#include <iostream>

#include<ctime>

#include <clocale>

using namespace std;

struct Stack

{

int info;

Stack \*adr;

};

void Watch(Stack \*head); // выводик элементов

int menu(Stack \*head); // менюшечка

Stack\* DopRand(Stack \*head); // добавление рандомных элементов

Stack\* InStack(Stack \*p, int in); // добавление элемента

void Sort(Stack \*head); // сортировочка

Stack\* Mission(Stack \*head); // выполнение задания (удаление отрицательных элементов)

Stack\* Clear(Stack \*head); // очистка стека

int main()

{

srand(time(0));

setlocale(LC\_ALL, "rus");

Stack \*head = NULL;

while (true)

{

int number = menu(head);

switch (number)

{

case 1: //Добавление рандомных элементов

head = DopRand(head);

break;

case 2: //Добавление элемента пользователя

int x;

system("cls");

cout << "Введите элемент, который желаете добавить:\t";

cin >> x;

head = InStack(head, x);

break;

case 3: //сортировка стека

if (head == NULL || head->adr == NULL)

break;

Sort(head);

break;

case 4: //удаление всех отрицательных элементов

head = Mission(head);

break;

case 5: //очистка стека

head = Clear(head);

break;

case 6:

case 0:

return 0;

}

}

}

void Watch(Stack \*head)

{

cout << "Cтек:\t";

Stack \*sp = head;

if (sp == NULL)

cout << "Cтек пуст";

while (sp != NULL)

{

cout << sp->info;

if (sp->adr != NULL)

cout << ", ";

sp = sp->adr;

}

if (sp == NULL)

cout << ".";

}

int menu(Stack \*head)

{

setlocale(LC\_ALL, "rus");

system("cls");

Watch(head);

int number;

cout << endl << " Выберите действие:";

cout << "\n 1. Добавить случайных элементов в стек.\n 2. Добавить элемент в стек.\n 3. Сортировка стека.\n 4. Выполнение задания (удалить все отрицательные элементы).\n 5. Очистить стек.\n 0. Выход.\n";

cout << "-->";

cin >> number;

return number;

}

Stack\* DopRand(Stack \*head)

{

system("cls");

int kol, x;

cout << "Введите количество элементов, которое желаете добавить:\t";

cin >> kol;

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

{

x = -9 + rand() % 18;

head = InStack(head, x);

}

return head;

}

Stack\* InStack(Stack \*p, int in)

{

Stack \*t = new Stack;

t->info = in;

t->adr = p;

return t;

}

void Sort(Stack \*head)

{

Stack \*t = NULL;

Stack \*t1;

int x;

do

{

for (t1 = head; t1->adr != t; t1 = t1->adr)

if (t1->info > t1->adr->info)

{

x = t1->info;

t1->info = t1->adr->info;

t1->adr->info = x;

}

t = t1;

} while (head->adr != t);

}

Stack\* Mission(Stack\* head)

{

Stack \*P = head;

if (head == NULL)

cout << "Ваш стек пустой!";

while (P->adr != NULL)

{

//-------------------

if (P->info < 0)

{

Stack \*st = head;

head = head->adr;

P = P->adr;

delete st;

}

//--------------------

else

{

if (P->adr->info < 0)

{

Stack \*st = P->adr;

P->adr = P->adr->adr;

delete st;

}

else P = P->adr;

}

//--------------------

}

if (head->adr == NULL && head->info<0)

{

Stack \*st = head;

head = head->adr;

delete st;

}

return head;

}

Stack\* Clear(Stack \*head)

{

Stack \*sp = head;

Stack \*st;

while (sp != NULL)

{

st = sp;

sp = sp->adr;

delete st;

}

return sp;

}