

Семинар #1: Основы C++. Домашнее задание.

Пространства имён, ссылки, перегрузка функций, `std::string` и `std::vector`.

Задача 1. Пространства имён

Пусть есть такой участок кода:

```
namespace mipt
{
    namespace fefm
    {
        struct Point
        {
            int x, y;
        };
    }

    namespace frtk
    {
        void print(fefm::Point p)
        {
            std::cout << p.x << " " << p.y << std::endl;
        }
    }
}
```

Вам нужно сделать следующее:

- В функции `main` создать переменную типа `Point` (из пространства имён `mipt::fefm`) и инициализировать её поля значениями `x = 10` и `y = 20`.
- Вызвать функцию `print` из пространства имён `mipt::frtk`, передав ей созданную структуру.

Решите эту задачу тремя способами:

1. Без использования ключевого слова `using`.
2. С использованием директив `using namespace`.
3. С использованием `using-объявлений`.

Задача 2. Куб

Напишите функцию `cube`, которая будет принимать одно число типа `int` по ссылке и возводить это число в куб. Вызовите эту функцию из функции `main`, чтобы возвести переменную типа `int` в куб.

```
#include <iostream>
// Тут нужно написать функцию cube

int main()
{
    int a = 5;
    cube(a);
    std::cout << a << std::endl; // Должно напечатать 125
}
```

Задача 3. Обмен

Напишите функцию `swap`, которая будет обменивать значения двух переменных типа `int`.

```
#include <iostream>
// Тут нужно написать функцию swap

int main()
{
    int a = 10;
    int b = 20;
    std::cout << a << " " << b << std::endl; // Должно напечатать 10 20

    swap(a, b);
    std::cout << a << " " << b << std::endl; // Должно напечатать 20 10
}
```

Задача 4. Возврат ссылки на максимум

Написан следующий код, в котором используются указатели:

```
#include <iostream>
int* getPointerToMax(int* pa, int* pb)
{
    if (*pa > *pb)
        return pa;
    else
        return pb;
}

int main()
{
    int a = 10;
    int b = 20;

    *getPointerToMax(&a, &b) += 1;

    std::cout << a << " " << b << std::endl;
}
```

Перепишите этот код, но вместо указателей используйте ссылки. В коде не должно остаться ни одного указателя. Название функции `getPointerToMax` измените на `getRefToMax`.

Задача 5. Передача структуры по константной ссылке

Пусть у нас есть следующая структура:

```
struct Book
{
    std::string title;
    int pages;
    float price;
};
```

Напишите функцию `isExpensive`, которая будет принимать на вход структуру `Book` по константной ссылке. Эта функция должна возвращать значение типа `bool`. Если цена книги больше чем 1000, то функция должна вернуть `true`, иначе функция должна вернуть `false`. Протестируйте эту функцию в функции `main`.

Задача 6. Выбор перегрузки

Пусть есть следующие перегруженные функции:

```
void cat(char x)      {std::cout << "Char" << std::endl;}
void cat(int x)       {std::cout << "Int" << std::endl;}
void cat(long long x) {std::cout << "Long Long" << std::endl;}
```

Какая из перегрузок будет выбрана (или будет ошибка), если вызвать `cat` следующим образом:

1. `cat(10)`
2. `cat(10LL)`
3. `cat(static_cast<short>(10))`
4. `char a = 10;`
`char b = 20;`
`cat(a + b);`
5. `cat(1.0f)`

Пусть также есть следующие перегруженные функции:

```
void dog(int& x)      {std::cout << "Ref" << std::endl;}
void dog(const int& x) {std::cout << "CRef" << std::endl;}
```

Какая из перегрузок будет выбрана (или будет ошибка), если вызвать `dog` следующим образом:

6. `int a = 10;`
`dog(a);`
7. `const int a = 10;`
`dog(a);`
8. `dog(10);`
9. `int a = 10;`
`dog(static_cast<int>(a));`
10. `int a = 10;`
`dog(static_cast<int&>(a));`
11. `int a = 10;`
`dog(+a);`
12. `int func() {return 10;}`
`int main()`
{
 dog(func());
}

Пусть также есть следующие перегруженные функции:

```
void cow(const char* str)      {std::cout << "C-string" << std::endl;}
void cow(const std::string& str) {std::cout << "std::string" << std::endl;}
```

Какая из перегрузок будет выбрана (или будет ошибка), если вызвать `cow` следующим образом:

13. `char a[10] = "Hello";`
`cow(a);`
14. `std::string a = "Hello";`
`cow(a);`

```

15. cow("Hello");
16. cow(std::string("Hello"));
17. using namespace std::string_literals;
   int main()
   {
      cow("Hello"s);
   }
18. std::string a = "Hello";
   cow(a.c_str());

```

Для того, чтобы сдать эту задачу нужно создать файл в формате .txt и, используя любой текстовый редактор, записать в него ответы в следующем формате (ответы ниже неверны):

- 1) Char
- 2) C-string
- 3) Error

После этого, файл нужно поместить в ваш репозиторий на github.

Задача 7. Добавление скобок

Напишите функцию addBrackets, которая будет добавлять к строке квадратные скобки. Протестируйте эту функцию на следующем коде:

```

#include <iostream>
#include <string>
int main()
{
    std::string a = "Cat";
    addBrackets(a);
    std::cout << a << std::endl; // Должно напечатать [Cat]

    addBrackets(a);
    std::cout << a << std::endl; // Должно напечатать [[Cat]]
}

```

Задача 8. Подсчёт символов

Напишите функцию:

```
void countLetters(const std::string& str, int& numLetters, int& numDigits)
```

которая будет принимать на вход строку str и подсчитывать число букв и цифр в этой строке. Количество букв нужно записать по ссылке numLetters, а количество цифр – по ссылке numDigits. Используйте библиотеку `cctype`. Вызвать эту функцию из функции main.

Задача 9. Повтор числа в строке

Напишите функцию repeat, которое будет принимать на вход некоторое целое число n и возвращать строку, содержащую это число в десятичной записи, повторённое n раз. Протестируйте функцию на следующем коде:

```

#include <iostream>
int main()
{
    std::cout << repeat(5) << std::endl; // Должно напечатать 55555
    std::cout << repeat(10) << std::endl; // Должно напечатать 101010101010101010
    std::cout << repeat(-1) << std::endl; // Не должно ничего печатать
}

```

Задача 10. Доменное имя

Напишите функцию `isDomainName`, которая принимает на вход строку и если строка начинается на `www.` и заканчивается на `.com`, то эта функция должна вернуть `true`, иначе `false`. Используйте методы строки `starts_with` и `ends_with`.

```
#include <iostream>
#include <string>
// Тут нужно написать функцию isDomainName

int main()
{
    std::cout << isDomainName("www.google.com") << std::endl;           // Напечатает 1
    std::cout << isDomainName("abc") << std::endl;                         // Напечатает 0
    std::cout << isDomainName("hello.com") << std::endl;                   // Напечатает 0
}
```

Задача 11. Удвоение вектора

Напишите функцию `doubling`, которая будет принимать на вход вектор целых чисел и увеличивать вектор в два раза, путём повтора всех элементов.

```
#include <iostream>
#include <string>
#include <vector>
void print(const std::vector<int>& v)
{
    for (std::size_t i = 0; i < v.size(); ++i)
        std::cout << v[i] << " ";
    std::cout << std::endl;
}
// Тут нужно написать функцию doubling

int main()
{
    std::vector<int> v {10, 20, 30};
    doubling(v);
    print(v); // Должно напечатать 10 20 30 10 20 30
}
```

Задача 12. Склейка строк

Напишите функцию `concatenate`, которая будет принимать вектор из строк и возвращать строку – результат конкатенации всех строк вектора.

```
#include <iostream>
#include <string>
#include <vector>
// Тут нужно написать функцию concatenate

int main()
{
    std::vector<std::string> v {"Cat", "Dog", "Mouse", "Tiger", "Elk"};
    std::cout << concatenate(v) << std::cout // Должно напечатать CatDogMouseTigerElk
}
```

Задача 13. Вектор префиксов

Напишите функцию `prefixes`, которая должна принимать на вход строку, а возвращать вектор, содержащий все подстроки, начинающиеся с начала строки.

```
#include <iostream>
#include <string>
#include <vector>
void print(const std::vector<std::string>& v)
{
    for (std::size_t i = 0; i < v.size(); ++i)
        std::cout << v[i] << " ";
    std::cout << std::endl;
}
// Тут нужно написать функцию prefixes

int main()
{
    std::vector<std::string> v = prefixes("Mouse");
    print(v); // Должно напечатать M Mo Mou Mous Mouse
}
```

Задача 14. Вектор индексов подстрок

Напишите функцию `substringIndexes`, которая будет принимать на вход две строки и будет искать индексы всех вхождений второй строки в первую. Функция должна возвращать вектор этих индексов.

```
#include <iostream>
#include <string>
#include <vector>
void print(const std::vector<std::size_t>& v)
{
    for (std::size_t i = 0; i < v.size(); ++i)
        std::cout << v[i] << " ";
    std::cout << std::endl;
}
// Тут нужно написать функцию substringIndexes

int main()
{
    std::vector<std::size_t> v1 = substringIndexes("cat and dog and cat", "cat");
    print(v1); // Должно напечатать 0 16

    std::vector<std::size_t> v2 = substringIndexes("look, cats were here", "cat");
    print(v2); // Должно напечатать 6

    std::vector<std::size_t> v3 = substringIndexes("catcatcatcatcatcat", "cat");
    print(v3); // Должно напечатать 0 3 6 9 12 15

    std::vector<std::size_t> v4 = substringIndexes("dog mouse elephant", "cat");
    print(v4); // Не должно ничего печатать
}
```

Задача 15. Получение элемента

Напишите три перегрузки функции под названием `get`, которые могли бы принимать некоторый контейнер и индекс. Функции должны возвращать элемент по индексу. Если значение индекса находится вне допустимых значений, функция должна печатать сообщение об ошибке и выходить из программы (используйте для этого функцию `std::exit` из `cstdlib`). Нужно написать три перегрузки функций `get`:

1. Перегрузка, которая принимает строку `std::string` и индекс.
2. Перегрузка, которая принимает вектор целых чисел `std::vector<int>` и индекс.
3. Перегрузка, которая принимает обычный массив целых чисел типа `int`, его размер и индекс.

```
#include <iostream>
#include <string>
#include <vector>
void print(const std::vector<int>& v)
{
    for (std::size_t i = 0; i < v.size(); ++i)
        std::cout << v[i] << " ";
    std::cout << std::endl;
}

void print(const int* a, std::size_t n)
{
    for (std::size_t i = 0; i < n; ++i)
        std::cout << a[i] << " ";
    std::cout << std::endl;
}
// Тут нужно написать перегрузки функции get

int main()
{
    std::vector<int> v {10, 20, 30, 40, 50};
    get(v, 2) += 1;
    print(v); // Напечатает 10 20 31 40 50

    std::string s = "Cat";
    get(s, 0) = 'B';
    std::cout << s << std::endl; // Напечатает Bat

    int a[5] = {10, 20, 30, 40, 50};
    get(a, 5, 2) += 1;
    print(a, 5); // Напечатает 10 20 31 40 50

    get(v, 10) = 0; // Должен напечатать сообщение об ошибке и выйти из программы
}
```

Задача 16. Парсинг по разделителю

Напишите функцию `split`, которая принимает строку и символ-разделитель и возвращает `std::vector<std::string>`, содержащий все подстроки, разделенные этим символом. Используйте методы строки `find` и `substr`.

```
#include <iostream>
#include <string>
#include <vector>
// Тут нужно написать функцию split

int main()
{
    std::string data = "apple,banana,cherry,durian";
    char delimiter = ',';

    std::vector<std::string> parts = split(data, delimiter);
    for (std::size_t i = 0; i < parts.size(); ++i)
        std::cout << parts.at(i) << std::endl;
}
```

Задача 17. Длина слов в файле

Напишите функцию:

```
std::vector<std::vector<std::size_t>> calculate_word_lengths(const std::string& filename)
```

которая будет считывать файл и считать количество символов в каждом слове файла. Под "словом" считается любая последовательность непробельных символов, ограниченная пробельными символами или началом/концом файла. Функция должна возвращать вектор, содержащий вектора количества символов слов для каждой строки. Например, если файл содержит:

```
one two three
hello
axolotl bison cat dog eagle
```

то функция должна вернуть вектор, содержащий:

```
{{{3, 3, 5}, {5}, {7, 5, 3, 3, 5}}}
```


Необязательные задачи (не входят в ДЗ, никак не учитываются)

Задача 1. Измени регистр первой буквы

Напишите функцию, которая будет принимать на вход строку типа `std::string` и возвращать строку с изменинным регистром первой буквы. Например, если на вход пришла строка "Cat", то функция должна вернуть строку "cat". Если на вход пришла строка "dog", то функция должна вернуть строку "Dog". Если на вход пришла пустая строка, то функция должна вернуть простую строку.

Задача 2. Удвоение

Напишите функции, которые будут принимать на вход строку и возвращать строку, повторённую два раза. То есть, если на вход этой функции приходит строка "Cat", то функция должна вернуть "CatCat". При этом нужно написать несколько функций, которые должны делать одно и то же, но возвращать результат разными методами.

- `std::string repeat1(const std::string& s)` Должна принимать на вход строку и возвращать результат.
- `void repeat2(std::string& s)` Должна принимать на вход строку по ссылке и изменять эту строку.
- `void repeat3(std::string* ps)` Должна принимать на вход указатель на строку и изменять строку, чей адрес хранит этот указатель.
- `std::string* repeat4(const std::string& s)` Эта функция должна создавать удвоенную строку в куче с помощью оператора `new` и возвращать указатель на неё. После вызова функции `repeat4` программист, который будет использовать эту функцию, сам должен позаботиться об её удалении.

Протестируйте эти функции в `main`.

Задача 3. Умножение строки

Напишите перегруженный оператор умножения, которая будет принимать на вход строку `std::string` и некоторое целое число `n` и возвращать строку, повторённую `n` раз. Протестируйте эту функцию в функции `main`.

Задача 4. Усечение

Напишите функцию `void truncateToDot(std::string& s)`, которая будет принимать строку по ссылке и усекать её до первого символа точки. Размер и вместимость строки должны стать как можно более маленькими, для этого используйте метод `shrink_to_fit`. Функция не должна ничего возвращать. Протестируйте функцию в `main`.

до	после
"cat.dog.mouse.elephant.tiger.lion"	"cat"
"wikipedia.org"	"wikipedia"
".com"	" "

Задача 5. Сумма чётных

Напишите функцию `sumEven`, которая будет принимать вектор чисел типа `int` по константной ссылке и возвращать сумму всех чётных чисел этого вектора.

аргумент	возвращаемое значение
{4, 8, 15, 16, 23, 42}	70

Задача 6. Последние цифры

Напишите функцию `lastDigits` которая должны будут принимать на вход вектор чисел типа `int` по ссылке и заменять все его элементы на последние цифры этих элементов.