NAMA : MOCH. JOHAN BINTANG PRATAMA

NIM : 1203230063

Code.c

```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;
}
```

Penjelasan

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

Bagian tersebut adalah bagian awal dari sebuah program dalam bahasa C. Dua direktif praprosesor (#include <stdio.h> dan #include <stdlib.h>) digunakan untuk menyertakan file header yang diperlukan untuk fungsi input/output standar dan alokasi memori. Kemudian, fungsi main() dideklarasikan sebagai titik masuk utama untuk program, yang akan dieksekusi secara otomatis oleh sistem operasi.

```
typedef char DataType;
```

Pernyataan `typedef char DataType;` digunakan untuk membuat alias tipe data baru dalam bahasa C. Dalam hal ini, tipe data `char` diberi nama baru yaitu `DataType`. Ini memungkinkan programmer untuk menggunakan `DataType` sebagai pengganti untuk tipe data `char` di seluruh kode program, membuat kode lebih mudah dibaca dan lebih mudah diubah jika diperlukan.

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

Pernyataan `typedef struct StackNode { DataType data; struct StackNode *next; } StackNode;` digunakan untuk mendefinisikan sebuah struktur yang disebut `StackNode`. Struktur ini memiliki dua anggota: `data`, yang merupakan variabel dari tipe data yang diberi alias `DataType`, dan `next`, yang merupakan pointer ke struktur `StackNode` berikutnya. Kemudian, alias `StackNode` digunakan untuk menyederhanakan pemanggilan struktur ini di seluruh program.

```
typedef struct Stack {
  StackNode *top;
} Stack;
```

Pernyataan `typedef struct Stack { StackNode *top; } Stack;` digunakan untuk mendefinisikan sebuah struktur yang disebut `Stack`. Struktur ini memiliki satu anggota yaitu `top`, yang merupakan pointer ke `StackNode` teratas dalam tumpukan. Dengan demikian, struktur `Stack` mengelompokkan pointer `top` yang menunjuk ke elemen teratas dalam tumpukan. Selanjutnya, alias `Stack` digunakan untuk membuat pemanggilan struktur ini menjadi lebih sederhana di seluruh program. Ini memungkinkan untuk merepresentasikan dan mengelola tumpukan dengan lebih mudah dan terorganisir.

```
void push(Stack *stack, DataType data);
```

Fungsi ini menambahkan elemen baru ke tumpukan,memiliki dua parameter yaitu stack (pointer ke tumpukan di mana elemen akan ditambahkan) dan data (data yang akan dimasukkan ke dalam tumpukan)

```
DataType pop(Stack *stack);
```

Fungsi ini menghapus dan mengembalikan elemen teratas dari tumpukan, memiliki satu parameter: stack (pointer ke tumpukan dari mana elemen akan dihapus),mengembalikan nilai dari tipe data DataType, yaitu data yang dihapus dari tumpukan.

```
int isEmpty(Stack *stack);\
```

Fungsi ini memeriksa apakah tumpukan kosong atau tidak, memiliki satu parameter: stack (pointer ke tumpukan yang akan diperiksa), mengembalikan nilai integer: 1 jika tumpukan kosong, dan 0 jika tumpukan tidak kosong

```
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;
      }
    }
  }
```

Fungsi ini menerima parameter str, yang merupakan pointer ke string yang akan diperiksa, sebuah tumpukan (stack) dibuat menggunakan variabel lokal,jika karakter saat ini adalah tanda buka kurung ((, {, atau [), karakter tersebut dimasukkan ke dalam tumpukan menggunakan fungsi push.

Jika karakter saat ini adalah tanda tutup kurung (), }, atau]), dilakukan pemeriksaan:

Jika tumpukan kosong, ini berarti ada tanda kurung tutup tanpa pasangan yang sesuai, sehingga fungsi mengembalikan 0 (tidak seimbang).

Jika tumpukan tidak kosong, karakter teratas dari tumpukan dihapus menggunakan fungsi pop, dan dilakukan pemeriksaan apakah tanda kurung tersebut berpasangan dengan karakter saat ini. Jika tidak berpasangan, fungsi mengembalikan 0 (tidak seimbang).

Setelah iterasi selesai, jika tumpukan tidak kosong, itu berarti masih ada tanda kurung buka tanpa pasangan yang sesuai, sehingga fungsi mengembalikan 0 (tidak seimbang).

Jika tumpukan kosong setelah iterasi, itu berarti semua tanda kurung telah dipasangkan dengan benar, sehingga fungsi mengembalikan 1 (seimbang).

```
 if (!isEmpty(&stack)) {
```

```
    return 0;
  }

  return 1;
}
```
Pada blok kode tersebut, setelah iterasi selesai, dilakukan pengecekan apakah tumpukan tidak kosong. Jika tumpukan tidak kosong, artinya masih ada tanda kurung buka yang tidak memiliki pasangan yang sesuai, sehingga fungsi mengembalikan nilai 0 (tidak seimbang). Jika tumpukan kosong setelah iterasi, artinya semua tanda kurung telah dipasangkan dengan benar, sehingga fungsi mengembalikan nilai 1 (seimbang). Dengan demikian, blok kode tersebut memastikan bahwa tanda kurung dalam string `str` seimbang atau tidak.

```
void push(Stack *stack, DataType data) {
  StackNode *newNode = (StackNode *)malloc(sizeof(StackNode));
  newNode->data = data;
  newNode->next = stack->top;
  stack->top = newNode;
}
```
Fungsi menerima dua parameter: stack, yang merupakan pointer ke tumpukan di mana elemen akan ditambahkan, dan data, yang merupakan data yang akan dimasukkan ke dalam tumpukan.

Pertama, sebuah node baru (newNode) dialokasikan di dalam memori menggunakan malloc dengan ukuran yang sesuai dengan StackNode.

Data yang diberikan (parameter data) disimpan dalam atribut data dari node baru tersebut.

Pointer next dari node baru diatur untuk menunjuk ke elemen teratas (top) dari tumpukan.

Pointer top dari tumpukan diperbarui untuk menunjuk ke node baru, sehingga node baru tersebut menjadi elemen teratas dalam tumpukan.

```
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;
}
```
Fungsi `pop` digunakan untuk menghapus dan mengembalikan elemen teratas dari tumpukan. Berikut adalah penjelasan singkat dari implementasi fungsi ini:

Fungsi menerima satu parameter, yaitu `stack`, yang merupakan pointer ke tumpukan dari mana elemen akan dihapus.Pertama, dilakukan pemeriksaan apakah tumpukan kosong

menggunakan fungsi `isEmpty`. Jika tumpukan kosong, fungsi mengembalikan nilai null karakter (`'\0'`), menandakan bahwa tidak ada elemen yang dapat dihapus dari tumpukan.Jika tumpukan tidak kosong, node teratas dari tumpukan dihapus. Variabel `temp` digunakan untuk menyimpan alamat dari node teratas.Data dari node teratas disimpan dalam variabel `data`.Pointer `top` dari tumpukan diperbarui untuk menunjuk ke node berikutnya, sehingga node teratas dihapus dari tumpukan.Node yang sudah tidak dibutuhkan kemudian dibebaskan dari memori menggunakan fungsi `free`.Fungsi mengembalikan data yang dihapus dari tumpukan.

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

Fungsi `isEmpty` digunakan untuk memeriksa apakah tumpukan kosong atau tidak.

Fungsi menerima satu parameter, yaitu `stack`, yang merupakan pointer ke tumpukan yang akan diperiksa,lalu mengembalikan nilai hasil perbandingan antara `stack->top` dan `NULL`. Jika `stack->top` adalah `NULL`, maka tumpukan dianggap kosong dan fungsi mengembalikan nilai `1`. Jika `stack->top` tidak `NULL`, maka tumpukan dianggap tidak kosong dan fungsi mengembalikan nilai 0, dengan demikian, fungsi ini secara efektif memeriksa apakah tumpukan memiliki elemen atau tidak dengan menguji apakah pointer `top`-nya null atau tidak.

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

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

  return 0;
}
```

Fungsi `main()` adalah fungsi utama dari program.Pertama, sebuah array karakter `str` dengan ukuran 100 dibuat untuk menyimpan string, lalu string dibaca menggunakan fungsi `scanf()` dan disimpan dalam array `str`, kemudian, fungsi `isBalanced()` dipanggil dengan argumen `str` untuk memeriksa apakah urutan tanda kurung dalam string tersebut seimbang atau tidak, hasil dari pemanggilan `isBalanced()` diuji menggunakan struktur pengkondisian `if-else`, jika hasilnya benar (seimbang), maka program mencetak "YES", jika hasilnya salah (tidak seimbang), maka program mencetak "NO", akhirnya, fungsi `main()` mengembalikan nilai 0, menandakan bahwa program telah berakhir dengan sukses kepada sistem operasi.

Output

```
{[()]}
YES
PS D:\IT TELKOM\Tugas\Semester 2\Algoritma Pemrograman\Praktikum 6> cd "d:\IT TELKOM\Tugas\Semester 2\Algoritma Pemrograman\Praktikum 6\" ; if ($?) { gcc praktikum6.c -o praktikum6 } ; if ($?) { .\prak
tikum6 }
{[()]}
NO
PS D:\IT TELKOM\Tugas\Semester 2\Algoritma Pemrograman\Praktikum 6> cd "d:\IT TELKOM\Tugas\Semester 2\Algoritma Pemrograman\Praktikum 6\" ; if ($?) { gcc praktikum6.c -o praktikum6 } ; if ($?) { .\prak
tikum6 }
{{[[(())]]}}
YES
PS D:\IT TELKOM\Tugas\Semester 2\Algoritma Pemrograman\Praktikum 6> cd "d:\IT TELKOM\Tugas\Semester 2\Algoritma Pemrograman\Praktikum 6\" ; if ($?) { gcc praktikum6.c -o praktikum6 } ; if ($?) { .\prak
tikum6 }
{()[]}()
YES
```