Шаблонные функции и классы

Мстоян Амиран

Разбор домашнего задания

Мотивация использования шаблонных функци

Поиск максимума

```
1 int max(int a, int b)
2 {
3     return (a > b) ? a : b;
4 }
```

```
1 double max(double a, double b)
2 {
3     return (a > b) ? a : b;
4 }
```

Шаблонный вариант

```
1 template <typename T> // объявление параметра шаблона функции
2 T max(T a, T b)
3 {
4     return (a > b) ? a : b;
5 }
```

Применение

```cpp
#include <iostream>

template <typename T>
const T& max(const T& a, const T& b)
{
    return (a > b) ? a : b;
}

int main()
{
    int i = max(4, 8);
    std::cout << i << '\n';

    double d = max(7.56, 21.434);
    std::cout << d << '\n';

    char ch = max('b', '9');
    std::cout << ch << '\n';

    return 0;
}
```

Работа шаблонов класса с пользовательскими классами

```
1  template <typename T> // объявление параметра шаблона функции
2  const T& max(const T& a, const T& b)
3  {
4      return (a > b) ? a : b;
5  }
6
7  class Dollars
8  {
9  private:
10     int m_dollars;
11 public:
12     Dollars(int dollars)
13         : m_dollars(dollars)
14     {
15     }
16 };
17
18 int main()
19 {
20     Dollars seven(7);
21     Dollars twelve(12);
22
23     Dollars bigger = max(seven, twelve);
24
25     return 0;
26 }
```

```
1 const Dollars& max(const Dollars &a, const Dollars &b)
2 {
3     return (a > b) ? a : b;
4 }
```

```
class Dollars
{
private:
    int m_dollars;
public:
    Dollars(int dollars)
        : m_dollars(dollars)
    {
    }

    friend bool operator>(const Dollars &d1, const Dollars &d2)
    {
        return (d1.m_dollars > d2.m_dollars);
    }
};
```

Еще один пример

```
1  #include <iostream>
2
3  template <class T>
4  T average(T *array, int length)
5  {
6      T sum = 0;
7      for (int count=0; count < length; ++count)
8          sum += array[count];
9
10     sum /= length;
11     return sum;
12 }
13
14 int main()
15 {
16     int array1[] = { 6, 4, 1, 3, 7 };
17     std::cout << average(array1, 5) << '\n';
18
19     double array2[] = { 4.25, 5.37, 8.44, 9.25 };
20     std::cout << average(array2, 4) << '\n';
21
22     return 0;
23 }
```

```cpp
#include <iostream>

class Dollars
{
private:
    int m_dollars;
public:
    Dollars(int dollars)
        : m_dollars(dollars)
    {
    }

    friend bool operator>(const Dollars &d1, const Dollars &d2)
    {
        return (d1.m_dollars > d2.m_dollars);
    }
};
```

```cpp
int main()
{
    Dollars array3[] = { Dollars(7), Dollars(12), Dollars(18), Dollars(15) };
    std::cout << average(array3, 4) << '\n';

    return 0;
}
```

```
class Dollars
{
private:
    int m_dollars;
public:
    Dollars(int dollars)
        : m_dollars(dollars)
    {
    }

    friend bool operator>(const Dollars &d1, const Dollars &d2)
    {
        return (d1.m_dollars > d2.m_dollars);
    }

    friend std::ostream& operator<< (std::ostream &out, const Dollars &dollars)
    {
        out << dollars.m_dollars<< " dollars ";
        return out;
    }

    Dollars& operator+=(Dollars dollars)
    {
        m_dollars += dollars.m_dollars;
        return *this;
    }

    Dollars& operator/=(int value)
    {
        m_dollars /= value;
        return *this;
    }
};
```

Шаблоны классов

- Практическое применение
- Мотивация использования
- Плюсы и минусы

```
class ArrayInt
{
private:
    int m_length;
    int *m_data;

public:
    ArrayInt()
    {
        m_length = 0;
        m_data = nullptr;
    }

    ArrayInt(int length)
    {
        assert(length > 0);
        m_data = new int[length];
        m_length = length;
    }

    ~ArrayInt()
    {
        delete[] m_data;
    }

    void Erase()
    {
        delete[] m_data;
        // Присваиваем значение nullptr для m_data, чтобы на выходе не получить висячий указатель!
        m_data = nullptr;
        m_length = 0;
    }

    int& operator[](int index)
    {
        assert(index >= 0 && index < m_length);
        return m_data[index];
    }

    int getLength() { return m_length; }
};
```

```
class ArrayDouble
{
private:
    int m_length;
    double *m_data;

public:
    ArrayDouble()
    {
        m_length = 0;
        m_data = nullptr;
    }

    ArrayDouble(int length)
    {
        assert(length > 0);
        m_data = new double[length];
        m_length = length;
    }

    ~ArrayDouble()
    {
        delete[] m_data;
    }

    void Erase()
    {
        delete[] m_data;
        // Присваиваем значение nullptr для m_data, чтобы на выходе не получить висячий указатель!
        m_data = nullptr;
        m_length = 0;
    }

    double & operator[](int index)
    {
        assert(index >= 0 && index < m_length);
        return m_data[index];
    }

    int getLength() { return m_length; }
};
```

```cpp
template <class T> // это шаблон класса с T вместо фактического (передаваемого) типа данных
class Array
{
private:
    int m_length;
    T *m_data;

public:
    Array()
    {
        m_length = 0;
        m_data = nullptr;
    }

    Array(int length)
    {
        m_data = new T[length];
        m_length = length;
    }

    ~Array()
    {
        delete[] m_data;
    }

    void Erase()
    {
        delete[] m_data;
        // Присваиваем значение nullptr для m_data, чтобы на выходе не получить висячий указатель!
        m_data = nullptr;
        m_length = 0;
    }


    T& operator[](int index)
    {
        assert(index >= 0 && index < m_length);
        return m_data[index];
    }

    // Длина массива всегда является целочисленным значением, она не зависит от типа элементов массива
    int getLength(); // определяем метод и шаблон метода getLength() ниже
};
```

```
template <typename T> // метод, определенный вне тела класса, нуждается в собственном определении шаблона метода
int Array<T>::getLength() { return m_length; } // обратите внимание, имя класса - Array<T>, а не просто Array
```

Домашнее задание

- Задача1 :
  - реализовать методы шаблонного ікласса Complex_number

- Задача 2:
  -реализовать методы шаблонного класса Auto_ptr1