#include <stdio.h>

#include <conio.h>

#include <iostream>

#include <math.h>

using namespace std;

struct Stack //объявление структутры стека

{

int info; //информационная часть элемента, например int

Stack \*next; //адресная часть - указатель на следующий элемент

} \*begin, \*t; //указатель вершины стека (\*begin), временный (\*t)

Stack\* InStack(Stack\*, int); //функция формирования элемента стека

void View(Stack\*); //функция просмотра стека

void Delete(Stack\*\*); //функция освобождения памяти, занятой стеком

int rand(void); //рандом

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

{

Stack \*t = new Stack; //Захватываем память для элемента

t->info = in; //формируем информационную часть

t->next = p; //формируем адресную часть

return t;

}

void View(Stack \*p)

{

Stack \*t = p; //устанавливаем текущий указатель на начало списка

while(t != NULL) //начинаем цикл, работающий до тех пор, пока указатель t не равен NULL (признак окончания списка)

{

cout << " " << t->info << endl; //вывод на экран информационной части

t = t->next; //текущий указатель переставляем на следующий элемент

}

}

void Delete(Stack \*\*p)

{

while(\*p != NULL)

{

t = \*p;

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

delete t;

}

}

void main()

{

int i, in, n, kod;

while(true)

{

cout << "\n\tCreate - 1.\n\tAdd - 2.\n\tView - 3.\n\tDelete - 4.\n\tEXIT - 0. : ";

cin >> kod;

switch(kod)

{

case 1: case 2:

if (kod == 1 && begin != NULL)

{

//Если создаем новый стек, должны освободить память, занятую предыдущим

cout << "Clear memory!" << endl;

break;

}

cout << "Input kol = ";

cin >> n;

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

{

in = rand();

begin = InStack(begin, in); //добавление нового элемента в стек, вершиной которого является указатель begin

}

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

else cout << "Add " << n << endl;

break;

case 3:

if (!begin)

{

cout << "Stack pust!" << endl;

break;

}

cout << "---Stack---" << endl;

View(begin);

break;

case 4:

Delete(&begin); //после выполнения функции указатель на вершину будет равен NULL

cout << "Memory free!" << endl;

break;

case 0:

if (begin != NULL)

Delete(&begin);

return;

}

}

}