C++

03

Функции

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;

int sum(int,int); //прототип

int sum(int a, int b){ //описание
    cout <<"before return"<<endl;
    return a+b;
    cout <<"after return"<<endl;
}



int main(int argc, char const *argv[])
{
    cout<<sum(1,2);
    return 0;
}
```

```
before return
3[Finished in 0.3s]
```

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;

int calc(int a, int b){
    return sum(a,b);
}

int sum(int a, int b){ //описание
    return a+b;
}



int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    int a=10;
    cout<<calc(a,a*2);
    return 0;
}
```

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;

int calc(int a, int b){
    return sum(a,b);
}

int sum(int a, int b){ //описание
    return a+b;
}



int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    int a=10;
    cout<<calc(a,a*2);
    return 0;
}
```

error: use of undeclared identifier 'sum'

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;

int sum(int,int);
int calc(int,int);

int calc(int a, int b){
    return sum(a,b);
}

int sum(int a, int b){ //описание
    return a+b;
}



int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    int a=10;
    cout<<calc(a,a*2);
    return 0;
}
```

```
30
```

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;

int sum(int,int);
int calc(int,int);

int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    int a=10;
    cout<<calc(a,a*2);
    return 0;
}

int calc(int a, int b){
    return sum(a,b);
}

int sum(int a, int b){ //описание
    return a+b;
}
```

```
30
```

Перегружаемая функция 1

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;

int sum(int,int);
int calc(int,int);

int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    cout<<calc(20,10)<<endl<<calc(20.0,10.0);
    return 0;
}

int calc(int a, int b){
    return sum(a,b);
}

int sum(int a, int b){ //описание
    return a+b;
}

double calc(double a, double b){
    return a-b;
}
```

```
1  30
2  30
```

Перегружаемая функция 2

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;

int sum(int,int);
int calc(int,int);
double calc(double,double);

int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    cout<<calc(20,10)<<endl<<calc(20.0,10.0);
    return 0;
}

int calc(int a, int b){
    return sum(a,b);
}

int sum(int a, int b){ //описание
    return a+b;
}

double calc(double a, double b){
    return a-b;
}
```

```
1  30
2  10
```

Поменять 2 числа местами

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;

void swap(int a,int b){
    int t;
    t=a;
    a=b;
    b=t;
}

int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    int a=1,b=2;
    cout<<a<<" "<<b<<endl;
    swap(a,b);
    cout<<a<<" "<<b<<endl;

    return 0;
}
```

```
1   1 2
2   1 2
3
```

& взятие адреса

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;

void swap(int &a,int &b){
    int t;
    t=a;
    a=b;
    b=t;
}

int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    int a=1,b=2;
    cout<<a<<" "<<b<<endl;
    swap(a,b);
    cout<<a<<" "<<b<<endl;

    return 0;
}
```

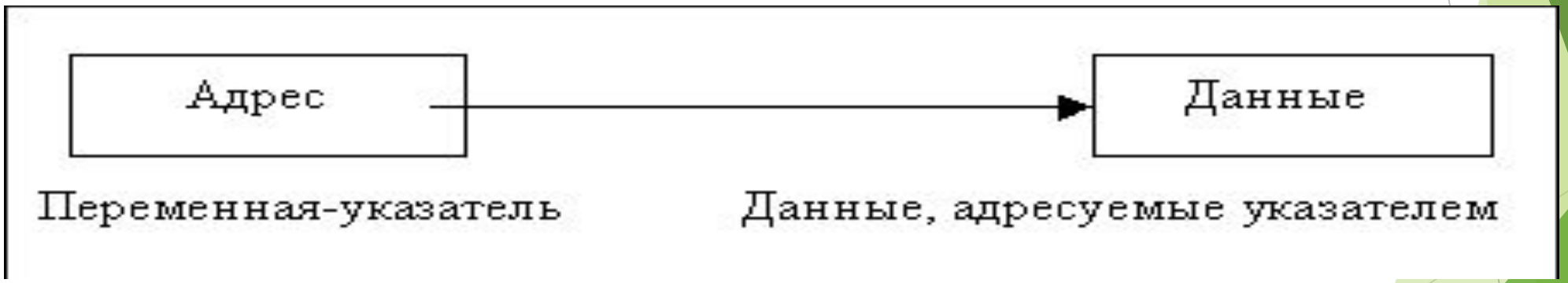
```
1 2
2 1
```

Указатели позволяют

- обрабатывать многомерные и одномерные массивы, строки, символы, структуры и массивы структур.
- динамически создавать новые переменные в процессе выполнения программы.
- обрабатывать связанные структуры: стеки, очереди, списки, деревья, сети.
- передавать функциям адреса фактических параметров.
- передавать функциям адреса функция в качестве параметров.

Определение

- **Указатель**-это переменная или константа, которая содержит значение адреса другой переменной.



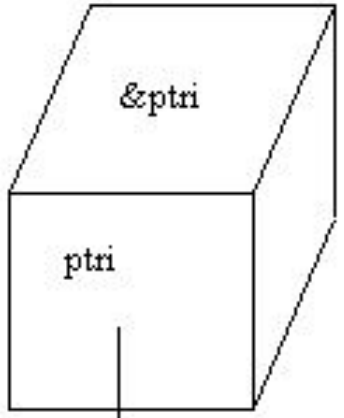
Объявление указателей и основные операции над ними

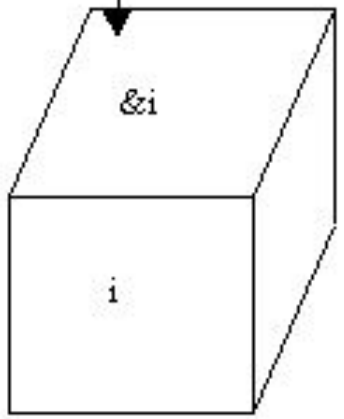
- **тип [модификатор] *<имя-указателя>**
- **тип**-имя типа переменной, адрес которой будет содержать переменная-указатель.(например integer, char, long)
- **имя-указателя** –идентификатор переменной типа указатель.(имя собственное)
- ***- определяет переменную типа указатель.

- Значение переменной-указателя-это адрес некоторой величины, целое без знака.
- Указатель содержит адрес первого байта переменной определённого типа.
- Тип адресуемой переменной, на которую ссылается указатель, определяет объём оперативной памяти, выделяемой переменной, связанной с указателем.

- указатель содержит адрес нулевого байта этой переменной
- тип адресуемой переменной определяет, сколько байтов, начиная с нулевого (адреса, определённого указателем) занимает это значение

Примеры объявлений указателей





&ptri	-адрес указателя.
ptri=&i;	-значение указателя, адрес i.
&i	-адрес переменной i.
i==*ptri;	-значение переменной.

& и *

- **&-получение адреса переменной.**
- ***-извлечение значения, расположенного по этому адресу.**

&-имя переменной

- получение адреса, **определяет адрес** размещения значения переменной определённого типа. Операндом операции **&** должно быть имя переменной того же типа, для которой определён и указатель левой части оператора присваивания, получающий значение этого адреса.

*-имя указателя

- получение значения определённого типа по заданному адресу. Определяет содержимое, находящееся по адресу, который содержится в указателе-переменной или указателе-константе. Иначе: **косвенная адресация**.

Косвенная адресация

- помощью операции * осуществляет доступ к значению по указателю, то есть извлечение значения, расположенного по адресу-содержимому указателя. Операнд * (т.е имя после) должно быть типа указатель (ранее объявлено).

Инициализация указателя

- **имя указателя_переменной=&имя_переменной**
- **int *ptri,i;**
- //объявление указателя и переменной типа int
- **ptri=&i;**
- //ptri получает значение адреса ‘i’

- оператор присваивания, использующий имя указателя и * операцию косвенной адресации:
- **Имя_переменной=*имя_указателя**
- Имя указателя –это переменная или константа, которая содержит адрес размещаемого значения, требуемого для переменной левой части оператора присваивания

- **i=*ptri;**
- **//** ‘i’ получает значение, расположенное по адресу
- // содержащемся в указателе ‘ptri’

Взаимосвязь указателя, адреса и значения переменной

```
int *pi          //указатель-переменная на данные типа integer
char *pc         //указатель-переменная на данные типа char
float *pf        //указатель-переменная на данные типа float
int m1[5]        //объявление массива m1 типа int на 5 значений
                 //m1-указатель-константа
int *m2[10]      //m2-имя массива на 10 значений типа указатель на
                 //значения типа int
```

Указатели можно использовать

- ***ptri**-значение переменной, находящейся по адресу, содержащемуся в указателе ptri
- **ptri**-значение адреса переменной
- **&ptri**-адрес местоположения самого указателя

- int i=123, j, *ptri;
- //объявление переменных и указателя
- ptri=&i;
- //инициализация указателя(присвоение адреса i)
- j=*ptri+1;
- //переменной i (*ptri) присваивается значение
- //переменной i и к её содержимому прибавляется единичка.

Многоуровневая адресация

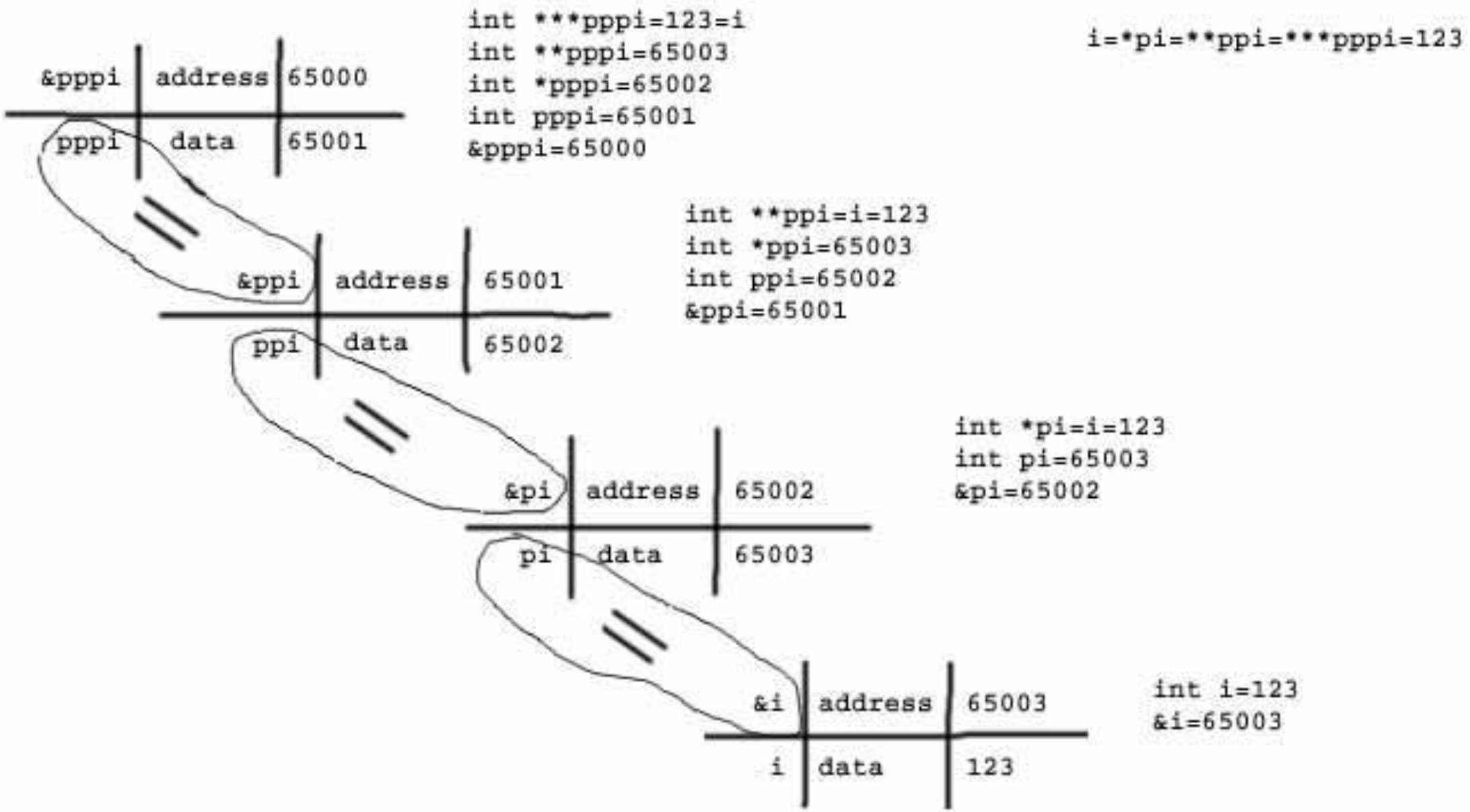
- **int i=123;**
- //где i-имя переменной
- **int *pi=&i;**
- //pi –указатель на переменную
- **int **ppi=π**
- //ppi-указатель на ‘указатель на переменную’
- **int ***pppi=&ppi;**
- //pppi-указатель на ‘указатель на ‘указатель на переменную’’.

Правила

- Полное количество звёздочек косвенной адресации, равное количеству звёздочек при объявлении указателя, определяет значение переменной.
- Уменьшение количества звёздочек косвенной адресации добавляет к имени переменной слово ‘**указатель**’, причём этих слов может быть столько, сколько может быть уровней косвенной адресации для этих имён указателей, то есть столько, сколько звёздочек стоит в объявлении указателя.

Соответствие между количеством уточнений (*) и результатом обращения к значению с помощью указателя

Обращение	Результат обращения	У.К.А.
i	Значение переменной i	0
*pi	Значение переменной, на которую указывает pi	1
pi	Указатель на переменную типа int, знпчение pi	0
**ppi	Значение переменной типа int	2
*ppi	Указатель на переменную типа int	1
ppi	Указатель на ‘указатель на переменную типа int’, значение указателя ppi	0
***pppi	Значение переменной i типа int	3
**pppi	Указатель на переменную типа int	2
*pppi	Указатель на ‘указатель на переменную типа int’	1
pppi	Указатель на ‘указатель на ‘указатель на переменную типа int’’,значение указателя pppi	0


int ***pppi=123=i
int **pppi=65003
int *pppi=65002
int pppi=65001
&pppi=65000
i=*pi=**ppi=***pppi=123
&pppi address 65000
pppi data 65001
int **ppi=i=123
int *ppi=65003
int ppi=65002
&ppi=65001
&ppi address 65001
ppi data 65002
int *pi=i=123
int pi=65003
&pi=65002
&pi address 65002
pi data 65003
int i=123
&i=65003
&i address 65003
i data 123

Операции над указателями

- Присвоить указателю значение адреса данных, или нуль.
- Увеличить (уменьшить) значение указателя
- Прибавить (вычесть) из значения указателя целое число
- Сложить или вычесть значение одного указателя из другого
- Сравнить два указателя с помощью операций отношения.

задать значение Переменной-указателю

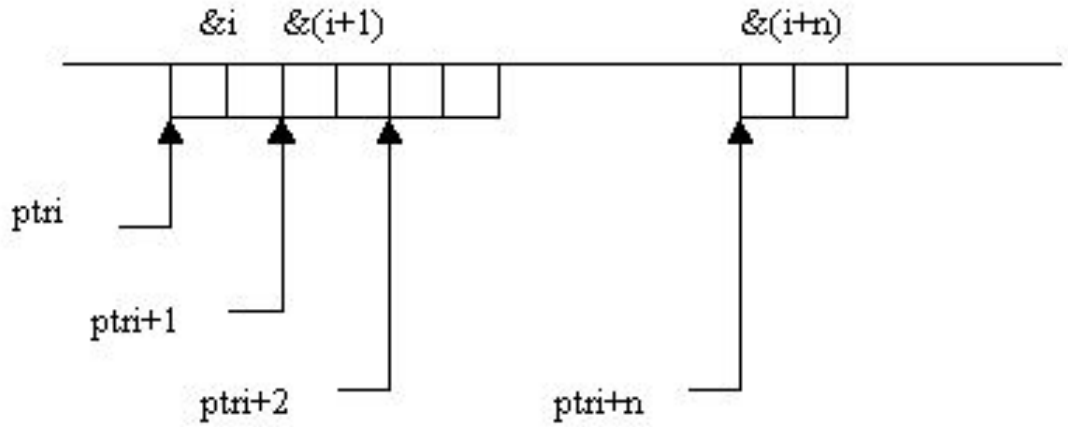
- Присвоить указателю адрес переменной, имеющей место в оперативной памяти, или нуль:
- **ptri=&i;**
- **ptri=NULL;**
- Объявить указатель вне функции (в том числе main) либо в любой функции, снабдив его описателем **stastic,** при этом начальным значением указателя является нулевой адрес (**NULL**)

- Присвоить указателю значение другого указателя, который к этому времени уже инициализирован (имеет определённое значение), например: **ptri=ptrj;** -это двойное указание одной и той же переменной.
- Присвоить переменной-указателю значение с помощью функций **malloc** и **calloc.**

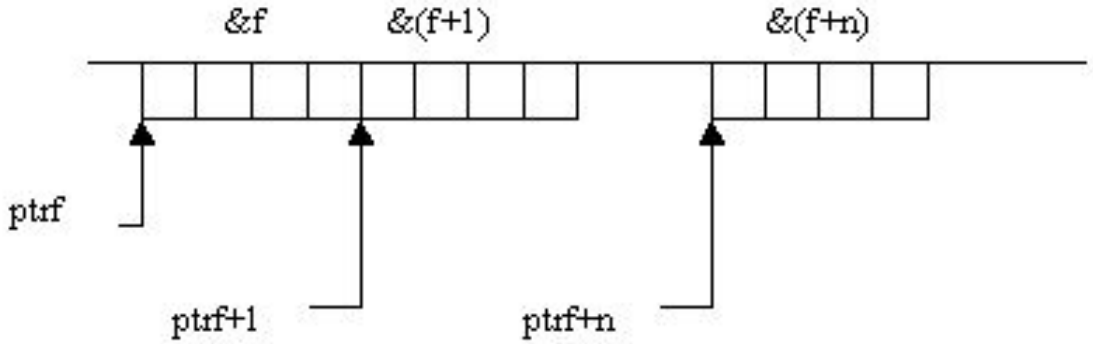
Изменение значений указателя

- +,
- ++,
- -,
- --

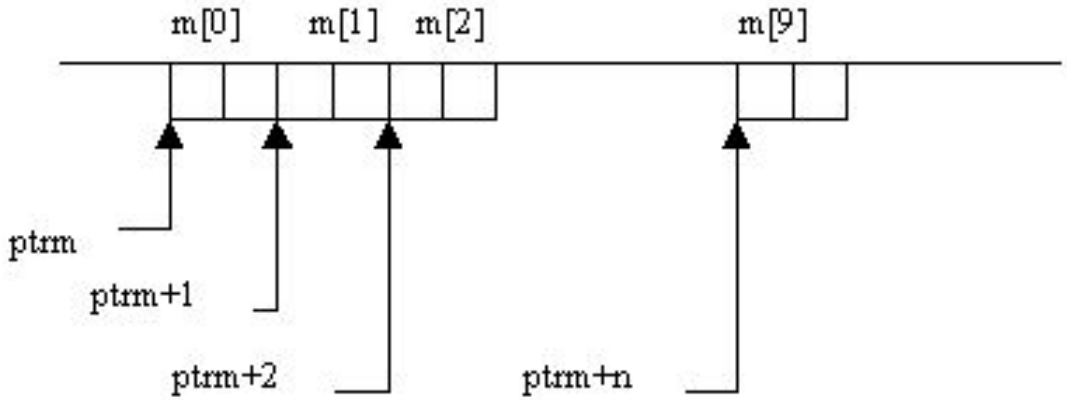
```
int i,*ptri;
ptri=&i;
ptri++;
```




```
float f,*ptrf;
ptrf=&f;
ptrf++;
```




```
int m[10],*ptrm;
ptrm=&m[0];
ptrm++;
```



Связь с массивами

```
int mas[10],*ptrm;
ptrm=&mas[0];
*prtm==mas[0]==*(mas+0) ;
//значение нулевого элемента массива mas
*(ptrm+i)==mas[i]==*(mas+i);
//значение i-го элемента массива mas
```

*mas+2==mas[0]+2;

*(mas+i)-3==mas[i]-3;

*(&(mas[i+1])+2)++;

- ptrm==&mas[i+1];
- //упрощение выражения, i не играет роли
- ptrm+2==&(mas[i+1])+2;
- //указатель переводится на 2 элемента вперёд
- ***ptrm++==**(*ptrm=*ptrm+1);
- //содержимое ячейки массива извлекается и к нему прибавляется единичка

префиксные (слева от имени указателя) **постфиксные** (справа от имени указателя)

- Префиксные операции в последовательности справа налево.
- Использование значения, полученного после выполнения префиксных операций
- Постфиксная операция над указателем.

- ***p++** сначала выполняется префиксная операция над указателем ,то есть определяется значение ***p**-содержимое, расположенное по адресу **px**, а затем выполняется постфиксная операция **++** увеличение значения указателя на квант памяти, то есть на 2 байта (если указатель типа int)

- **(++(*p)+2)** сначала:
  - *p -так как префиксные операции выполняются справа налево.
  - ***p=***p+1 -самая ‘левая’ префиксная операция
  - **+2 -**выполнение постфиксной операции

Проблемы

- Попытка работать с неинициализированными указателям, то есть с указателем, не содержащим адреса оперативной памяти (ОП), выделенной переменной.
- Потеря ссылки, то есть значения указателя из-за присваивания ему нового значения до освобождения ОП, которою он адресует.
- Не освобождение ОП, запрошенной с помощью функции **malloc**

Синтаксис указателей

- data_type *pointerName;
- data_type — тип данных,
- pointerName — имя указателя.

- int *integerPointer;

```
// Объявление указателя и простой переменной в одной строке
int *pointer1,   // это указатель
    variable;    // это обычная переменная типа int

// Объявление двух указателей в одно строке
int *pointer1, // это указатель с именем pointer1
    *pointer2; // это указатель с именем pointer2
```

два способа использования указателя

1. Использовать имя указателя без символа *, таким образом можно получить фактический адрес ячейки памяти, куда ссылается указатель.
2. Использовать имя указателя с символом *, это позволит получить значение, хранящееся в памяти. В рамках указателей, у символа * есть техническое название — операция разыменования. По сути, мы принимаем ссылку на какой-то адрес памяти, чтобы получить фактическое значение.

Объявление указателя, получение адреса переменной

- Для того чтобы объявить указатель, который будет ссылаться на переменную,
- необходимо сначала получить адрес этой переменной.
- Чтобы получить адрес памяти переменной (её расположение в памяти), нужно использовать знак & перед именем переменной.
- Это позволяет узнать адрес ячейки памяти, в которой хранится значение переменной.
- Эта операция называется — операция взятия адреса

- int var = 5;
- // простое объявление переменной с предварительной инициализацией
- int *ptrVar;
- // объявили указатель, однако он пока ни на что не указывает
- ptrVar = &var;
- // теперь наш указатель ссылается на адрес в памяти, где хранится число 5

```
#include <stdio.h>

int main()
{
    int var;      // обычная целочисленная переменная
    int *ptrVar; // целочисленный указатель (ptrVar должен быть типа int, так как он будет ссылаться на переменную типа int)

    ptrVar = &var;        // присвоили указателю адрес ячейки в памяти, где лежит значение переменной var
    scanf( "%d", &var );  // в переменную var положили значение, введенное с клавиатуры
    printf( "%d\n", *ptrVar ); // вывод значения через указатель
    getchar();
}
```

&

- & - унарный оператор, возвращающий адрес операнда в памяти
- m = &count;

  //помещает в m адрес переменной count
- q = *m;

  //помещает значение count в q

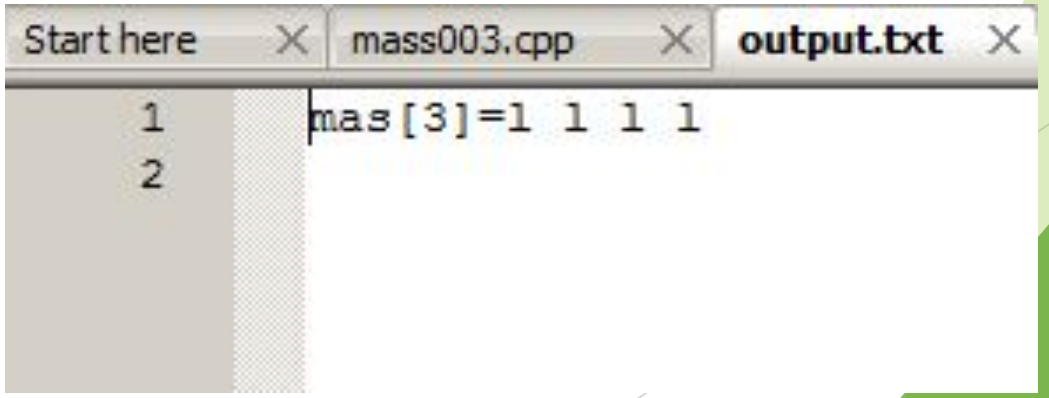
```cpp
int main(int argc, char const *argv[]){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    int target, source;
    int *memory;

    cin>>source;
    memory= &source;
    target=*memory;
    cout<<source<<endl
        <<memory<<endl
        <<target<<endl;

    return 0;
}
```

```
1   110
```

```
1   110
2   0x7fff51ce059c
3   110
4
```

Варианты обращения к элементам массива

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <stdlib.h>
#include <time.h>
//+79217811100
using namespace std;
int main(){
    ofstream cout("output.txt");
    ifstream cin("input.txt");
    char mas1[]="hello";
    char mas2[]={'h','e','l','l','o'};

    cout<<"mas[3]="<<mas1[3]<<" "<<(mas1+3)[0]<<" "<<(mas1+0)[3]<<" "<<(mas1-5)[8]<<endl;


    return 0;
}
```



Обращение через указатели

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <stdlib.h>
#include <time.h>
//+79217811100
using namespace std;
int main(){
    ofstream cout("output.txt");
    ifstream cin("input.txt");
    char mas1[]="hello";
    char mas2[]={'h','e','l','l','o'};

    //cout<<*(mas1+0)<<endl;
    for (int i=0; i<5;i++){
        cout<<*(mas1+i)<<" ";
    }

    return 0;
}
```

Использование адресации

```
using namespace std;
int main(){
    ofstream cout("output.txt");
    ifstream cin("input.txt");
    char mas1[]="hello";
    char mas2[]={'h','e','l','l','o'};

    //cout<<*(mas1+0)<<endl;
    for (int i=0; i<5;i++){
        cout<<*(mas1+i)<<" ";
    }
    cout<<endl;
    cout<<&mas1[0]<<endl;
    cout<<mas1+0<<endl;
    for (int i=0; i<5;i++){
        cout<<&mas1[i]<<" ";
    }
    cout<<endl;

    return 0;
}
```

```
1    h e l l o
2    hello
3    hello
4    hello ello llo lo o
5
```

Структуры

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;

struct Points{
    double x,y;
};

Points med(Points A,Points B){
    Points RES;
    RES.x=(A.x+B.x)/2;
    RES.y=(A.y+B.y)/2;
    return RES;
}

int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    Points A,B,M;
    cin>>A.x>>A.y>>B.x>>B.y;
    M=med(A,B);
    cout<<M.x<<" "<<M.y;

    return 0;
}
```

```
1   12 44 2 47
```

```
1   7 45.5
```

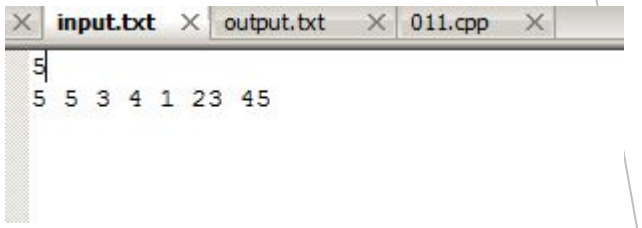
Общий вид для структур

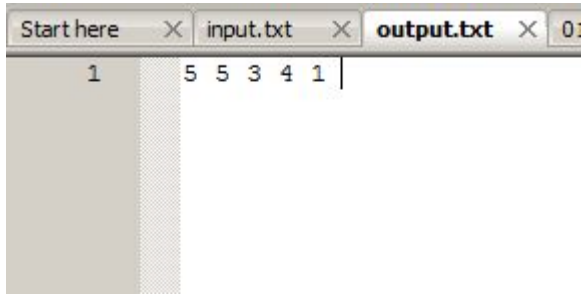
```
#include<iostream>
#include<fstream>

using namespace std;

struct TElement{
    int a;

    TElement& read(istream &cin){
        cin>>a;
        return *this;
    }
    TElement& write(ostream &cout){
        cout<<a;
        return *this;
    }
};

istream& operator >> (istream &cin, TElement &a){
    a.read(cin);
    return cin;
}

ostream& operator << (ostream &cout, TElement &a){
    a.write(cout);
    return cout;
}
```

```
int main (){
    ifstream cin("input.txt");
    ofstream cout("output.txt");

    int n;
    return 0;
}
```

Ввод и вывод массива сложный

```
#include<iostream>
#include<fstream>

using namespace std;

struct TElement{
    int a;

    TElement& read(istream &cin){
    TElement& write(ostream &cout){
};

istream& operator >> (istream &cin, TElement &a){

ostream& operator << (ostream &cout, TElement &a){

int main (){
    ifstream cin("input.txt");
    ofstream cout("output.txt");
    TElement a[100];
    int n;
    cin>>n;
    for (int i=0;i<n;i++){
        cin >> a[i];
    }
    for(int i=0;i<n;i++){
        cout << a[i];
    }
    return 0;
}
```





Массив структур

```
struct TElement{
    int a;
    TElement& read (istream &cin){
        cin>>a;
        return *this;
    }
    TElement& write(ostream &cout){
        cout<<a<<" ";
        return *this;
    }
};

istream& operator >>(istream &cin, TElement &a){
    a.read(cin);
    return cin;
}
ostream& operator << (ostream &cout, TElement &a){
    a.write(cout);
    return cout;
}
```

```
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TElement A[100];
    int N;



    return 0;
}
```

Ввод и вывод массива структур

```
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TElement A[100];
    int N;
    cin >>N;
    for (int i=0;i<N;i++){
        cin>>A[i];
    }
    for (int i=0;i<N;i++){
        cout<<A[i];
    }
    cout<<endl;
    for (int i=N;i>=0;i--){
        cout<<A[i];
    }

    return 0;
}
```

```
1   5
2   1 2 3 4 5 6 7 8 9 10
```

```
1   1 2 3 4 5
2   5 4 3 2 1
```

Решение

```
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TElement A[100];
    int N;
    cin >>N;
    for (int i=0;i<N;i++){
        cin>>A[i];
    }
    for (int i=0;i<N;i++){
        cout<<A[i];
    }
    cout<<endl;
    for (int i=N-1;i>=0;i--){
        cout<<A[i];
    }

    return 0;
}
```

Массив из более сложных структур

```
struct TElement{
    int h,m,s;
    TElement& read (istream &cin){
        cin>>h>>m>>s;
        return *this;
    }
    TElement& write(ostream &cout){
        cout<<h<<" "<<m<<" "<<s<<endl;
        return *this;
    }
};

istream& operator >>(istream &cin, TElement &a){
    a.read(cin);
    return cin;
}
ostream& operator << (ostream &cout, TElement &a){
    a.write(cout);
    return cout;
}
```

```
1   5
2   1 2 3
3   4 5 6
4   7 8 9
5   10 11 12
6   13 14 15
```

```
1   1 2 3
2   4 5 6
3   7 8 9
4   10 11 12
5   13 14 15
6
7   13 14 15
8   10 11 12
9   7 8 9
10  4 5 6
11  1 2 3
12
```

Более сложный вариант вывода

```
struct TElement{
    int h,m,s;
    TElement& read (istream &cin){
        cin>>h>>m>>s;
        return *this;
    }
    TElement& write(ostream &cout){
        cout<<h/10<<h%10<<":"<<
            m/10<<m%10<<":"<<
            s/10<<s%10<<endl;
        return *this;
    }
};
```

```
 1  01:02:03
 2  04:05:06
 3  07:08:09
 4  10:11:12
 5  13:14:15
 6
 7  13:14:15
 8  10:11:12
 9  07:08:09
10  04:05:06
11  01:02:03
```

Поиск по простой структуре

```
struct TElement{
    int a;
    TElement& read (istream &cin){
        cin>>a;
        return *this;
    }
    TElement& write(ostream &cout){
        cout<<a<<" ";
        return *this;
    }
    bool operator == (TElement b){
        return a==b.a;
    }
    bool operator != (TElement b){
        return !(*this==b);
    }
};
```

```cpp
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TElement A[100],K;
    int N,i;
    cin >>N;
    for (int i=0;i<N;i++){
        cin>>A[i];
    }
    cin>>K;
    for (int i=0;i<N && K!=A[i];i++);
    cout<<(i+1)%(N+1);

    return 0;
}
```

```
10
1 2 3 4 5 6 7 8 9 10
5

```

```cpp
    for (i=0;i<N && K!=A[i];i++);
    //cout<<(i+1)%(N+1);
    cout<<i;
```

```
4
```

```
    bool operator != (TElement b){
        //return !(*this==b);
        return a!=b.a;
    }
};
```

```
10
1 2 3 4 5 6 7 8 9 10
5

```

```
for (i=0;i<N;i++){
    if (A[i]==K){
        cout<<i<<" ";
    }
}
```

```
4
```

Поиск по более сложной структуре

```
struct TElement{
    int h,m,s;
    TElement& read (istream &cin){
        cin>>h>>m>>s;
        return *this;
    }
    TElement& write(ostream &cout){
        cout<<h/10<<h%10<<":"<<
            m/10<<m%10<<":"<<
            s/10<<s%10<<endl;
        return *this;
    }
    bool operator == (TElement b){
        return h==b.h && m==b.m && s==b.s;
    }

    bool operator != (TElement b){
        return !(*this==b);
    }
};
```

```
5
1 2 3
4 5 6
7 8 9
10 11 12
13 14 15

4 5 6
```

```
1   1
```

Поиск максимума. Перегрузка логических операторов

```
    bool operator == (TElement b){
        return a==b.a;
    }
    bool operator != (TElement b){
        return !(*this==b);
        //return a!=b.a;
    }
    bool operator > (TElement b){
        return a>b.a;
    }
    bool operator>=(TElement b){
        return *this>b||*this==b;
    }
    bool operator<(TElement b){
        return !(*this>=b);
    }
    bool operator<=(TElement b){
        return !(*this>b);
    }
```

Поиск минимума

```
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TElement A[100];
    int N,res=0;
    cin >>N;
    for (int i=0;i<N;i++){
        cin>>A[i];
    }
    for (int i=0;i<N;i++){
        if(A[i]<A[res]){
            res=i;
        }
    }
    cout <<A[res]<<" "<<res;

    return 0;
}
```

```
1   10
2   1 2 3 4 0 6 7 8 9 10
```

```
1   0  4
```

Поиск максимума

```
TElement A[100];
int N,res=0;
cin >>N;
for (int i=0;i<N;i++){
    cin>>A[i];
}
for (int i=0;i<N;i++){
    if(A[i]>A[res]){
        res=i;
    }
}
cout <<A[res]<<" "<<res;
```

```
1   10
2   1 2 3 4 0 6 7 8 9 10
```

```
1   10  9
```

Для более сложной структуры

```
    bool operator == (TElement b){
        return h==b.h && m==b.m && s==b.s;
    }
    bool operator != (TElement b){
        return !(*this==b);
        //return a!=b.a;
    }
    bool operator > (TElement b){
        return h*3600+m*60+s>b.h*3600+b.m*60+s;
    }
    bool operator>=(TElement b){
        return *this>b||*this==b;
    }
    bool operator<(TElement b){
        return !(*this>=b);
    }
    bool operator<=(TElement b){
        return !(*this>b);
    }
```

```
5
22 11 11
11 22 33
33 11 10
9 8 7
6 5 4
```

```
    for (int i=0;i<N;i++){
        if(A[i]>A[res]){
            res=i;
        }
    }
    cout <<A[res]<<" "<<res;
```

```
1   33:11:10
2   2
```

Возможная оптимизация - подбор ключа

```
struct TElement{
    int h,m,s,k;
    TElement& read (istream &cin){
        cin>>h>>m>>s;
        k=3600*h+60*m+s;
        return *this;
    }
    TElement& write(ostream &cout){
        cout<<h/10<<h%10<<":"<<
            m/10<<m%10<<":"<<
            s/10<<s%10<<endl;
        return *this;
    }
    bool operator == (TElement b){
        return k=b.k;
    }
    bool operator != (TElement b){
        return !(*this==b);
        //return a!=b.a;
    }
    bool operator > (TElement b){
        return k>b.k;
```

Предоставление массива в виде структуры

```
struct TArray{
    TElement A[100];
    int N;

    TArray& read (istream &cin){
        cin>>N;
        for (int i=0;i<N;i++){
            cin >>A[i];
        }
        return *this;
    }
    TArray& write(ostream &cout){
        for (int i=0;i<N;i++){
            cout<<A[i];
        }
        return *this;
    }
};

istream& operator >> (istream &cin, TArray &a){
    a.read(cin);
    return cin;
}
ostream& operator <<(ostream &cout,TArray &a){
    a.write(cout);
    return cout;
}
```

```cpp
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TArray A;
    cin>>A;
    cout<<A;

    return 0;
}
```

```
1   10
2   1 2 3 4 5 6 7 8 9 10
```

```
1   1 2 3 4 5 6 7 8 9 10
```

Поиск экстремумов

```
struct TArray{
    TElement A[100];
    int N,iMax,iMin;

    TArray& read (istream &cin){
        cin>>N;
        for (int i=0;i<N;i++){
            cin >>A[i];
        }
        findE();
        return *this;
    }
    void findE(){
        iMin=0;
        iMax=0;
        for (int i=0;i<N;i++){
            if (A[i]<A[iMin]) iMin=i;
            if (A[i]>A[iMax]) iMax=i;
        }
    }
```

```cpp
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TArray A;
    TElement K;
    cin>>A;
    cin>>K;
    cout<<A<<endl;;
    cout<<A.A[A.iMax]<<" "<<A.iMax<<endl;
    cout<<A.A[A.iMin]<<" "<<A.iMin<<endl;

    return 0;
}
```

```
1 2 3 4 5 6 7 8 9 10
10  9
1  0
```

Передача по значению

```
int sqr(int x);

int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");
    int a,b;
    cin>>a>>b;
    b=sqr(a);
    cout<<a<<" "<<b;
    return 0;
}

int sqr(int x){
    return x*x;
}
```

```
1   11 20
```

```
1   11 121
```

Передача по ссылке

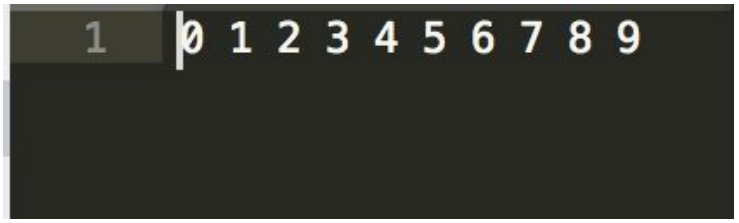
```cpp
#include <iostream>
#include <fstream>
#include <locale>

using namespace std;

void swap (int *x, int *y){
    int temp;
    temp=*x;
    *x=*y;
    *y=temp;
}

int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");
    int a,b;
    cin>>a>>b;
    swap(&a,&b);
    cout<<a<<" "<<b;
    return 0;
}
```

```
1   11 20
```

```
1   20 11
```

Вывод массива

```
void display(int a[10]);

int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    int t[10];
    for (int i=0;i<10;i++){
        t[i]=i;
    }
    display(t);
    return 0;
}

void display(int t[10]){
    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    for (int i=0;i<10;i++){
        cout<<t[i]<<" ";
    }
}
```



Массив неизвестного размера

```
void display(int t[]){
    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    for (int i=0;i<10;i++){
        cout<<t[i]<<" ";
    }
}
```

Использование указателя

```
void display(int  *t){
    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    for (int i=0;i<10;i++){
        cout<<t[i]<<" ";
    }
}
```

```
void display(int a[10]);
void sqr(int a[10]);

int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    int t[10];
    for (int i=0;i<10;i++){
        t[i]=i;
    }
    //display(t);
    sqr(t);
    display(t);
    return 0;
}

void display(int  *t){
    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    for (int i=0;i<10;i++){
        cout<<t[i]<<" ";
    }
}


void sqr(int *t){
    for (int i=0;i<10;i++){
        t[i]*=t[i];
    }
}
```

```
int fact1(int n){
    int answ;
    if(n==1||n==0){
        return 1;
    }
    answ=fact1(n-1)*n;
    return answ;
}

int fact2(int n){
    int t,answ;
    answ=1;
    for (t=1;t<=n;t++){
        answ=answ*t;
    }
    return answ;
}

int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");
    int a,b,af,bf;
    cin>>a>>b;
    af=fact1(int(a));
    bf=fact2(int(b));
    cout<<a<<" "<<af<<endl;
    cout<<b<<" "<<bf<<endl;

    return 0;
}
```

Рекурсивный и нерекурсивый факториал

И вновь простая обработка массива с сортировкой

```cpp
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    int N, arr[100];
    cin >>N;
    for (int i=0;i<N;i++){
        cin>>arr[i];
    }

    for (int i=0;i<N-1; i++){
        if (arr[i]>arr[i+1]){
            swap(arr[i],arr[i+1]);
            i=i-2*(i>0);
        }
    }

    for (int i=0;i<N;i++){
        cout<<arr[i]<<" ";
    }

    return 0;
}
```

```
10
1 2 3 4 0 6 7 8 11 10
```

```
0 1 2 3 4 6 7 8 10 11
```

switch case

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;

int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    char op;
    int a,b;
    cin>>a>>op>>b;
    cout<<a<<op<<b<<"=";
    switch(op){
        case'-':cout<<a-b<<endl;
        case'+':cout<<a+b<<endl;
        case'*':cout<<a*b<<endl;
        case'/':cout<<a/b<<endl;
    }
    return 0;
}
```

```
1   10/4
```

```
1   10-4
```

```
1   10/4=2
```

```
1   10-4=6
2   14
3   40
4   2.5
5
```

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;

int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    char op;
    int a,b;
    cin>>a>>op>>b;
    cout<<a<<op<<b<<"=";
    switch(op){
        case'-':cout<<a-b<<endl; break;
        case'+':cout<<a+b<<endl; break;
        case'*':cout<<a*b<<endl; break;
        case'/':cout<<a/1.0/b<<endl; break;
    }
    return 0;
}
```

```
10-4=6
```

```
switch(op){
    case'-':cout<<a-b<<endl; break;
    case'+':cout<<a+b<<endl; break;
    case'*':cout<<a*b<<endl; break;
    case'/':cout<<a/1.0/b<<endl; break;
    default: cout<<"error"<<endl;
}
```

```
1  10^4=error
2
```

Локальная область видимости внутри case

```
switch(op){
    default: cout<<"error"<<endl;
    case'-':{
        int res=a-b;
        cout<<res<<endl;
        break;
    }
    case'+':cout<<a+b<<endl; break;
    case'*':cout<<a*b<<endl; break;
    case'/':cout<<a/1.0/b<<endl; break;

}
return 0;
```

Структуры

```cpp
#include <iostream>
#include <fstream>
#include <locale>


using namespace std;

int main(int argc, char const *argv[]){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");

    setlocale(LC_ALL,"Russian");
    struct Car{
        char name[50];
        char color[50];
        int speed;
        int age;
    }car01;

    Car car03;

    struct Car car02={'F','F',100,1};

    cin>>car01.name>>car01.color>>car01.speed>>car01.age;

    cout<<car01.name<<car01.color<<car01.speed<<car01.age<<endl;
    cout<<car02.name<<car02.color<<car02.speed<<car02.age<<endl;

    return 0;
}
```

Структура для времени

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;
struct Time{
    int h,m,s;
};

int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    Time t;
    cin>>t.h>>t.m>>t.s;
    cout<<t.h<<":"<<t.m<<":"<<t.s;
    return 0;
}
```

```
1   8 1 12
```

```
1   8:1:12
```

Функции внутри структур

```
1   8 1 12
```

```
1   08:01:12
2
```

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <iomanip>

using namespace std;
struct Time{
    int h,m,s;
    void read (istream &cin){
        cin>>h>>m>>s;
    }
    void write(ostream &cout){
        cout<<h/10<<h%10<<":"
            <<m/10<<m%10<<":"
            <<s/10<<s%10<<endl;
    }
};

int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    Time t;
    t.read(cin);
    t.write(cout);
    return 0;
}
```

```
struct Time{
    int h,m,s;
    void read (istream &cin){
        cin>>h>>m>>s;
    }
    void write(ostream &cout){
        cout<<h/10<<h%10<<":"
            <<m/10<<m%10<<":"
            <<s/10<<s%10<<endl;
    }
    void correct(){
        m+=s/60;
        s%=60;
        h+=m/60;
        m%=60;
        h%=24;
    }
};
```

```
1  80 1111 12
```

```
1  02:31:12
2
```

```
struct Time{
    int h,m,s;
    void read (istream &cin){
        cin>>h>>m>>s;
    }
    void write(ostream &cout){
        cout<<h/10<<h%10<<":"
            <<m/10<<m%10<<":"
            <<s/10<<s%10<<endl;
    }
    void correct(){
        m+=s/60;
        s%=60;
        h+=m/60;
        m%=60;
        h%=24;
    }
    bool iscorrect(){
        return h>=0&&h<24&&m>=0&&m<60&&s>=0&&s<60;
    }
};
```

```
int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    Time t;
    t.read(cin);
    if (!t.iscorrect()) cout<<"incorrect input \n";
    t.correct();
    t.write(cout);
    return 0;
}
```

```
struct Time{
    int h,m,s,err;
    void read (istream &cin){
        cin>>h>>m>>s;
        err=!iscorrect();
        correct();
    }
    void write(ostream &cout){
        cout<<h/10<<h%10<<":"
            <<m/10<<m%10<<":"
            <<s/10<<s%10<<endl;
    }
    void writeerror(ostream &cout){
        if (err==1)cout<<"incorrect input \n";
    }
    void correct(){
        m+=s/60;
        s%=60;
        h+=m/60;
        m%=60;
        h%=24;
    }
    bool iscorrect(){
        return h>=0&&h<24&&m>=0&&m<60&&s>=0&&s<60;
    }
};
```

```
int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    Time t;
    t.read(cin);
    t.writeerror(cout);
    t.write(cout);
    return 0;
}
```

```
1  incorrect input
2  02:31:12
3
```

```
struct Time{
    int h,m,s,err;
    Time& read(istream &cin){
        cin>>h>>m>>s;
        err=!iscorrect();
        correct();
        return *this;
    }
    Time& write(ostream &cout){
        cout<<h/10<<h%10<<":"
            <<m/10<<m%10<<":"
            <<s/10<<s%10<<endl;
        return *this;
    }
    Time& writeerror(ostream &cout){
        if (err==1)cout<<"incorrect input \n";
        return *this;
    }
    void correct(){
        m+=s/60;
        s%=60;
        h+=m/60;
        m%=60;
        h%=24;
    }
    bool iscorrect(){
        return h>=0&&h<24&&m>=0&&m<60&&s>=0&&s<60;
    }
};
```

```
int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    Time t;
    t.read(cin)
    .writeerror(cout)
    .write(cout);
    return 0;
}
```

```
1  incorrect input
2  02:31:12
3
```

Сравнение двух времен в лоб

```
struct Time{
    int h,m,s,err;
    Time& read(istream &cin){ //метод возвращает исходный объект
        cin>>h>>m>>s;
        err=!iscorrect();
        correct();
        return *this;
    }
    Time& write(ostream &cout){
        cout<<h/10<<h%10<<":"
            <<m/10<<m%10<<":"
            <<s/10<<s%10<<endl;
        return *this;
    }
    Time& writeerror(ostream &cout){
        if (err==1)cout<<"incorrect input \n";
        return *this;
    }
    void correct(){
        m+=s/60;
        s%=60;
        h+=m/60;
        m%=60;
        h%=24;
    }
    bool iscorrect(){
        return h>=0&&h<24&&m>=0&&m<60&&s>=0&&s<60;
    }
    bool ismorethen(Time B){
        return 3600*h+60*m+s>3600*B.h+60*B.m+B.s;
    }
};
```

```
int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    Time t1,t2;
    t1.read(cin);
    t2.read(cin);
    if(t1.ismorethen(t2)) cout<<"t1>t2";
        else cout<<"t2>t1";
    return 0;
}
```

```
1   80 1111 12
2   17 111 11
```

```
1   t2>t1
```

Сравнение двух времен через перегрузку оператора

```
bool operator > (Time A,Time B){
    return A.ismorethen(B);
}

int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    Time t1,t2;
    t1.read(cin);
    t2.read(cin);
    if(t1>t2) cout<<"t1>t2";
        else cout<<"t2>t1";
    return 0;
}
```

Перегрузка ввода и вывода

```
istream& operator>>(istream &cin, Time &A){
    A.read(cin);
    return cin;
}
ostream& operator<<(ostream &cout, Time &A){
    A.write(cout);
    return cout;
}


int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    Time t1,t2;
    cin>>t1>>t2;
    cout<<t1<<t2;
    if(t1>t2) cout<<"t1>t2";
        else cout<<"t2>t1";
    return 0;
}
```

```
000 01 01
222 22 22
```

```
1   00:01:01
2   06:22:22
3   t2>t1
```

Массив структур

```
struct TElement{
    int a;
    TElement& read (istream &cin){
        cin>>a;
        return *this;
    }
    TElement& write(ostream &cout){
        cout<<a<<" ";
        return *this;
    }
};

istream& operator >>(istream &cin, TElement &a){
    a.read(cin);
    return cin;
}
ostream& operator << (ostream &cout, TElement &a){
    a.write(cout);
    return cout;
}
```

```
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TElement A[100];
    int N;



    return 0;
}
```

Ввод и вывод массива структур

```
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TElement A[100];
    int N;
    cin >>N;
    for (int i=0;i<N;i++){
        cin>>A[i];
    }
    for (int i=0;i<N;i++){
        cout<<A[i];
    }
    cout<<endl;
    for (int i=N;i>=0;i--){
        cout<<A[i];
    }

    return 0;
}
```

```
1   5
2   1 2 3 4 5 6 7 8 9 10
```

```
1   1 2 3 4 5
2   5 4 3 2 1
```

Решение

```
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TElement A[100];
    int N;
    cin >>N;
    for (int i=0;i<N;i++){
        cin>>A[i];
    }
    for (int i=0;i<N;i++){
        cout<<A[i];
    }
    cout<<endl;
    for (int i=N-1;i>=0;i--){
        cout<<A[i];
    }

    return 0;
}
```

Массив из более сложных структур

```
struct TElement{
    int h,m,s;
    TElement& read (istream &cin){
        cin>>h>>m>>s;
        return *this;
    }
    TElement& write(ostream &cout){
        cout<<h<<" "<<m<<" "<<s<<endl;
        return *this;
    }
};

istream& operator >>(istream &cin, TElement &a){
    a.read(cin);
    return cin;
}
ostream& operator << (ostream &cout, TElement &a){
    a.write(cout);
    return cout;
}
```

```
1   5
2   1 2 3
3   4 5 6
4   7 8 9
5   10 11 12
6   13 14 15
```

```
1   1 2 3
2   4 5 6
3   7 8 9
4   10 11 12
5   13 14 15
6
7   13 14 15
8   10 11 12
9   7 8 9
10  4 5 6
11  1 2 3
12
```

Более сложный вариант вывода

```
struct TElement{
    int h,m,s;
    TElement& read (istream &cin){
        cin>>h>>m>>s;
        return *this;
    }
    TElement& write(ostream &cout){
        cout<<h/10<<h%10<<":"<<
            m/10<<m%10<<":"<<
            s/10<<s%10<<endl;
        return *this;
    }
};
```

```
 1  01:02:03
 2  04:05:06
 3  07:08:09
 4  10:11:12
 5  13:14:15
 6
 7  13:14:15
 8  10:11:12
 9  07:08:09
10  04:05:06
11  01:02:03
```

Поиск по простой структуре

```
struct TElement{
    int a;
    TElement& read (istream &cin){
        cin>>a;
        return *this;
    }
    TElement& write(ostream &cout){
        cout<<a<<" ";
        return *this;
    }
    bool operator == (TElement b){
        return a==b.a;
    }
    bool operator != (TElement b){
        return !(*this==b);
    }
};
```

```cpp
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TElement A[100],K;
    int N,i;
    cin >>N;
    for (int i=0;i<N;i++){
        cin>>A[i];
    }
    cin>>K;
    for (int i=0;i<N && K!=A[i];i++);
    cout<<(i+1)%(N+1);

    return 0;
}
```

```
10
1 2 3 4 5 6 7 8 9 10
5

```

```cpp
    for (i=0;i<N && K!=A[i];i++);
    //cout<<(i+1)%(N+1);
    cout<<i;
```

```
4
```

```
    bool operator != (TElement b){
        //return !(*this==b);
        return a!=b.a;
    }
};
```

```
10
1 2 3 4 5 6 7 8 9 10
5

```

```
for (i=0;i<N;i++){
    if (A[i]==K){
        cout<<i<<" ";
    }
}
```

```
4
```

Поиск по более сложной структуре

```
struct TElement{
    int h,m,s;
    TElement& read (istream &cin){
        cin>>h>>m>>s;
        return *this;
    }
    TElement& write(ostream &cout){
        cout<<h/10<<h%10<<":"<<
            m/10<<m%10<<":"<<
            s/10<<s%10<<endl;
        return *this;
    }
    bool operator == (TElement b){
        return h==b.h && m==b.m && s==b.s;
    }

    bool operator != (TElement b){
        return !(*this==b);
    }
};
```

```
5
1 2 3
4 5 6
7 8 9
10 11 12
13 14 15

4 5 6
```

```
1   1
```

Поиск максимума. Перегрузка логических операторов

```
    bool operator == (TElement b){
        return a==b.a;
    }
    bool operator != (TElement b){
        return !(*this==b);
        //return a!=b.a;
    }
    bool operator > (TElement b){
        return a>b.a;
    }
    bool operator>=(TElement b){
        return *this>b||*this==b;
    }
    bool operator<(TElement b){
        return !(*this>=b);
    }
    bool operator<=(TElement b){
        return !(*this>b);
    }
```

Поиск минимума

```
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TElement A[100];
    int N,res=0;
    cin >>N;
    for (int i=0;i<N;i++){
        cin>>A[i];
    }
    for (int i=0;i<N;i++){
        if(A[i]<A[res]){
            res=i;
        }
    }
    cout <<A[res]<<" "<<res;

    return 0;
}
```

```
1   10
2   1 2 3 4 0 6 7 8 9 10
```

```
1   0  4
```

Поиск максимума

```
TElement A[100];
int N,res=0;
cin >>N;
for (int i=0;i<N;i++){
    cin>>A[i];
}
for (int i=0;i<N;i++){
    if(A[i]>A[res]){
        res=i;
    }
}
cout <<A[res]<<" "<<res;
```

```
1   10
2   1 2 3 4 0 6 7 8 9 10
```

```
1   10  9
```

Для более сложной структуры

```
    bool operator == (TElement b){
        return h==b.h && m==b.m && s==b.s;
    }
    bool operator != (TElement b){
        return !(*this==b);
        //return a!=b.a;
    }
    bool operator > (TElement b){
        return h*3600+m*60+s>b.h*3600+b.m*60+s;
    }
    bool operator>=(TElement b){
        return *this>b||*this==b;
    }
    bool operator<(TElement b){
        return !(*this>=b);
    }
    bool operator<=(TElement b){
        return !(*this>b);
    }
```

```
5
22 11 11
11 22 33
33 11 10
9 8 7
6 5 4
```

```
    for (int i=0;i<N;i++){
        if(A[i]>A[res]){
            res=i;
        }
    }
    cout <<A[res]<<" "<<res;
```

```
1   33:11:10
2   2
```

Возможная оптимизация - подбор ключа

```
struct TElement{
    int h,m,s,k;
    TElement& read (istream &cin){
        cin>>h>>m>>s;
        k=3600*h+60*m+s;
        return *this;
    }
    TElement& write(ostream &cout){
        cout<<h/10<<h%10<<":"<<
            m/10<<m%10<<":"<<
            s/10<<s%10<<endl;
        return *this;
    }
    bool operator == (TElement b){
        return k=b.k;
    }
    bool operator != (TElement b){
        return !(*this==b);
        //return a!=b.a;
    }
    bool operator > (TElement b){
        return k>b.k;
```

Предоставление массива в виде структуры

```
struct TArray{
    TElement A[100];
    int N;

    TArray& read (istream &cin){
        cin>>N;
        for (int i=0;i<N;i++){
            cin >>A[i];
        }
        return *this;
    }
    TArray& write(ostream &cout){
        for (int i=0;i<N;i++){
            cout<<A[i];
        }
        return *this;
    }
};

istream& operator >> (istream &cin, TArray &a){
    a.read(cin);
    return cin;
}
ostream& operator <<(ostream &cout,TArray &a){
    a.write(cout);
    return cout;
}
```

```cpp
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TArray A;
    cin>>A;
    cout<<A;

    return 0;
}
```

```
1   10
2   1 2 3 4 5 6 7 8 9 10
```

```
1   1 2 3 4 5 6 7 8 9 10
```

Поиск экстремумов

```
struct TArray{
    TElement A[100];
    int N,iMax,iMin;

    TArray& read (istream &cin){
        cin>>N;
        for (int i=0;i<N;i++){
            cin >>A[i];
        }
        findE();
        return *this;
    }
    void findE(){
        iMin=0;
        iMax=0;
        for (int i=0;i<N;i++){
            if (A[i]<A[iMin]) iMin=i;
            if (A[i]>A[iMax]) iMax=i;
        }
    }
```

```cpp
int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    TArray A;
    TElement K;
    cin>>A;
    cin>>K;
    cout<<A<<endl;;
    cout<<A.A[A.iMax]<<" "<<A.iMax<<endl;
    cout<<A.A[A.iMin]<<" "<<A.iMin<<endl;

    return 0;
}
```

```
1 2 3 4 5 6 7 8 9 10
10  9
1  0
```

Передача по значению

```
int sqr(int x);

int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");
    int a,b;
    cin>>a>>b;
    b=sqr(a);
    cout<<a<<" "<<b;
    return 0;
}

int sqr(int x){
    return x*x;
}
```

```
1   11 20
```

```
1   11 121
```

Передача по ссылке

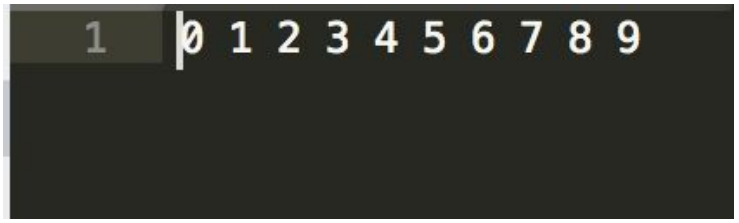
```cpp
#include <iostream>
#include <fstream>
#include <locale>

using namespace std;

void swap (int *x, int *y){
    int temp;
    temp=*x;
    *x=*y;
    *y=temp;
}

int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");
    int a,b;
    cin>>a>>b;
    swap(&a,&b);
    cout<<a<<" "<<b;
    return 0;
}
```

```
1   11 20
```

```
1   20 11
```

Вывод массива

```
void display(int a[10]);

int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    int t[10];
    for (int i=0;i<10;i++){
        t[i]=i;
    }
    display(t);
    return 0;
}

void display(int t[10]){
    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    for (int i=0;i<10;i++){
        cout<<t[i]<<" ";
    }
}
```



Массив неизвестного размера

```
void display(int t[]){
    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    for (int i=0;i<10;i++){
        cout<<t[i]<<" ";
    }
}
```

Использование указателя

```
void display(int  *t){
    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    for (int i=0;i<10;i++){
        cout<<t[i]<<" ";
    }
}
```

```
void display(int a[10]);
void sqr(int a[10]);

int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");

    int t[10];
    for (int i=0;i<10;i++){
        t[i]=i;
    }
    //display(t);
    sqr(t);
    display(t);
    return 0;
}

void display(int  *t){
    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    for (int i=0;i<10;i++){
        cout<<t[i]<<" ";
    }
}


void sqr(int *t){
    for (int i=0;i<10;i++){
        t[i]*=t[i];
    }
}
```

```
int fact1(int n){
    int answ;
    if(n==1||n==0){
        return 1;
    }
    answ=fact1(n-1)*n;
    return answ;
}

int fact2(int n){
    int t,answ;
    answ=1;
    for (t=1;t<=n;t++){
        answ=answ*t;
    }
    return answ;
}

int main(){

    ifstream cin("001in.txt");
    ofstream cout("001out.txt");
    setlocale(LC_ALL,"Russian");
    int a,b,af,bf;
    cin>>a>>b;
    af=fact1(int(a));
    bf=fact2(int(b));
    cout<<a<<" "<<af<<endl;
    cout<<b<<" "<<bf<<endl;

    return 0;
}
```

Рекурсивный и нерекурсивый факториал

Указатель на функцию

- void f() { }
- void (*pf)() = &f;
- pf();

f - функция.
переменная pf, является *указателем на функцию, которая ничего не возвращает и не принимает ни одного аргумента*.
В определении pf ей присваивается адрес функции f.
В третьей строке вызывается функция по указатели pf(в данном случае будет вызвана функция f)

```
void f(int a) { }
void g(int b) { }

void (*pf)(int) = &f;
pf(10); // Вызывается f(10)

pf = &g;
pf(20); // Вызывается g(20)
```

```
void f() { }
void g() { }

void (*pf) = &f; // Верно, &f - указатель на функцию f
pf = g; // Тоже верно, имя функции(g) автоматически приводится к указателю на функцию.
```

Чтение произвольного числа символов

```
int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    int a;
    while (cin>>a)cout<<a<<" ";
    return 0;
}
```

```
int main(int argc, char const *argv[])
{
    ifstream cin("003in.txt");
    ofstream cout("005out.txt");
    int a;
    for (cin>>a;a;cin>>a)cout<<a<<" ";
    return 0;
}
```