Курс c++

Царьков Олег

Описание тикета OTSN-10

Статичные переменные и функции класса.

Переменные и функции, объявленые в классе, считаются принадлежащими каждой переменной данного типа.

Для разных объектов класса переменные принимают разные значения.

```
class MyClass {
private:
    int a;
};

int main() {
    MyClass var1;
    MyClass var2;
    return 0;
}
```

Тут переменные $var1.a$ и $var2.a$ живут разной жизнью. Одна — это переменная a объекта $var1$, вторая — переменная a объекта $var2$.

Статичная переменная — это переменная, являющаяся общей для всех переменных класса.

Статичная функция — это как будто функция, находящаяся вне класса.

```
#include <iostream>

class MyClass {
public:
    static const int CONST_VAR = 5;
private:
    int a;
public:
    static void SomeFunc() {
        std::cout << "text\n";
        return;
    }
};

int main() {
    MyClass var1;
```

```
    MyClass var2;
    std :: cout << MyClass :: CONST_VAR << "\ n";
    MyClass :: SomeFunc();
    return 0;
}
```

Доступ к статичным объектам класса производится через ::, перед которыми написано имя класса, а не имя объекта.

Таким образом, функция $SomeFunc()$ и переменная $CONST\_VAR$ в данном случае — просто общие для всех объектов класса.

Задание 1. Написать класс , в котором есть статичная функция $InstanceNumber()$, возвращающая количество переменных класса, объявленых с момента начала программы до момента вызова этой функции.

Объявление типа с помощью typedef.

```
#include < iostream >

int main() {
    typedef int MyType;
    MyType a;
    a = 4;
    a += 1;
    std :: cout << a << "\ n";
    return 0;
}
```

Строка $typedef\ int\ MyType;$ объявляет новое название типа. Тип, написанный вторым после слова $typedef$ будет новым названием для типа, написанного первым после слова $typedef$.

Задание 2. Ввести новое название для типа $std :: vector < int >$.

Шаблон проектирования "итератор".

Контейнер — объект, содержащий в себе неопределенное число других объектов.

Например, $std :: vector$ — это контейнер. Он может содержать много переменных сразу, причем сколько именно — не известно заранее, его размер может меняться.

Итератор — это объект некоторого класса, который ведет себя как указатель для элементов контейнера.

Имеется ввиду следующее:

• Имеется некий контейнер, и еще один объект класса(итератор), притворяющийся указателем на элементы контейнера

• У него есть *operator++*, который заставляет указывать его на следующий элемент контейнера

• У него есть унарный *operator**, который возвращает элемент контейнера(притворяется операцией разыменовывания)

• *operator* == и *operator*! = сравнивающий указывают ли два итератора на один и тот же элемент

• у контейенра есть две функции *begin*(), возвращающая итератор на первый элемент, и *end*(), возвращающая итератор на элемент после последнего.

Рассмотрим пример с итератором для вектора.

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

int main() {
    std::vector<int> vector(3);
    vector[0] = 1;
    vector[1] = 2;
    vector[2] = 3;
    for (std::vector<int>::iterator it = vector.begin(); it != vector.end(); ++
it) {
        std::cout << *it << " ";
    }
    std::cout << "\n";
    const std::vector<int> vector2(3, 5);
    for (std::vector<int>::const_iterator it = vector2.begin(); it != vector2.end(); ++
it) {
        std::cout << *it << " ";
    }
    std::cout << "\n";
}
```

Единственная сложность, что нужно отдельно писать класс итераторов для константного контейнера и для неконстантного, так как итератор на константный объект не должен позволять менять этот объект и в определении его функций будет везде слово *const*, но для неконстантных объектов нужен итератор, позволяющий менять объект, поэтому нужно писать два разных класса *iterator* и *const_iterator*.

Задание 3. Написать итераторы для класса матриц и класса *Array*, написанных ранее.