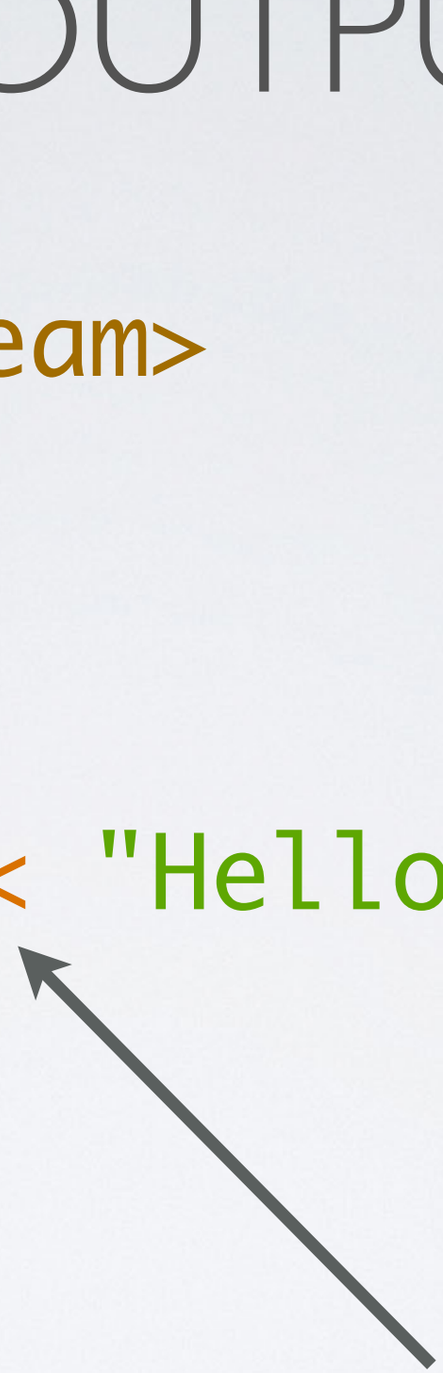
ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Лекция № | / **05**
11.03.2019 г.

STANDARD OUTPUT STREAM

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";

    return 0;
}
```



Insertion operator:
operator<<

STANDARD INPUT/OUTPUT STREAM

```
#include <iostream>

int main()
{
    int x, y;

    std::cin >> x >> y;
    std::cout << "x=" << x << ", y=" << y << std::endl;

    return 0;
}
```

Extraction operator:
operator>>

Manipulator:
'\n' + std::flush

STANDARD INPUT/OUTPUT STREAM

```cpp
#include <iostream>

int main()
{
   int x, y;
   std::cout << "Please Enter two numbers:\n> ";
   std::cin >> x >> y;
   std::cout << "x=" << x << ", y=" << y << '\n';

   return 0;
}
```

Console:

Please Enter two numbers:

> 1 2

x=1, y=2

STANDARD INPUT/OUTPUT STREAM

```cpp
#include <iostream>

int main()
{
   int x, y;
   std::cout << "Please Enter two numbers:\n> ";
   std::cin >> x >> y;
   std::cout << "x=" << x << ", y=" << y << '\n';

   return 0;
}
```

Console:

Please Enter two numbers:

> a b ???

STANDARD INPUT/OUTPUT STREAM

```
#include <iostream>

int main()
{
    int x, y;
    std::cout << "Please Enter two numbers:\n> ";
```

Console:

```
Please Enter two numbers:
> a b
Incorrect input. Try again.
>
```

```
    if (std::cin >> x >> y)
        std::cout << "x=" << x << ", y=" << y << '\n';
    else
    {
        std::cout << "Incorrect input. Try again.\n> ";
        std::cin.clear();
        std::cin.ignore(
                std::numeric_limits<std::streamsize>::max(),
                '\n');
        ...
    }
    return 0;
}
```

```
class ostream {
    // ...
    ostream &operator<<(int val) {
        // ...
        return *this;
    }
    ostream &operator<<(const char *v);

    // pf - manipulator
    ostream &operator<<(ostream & (*pf)(ostream &)) {
        return pf(*this);
    }
}

ostream &endl(ostream& os) {
    return os.put('\n').flush();
}

extern ostream cout;
```

MANIPULATORS <IOMANIP>

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

void printAligned(int val, size_t width, char fillChar = ' ')
{
   std::cout << "================";
   std::cout << std::setfill(fillChar);
   std::cout << std::left << std::setw(width) << val << '\n';
   std::cout << std::right << std::setw(width) << val << '\n';
}
```

MANIPULATORS <IOMANIP>

```
...

int main()
{
   printAligned(123456, 15);
   printAligned(123456, 15, '_');

   return 0;
}
```

Console:

```
===============
123456
              123456
===============
123456_________
_________123456
```

MANIPULATORS <IOMANIP>

```
...

int main()
{
   std::cout << std::hex << std::showbase << std::uppercase;
   printAligned(0x123abc, 15);

   return 0;
}
```

Console:

```
===============
0X123ABC
        0X123ABC
```

MANIPULATORS <IOMANIP>

```
...

int main()
{
   std::cout << std::boolalpha
         << "true/false values: "
         << true << ", " << false << '\n';

   return 0;
}
```

Console:

true/false values: true, false

MANIPULATORS <IOMANIP>

```
...

int main()
{
   std::cout << "doubles: " << 12.3 << ", " << 12.0 << ", "
         << std::showpoint << 12.0 << '\n';

   return 0;
}
```

Console:

doubles: 12.3, 12, 12.0000

MANIPULATORS <IOMANIP>

```
...

int main()
{
   std::cout << "scientific double: " << std::scientific <<
        << 12300.123 << '\n';
   std::cout << "fixed double: " << std::fixed <<
        << 12300.123 << '\n';

   return 0;
}
```

Console:

scientific double: 1.230012e+04
fixed double: 12300.123000

MANIPULATORS <IOMANIP>

```
...

int main()
{
   std::cout << "Very precise double: "
        << std::setprecision(10) << 0.0000000001 << '\n';
   std::cout << "Less precise double: "
        << std::setprecision(1) << 0.0000000001 << '\n';

   return 0;
}
```

Console:

Very precise double: 0.0000000001

Less precise double: 0.0

MANIPULATORS <IOMANIP>

```
...

int main()
{
   std::cout << "now please enter some "
             "comma-separated names:\n> ";

   for (std::string s;
         std::getline(std::cin >> std::ws, s, ',');)
   {
      if (s.empty()) { break; }
      std::cout << "name: \"" << s << "\"\n";
   }

   return 0;
}
```

Console:

```
now please enter some comma-separated names:
> Vasya, Petya, Vanya
name: "Vasya"
name: "Petya"
name: "Vanya"
```

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

using namespace std;

int main()
{
    ifstream inp("test.txt");

    if (!inp) {
        cerr << "File not found!" << endl;
        return 1;
    }

    string s;

    for (int n = 1; getline(inp, s); ++n)
        cout << "Line " << n << ": " << s << endl;

    return 0;
}
```

Построчное чтение файла

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

using namespace std;

string getLine(int n) {
    ostringstream oss;

    oss << n << "^2=" << n * n;
    return oss.str();
}

int main() {
    ofstream of;

    of.open("out.txt", ofstream::out);

    if (!of) {
        cerr << "Cannot open file for writing!" << endl;
        return 1;
    }

    for (int i = 0; i < 10; ++i) {
        string line = getLine(i);

        of << line << endl;
    }

    of.close();
}
```

Запись в файл и stringstream

```cpp
#include <iostream>
#include <string>

using namespace std;

const std::string A1("a1");

int main() {
    string s("abcd");

    cout << "String size = " << s.size() << endl;

    s.clear();
    if (s.empty())
        cout << "s is empty now!" << endl;

    s += "1234";
    cout << s[3] << endl;              // 4

    s.insert(1, "abc");                // 1abc234
    s.erase(2, 2);                     // 1a234
    s.replace(0, 2, A1);               // a1234

    cout << s.substr(2)    << endl;    // 234
    cout << s.substr(2, 1) << endl;    // 2

    return 0;
}
```

Работа со строками

```cpp
#include <iostream>
#include <string>
#include <stdio.h>

using namespace std;

int main() {
    string s("abcdab");

    puts(s.c_str());          // const char* c_str() const;

    char sum = 0;
    const char *p0 = s.data();
    for (const char *p = p0; p < p0 + s.size(); ++p)
        sum += *p;

    cout << "Sum = " << int(sum) << endl;

    size_t pos = s.find("ab");
    cout << pos << endl;          // 0

    size_t rpos = s.rfind("ab");
    cout << rpos << endl;         // 4

    size_t what = s.find("wat");
    if (what == string::npos)
        cout << "not found" << endl;

    return 0;
}
```

Работа по указателю и поиск

```cpp
#include <iostream>
#include <string>

using namespace std;

int main() {
    string s;
    getline(cin, s);        // <--- "  12  911 358  "

    for (size_t pos = 0;;) {
        // Ищем следующий непробельный символ
        pos = s.find_first_not_of(" \t", pos);
        if (pos == string::npos)
            break;  // Ничего не найдено

        // Ищем следующий пробельный символ
        size_t pos1 = s.find_first_of(" \t", pos);
        size_t len = (pos1 == string::npos) ? string::npos : pos1 - pos;

        string word(s.substr(pos, len));
        cout << "word = '" << word << "'\n";

        if (pos1 == string::npos)
            break;

        pos = pos1;
    }

    return 0;
}
```

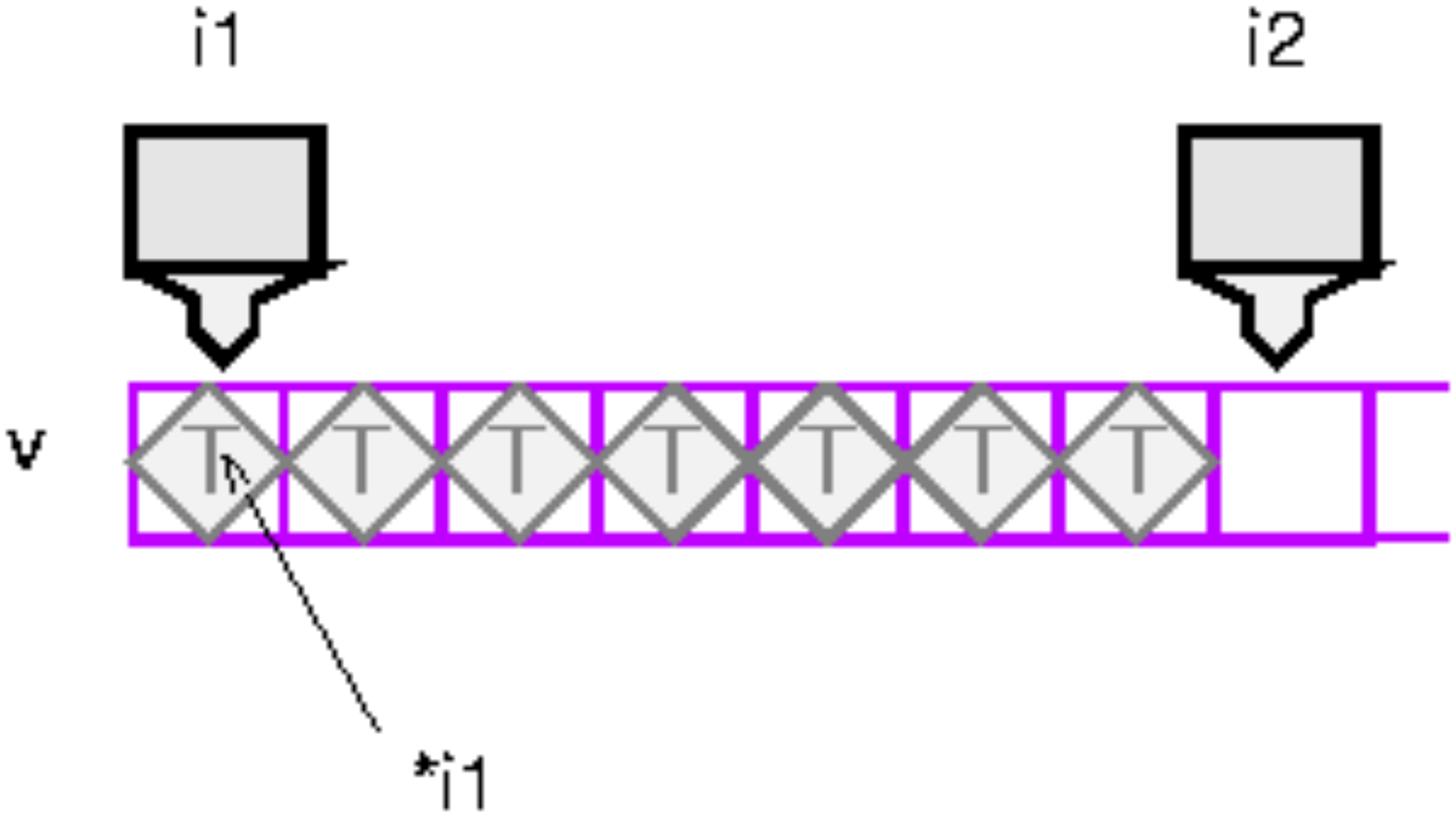
Разбиение строки на слова

```cpp
#include <iostream>
#include <string>

using namespace std;

// int stoi(const string &str, size_t *idx = 0, int base = 10);
// double stod(const string &str, size_t *idx = 0);

int main() {
    string s("41abc");

    cout << stoi(s) << endl;                    // 41
    cout << stoi(s, nullptr, 16) << endl;       // 268988

    size_t next = 0;
    cout << stoi(s, &next) << " and [" << s.substr(next) << "]\n";
    // 41 and [abc]

    // base==0: autodetect
    cout << stoi("0x7f", nullptr, 0) << endl;   // 127

    // Числа с плавающей точкой
    cout << stod("1.1e15")  << endl;             // 1.1e+15

    s = "1100efun";
    cout << stod(s, &next) << endl;     // 1100
    cout << s.substr(next) << endl;     // efun

    return 0;
}
```

Конвертация строк в числа

```cpp
#include <vector>
#include <iostream>
using namespace std;

int main() {
    vector<int> v;

    v.push_back(10);
    v.push_back(20);
    v.push_back(30);

    for (int i = 0; i < v.size(); ++i)
        cout << v[i] << endl;                          // 10… 20… 30

    vector<int> zeroes(100), fives(10, 5);
    v = zeroes;

    cout << v.size()     << ' ' << v.front()    << endl;     // 100 0
    cout << fives.size() << ' ' << fives.back() << endl;     // 10 5

    return 0;
}
```

std::vector<T> – динамический массив



ИТЕРАТОРЫ

```
int arr[N];

for (int i = 0; i < N; i++)
    func(arr[i]);

////////////////////////////////

struct List {
    int val;
    List *next;
};

for (List *p = head; p; p = p->next)
    func(p);

////////////////////////////////

struct Tree {
    int val;

    Tree *left, *right;
};

for (Tree *p = root; ....) {
    // ....
}
```

Индекс в массиве (указатель)

Указатель на элемент списка

Указатель на узел дерева

Указатель в массиве

Указатель в списке

Указатель в дереве

А б с т р а к ц и я

Итератор

Операции

- Обращение к элементу
- Сдвиг вперёд на 1
- Сдвиг назад на 1
- Сдвиг вперёд на N
- Сдвиг назад на N
- Разность между двумя итераторами

Итераторы с произвольным доступом

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<int> a = { 1, 2, 3, 4 };

    for (vector<int>::const_iterator it = a.begin(); it != a.end(); ++it)
        cout << *it << ' ';

    return 0;
}
```

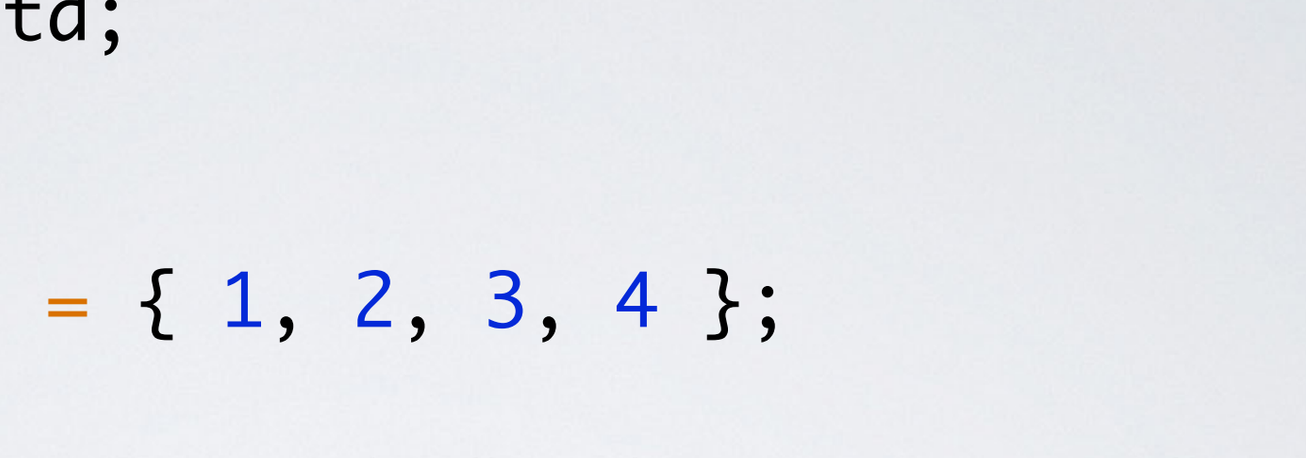
Итераторы у std::vector<T>

```cpp
vector<int> a = { 1, 2, 3, 4 };

for (auto it = a.cbegin(); it != a.cend(); ++it)
    cout << *it << ' ';      // 1 2 3 4

for (auto &x: a)
    ++x;

for (auto x: a)
    cout << x << ' ';        // 2 3 4 5
```

Range-based
for-loop

auto и range-based for-loop (C++11)

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<int> a = { 1, 2, 3, 4 };

    for (auto it = a.rbegin(); it != a.rend(); ++it)
        cout << *it << ' ';      // 4 3 2 1

    cout << endl;
    return 0;
}
```



Обратные итераторы

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<int> a = { 1, 2, 3, 4, 5 };

    a.resize(6);          // 1 2 3 4 5 0
    a.resize(8, 99);      // 1 2 3 4 5 0 99 99
    a.resize(3);          // 1 2 3
    a.pop_back();         // 1 2

    a.insert(a.begin() + 1, 7);   // 1 7 2

    vector<int> b = { -1, -2 };

    a.insert(a.begin(), b.begin(), b.end());  // -1 -2 1 7 2
    //         куда       откуда    докуда

    a.insert(a.end(), b.rbegin(), b.rend());  // -1 -2 1 7 2 -2 -1

    a.erase(a.begin() + 2);                   // -1 -2 7 2 -2 -1
    a.erase(a.begin() + 3, a.end());          // -1 -2 7

    a.swap(b);  // a <==> b
    a.clear();  // очистка массива

    for (int x: a)
        cout << x << endl;

    return 0;
}
```

resize

pop_back

insert

erase

swap

clear

Операции с std::vector<T>

```cpp
#include <iostream>
#include <algorithm>
#include <iterator>
#include <vector>
using namespace std;

bool cmp(int x, int y) { return y < x; }
bool isEven(int x)      { return x % 2 == 0; }

int main() {
    vector<int> a = { 1, 4, 9 };
    a.resize(6);    // 1 4 9 0 0 0

    //    откуда       докуда            куда
    copy(a.begin(), a.begin() + 3, a.begin() + 3);  // 1 4 9 1 4 9

    vector<int> b;
    copy(a.rbegin(), a.rend(), back_inserter(b));   // b: 9 4 1 9 4 1

    sort(a.begin(), a.end());                       // a: 1 1 4 4 9 9
    sort(b.begin(), b.end(), cmp);                  // b: 9 9 4 4 1 1

    partition(a.begin(), a.end(), isEven);          // a: 4 4 1 1 9 9
    rotate(a.begin(), a.begin() + 3, a.end());      // a: 1 9 9 4 4 1

    for (int x: a)
        cout << x << endl;

    return 0;
}
```

Итераторы и алгоритмы

http://www.cplusplus.com/

КОНЕЦ ПЯТОЙ ЛЕКЦИИ

```
std::cout << "Bye!" << endl;
```