



Язык C++

Лекция №6

Введение в C++

Первая программа C++

```
#include <iostream.h>
int n = 5; // глобальная переменная
void main()
{
int n; // локальная переменная
cout << "Enter n" ; // вывод в поток
cin >> n; // ввод из потока

// вывод переменной в поток
cout << "n = " << n;

// вывод глобальной переменной
cout << "global n = " << ::n;
}
```

ПОТОКИ ВВОДА/ВЫВОДА

```
#include <iostream>
using namespace std;
int main()
{   int a;
    cout << "Hello, world!" << endl;
    cout.width(10);
    cout << "ten" << "four";
    cin >> a;
    return 0;
}
```

Функции `iostream`

`width(int x)`

минимальное число знаков до следующего вывода

`fill(char x)`

устанавливает символ-заполнитель и возвращает предыдущий символ-заполнитель. По умолчанию в качестве символа-заполнителя используется пробел

`precision(int x)`

устанавливает число значащих знаков для чисел с плавающей точкой

Манипуляторы iostream

endl	перевод строки
flush	выгружает содержимое буфера в поток
dec	меняет формат вывода числа на десятичный
oct	меняет формат вывода числа на восьмеричный
hex	меняет формат вывода числа на шестнадцатеричный
ws	извлекает и удаляет символы пустых промежутков из потока
showpos	показывает '+' перед неотрицательными числами
showpoint	показывает десятичную точку
noshowpoint	скрывает десятичную точку

Пространства имен

```
namespace foo  
{  
    int bar;  
}
```

```
using namespace foo;
```

КОНСТАНТЫ

```
const int n = 10;
```

```
const double pi =  
    3.1415926535897932384626433832795;
```

```
int sqr(const int n)  
{ return n*n;  
}
```

```
int strlen (const char *s)  
{ char *p = (char *) s;  
  while (*p++);  
  return p-s;  
}
```

Перегрузка функций

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

```
float sqr(float n)
{
    return n*n;
}
```

```
double sqr(double n)
{
    return n*n;
}
```

...

```
n=sqr(2) ;  
x=sqr(2.0)
```

...

Передача параметров функциям

// По значению

```
void noswap(int a, int b)
{
    int c = a;
    a = b; b = c;
}
```

// По адресу

```
void swap(int* a, int* b)
{
    int c = *a;
    *a = *b; *b = c;
}
```

// По ссылке

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

```
swap(x, y);
```

```
cout << x << ' ' << y << endl;
```

```
swap(&x, &y);
```

```
cout << x << ' ' << y << endl;
```

```
noswap(x, y);
```

```
cout << x << ' ' << y << endl;
```

Передача параметров функциям

```
double power(double x, int n = 2)
{ double z = 1;
  for (int i=1; i<=abs(n) ; i++) z*=x;
  if (n<0) z=1.0/z;
  return z;
}
```

```
a=power(2.0,10);
```

```
b=power(2.0);
```



Перерыв 10 мин

Описание класса

```
class vector
{private:
    float *p; // указатель на начало вектора
    int n; // количество элементов в векторе
public:
    vector(int i = 3); // конструктор
                        // (назначен параметр по умолчанию)
    ~vector(); // деструктор (не может иметь параметров)
    float item(int i); // возвращает указанный элемент
    void assign(int i, float x); // назначение элемента
    float num()
        { return n; }; // возвращает число элементов (inline)
    float norm(); // возвращает квадрат нормы вектора
};
```

Реализация методов класса

```
vector::vector(int i)
```

```
{ int j;  
  n=i;  
  p= new float[n];  
  for (j=0; j<n; j++) p[j]=0;  
  cout << "vector created " << n << "\n";  
}
```

```
vector::~~vector()
```

```
{ delete p;  
  cout << "vector destroyed\n";  
}
```

```
float vector::item(int i)
```

```
{ if ((i>=0) && (i<n)) return p[i];  
  else {cout << "Error in vector::item"; return 0;}  
}
```

Реализация методов класса

```
void vector::assign(int i, float x)
{
    if ((i>=0) && (i<n)) p[i]=x;
    else cout << "Error in vector::assign";
}
```

```
float vector::norm()
{ int i;
  float x=0;
  for (i=0; i<n; i++) x+=p[i]*p[i];
  return x;
}
```

Использование объекта

```
main ()
{ int i;
  vector a(100);
  vector b;
  for (i=0; i<a.num(); i++) a.assign(i,i);
  cout << a.norm() << "\n";
  a.~vector();
}
```

```
vector created 100
vector created 3
328350
vector destroyed 100
vector destroyed 3
```

Перегрузка операций

```
{  
    ...  
    float operator() (int i);    // возвращает указанный элемент  
    void operator=(vector &x); // присваивает значение одного  
        ...                // вектора другому  
}
```

```
float vector::operator() (int i)  
{ if ((i>=0) && (i<n)) return p[i];  
    else {cout << "Error in vector::item"; return 0;}  
}
```

```
void vector::operator=(vector &x)  
{ if (x.num()==n)  
    for (int i=0; i<n; i++) p[i]=x(i);  
    else cout << "Error in operator =\n";  
}
```

```
// в main():  
c=a;  
cout << c(10);
```


Улучшенный класс Vector

```
class vector
{ ...
  public:
    vector (int i = 3);           // конструктор
    vector (float x, float y, float z); // второй конструктор
    vector& operator=(vector &x);
        // присваивает значение одного в. другому
    vector& operator*(float c); // умножение вектора на скаляр
    float operator*(vector &x); // умножение вектора на вектор
    float& operator[(int i); // возвращает ссылку на элемент
}
```

Реализация класса Vector

```
vector::vector(float x, float y, float z)
{
    n=3;
    p=new float[n];
    vector::assign(0,x);
    vector::assign(1,y);
    vector::assign(2,z);
    cout << "3d vector created " << n << "\n";
}
```

```
vector& vector::operator=(vector &x)
{
    if (x.num()==n)
        for (int i=0; i<n; i++) p[i]=x(i);
    else cout << "Error in operator =\n";
    return *this;
}
```

Реализация класса Vector

```
// умножение вектора на скаляр
vector & vector::operator*(float c)
{ for (int i=0; i<n; i++) p[i]*=c;
  return *this;
}

// умножение вектора на вектор
float vector::operator*(vector &x)
{ float s=0.0;
  for (int i=0; i<n; i++) s+=operator()(i)*x(i);
  return s;
}

// Операция [ ]
float & vector::operator[](int i)
{ if ((i>=0) && (i<n)) return p[i];
  else {cout << "Error in vector::item"; return p[0];}
}
```

Использование класса Vector

```
main()
{ vector b;
  vector d(1,1,1);
  b=d*2.0;
  b[0]=3;
  for (i=0; i<b.num(); i++) cout << b(i) << " ";
}
```

3 2 2

Наследование

```
class matrix: public vector
{protected:
    int M,N;
    int lineaddres(int i, int j) { return i*N+j; };
public:
    matrix(int m=2, int n=2): vector(m*n)
        { M=m; N=n; }
    float item(int i, int j);
    void assign(int i, int j, float x);
    int m() { return M;};
    int n() { return M;};
};
```

```
float matrix::item (int i, int j)
{ return vector::operator()(lineaddres(i,j)); }
```

```
void matrix::assign(int i, int j, float x)
{ vector::assign(lineaddres(i,j),x); }
```

Наследование

```
main ()
{ int i, j;

  for (i=0; i<z.m(); i++)
  {
    for (j=0; j<z.n(); j++)
    {
      z.assign(i, j, (i+1)*10+j+1);
      cout << z.item(i, j) << " ";
    }
    cout << "\n";
  }
}
```

```
11 12 13
21 22 23
31 32 33
```

Атрибуты наследования

- **private** – доступны только в данном классе
- **protected** – доступны только в данном классе и потомках
- **public** – доступны для всеобщего использования