

```
//_____ file main.cpp
```

```
#include "stack.h"
```

```
template <class Type=int>
```

```
class stack{//объявления шаблона стек элементов типа Type, по умолчанию -- стек целых чисел
```

```
    Type *body;//массив, в котором хранятся элементы стека
```

```
    int top;// индекс вершины -1, т.е. место, куда можно положить элемент
```

```
    Maxsize;// глубина стека
```

```
    public:
```

```
    stack (int);//конструктор -- строит пустой стек заданной глубины
```

```
    ~stack();//деструктор -- очистка памяти
```

```
    stack <Type >& operator=(const stack<Type> &q);//перегрузка присваивания
```

```
    stack(const stack &);//перегрузка конструктора копирования
```

```
    int push (Type);// возвращает 0, если не удалось элемент положить в стек и не 0(истина), если удалось
```

```
    int pop (Type &);//возвращает 0, если не достать элемент из стека и не 0(истина), если удалось  
}; // конец в объявлении шаблона стек, далее реализации методов
```

```
template <class Type>
```

```
stack <Type >& stack<Type>::operator=(const stack<Type> &q)
```

```
{
```

```
    if(this==&q)
```

```
        return *this;
```

```
    delete []body;
```

```
    body=new Type[q.Maxsize];
```

```
    Maxsize=q.Maxsize;
```

```
    top=q.top;
```

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

```
        body[i]=q.body[i];
```

```
    return *this;
```

```
}
```

```
template <class Type>
```

```

stack<Type>::stack(int n){
    Maxsize=n;
    top=0;
    body= new Type [n]; //место под массив
}

```

```

template <class Type> stack<Type>::stack(const stack <Type>&q){
    Maxsize=q.Maxsize;
    top=q.top;
    body= new Type [q.top]; //место под массив
    for(int i=0;i<top;i++)
        body[i]=q.body[i]; //копирование элементов
}

```

```

template <class Type>
stack<Type>::~~stack(){
    delete []body;
}

```

```

template <class Type> int stack<Type>::push(Type v){ //добавление элемента в стек
    return top==Maxsize?
    0: (*(body+top)=v, ++top);
}

```

```

template <class Type> int stack<Type>::pop(Type &v){ //удаление элементов из стека
    return top?(v=*(body+top-1), top--):0;
}

```

```

template <class Type> // печать элементов стека, стек не изменяется
void pr(stack <Type>q){
    Type u;
    while(q.pop(u))

```

```

        cout <<u<<" ";
    cout<<"\n";
}

```

template <class Type>//возвращает перевернутый стек, исходный не меняется

```

stack<Type>invert(stack <Type>q){
    Type u; stack <Type> r(q);
    while(q.pop(u))
        ;
    while(r.pop(u))
        q.push(u);
    return q;
}

```

```

int main(){
    stack <>q(8);//стек целых чисел
    int k; cin>>k;
    while(q.push(rand()%11)&&--k>0) //заполнение стека случайными числами
        ;
    pr(q);
    q=invert(q);//перевернули
    pr(q);//распечатали
    q=invert(q);
    pr(q);vec b;
    stack<vec>q1(11); //стек пар чисел
    do{
        b.x=rand()%11; b.y=rand()%11;
    }
    while(q1.push(b)); //заполнение стека парами случайных чисел
    pr(q1);
    q1=invert(q1);//перевернули
    pr(q1);//распечатали
}

```

```

    q1=invert(q1);

    pr(q1);

    str s;

    stack<str>q2(4);//стек строк ограниченной длины
do{

    int n=rand()%25,i;

    for (i=0;i<n;i++)

        s.s[i]='a'+rand()%26;

        s.s[i]=0;

    }

    while(q2.push(s));//заполнение стека случайными строчными англ. буквами

    pr(q2);

    q2=invert(q2);//перевернули

    pr(q2);//распечатали

    q2=invert(q2);

    pr(q2);

    return 0;

}

//_____file stack.h шаблон нельзя компилировать отдельно, поэтому он определен в main

#include <iostream>

#include<stdlib.h>

using namespace std;

struct str{

    char s[256]; //строки ограниченной длины

    friend ostream&operator<<(ostream&,const struct str&); // для вывода строки

};

struct vec{

    int x,y; //пары целых чисел

    friend ostream&operator<<(ostream&,const struct vec&); // вывод на экран пар

};

```

```
// _____ file stack.cpp
#include "stack.h"

ostream &operator<<(ostream &cout,const str &x)
{
    cout<<x.s;
    return cout;
}

ostream &operator<<(ostream &cout,const vec &g)
{
    cout<<"("<<g.x<<" "<<g.y<<")";
    return cout;
}
```