class ArrayStack /\* стек на массиве \*/

{

private:

int size; /\* размерность массива \*/

int\* p; /\* указатель на массив \*/

int top; /\* верхушка стека \*/

public:

ArrayStack(const int& \_size); /\* конструктор \*/

ArrayStack(const ArrayStack& s); // конструктор копирования

~ArrayStack(); /\* деструктор \*/

void push(const int& n); /\* втолкнуть элемент в стек \*/

int pop(); /\* вытолкнуть элемент из стека \*/

bool isEmpty(); /\* пустой стек? \*/

bool isFull(); /\* полный стек? \*/

};

#include <iostream>

#include "stack.h"

//Выполнил Яскевич Сергей

ArrayStack::ArrayStack(const int& \_size)

{

size=\_size;

p=new int[size];

top=0;

}

ArrayStack::~ArrayStack()

{

size=0;

top=0;

delete[] p;

p=NULL;

}

bool ArrayStack::isFull()

{

if(top==(size))

return true;

return false;

}

bool ArrayStack::isEmpty()

{

if(top==0)

return true;

return false;

}

void ArrayStack::push(const int& n)

{

if(!isFull())

{

p[top]=n;

top++;

}

}

int ArrayStack::pop()

{

if(!isEmpty())

{

int n;

top--;

n=p[top];

return n;

}

return 0;

}

ArrayStack::ArrayStack(const ArrayStack& s)

{

size=s.size;

top=s.top;

p=new int[size];

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

{

p[i]=s.p[i];

}

}

/\*Лабораторная работа №3

Стек на массиве\*/

#include <iostream>

#include "stack.h"

using namespace std;

//Выполнил Яскевич Сергей

void main ()

{

int k;

printf("Checking basic features. Input num\n");

cin>>k;

ArrayStack m(k);

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

{

m.push(i);

}

ArrayStack n(m);

for(int i=0;i<=2\*k;i++)

{

cout<<m.pop()<<" "<<n.pop()<<endl;

n.push(i);

}

}

/\*Лабораторная работа №3

Стек на массиве\*/

#include <iostream>

#include "stack.h"

using namespace std;

//Выполнил Яскевич Сергей

void main ()

{

int k;

printf("Checking basic features. Input num\n");

cin>>k;

ArrayStack m(k);

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

{

m.push(i);

}

ArrayStack n(m);

for(int i=0;i<=2\*k;i++)

{

cout<<m.pop()<<" "<<n.pop()<<endl;

n.push(i);

}

}

#include <iostream>

struct Node

{

int value;

Node\* prev;

};

class Stack

{

private:

Node\* head;

public:

Stack();

~Stack();

void Push(int);

char Pop();

char Peek();

bool isEmpty();

};

Stack::Stack()

{

head=NULL;

}

void Stack::Push(int a)

{

Node\* temp=new Node;

temp->prev=head;

temp->value=a;

head=temp;

}

char Stack::Pop()

{

char a;

Node\* temp=head;

if(isEmpty())

return NULL;

else

a=temp->value;

head=head->prev;

delete temp;

return a;

}

char Stack::Peek()

{

if(!isEmpty())

return head->value;

else

return NULL;

}

bool Stack::isEmpty()

{

if(head==NULL)

return true;

else

return false;

}

Stack::~Stack()

{

Node\*temp;

while(head)

{

temp=head;

head=head->prev;

delete temp;

}

}

/\*

1.2 Описать процедуру, которая подсчитывает количество элементов стека,

//у которых равные "соседи".

\*/

#include <iostream>

#include "Stack.h"

#include<string.h>

using namespace std;

using namespace std;

int main()

{

setlocale(LC\_ALL,".1251");

Stack a;

int l,g,x1,x2,x3,n;

FILE \*IN,\*OUT;

if((IN=fopen("input.txt","rt"))==NULL)

{

printf("Проверьте наличие файла \"input.txt\"\n");

return 0;

}

char tmp[100];

char\* c=(char\*)malloc(100);

n=0;

while(!feof(IN))

{

fgets(tmp,100,IN);

c=strtok(tmp," ");

while (c)

{

n++;

g=atoi(c);

a.Push(g);

c=strtok(NULL," ");

}

}

l=0;

OUT=fopen("output.txt","w");

if (n<=2)

{

fprintf(OUT,"%d",0);

return 0;

}

x1=a.Pop();

x2=a.Pop();

while (!(a.isEmpty()))

{

x3=a.Pop();

if (x1==x3) l++;

x1=x2;

x2=x3;

}

fclose(IN);

fprintf(OUT,"%d",l);

fclose(OUT);

return 0;

}