

Объектно-ориентированное программирование на алгоритмическом языке C++

МИРЭА, Институт Информационных
технологий, кафедра Вычислительной техники

Список

Список - это контейнер с двунаправленным последовательным доступом к его элементам

```
#include <list>
```

```
list < тип > «имя списка»
```

```
list < тип > :: iterator «имя итератора»
```



Некоторые методы списка

Метод	Описание
<code>assign (m, const T & значение = T())</code>	Присваивает списку <code>m</code> элементов, каждый элемент равно параметру <code>значение</code>
<code>back ()</code>	Возвращает ссылку на последний элемент списка
<code>begin ()</code>	Возвращает итератор первого элемент списка
<code>clear ()</code>	Удаляет все элементы списка
<code>empty ()</code>	Возвращает истинное значение если список пуст, иначе ложь
<code>end ()</code>	Возвращает итератор конца списка
<code>Erase (iterator it)</code>	Удаляет элемент, на который указывает итератор <code>it</code> . Возвращает итератор, указывающий на элемент, который расположен после удаленного
<code>erase (iterator start, iterator end)</code>	Удаляет элементы, заданные между итераторами <code>start</code> и <code>end</code> . Возвращает итератор, указывающий на элемент, который расположен за последним удаленным
<code>front ()</code>	Возвращает ссылку на первый элемент списка
<code>insert (iterator it, const T & val = T())</code>	Вставляет параметр <code>val</code> перед элементом, заданным итератором <code>it</code> . Возвращает итератор элемента
<code>pop_back ()</code>	Удаляет последний элемент списка
<code>push_back (const T & val)</code>	Добавляет в конец списка элемент, значение которого равно параметру <code>val</code>

Пример использования списка

```
#include <iostream>
#include <list>
using namespace std;
int main()
{
    list < char > lst; // создание пустого списка
    int i;

    for ( i = 0; i < 10; i++ ) lst.push_back ( 'A' + i );
    cout << "Размер = " << lst.size () << endl;

    list < char > :: iterator p;
    cout << "Содержимое: ";
    while ( ! lst.empty ( ) ) {
        p = lst.begin ( );
        cout << *p;
        lst.pop_front ( );
    }
    return 0;
}
```

Размер =10

Содержимое: ABCDEFGHIJ

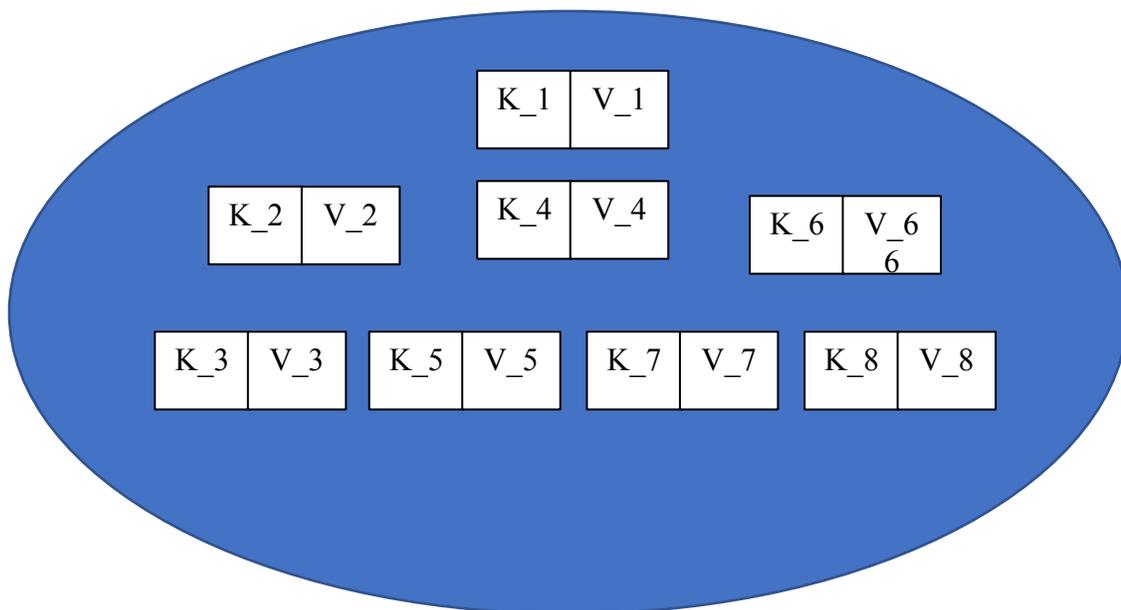
Ассоциативный список

Ассоциативный список - контейнер, в котором уникальным ключам соответствуют определенные значения

```
#include <map>
```

```
map < тип key, тип val > «имя ассоциативного списка»
```

```
map < тип key, тип val > :: iterator «имя итератора»
```



Некоторые методы ассоциативного списка

Метод	Описание
<code>assign (m, const T & значение = T())</code>	Присваивает списку <code>m</code> элементов, каждый элемент равно параметру <code>значение</code>
<code>back ()</code>	Возвращает ссылку на последний элемент списка
<code>begin ()</code>	Возвращает итератор первого элемент списка
<code>clear ()</code>	Удаляет все элементы списка
<code>empty ()</code>	Возвращает истинное значение если список пуст, иначе ложь
<code>end ()</code>	Возвращает итератор конца списка
<code>Erase (iterator it)</code>	Удаляет элемент, на который указывает итератор <code>it</code> . Возвращает итератор, указывающий на элемент, который расположен после удаленного
<code>erase (iterator start, iterator end)</code>	Удаляет элементы, заданные между итераторами <code>start</code> и <code>end</code> . Возвращает итератор, указывающий на элемент, который расположен за последним удаленным
<code>front ()</code>	Возвращает ссылку на первый элемент списка
<code>insert (iterator it, const T & val = T())</code>	Вставляет параметр <code>val</code> перед элементом, заданным итератором <code>it</code> . Возвращает итератор элемента
<code>pop_back ()</code>	Удаляет последний элемент списка
<code>push_back (const T & val)</code>	Добавляет в конец списка элемент, значение которого равно параметру <code>val</code>

Пример использования ассоциативного списка

```
#include <iostream>
#include <map>
using namespace std;
int main ( ) {
    map < char, int > m;
    int i;
    // размещение пар в ассоциативном списке
    for ( i=0; i<10; i++ ) {
        m.insert ( pair < char, int > ( 'A' + i, i ) );
    }
    char ch;
    cout << "Введите ключ: ";
    cin >> ch;

    map < char, int > :: iterator p;
    p = m.find ( ch ); // поиск значения по заданному ключу
    if ( p != m.end ( ) )
        cout << p -> second;
    else
        cout << "Такого ключа в ассоциативном списке нет\n";

    return 0;
}
```

Алгоритмы

Алгоритмы обрабатывают данные содержащиеся в контейнерах

`#include <algorithm>`

<code>copy</code>	Копирует последовательность
<code>count</code>	Возвращает число элементов в последовательности
<code>count_if</code>	Возвращает число элементов в последовательности, удовлетворяющих некоторому предикату
<code>equal</code>	Определяет идентичность двух диапазонов
<code>fill</code> <code>fill_n</code>	Заполняет диапазон заданным значением
<code>find</code>	Выполняет поиск диапазона для значения и возвращает первый найденный элемент
<code>includes</code>	Определяет, включает ли одна последовательность все элементы другой последовательности
<code>max</code>	Возвращает максимальное из двух значений
<code>max_element</code>	Возвращает итератор максимального элемента внутри диапазона

Алгоритмы

merge	Выполняет слияние двух упорядоченных последовательностей, а результат размещает в третьей последовательности
min	Возвращает минимальное из двух значений
partial_sort	Сортирует диапазон
random_shuffle	Беспорядочно перемешивает последовательность
remove	Удаляет элементы из заданного диапазона
replace	Заменяет элементы внутри диапазона
reverse	Меняет порядок сортировки элементов диапазона на обратный
rotate	Выполняет циклический сдвиг влево элементов в диапазоне
search	Выполняет поиск подпоследовательности внутри последовательности
sort	Сортирует диапазон
swap	Меняет местами два значения
transform	Назначает функцию диапазону элементов и сохраняет результат в новой последовательности
unique	Удаляет повторяющиеся элементы из диапазона

Пример использования алгоритма

```
// Демонстрация алгоритмов count и count_if
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

/* это унарный предикат, который определяет, является ли значение четным */
bool even ( int x ) { return ! ( x % 2 ); }

int main ( ) {
    vector < int > v;
    int    i;

    for ( i=0; i<20; i++ ) {
        if ( i % 2 ) v.push_back ( 1 );
        else      v.push_back ( 2*i );
    }

    cout << "Последовательность: ";
    for ( i=0; i < v.size (); i++ ) cout << v[i] << " ";
    cout << endl;
}
```

Пример использования алгоритма

```
int n;  
  
n = count ( v.begin (), v.end (), 1 );  
cout << n << " элементов равно 1\n";  
  
n = count_if ( v.begin (), v.end (), even );  
// int count_if <T1> ( vector <T1>::iterator , vector <T1>::iterator ,  
bool (*p) ( <T1> )) ;  
cout << n << " четных элементов \n";  
  
return 0 ;  
}
```

После выполнения программы на экране появится следующее:

Последовательность: 0 1 2 1 4 1 8 1 12 1 16 1 20 1 24 1 28 1 32 1 36 1

10 элементов равно 1

10 четных элементов

Пример использования алгоритма

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

int main(){
    vector < int > v;
    int i;

    for ( i=0; i<20; i++ )
        v.push_back ( rand()%20 );

    cout << "Последовательность: ";
    for ( i=0; i < v.size (); i++ )
        cout << v[i] << " ";
    cout << endl;
```

Пример использования алгоритма

```
sort (v.begin()+5,v.end()-5);  
cout << "Первая сортировка: ";  
for ( i=0; i < v.size (); i++ ) cout << v[i] << " ";  
    cout << endl;
```

```
sort (v.begin(),v.end());  
cout << "Вторая сортировка: ";  
for ( i=0; i < v.size (); i++ ) cout << v[i] << " ";  
    cout << endl;
```

```
return 0;  
} //int main()
```

После выполнения программы на экране появится следующее:

```
Последовательность: 1 7 14 0 9 4 18 18 2 4 5 5 1 7 1 11 15 2 7 16  
Первая сортировка: 1 7 14 0 9 1 1 2 4 4 5 5 7 18 18 11 15 2 7 16  
Вторая сортировка: 0 1 1 1 2 2 4 4 5 5 7 7 7 9 11 14 15 16 18 18
```

Сортировка произвольного класса

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

class vect3{
public:
    double x,y,z;
    vect3(double xx=0,double yy=0,double zz=0){x=xx;y=yy;z=zz;}
    void show();
    double mod();
};

void vect3::show(){
    cout<<"{"<<x<<" ";<<y<<" ";<<z<<" }";
}

double vect3::mod(){
    return sqrt (x*x + y*y + z*z);
}

bool comp(vect3 A, vect3 B){
    if(A.mod()<B.mod())return true;
    else return false;
}

}
```

Сортировка произвольного класса

```
int main(){
    vector < vect3 > v;

    for(int i=0;i<5;i++)
        v.push_back( vect3(rand()%10,rand()%10,rand()%10) );

    vector <vect3>::iterator i;
    for(i = v.begin();i!=v.end();i++){
        i->show();
        cout<<"mod = "<<i->mod()<<endl;
    }

    sort(v.begin(), v.end(), comp);

    cout<<"после сортировки:"<<endl;
    for(i = v.begin();i!=v.end();i++){
        i->show();
        cout<<"mod = "<<i->mod()<<endl;
    }

    return 0;
}
```

```
{4;7;1}mod = 8.12404
{4;9;0}mod = 9.84886
{2;8;8}mod = 11.4891
{5;5;4}mod = 8.12404
{1;7;1}mod = 7.14143
после сортировки:
{1;7;1}mod = 7.14143
{4;7;1}mod = 8.12404
{5;5;4}mod = 8.12404
{4;9;0}mod = 9.84886
{2;8;8}mod = 11.4891
```