**链栈的操作**

1. 定义
   1. 参照线性表的操作
2. 代码

#include <iostream>

#include <stdio.h>

#include <iomanip>

#include <string>

#include <assert.h>

#include <time.h>

#include <string>

template <typename T>

class SQLstack{

struct SLNode{

T data;

SLNode \*next;

};

SLNode \*head;

public:

//初始化函数

SQLstack(){

head = new SLNode();

head->next = 0;

}

virtual ~SQLstack(){

SLNode \*p = new SLNode();

p = head->next;

while(p){

head->next = p->next;

delete p;

p = head;

}

delete head;

}

//入栈函数

bool push(T e){

SLNode \*p = new SLNode();

p->data = e;

p->next = head->next;

head->next = p;

return true;

}

//出栈函数

bool pop(){

//栈是否为空

if(isEmpty()) return false;

SLNode \*p = head->next;

head->next = p->next;

delete p;

return true;

}

//输出栈顶元素

T& top(){

if(isEmpty()) throw "empty";

return head->next->data;

}

bool isEmpty(){

return head->next == 0;

}

void traverse(void (\*func)(T e)){

SLNode \*p = head;

while(p){

func(p->data);

p = p->next;

}

}

//private:

// //增加内存函数

// void relloc(){

// T \*p = new T[2\*capacity]; // 增加内存空间

// for(int i = 0; i < capacity; i++){

// p[i] = data[i];

// delete[] data;

// data = p;

// }

// }

};

template <typename T>

void printStack(T e){

std::cout<<e<<" ";

}

int main()

{

SQLstack<char> sql;

sql.push('A');

sql.push('B');

// sql.traverse(printStack);

// std::cout<<std::endl;

// sql.pop();

// sql.traverse(printStack);

// std::cout<<std::endl;

// char e = sql.top();

// std::cout<<e<<std::endl;

//遍历也可以

while(!sql.isEmpty()){

char e = sql.top();

std::cout<<e<<" ";

sql.pop();

}

}