

代码：

typedef char STDataType;

typedef struct Stack

{

    STDataType\* a;

    int top;  //栈顶

    int capacity;  //容量,方便增容

}Stack;

void StackInit(Stack\* pst);

void StackDestory(Stack\* pst);

//性质决定在栈顶出入数据

void StackPush(Stack\* pst,STDataType x);

void StackPop(Stack\* pst);

STDataType StackTop(Stack\* pst);

bool StackEmpty(Stack\* pst);

int StackSize(Stack\* pst);

void StackInit(Stack\* pst)

{

    assert(pst);

    //pst->a = NULL;

    //pst->top = 0;

    //pst->capacity = 0;

    //或者一上来给一些空间

    pst->a = (STDataType\*)malloc(sizeof(STDataType) \* 4);

    pst->top = 0;

    pst->capacity = 4;

}

void StackDestory(Stack\* pst)

{

    assert(pst);

    free(pst->a);

    pst->a = NULL;

    pst->capacity = pst->top = 0;

}

void StackPush(Stack\* pst, STDataType x)

{

    assert(pst);

    if (pst->top == pst->capacity)

    {

        STDataType\* tmp = (STDataType\*)realloc(pst->a,sizeof(STDataType) \* pst->capacity \* 2);

        if (tmp == NULL)

        {

            printf("realloc fail\n");

            exit(-1);  //结束整个程序

        }

        pst->a = tmp;

        pst->capacity \*= 2;

    }

    //入数据

    pst->a[pst->top] = x;

    pst->top++;

}

void StackPop(Stack\* pst)

{

    assert(pst);

    assert(!StackEmpty(pst));

    pst->top--;

}

STDataType StackTop(Stack\* pst)

{

    assert(pst);

    assert(!StackEmpty(pst));

    return pst->a[pst->top - 1];

}

bool StackEmpty(Stack\* pst)

{

    assert(pst);

    return pst->top == 0;

}

int StackSize(Stack\* pst)

{

    assert(pst);

    return pst->top;

}

bool isValid(char \* s){

    Stack st;

    StackInit(&st);

    while(\*s)

    {

        //左括号入栈,右括号出栈

        //右括号找最近的左括号匹配

        if((\*s == '(') ||　(\*s == '[') || (\*s == '{'))

        {

            StackPush(&st,\*s);

            ++s;

        }

        else

        {

            //为空,说明没有前括号

            if(StackEmpty(&st))

               StackDestory(&st);

               return false;

            char top = StcakTop(&st);

            if(top == '[' && \*s != ']')

               StackDestory(&st);

               return false;

            else if(top == '(' && \*s != ')')

               StackDestory(&st);

               return false;

            else if(top == '{' && \*s != '}')

               StackDestory(&st);

               return false;

            else

            {

               //走到这里才意味着匹配

               StackPop(&st);

               ++s;

            }

        }

    }

    bool ret = StackEmpty(&st);

    StackDestory(&st);

    return ret;

}