** node baru menjadi **top** (node teratas saat ini).
- Mengatur **top** menjadi node baru yang telah dibuat.

Fungsi pop()

```
char pop(struct Node** top) {
    if (isEmpty(*top)) {
        printf("Stack is empty\n");
        exit(EXIT_FAILURE);
    }
    struct Node* temp = *top;
    char popped = temp->data;
    *top = (*top)->next;
    free(temp);
    return popped;
}
```

- Fungsi ini digunakan untuk menghapus elemen dari atas stack dan mengembalikan nilainya.
- Pertama, memeriksa apakah stack kosong menggunakan fungsi **isEmpty()**. Jika iya, mencetak pesan kesalahan dan keluar dari program.
- Jika stack tidak kosong, menyimpan node teratas dalam variabel sementara **temp**.
- Mengambil karakter dari node teratas dan menyimpannya dalam variabel **popped**.
- Menggeser **top** ke node berikutnya.
- Membebaskan memori yang dialokasikan untuk node teratas.
- Mengembalikan karakter yang diambil dari node teratas.

Fungsi isBalanced()

```
int isBalanced(char expr[]) {
    struct Node* stack = NULL;
    for (int i = 0; expr[i] != '\0'; i++) {
        if (expr[i] == '(' || expr[i] == '[' || expr[i] == '{') {
            push(&stack, expr[i]);
        } else if (expr[i] == ')' || expr[i] == ']' || expr[i] == '}')
{
            if (isEmpty(stack)) {
                return 0;
            }
            char popped = pop(&stack);
```

```
            if ((expr[i] == ')' && popped != '(') ||
                (expr[i] == ']' && popped != '[') ||
                (expr[i] == '}' && popped != '{')) {
                return 0;
            }
        }
    }
    return isEmpty(stack);
}
```

- Ini adalah fungsi inti yang memeriksa apakah urutan tanda kurung dalam string **expr** seimbang atau tidak.
- Iterasi melalui string **expr** menggunakan loop **for**.
- Jika karakter saat ini adalah tanda kurung pembuka ((, **[**, atau {), itu ditambahkan ke dalam stack menggunakan fungsi **push()**.
- Jika karakter saat ini adalah tanda kurung penutup (), **]**, atau }):
  - Jika stack kosong, berarti ada tanda kurung penutup tanpa tanda kurung pembuka yang sesuai, sehingga mengembalikan 0 (false).
  - Jika stack tidak kosong, mengambil tanda kurung pembuka dari stack menggunakan fungsi **pop()** dan memeriksa apakah tanda kurung penutup dan pembuka sesuai. Jika tidak sesuai, mengembalikan 0 (false).
- Setelah iterasi selesai, memeriksa apakah stack kosong atau tidak. Jika stack kosong, itu berarti semua tanda kurung berpasangan, sehingga mengembalikan 1 (true). Jika tidak kosong, itu berarti ada tanda kurung pembuka tanpa tanda kurung penutup yang sesuai, sehingga mengembalikan 0 (false).

Fungsi main()

```
int main() {
    char expr[100];
    printf("Masukkan urutan tanda kurung: ");
    scanf("%s", expr);

    if (isBalanced(expr)) {
        printf("YES\n");
    } else {
        printf("NO\n");
    }

    return 0;
}
```

- Fungsi utama dari program.
- Mendeklarasikan array **expr** untuk menyimpan urutan tanda kurung yang dimasukkan pengguna.
- Meminta pengguna untuk memasukkan urutan tanda kurung menggunakan **printf()** dan **scanf()**.

- Memanggil fungsi **isBalanced()** untuk memeriksa keseimbangan tanda kurung.
- Mencetak "YES" jika urutan tanda kurung seimbang dan "NO" jika tidak.
- Mengembalikan 0 untuk menandakan bahwa program berakhir dengan sukses.

Hasil Run Program :