**Черкасов А. А-06-19 Вариант 24**

1. Из значений элементов двух заданных стеков собрать новый стек, состоящий из элементов 1, 2 и еще раз первого стека.

7

5

4

0

7

4

4

Top11

Top21

Top21

7

4

7

23)

7

4

Top11

7

0

4

Top31

5

4

7

24)

**Предопределенные элементы**

typedef int TInfo; //удобство для написания абстрактных функций

struct stack{ //Элемент стэка

TInfo data; //содержание

stack\* back; //указатель на предыдущий

};

**Функциональные тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Исходные данные** | **Ожидаемый результат (стек слева -> направо = снизу -> вверх)** | **Смысл теста** |
| **1** | 1  2 | Стек 1: 1  Стек 2: 2  Результат задачи:1 2 1 | Один элемент в каждом стеке. Проверка работоспособности |
| **2** | 1 2 3  4 5 6 | Стек 1: 1 2 3  Стек 2: 4 5 6  Результат задачи: 3 2 1 6 5 4 3 2 1 | Одинаковое количество элементов в стеках. Проверка корректности создания 3-го |
| **3** | 4 5 6 7  11 12 13 | Стек 1: 4 5 6 7  Стек 2:11 12 13  Результат задачи: 7 6 5 4 13 12 11 7 6 5 4 | Разное количество элементов в стеках. 1>2 Проверка корректности создания 3-го |
| **4** | 1 2  4 5 6 | Стек 1: 1 2  Стек 2: 4 5 6  Результат задачи: 2 1 6 5 4 2 1 | Разное количество элементов в стеках. 1<2 Проверка корректности создания 3-го |
| **5** | 1 4 3 7 | Стек 1: 1 4 3 7  Стек 2: пустой стек  Результат задачи: 7 3 4 1 7 3 4 1 | Пустой второй стек  Создан третий, в котором двойное количество элементов первого |
| **6** | 4 5 1 2 | Стек 1: пустой стек  Стек 2: 4 5 1 2  Результат задачи: 2 1 5 4 | Пустой первый стек  Создан третий только из элементов второго |
| **7** |  | Стек 1: пустой стек  Стек 2: пустой стек  Результат задачи: пустой стек | Два пустых стека  Третий тоже пуст |

**Код**

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.16)

project(Lab10)

set(CMAKE\_CXX\_STANDARD 17)

add\_executable(Lab10 main.cpp)

**main.cpp**

#include "Func.h"

#include <cctype>

**int main**(int argc,char\*\* argv) { //Параметры входные файлы для стэков 1й и 2й соответственно

system("chcp 65001");//поддержка кириллицы

stack \*fStack,\*sStack,\*rStack;//стэки (1й 2й, результат)

InitNewStack(&fStack);

InitNewStack(&sStack);

InitNewStack(&rStack);

bool end = false;

printf("Лабораторная работа #10\n");

while(!end)

{

printf("\'N\' ввести стек с клавиатуры\n"

"\'M\' ввести стек из файла\n"

"\'V\' ввывести содержимое стека\n"

"\'T\' решение задачи\n"

"\'D\' освобождение стеков\n"

"\'X\' Выход\n");

int num;

char act;

scanf("%c",&act);

fflush(stdin);

act =(char)tolower(act);

switch(act)

{

case 'n':

printf("В 1й или 2й стек?\n");

scanf("%d",&num); fflush(stdin);

if (num==1) {InKeyboard(&fStack);}

else InKeyboard(&sStack);

break;

case 'm':

printf("В 1й или 2й стек?\n");

scanf("%d",&num); fflush(stdin);

if (num==1) InFile(&fStack, argv[1]);

else InFile(&sStack, argv[2]);

break;

case 'v':

printf("Первый стек:\n");

stackShow(&fStack);

printf("Второй стек:\n");

stackShow(&sStack);

printf("Результат задачи:\n");

stackShow(&rStack);

break;

case 't':

rStack=task(fStack,sStack);

break;

case 'd':

InitNewStack(&fStack);

InitNewStack(&sStack);

InitNewStack(&rStack);

break;

case 'x':

end=true;

break;

default:

printf("Неизвестная команда\n");

break;

}

printf("press ENTER\n"); getc(stdin); fflush(stdin);

}

End(&fStack,&sStack,&rStack); //Освобождение памяти

return 0;

}

**Func.h**

#include "Hstack.h"

#include <cstdio>

**void InitNewStack**(stack\*\* pstack){ //Новый стек

if (!pstack) s\_destructor(pstack); //Если не пустой - очищаем

(\*pstack) = nullptr;

}

**void InKeyboard**(stack\*\* pstack) { //Ввод по клавиатуре

InitNewStack(pstack);

int Data = 0,counter=0;

printf("Стоп сигнал \'-1\'\n");

while (Data!=-1)

{

scanf("%d",&Data); fflush(stdin);

if (Data!=-1) {

s\_push(pstack, Data);

counter++;

}

}

if (counter==0) printf("Создан пустой стек\n");

else printf("Создан стек с %d элементами\n",counter);

}

**void InFile**(stack\*\* pstack, char\* filename) { //ВВод из файла

InitNewStack(pstack); //пересоздание стэка

FILE \*pFile = fopen(filename,"r"); //подключение файла

int Data,counter=0;

do {

fscanf(pFile,"%d",&Data);

if (!feof(pFile)) {

s\_push(pstack, Data);

counter++;

}

}while(!feof(pFile));

if (counter==0) printf("Создан пустой стек\n");

else printf("Создан стек с %d элементами\n",counter);

}

**void stackShow**(stack\*\* pstack) { //Вывод стека

stack\* temp = (\*pstack); //Через доп. стек

if (!temp) printf("Пустой стек");

while (temp)

{

printf(" %d\n",temp->data);

temp=temp->back;

}

printf("\n");

free(temp);

}

**stack\* task**(stack\* pfstack, stack\* psstack) { //Задача

stack \*res; InitNewStack(&res); //Стек - результат

stack \*temp; InitNewStack(&temp);

temp = pfstack; //Временная ссылка на первый стек

while (temp) //Пока не пустой

{//Передаем результату данные первого, пока не дошли до конца

s\_push(&res,temp->data);

temp=temp->back;

}

temp = psstack;//Временная ссылка на второй стек

while (temp)

{//Передаем результату данные второго, пока не дошли до конца

s\_push(&res,temp->data);

temp=temp->back;

}

temp = pfstack;

while (temp) //Временная ссылка на первый стек

{//Передаем результату снова данные первого, пока не дошли до конца

s\_push(&res,temp->data);

temp=temp->back;

}

free(temp);

return res;

}

**void End**(stack\*\* fStack, stack\*\* sStack,stack\*\* rStack) {

s\_destructor(fStack); s\_destructor(sStack); s\_destructor(rStack); //Освобождаем память

}

**Hstack.h**

#include <cstdlib>

**typedef int TInfo**; //удобство для написания абстрактных функций

**struct stack**{ //Элемент стэка

TInfo data; //содержание

stack\* back; //указатель на предыдущий

};

**void s\_push**(stack\*\* pstack, TInfo Data) //Добавление элемента в стек

{

if ((\*pstack) == nullptr) { //Если стек пустой

stack\* newElem = new stack; //Новый адресс памяти

newElem->data = Data; //Передаем значение

newElem->back = nullptr; //Предыдущего нет

(\*pstack) = newElem; //Присваеваем новой адресс памяти текущему

} else{

stack\* newElem = new stack; //Новый адресс памяти

newElem->data = Data;

newElem->back = (\*pstack); //Текущий становиться предыдущим

(\*pstack) = newElem; //Присваеваем новой адресс памяти текущему

}

}

**void destr\_last**(stack\*\* pstack) //Удаление нынешнего элемента

{

stack\* temp = (\*pstack)->back; //Уходим назад

free(pstack); //Удаляем нынешний элемент

(\*pstack) =temp; //Присвпиваем предыдущий

}

**TInfo s\_pop**(stack\*\* pstack) //"Вытаскивание" элемента

// (в текущем не используется т.к. нет необходимости в "вытаскивании элемента", а только получение значений стека)

{

if ((\*pstack)==nullptr) throw 1; //Код ошибки на случай пустого

else {

TInfo value = (\*pstack)->data; //Получаем значение

destr\_last(pstack); //Удаляем нынешний

return value; //Возврашаем значение

}

}

**void s\_destructor**(stack\*\* pstack) //Полное уничтожение стека

{

while ((\*pstack))

{

destr\_last(pstack);

}

}