# 实验三：单链表

#### （一）实验题目

题目一**：利用顺序栈实现将10进制数转换为16进制数。**

* + 1. **第一组数据：4**
    2. **第二组数据：11**
    3. **第三组数据：254**
    4. **第四组数据：1357**

题目二：**对一个合法的数学表达式来说，其中的各大小括号“{”，“}”，“[”，“]”，“(”和“)”应是相互匹配的。设计算法对以字符串形式读入的表达式S，判断其中的各括号是否是匹配的。**

**\* 栈扩展实验**

* + **非必做内容，有兴趣的同学选做，**

**<1>（假设栈的输入序列为1、2、3、...、n，设计算法实现对给定的一个序列，判定其是否是此栈合法的输出序列。**

**<2>假设栈的输入序列为1、2、3、...、n，设计算法求出所有可能的出栈序列。**

**<3>利用栈求解算术表达式的值。**

#### （二） 实验源码

#include <iostream>

#include "stack1.cpp"

using namespace std;

void transform(){

int data;

cout<<"请输入一个十进制数：";

cin>>data;

stack<int>s;

while(data%16!=0){

if(s.fullstack()){

cout<<"can't push because the stack is full"<<endl;

break;

}

else{

s.push(data%16);

data=data/16;

}

}

cout<<"它的十六进制数为：0x";

while(!s.empty()){

cout<<s.pop();

}

cout<<endl;

}

void judgement(){

char str[100];

stack<char>s;

cout<<"请输入一个表达式：";

cin>>str;

for(int i=0;str[i]!=0;i++){

switch(str[i]){

case '{':

if(s.gettop()=='('||s.gettop()=='['||s.gettop()=='{'){

cout<<"该表达式不合法！"<<endl;

return ;

}

s.push(str[i]);

break;

case '[':

if(s.gettop()=='('||s.gettop()=='['){

cout<<"该表达式不合法！"<<endl;

return ;

}

s.push(str[i]);

break;

case '(':

if(s.gettop()=='('){

cout<<"该表达式不合法！"<<endl;

return ;

}

s.push(str[i]);

break;

case '}':

if(s.pop()!='{'){

cout<<"该表达式不合法！"<<endl;

return ;

}

break;

case ']':

if(s.pop()!='['){

cout<<"该表达式不合法！"<<endl;

return ;

}

break;

case ')':

if(s.pop()!='('){

cout<<"该表达式不合法！"<<endl;

return ;

}

break;

}

}

if(!s.empty()){

cout<<"该表达式不合法！"<<endl;

return ;

}

cout<<"该表达式合法"<<endl;

}

int main(int argc, char\*\* argv) {

transform();

judgement();

return 0;

}

#ifndef STACK1\_H

#define STACK1\_H

#include <iostream>

using namespace std;

template<class T>

class stack

{

public:

stack();

~stack();

T gettop(){return a[p];}

void push(T x);

bool fullstack();

bool empty();

T pop();

T \*a;

int p;

int maxsize;

};

template<typename T>

stack<T>::stack()

{

maxsize=10000;

a=new T[maxsize];

p=-1;

}

template<typename T>

stack<T>::~stack()

{

delete []a;

}

template<typename T>

bool stack<T>:: fullstack(){

if(p==maxsize){

return true;

}

return false;

}

template<typename T>

bool stack<T>::empty(){

if(p==-1){

return true;

}

return false;

}

template<typename T>

void stack<T>::push(T x){

if(fullstack()){

cout<<"the stack is full!"<<endl;

return ;

}

p++;

a[p]=x;

}

template<typename T>

T stack<T>::pop(){

if(empty()){

cout<<"the stack is empty!"<<endl;

return 0;

}

p--;

return a[p+1];

}

#endif

链式栈：

#include<iostream>

using namespace std;

template<typename T>

struct node{

T data;

node<T> \*next;

};

template<class T>

class stack

{

public:

stack();

~stack();

T gettop(){return p->data;}

void push(T x);

bool empty();

T pop();

node<T>\* p;

};

template<typename T>

stack<T>::stack()

{

p=NULL;

}

template<typename T>

stack<T>::~stack()

{

delete []p;

}

template<typename T>

bool stack<T>::empty(){

if(p==NULL){

return true;

}

return false;

}

template<typename T>

void stack<T>::push(T x){

node<T> \*new\_node =new node<T>;

new\_node->data=x;

new\_node->next=p;

p=new\_node;

}

template<typename T>

T stack<T>::pop(){

if(empty()){

cout<<"the stack is empty!"<<endl;

return 0;

}

node<T> \*pre=p;

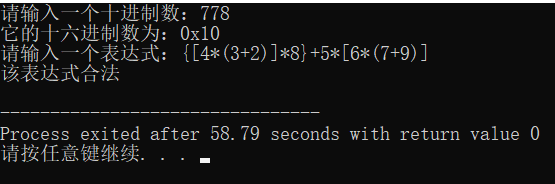
p=p->next;

return pre->data;

}

#endif

#### （三） 实验截图



#### （四） 心得体会