栈的应用之括号匹配

#include <stdio.h>

#include <stdlib.h>

# define MAXSIZE 100

typedef int datatype;

typedef struct{

datatype a[MAXSIZE];

int top;

}sequence\_stack;

//栈的初始化

void init(sequence\_stack \*st);

//判断栈是否为空

int empty(sequence\_stack st);

//读栈顶的节点值

datatype read(sequence\_stack st);

//栈的插入操作(进栈)

void push(sequence\_stack \*st,datatype x);

//栈的删除操作(出栈)

void pop(sequence\_stack \*st);

//括号匹配

int match\_kouhao(char c[]);

int main()

{

sequence\_stack Stu;

init(&Stu);

for(int i = 0; i < 10;i ++)

push(&Stu,i);

for(int i = 0;i < 10;i ++)

{

printf("%d",read(Stu));

pop(&Stu);

}

char c1[] = "{}[#";

char c2[] = "{}[]()#";

if(match\_kouhao(c1))

printf("\n\n括号匹配成功！\n");

else

printf("\n\n括号匹配失败！\n");

if(match\_kouhao(c2))

printf("\n\n括号匹配成功！\n");

else

printf("\n\n括号匹配失败！\n");

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*函数功能：栈的初始化 \*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void init(sequence\_stack \*st)

{

st->top = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*函数功能：判断栈是否为空 \*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int empty(sequence\_stack st)

{

return(st.top ? 0 : 1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*函数功能：读栈顶的节点值 \*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

datatype read(sequence\_stack st)

{

if(empty(st))

{

printf("\n栈是空的！");

exit(1);

}

else

return st.a[st.top - 1];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*函数功能：栈的插入操作(进栈) \*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void push(sequence\_stack \*st,datatype x)

{

if(st->top == MAXSIZE)

{

printf("\n栈已经满了！");

exit(1);

}

st->a[st->top] = x;

st->top ++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*函数功能：栈的删除操作(出栈) \*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void pop(sequence\_stack \*st)

{

if(st->top == 0)

{

printf("\n这是一个空的栈！");

exit(0);

}

st->top --;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*函数功能：栈的删除操作(出栈) \*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int match\_kouhao(char c[])

{

int i = 0;

sequence\_stack s;

init(&s);

while(c[i] != '#')

{

switch(c[i])

{

case '{':

case '[':

case '(':

push(&s,c[i]);

break;

case '}':

if(!empty(s) && read(s) == '{')

{

pop(&s);

break;

}

else

return 0;

case ']':

if(!empty(s) && read(s) == '[')

{

pop(&s);

break;

}

else

return 0;

case ')':

if(!empty(s) && read(s) == '(')

{

pop(&s);

break;

}

else

return 0;

}

i ++;

}

return (empty(s));

}

运行结果：

