#include <iostream>

#include <random>

#include <Windows.h>

using namespace std;

struct Stack { // Декларация структурного типа

int info;

Stack\* next;

} \*Begin, \* t;

Stack\* InStack(Stack\*, int); // Декларации прототипов функций пользователя

void View(Stack\*);

void Del\_All(Stack\*\*);

Stack\* InStack(Stack\* p, int in) { // Функция добавления элемента в Стек

Stack\* t = new Stack;

t->info = in;

t->next = p;

return t;

}

void View(Stack\* p) { // Функция просмотра Стека

Stack\* t = p;

while (t != NULL) {

cout << t->info << endl;

t = t->next;

}

}

void Del\_All(Stack\*\* p) { // Функция освобождения памяти

while (\*p != NULL) {

t = \*p;

\*p = (\*p)->next;

delete t;

}

}

void Zad() {

Stack\* max = Begin;

t = Begin->next;

while (t)

{

if (t->info > max->info)

max = t;

t = t->next;

}

while (Begin != max)

{

t = Begin;

Begin = Begin->next;

delete t;

}

}

int main()

{

SetConsoleCP(1251); SetConsoleOutputCP(1251); // Добавление русской локализация

int i, in, n, kod;

while (true) {

cout << "\n\tСоздать - 1.\n\tДобавить - 2.\n\tПосмотреть - 3.\n\t" << "Изменить - 4.\n\tУдалить - 5.\n\tВЫЙТИ - 0. : ";

cin >> kod; cout << endl;

switch (kod) {

case 1: case 2:

if (kod == 1 && Begin != NULL) { // Если создаем новый стек, должны освободить память, занятую предыдущим

cout << "Память очищена!" << endl;

break;

}

cout << "Введите количество чисел в СТЕКЕ = ";

cin >> n;

for (i = 1; i <= n; i++) {

cin >> in;

Begin = InStack(Begin, in);

}

if (kod == 1) cout << "Создано: " << n << endl;

else cout << "Добавлено: " << n << endl;

break;

case 3:

if (!Begin) {

cout << "Пусто!" << endl;

break;

}

cout << "<>< СТЕК ><>" << endl;

View(Begin);

break;

case 4:

{

if (!Begin) {

cout << "Пусто!" << endl;

break;

}

Zad();

cout << "СТЕК изменен!\n";

break;

}

case 5:

Del\_All(&Begin);

cout << "Память свободна!" << endl;

break;

case 0:

if (Begin != NULL)

Del\_All(&Begin);

return 0; // Выход

}

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#include <iostream>

#include <iomanip>

using namespace std;

struct Node {

int data;

Node\* prev;

Node\* next;

};

struct List {

Node\* top;

Node\* begin;

int amount;

};

List\* createList();

void pushToTheBegin(List\*, int);

void pushToTheEnd(List\*, int);

int pop(List\*);

void printListFromTheBegin(List\*);

void printListFromTheEnd(List\*);

void sort(List\*, List\*);

void deleteList(List\*);

List\* createList() {

List\* l = new List;

l->top = nullptr;

l->begin = nullptr;

l->amount = 0;

return l;

}

void pushToTheBegin(List\* l, int newNumb) {

Node\* n = new Node;

if (l->top == nullptr) {

n->prev = nullptr;

n->next = nullptr;

l->top = n;

l->begin = n;

n->data = newNumb;

l->amount++;

return;

}

if (l->top->prev == nullptr) {

n->prev = l->top->prev;

n->next = l->top;

l->top->prev = n;

l->begin = n;

n->data = newNumb;

l->amount++;

return;

}

n->prev = l->begin->prev;

n->next = l->begin;

l->begin->prev = n;

n->data = newNumb;

l->begin = n;

l->amount++;

}

void pushToTheEnd(List\* l, int newNumb) {

Node\* n = new Node;

if (l->top == nullptr) {

n->prev = nullptr;

n->next = nullptr;

l->top = n;

l->begin = n;

n->data = newNumb;

l->amount++;

return;

}

n->prev = l->top;

n->next = nullptr;

n->data = newNumb;

l->top->next = n;

l->top = n;

l->amount++;

}

int pop(List\* l) {

if (l == nullptr)

{

cout << endl << "Стек пустой";

return 1;

}

Node\* n = l->begin;

int numb;

if (n->prev == nullptr) {

numb = n->data;

delete n;

return numb;

}

l->begin->next = l->begin->prev;

numb = l->begin->data;

l->begin = l->begin->next;

delete n;

l->amount--;

return numb;

}

void printListFromTheBegin(List\* l) {

if (l == nullptr)

{

cout << endl << "Стек пустой";

return;

}

Node\* n = l->begin;

while (n != nullptr) {

cout << n->data << setw(15);

n = n->next;

}

cout << endl;

}

void printListFromTheEnd(List\* l) {

if (l == nullptr)

{

cout << endl << "Стек пустой";

return;

}

Node\* n = l->top;

while (n != nullptr) {

cout << n->data << setw(15);

n = n->prev;

}

cout << endl;

}

void sort(List\* l) {

int max = l->top->data;

Node\* temp = l->top;

Node\* toDelete = l->top;

while (temp->prev != nullptr) {

if (temp->prev->data > max) {

max = temp->prev->data;

toDelete = temp->prev;

}

temp = temp->prev;

}

if (max == l->top->data) {

Node\* z = l->top->prev;

l->top = z;

z->next = nullptr;

delete toDelete;

}

else if (max == l->begin->data) {

Node\* x = l->begin->next;

l->begin = x;

x->prev = nullptr;

delete toDelete;

}

else {

Node\* prevToDelete = toDelete->prev;

Node\* nextToDelete = toDelete->next;

nextToDelete->prev = toDelete->prev;

prevToDelete->next = toDelete->next;

toDelete->prev = nullptr;

toDelete->next = nullptr;

l->amount--;

delete toDelete;

}

int m;

bool x = true;

while (x) {

cout << "Вы хотите увидеть новую(все элементы между первым и максимальным) очередь: 1 - с начала, 2 - с конца: ";

cin >> m;

switch (m) {

case 1:

printListFromTheBegin(l);

x = false;

break;

case 2:

printListFromTheEnd(l);

x = false;

break;

default:

cout << "Неверный ввод";

continue;

}

}

}

void deleteList(List\*\* l) {

Node\* last = (\*l)->top;

Node\* second = nullptr;

while (last != nullptr) {

if (last->prev == nullptr) {

(\*l)->begin = nullptr;

}

second = last;

last = last->prev;

delete second;

(\*l)->amount--;

}

delete\* l;

\*l = nullptr;

}

int main() {

setlocale(LC\_ALL, "Russian");

List\* l = createList();

int temp = 0, amount;

cout << "Введите кол-во чисел, которое хотите ввести: ";

cin >> amount;

while (temp < amount) {

int k, numb;

cout << "Добавить число в: 1 - начало очереди, 2 - конец очереди: ";

cin >> k;

switch (k) {

case 1:

cout << "Введите число: ";

cin >> numb;

pushToTheBegin(l, numb);

break;

case 2:

cout << "Введите число: ";

cin >> numb;

pushToTheEnd(l, numb);

break;

default:

cout << "Неверный ввод";

continue;

}

temp++;

}

int c;

bool x = true;

while (x) {

cout << "Вы хотите увидеть очередь: 1 - с начала, 2 - с конца: ";

cin >> c;

switch (c) {

case 1:

printListFromTheBegin(l);

x = false;

break;

case 2:

printListFromTheEnd(l);

x = false;

break;

default:

cout << "Неверный ввод";

continue;

}

}

sort(l);

deleteList(&l);

return 0;

}