Araya Aliya Citranaya

IF 03-01 / 1203230002

OTS STACK

Tanda Kurung Aja Berpasangan

Tanda kurung terdiri dari salah satu karakter berikut: (,), {, }, [,].

Dua tanda kurung dianggap berpasangan jika sebuah kurung pembuka (yaitu (, {, atau [ ) terdapat di sebelah kiri kurung penutup (yaitu), },]) dengan jenis yang sama persis. Ada tiga jenis pasangan yaitu (), {}, dan [].

Dua tanda kurung dianggap tidak berpasangan jika set tanda kurung yang menyertainya tidak seimbang. Misalnya, {([)} dianggap tidak seimbang karena konten diantara { dan } tidak seimbang, [memiliki pasangan tanda kurung penutup yang tidak sesuai yaitu).

Beberapa contoh tanda kurung yang berpasangan adalah []{}(), [({})]{}(), dan ({(){}[]})[].

Berdasarkan logika ini, kita mengatakan bahwa sebuah urutan tanda kurung dianggap seimbang jika kondisi-kondisi berikut terpenuhi:

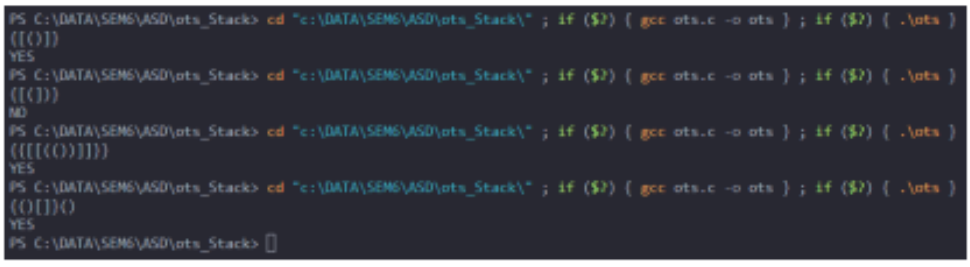
1. Tidak mengandung tanda kurung yang tidak seimbang.
2. Subset tanda kurung yang terdapat di antara pasangan tanda kurung yang seimbang juga merupakan pasangan tanda kurung yang seimbang.

Anda diminta untuk membuat fungsi **isBalanced** yang menerima input string berisikan satu set tanda kurung dan mengembalikan "YES" jika tanda kurung berpasangan, dan "NO" jika tidak.

Format Masukan:
Satu baris string yang berisikan suatu set tanda kurung.

Format Output:
Outputkan "YES" jika setiap urutan tanda kurung seimbang, atau "NO" jika tidak.

Catatan:
Gunakan fungsi-fungsi pada stack yang telah dijelaskan pada modul praktikum untuk menyelesaikan permasalahan ini.



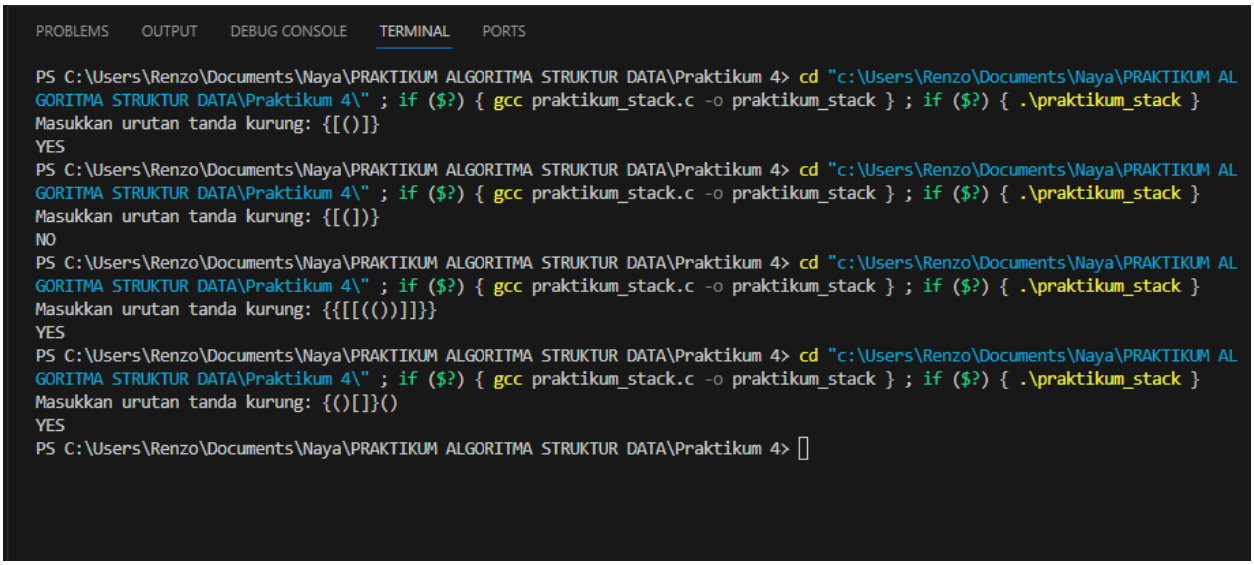
Source Code :

```c
#include <stdio.h> // untuk fungsi input-output standart
#include <stdbool.h> // untuk mendapatkan tipe data bool dan nilai konstan 'true' or 'false'


#define MAX_STACK_SIZE 100 // prosesor yang mendefinisikan konstanta 'max_stack_size' dengan nilai 100 yang digunakan untuk menentukan ukuran maksimum tumpukan

//mendefinisikan stack dengan struct
typedef struct stack { // untuk mendeklarasi tipe data turunan
    char items[MAX_STACK_SIZE]; // stack items untuk menyimpan elemen-elemen tumpukan
    int top; // untuk menyimpan indeks dari elemen teratas tumpukan
} Stack;

void initializeStack(Stack *stack) { // sebagai pointer ke tipe data stack, bertujuan untuk menginisialisasi atau menyiapkan sebuah tumpukan (stack) untuk digunakan
    stack->top = -1; // menandakan bahwa tumpukan paling atas kosong
}

// Fungsi untuk menambahkan elemen ke tumpukan
void push(Stack *stack, char item) { //menambah elemen berupa pointer stack ke tumppukan yang ingin di modifikasi dan item yang akan dimasukkan ke dalam tumpukan
    if (stack->top == MAX_STACK_SIZE - 1) { // untuk memeriksa tumpukan sudah penuh atau belum. 'stack-top' indeks teratas tumpukan. jika 'stack-top = max_stack_size -1' maka tumpukan dianggap penuh karena indeksnya sudah mencapai batas maximum
        printf("Tumpukan penuh\n"); // jika tumpukan sudah penuh, maka pesan "tumpukan penuh" akan dicetak
    } else { // jika kondisi sebelumnya tidak terpenuhi (belum penuh), maka kode dalam else akan dijalankan
        stack->items[++(stack->top)] = item; // 'stack-top' diinkremen tertampil indek berikutnya, kenudian item dimasukkan ke dalam tumpukan pada indeks baru yang diinkremen, dan 'stack-items' adalah array yang menyimpan elemen-elemen tumpukan dan 'stack-top' adalah indeks item akan dimasukkan
    }
}

// Fungsi untuk menghapus elemen dari tumpukan
char pop(Stack *stack) { // 'pop' untuk menghapus elemen dari tumpukan dan 'stack *stack' adalah pointer ke tumpukan yang ingin dimodifikasi
```

```
    if (stack->top == -1) { //untuk memeriksa apakah tumpukan kosong atau tidak 'stack-top' adalah indeks dari elemen teratas tumpukan. jika nilainya -1 berarti tumpukan kosong karena tidak ada elemen di dalamnya
        printf("Tumpukan kosong\n"); // jika tumpukan kosong, maka pesan "tumpukan kosong" akan dicetak
        return '\0'; // kembali ke karakter null, menandakan bahwa tidak ada nilai yang diambil dari
    } else {
        return stack->items[(stack->top)--]; // 'stack-top' di-dekremen menggunakan 'stack-top --' sehingga menunjukkan ke elemen teratas yang baru setelah penghapusan. karena nilai yang diambil dari tumpukan telah dikurangi dari 'stack-top' maka elemen tidak lagi menjadi bagian dari tumpukan
    }
}

bool isBalanced(const char* kurung) { // definisi fungsi 'isBalanced', menerima variabel kurung yang merupakan string dan akan diperiksa apakah urutan kurungnya seimbang atau tidak. tipe data bool adalah kembalian, yang akan mengindikasikan apakah string seimbang atau tidak
    Stack stack; // variabel stack adalah tumpukan yang digunakan untuk menyimpan kurung-kurung yang ditemukan dalam string
    initializeStack(&stack); // untuk menginisialisasi tumpukan sebelum digunakan, fungsi ini akan mengatur indeks teratas 'top' menjadi-1 yang menandakan bahwa tumpukan kosong

    for (int i = 0; kurung[i] != '\0'; i++) { // loop berlangsung selama karakter yang sedang diperiksa bukanlah karakter null terminator yang menandakan akhir dari string
        if (kurung[i] == '(' || kurung[i] == '[' || kurung[i] == '{') { //memeriksa apakah karakter saat ini adalah salah satu dari tiga jenis kurung buka. jika iya, karakter tersebut akan dimasukkan ke dalam tumpukan menggunakan fungsi'push'
            push(&stack, kurung[i]); // memanggil fungsi 'push' untuk memasukkan karakter kurung buka ke dalam tumpukan
        } else if (kurung[i] == ')' || kurung[i] == ']' || kurung[i] == '}') { // memeriksa apakah karakter saat ini adalah salah satu dari tiga jenis kurung tutup, jika iya akan dilakukan pengecekan apakah tumpukan kosong, karena harus ada kurung terbuka yang berpasangan untuk kurung tutup tersebut
            if (stack.top == -1) { // memeriksa apakah tumpukan kosong, jika iya, artinya tidak ada kurung terbuka yang berpasangan dengan kurung tutup saat ini, sehingga urutan kurung tidak seimbang
                return false; // maka akan mengembalikan nilai else dalam if di fungsi main atau akan langsung mengeprint no
            }
```

```
            char menjadi = pop(&stack); // memanggil fungsi 'pop' untuk menghapus
dan mengambil karakter teratas dari tumpukan. karakter ini akan menjadi kurung
terbuka yang berpasangan dengan kurung tutup yang sedang diproses saat ini
            if ((kurung[i] == ')' && menjadi != '(') || // jika varibel kurung
memiliki inisialisasi nilai = karakter ')' dan pasangan != '('
                (kurung[i] == ']' && menjadi != '[') || //atau varibel kurung
memiliki inisialisasi nilai = karakter ']' dan pasangan != '['
                (kurung[i] == '}' && menjadi != '{')) { //dan varibel kurung
memiliki inisialisasi nilai = karakter '}' dan pasangan != '{'
                return false; // akan mengembalikan nilai else dalam if di fungsi
main atau akan langsung mengeprint no
            }
        }
    }

    return stack.top == -1; // fungsi akan terus berjalan hingga nilai teratas
dalam stack habis
}

int main() { // fungsi main merupakan titik masuk utama program
    char kurung[MAX_STACK_SIZE]; // mendeklarasi array 'kurung' yang akan
digunakan untuk menyimpan urutan tanda kurung yang dimasukkan oleh pengguna.
ukuran array ini sama dengan 'max_stack_size' yang kemungkinan adalah konstanta
yang didefinisikan sebelumnya
    printf("Masukkan urutan tanda kurung: "); // mencetak pesan untuk meminta
pengguna memasukkan urutan tanda kurung
    scanf("%s", kurung); // membaca input dari pengguna dan menyimpannya dalam
array 'kurung'

    if (isBalanced(kurung)) { // memanggil fungsi'isBalanced' untuk memeriksa
apakah urutan tanda kurung yang dimasukkan oleh pengguna seimbang atau tidak.
jika true, artinya urutan kurang seimbang, maka blok kode dalam 'if' akan
dijalankan
        printf("YES\n"); // mencetak YES yang menandakan bahwa urutan tanda
kurung yang dimasukkan oleh pengguna seimbang
    } else { // jika hasil 'isBalanced(kurung)' adalah false artinya urutan
kurung tidak seimbang, maka blok kode dalam else akan di jalankan
        printf("NO\n"); // mencetak NO yang menandakan bahwa urutan tanda kurung
yang dimasukkan oleh pengguna tidak seimbang
    }

    return 0; // memberi nilai kembali pada sistem operasi, menandakan status
keluar dari program
}
```

Output :


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\Renzo\Documents\Naya\PRAKTIKUM ALGORITMA STRUKTUR DATA\Praktikum 4> cd "c:\Users\Renzo\Documents\Naya\PRAKTIKUM ALGORITMA STRUKTUR DATA\Praktikum 4\" ; if ($?) { gcc praktikum_stack.c -o praktikum_stack } ; if ($?) { .\praktikum_stack }
Masukkan urutan tanda kurung: {[()]}
YES
PS C:\Users\Renzo\Documents\Naya\PRAKTIKUM ALGORITMA STRUKTUR DATA\Praktikum 4> cd "c:\Users\Renzo\Documents\Naya\PRAKTIKUM ALGORITMA STRUKTUR DATA\Praktikum 4\" ; if ($?) { gcc praktikum_stack.c -o praktikum_stack } ; if ($?) { .\praktikum_stack }
Masukkan urutan tanda kurung: {[(])}
NO
PS C:\Users\Renzo\Documents\Naya\PRAKTIKUM ALGORITMA STRUKTUR DATA\Praktikum 4> cd "c:\Users\Renzo\Documents\Naya\PRAKTIKUM ALGORITMA STRUKTUR DATA\Praktikum 4\" ; if ($?) { gcc praktikum_stack.c -o praktikum_stack } ; if ($?) { .\praktikum_stack }
Masukkan urutan tanda kurung: {{[[(())]]}}
YES
PS C:\Users\Renzo\Documents\Naya\PRAKTIKUM ALGORITMA STRUKTUR DATA\Praktikum 4> cd "c:\Users\Renzo\Documents\Naya\PRAKTIKUM ALGORITMA STRUKTUR DATA\Praktikum 4\" ; if ($?) { gcc praktikum_stack.c -o praktikum_stack } ; if ($?) { .\praktikum_stack }
Masukkan urutan tanda kurung: {()[]}()
YES
PS C:\Users\Renzo\Documents\Naya\PRAKTIKUM ALGORITMA STRUKTUR DATA\Praktikum 4>