Nama: Rizkia Putra Raditya
NIM : 1203230083
Kelas : IF 03-03

Stack

Source Code:

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

typedef char DataType;

typedef struct StackNode {
  DataType data;
  struct StackNode *next;
} StackNode;

typedef struct Stack {
  StackNode *top;
} Stack;

void push(Stack *stack, DataType data);
DataType pop(Stack *stack);
int isEmpty(Stack *stack);

int isBalanced(char *str) {
  Stack stack;
  stack.top = NULL;

  for (int i = 0; str[i] != '\0'; i++) {
    if (str[i] == '(' || str[i] == '{' || str[i] == '[') {
      push(&stack, str[i]);
    } else if (str[i] == ')' || str[i] == '}' || str[i] == ']') {
      if (isEmpty(&stack)) {
        return 0;
      }

      char poppedChar = pop(&stack);

      if ((poppedChar == '(' && str[i] != ')') ||
          (poppedChar == '{' && str[i] != '}') ||
          (poppedChar == '[' && str[i] != ']')) {
        return 0;
      }
    }
```

```
  }

  if (!isEmpty(&stack)) {
    return 0;
  }

  return 1;
}

void push(Stack *stack, DataType data) {
  StackNode *newNode = (StackNode *)malloc(sizeof(StackNode));
  newNode->data = data;
  newNode->next = stack->top;
  stack->top = newNode;
}

DataType pop(Stack *stack) {
  if (isEmpty(stack)) {
    return '\0';
  }

  StackNode *temp = stack->top;
  DataType data = temp->data;
  stack->top = temp->next;
  free(temp);

  return data;
}

int isEmpty(Stack *stack) {
  return stack->top == NULL;
}

int main() {
  char str[100];
  scanf("%s", str);

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

  return 0;
}
```

Output:

```
PS C:\Users\Molly> cd "d:\TUGAS PUTE\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
{[()]}
YES
PS D:\TUGAS PUTE> cd "d:\TUGAS PUTE\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
{[)}]
NO
PS D:\TUGAS PUTE> cd "d:\TUGAS PUTE\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
{{[[()]]}}
NO
PS D:\TUGAS PUTE> cd "d:\TUGAS PUTE\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
{{[[()]]}}
YES
PS D:\TUGAS PUTE> cd "d:\TUGAS PUTE\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
{[]()}()
YES
PS D:\TUGAS PUTE>
```

Penjelasan:

1. push(Stack *stack, DataType data)

Fungsi: Memasukkan elemen baru (data) ke tumpukan (stack).

Parameter:

stack: Pointer ke struktur Stack yang merepresentasikan tumpukan.

data: Elemen yang akan dimasukkan ke tumpukan (bertipe DataType, biasanya karakter).

Proses:

Mengalokasikan memori dinamis untuk simpul tumpukan baru (newNode).

Menyetel data simpul baru ke nilai data yang diberikan.

Menyetel next simpul baru ke simpul teratas (top) tumpukan saat ini.

Memperbarui top tumpukan untuk menunjuk ke simpul baru, menjadikannya simpul teratas yang baru.

2. DataType pop(Stack *stack)

Fungsi: Menghapus dan mengembalikan elemen teratas dari tumpukan.

Parameter:

stack: Pointer ke struktur Stack yang merepresentasikan tumpukan.

Proses:

Memeriksa apakah tumpukan kosong (isEmpty(stack)).

Jika kosong, mengembalikan karakter null ('\0').

Jika tumpukan tidak kosong:

Menyimpan simpul teratas (top) tumpukan ke dalam variabel sementara (temp).

Menyimpan data dari simpul teratas (temp->data) ke variabel data.

Memperbarui top tumpukan untuk menunjuk ke simpul berikutnya (temp->next).

Membebaskan memori yang dialokasikan untuk simpul teratas lama (free(temp)).

Mengembalikan nilai data yang diambil dari simpul teratas yang dihapus.

3. int isEmpty(Stack *stack)

Fungsi: Memeriksa apakah tumpukan kosong.

Parameter:

stack: Pointer ke struktur Stack yang merepresentasikan tumpukan.

Proses:

Mengembalikan nilai 1 (TRUE) jika top tumpukan bernilai NULL, yang menandakan tumpukan kosong.

Mengembalikan nilai 0 (FALSE) jika top tidak NULL, yang menandakan tumpukan tidak kosong.

4. int isBalanced(char *str)

Fungsi: Menentukan apakah tanda kurung dalam string str berpasangan dengan benar.

Parameter:

str: Pointer ke string yang berisi karakter tanda kurung.

Proses:

Menginisialisasi tumpukan stack dengan top bernilai NULL.

Iterasi melalui setiap karakter dalam string str:

Jika karakter adalah tanda kurung buka ('(', '{', '['):

Memasukkan karakter tersebut ke tumpukan menggunakan push(stack, str[i]).

Jika karakter adalah tanda kurung tutup (')', '}', ']'):

Memeriksa apakah tumpukan kosong (isEmpty(stack)).

Jika kosong, mengembalikan 0 (tanda kurung tidak seimbang).

Menghapus karakter teratas dari tumpukan menggunakan char poppedChar = pop(stack).

Membandingkan karakter yang dihapus (poppedChar) dengan karakter saat ini (str[i]):

Jika tidak cocok (poppedChar == '(' && str[i] != ')', dst.), mengembalikan 0 (tanda kurung tidak seimbang).

Setelah iterasi selesai:

Memeriksa apakah tumpukan kosong (!isEmpty(stack)).

Jika tidak kosong, mengembalikan 0 (tanda kurung tidak seimbang).

Jika semua karakter telah diproses dan tumpukan kosong, mengembalikan 1 (tanda kurung berpasangan).

5. main()

Fungsi: Fungsi utama program.

Proses:

Mendeklarasikan variabel string str untuk menyimpan input pengguna (maksimal 100 karakter).

Membaca string dari pengguna menggunakan scanf("%s", str).

Memanggil fungsi isBalanced(str) untuk menentukan apakah tanda kurung dalam str berpasangan.

Berdasarkan hasil `