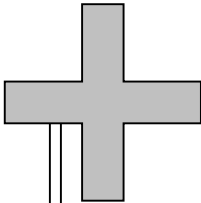



Стандартная библиотека шаблонов языка C++

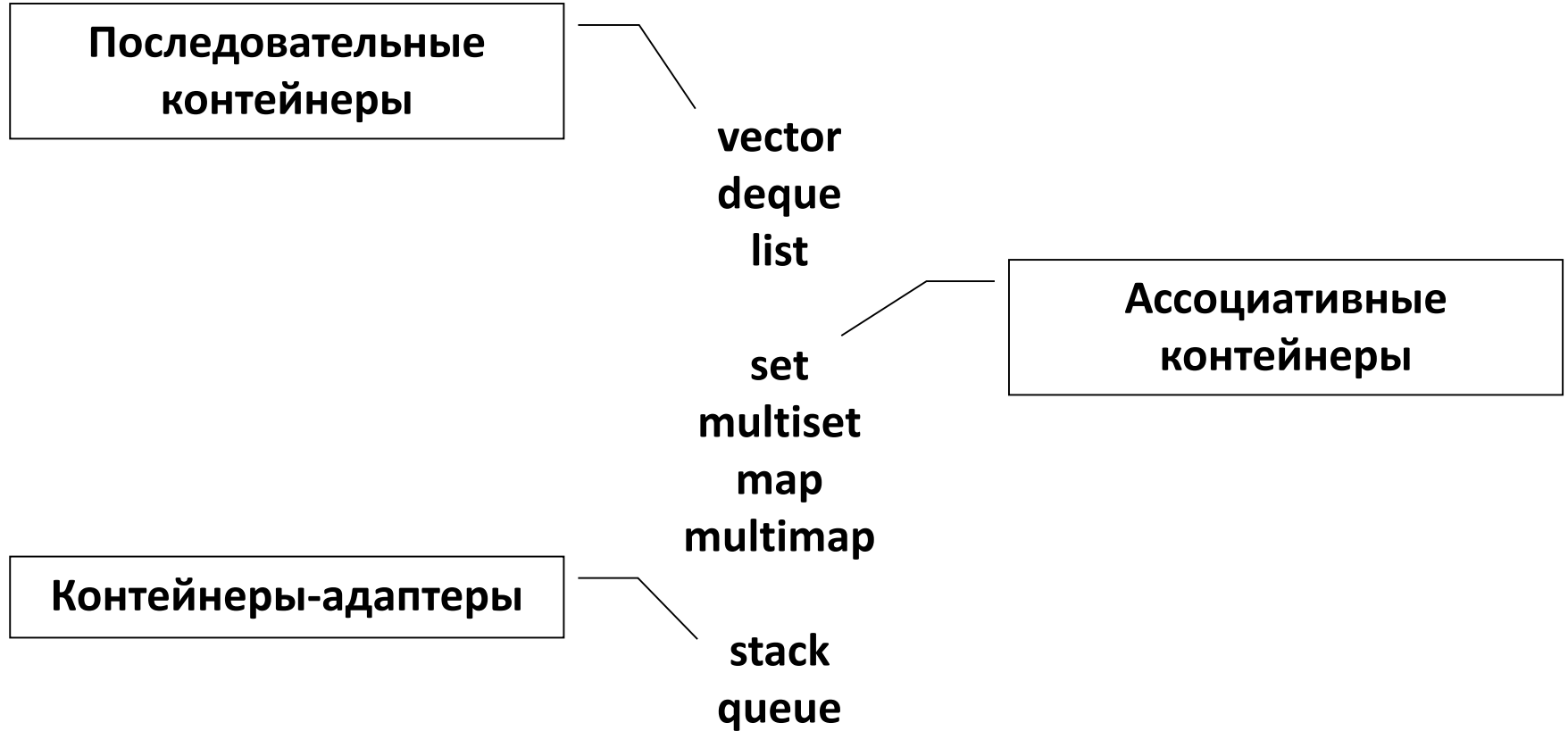


К счастью, с зарядками проблема была решена, теперь у нас у всех стандартизированные mini-USB. Или micro-USB? Чёрт.

- 
1. Код написан профессионалами.
 2. Код написан эффективно.
 3. Унифицированный интерфейс.
 4. Документация.
 5. Переносимость.

- 
1. Неприспособленность для структурных типов.
 2. Низкая эффективность решения частных задач.
 3. Сложность интерфейса для строк.
 4. Сложность управления пулом памяти.

Основные контейнеры STL



Пример использования контейнера vector

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

using namespace std;

void main() {
    vector<int> v1;
    v1.push_back(10);
    v1.push_back(20);
    cout << "The second integer of v1 is "
         << v1[1] << endl;
}
```

Пример использования контейнера deque

```
#include <deque>
#include <iostream>

using namespace std;

void main() {
    deque <int> d1;
    d1.push_front(2);
    d1.push_front(3);
    d1.push_back(1);
    for (int i = 0; i < d1.size(); i++)
        cout << d1[i] << ' ';
}
```

Пример использования контейнера list

```
#include <list>
#include <iostream>

using namespace std;

void main() {
    list<int> l1;
    l1.push_back(1);
    l1.push_front(1);
    l1.insert(++l1.begin(), 3);
    l1.push_back(10);
    l1.push_back(11);
    list<int>::iterator i;
    for (i = l1.begin(); i != l1.end(); ++i)
        cout << *i << ' ';
}
```

Иерархия старшинства итераторов

Итератор произвольного доступа

++, --, [], арифметические операции

Двунаправленный итератор

++, --

Однонаправленный итератор

++

Итератор ввода

read

Итератор вывода

write



Пример использования итератора

```
struct Circle { ... };

int main() {
    Circle c1(10), c2(20);
    map<string,Circle> circles;
    circles["one"] = c1;
    circles["two"] = c2;

    map<string,Circle>::iterator it = circles.begin();
    map<string,Circle>::const_iterator end = circles.end();
    for ( ; it != end; ++it) {
        cout << "NameIs: " << (it->first).c_str() << endl;
        cout << "Radius: " << (it->second).radius << endl;
    }
}
```


Пример использования алгоритма

```
#include <vector>
#include <algorithm>
#include <iterator>
#include <iostream>

int main() {
    using namespace std;
    const int n = 4;
    int a[n];
    generate(a, a+n, rand);
    copy(a, a+n, ostream_iterator<int>(cout, " "));
}
```

Функтором (или функциональным объектом) называется любой объект, к которому можно применить оператор вызова функции – `operator()`. Такой вызов в C++ применим к имени функции, указателю на функцию или объекту класса, который переопределяет `operator()`.

Предикатом называется функция (или функциональный объект), которая в зависимости от значения аргументов может возвращать либо `true`, либо `false`.

Пример реализации

```
bool  is_greater_than(int arg1,int arg2)
{
    return arg1 > arg2;
}
```

```
const int n=4;
int a[n]= {9,7,3,4};
sort(a,a+n, is_greater_than);
copy(a,a+n,std::ostream_iterator<int>
      (std::cout," "));
```

Пример реализации

```
const int n=4;  
int a[n]= {9,7,3,4};  
sort(a,a+n,std::greater<int>());  
copy(a,a+n,std::ostream_iterator<int>  
      (std::cout," "));
```

Пример реализации

```
bool  is_greater_than_7(int x) {  
    return x > 7;  
}
```

```
const int n=4;  
int a[n]= {9,7,3,4};  
int *it = std::find_if(a, a+n,  
                       is_greater_than_7);  
std::cout << *it;
```

Пример реализации

```
class GreaterThan {
    int limit;
public:
    GreaterThan(int limit): limit(limit) {}
    bool operator()(int x) const {
        return x > limit;
    }
};

const int n=4;
int a[n]= {2,9,7,3};
int *it = std::find_if(a, a+n,
                       GreaterThan(7));
std::cout << *it;
```

Пример реализации

```
const int n = 4;  
int a[n]= {2,9,7,3};  
int *it = find_if(a,a+n,  
    std::bind2nd(std::greater<int>(),7));  
cout << *it;
```