[Лабораторная работа №2. Знакомство со стандартной библиотекой шаблонов языка C++ 2](#_Toc128958051)

[Практические задания 2](#_Toc128958052)

[Важное замечание 2](#_Toc128958053)

[Обязательные задания 2](#_Toc128958054)

[Задание 1. Основы работы с контейнером vector. 20 баллов 2](#_Toc128958055)

[Бонус 10 баллов за сортировку элементов массива в порядке возрастания 3](#_Toc128958056)

[Задание 2 – работа с контейнером string 4](#_Toc128958057)

[Вариант 1 – TrimBlanks – 20 баллов 4](#_Toc128958058)

[Вариант 2 – RemoveExtraSpaces – 30 баллов 4](#_Toc128958059)

[Вариант 3 – FindAndReplace – 40 баллов 4](#_Toc128958060)

[Вариант 4 – HTML Encode – 40 баллов 4](#_Toc128958061)

[Вариант 5 – HTML Decode – 60 баллов 5](#_Toc128958062)

[Задание 3 – работа с контейнером map 6](#_Toc128958063)

[Вариант 1- Подсчет частоты встречаемости слов – 40 баллов 6](#_Toc128958064)

[Бонус 10 баллов за подсчет слов, записанных в case-insensitive формате 6](#_Toc128958065)

[Вариант 2 – мини-словарь – 90 баллов 6](#_Toc128958066)

[Бонус 10 баллов за возможность распознавания слов, записанных в разном регистре 7](#_Toc128958067)

[Бонус 20 баллов за реализацию двунаправленного перевода 7](#_Toc128958068)

[Задание 4. Работа с контейнером set 8](#_Toc128958069)

[Вариант 1 – Cross Sets - 30 баллов 8](#_Toc128958070)

[Вариант 2 – Спортшкола – 40 баллов 8](#_Toc128958071)

[Вариант 3 – Фильтр нецензурных слов – 60 баллов 8](#_Toc128958072)

[Бонус в 10 баллов за распознавание нецензурных слов, записанных в разном регистре 9](#_Toc128958073)

[Вариант 4 – Генератор простых чисел – 100 баллов 9](#_Toc128958074)

[Дополнительные задания 10](#_Toc128958075)

[Задание 5. – регулярные выражения 10](#_Toc128958076)

[Вариант 1 – парсер URL-ов – 80 баллов 10](#_Toc128958077)

[Задание 6 11](#_Toc128958078)

[Вариант 1 – Expand Template - 100 баллов 11](#_Toc128958079)

[Бонус в 150 баллов за эффективное решение данной задачи с использованием алгоритма «Ахо-Корасик» 12](#_Toc128958080)

[Задание 7 12](#_Toc128958081)

[Вариант 1 — разбор выражения — 100 баллов 12](#_Toc128958082)

[Бонус в 50 баллов за обработку ошибок 13](#_Toc128958083)

[Бонус в 80 баллов за нерекурсивное решение задачи 13](#_Toc128958084)

# Лабораторная работа №2. Знакомство со стандартной библиотекой шаблонов языка C++

## Практические задания

Для получения оценки «**удовлетворительно**» необходимо набрать не **менее 80 баллов**.

Для получения оценки «**хорошо**» необходимо набрать не **менее 300 баллов.**

Для получения оценки «**отлично**» необходимо набрать не **менее 500 баллов**.

**Внимание**, дополнительные задания принимаются только после успешной защиты обязательных заданий.

### Важное замечание

**При выполнении заданий следует отдавать предпочтение использованию алгоритмов STL, нежели написанию «сырых» циклов. Рекомендуется посмотреть доклад (англ.) C++ Seasoning c конференции Going Native 2013, доступный по ссылке**:

<https://channel9.msdn.com/Events/GoingNative/2013/Cpp-Seasoning>

### Обязательные задания

#### Задание 1. Основы работы с контейнером vector. 20 баллов

Ознакомьтесь с возможностями класса [**vector**](https://en.cppreference.com/w/cpp/container/vector) библиотеки STL, а также с основными [алгоритмами STL](https://en.cppreference.com/w/cpp/algorithm/transform).

Разработайте программу, выполняющую считывание массива чисел с плавающей запятой, разделяемых пробелами, из стандартного потока ввода в vector, обрабатывающую его согласно заданию Вашего варианта и выводящую в стандартный поток полученный массив (разделенный пробелами). В программе должны быть выделены функции, выполняющие считывание массива, его обработку[[1]](#footnote-1) и вывод результата.

**Подсказка**: чтобы вывести числа с желаемой точностью, воспользуйтесь манипуляторами вывода в поток: [std::setprecision](https://en.cppreference.com/w/cpp/io/manip/setprecision), и [std::fixed](https://en.cppreference.com/w/cpp/io/manip/fixed).

Пустой массив, переданный программе – допустимые входные данные. При его обработке пустой массив должен оставаться пустым.

**Для тестирования разрабатываемых функций должны быть разработаны тесты, проверяющие корректность их работы на некотором разумном наборе входных параметров.**

|  |  |
| --- | --- |
| **Вариант** | **Выводимое значение** |
| 1 | Прибавить к каждому элементу массива среднее арифметическое его положительных элементов  Подсказка: используйте алгоритм std::accumulate, чтобы найти сумму положительных элементов. |
| 2 | Каждый элемент массива должен быть умножен на минимальный элемент исходного массива  Подсказка: используйте алгоритм std::min\_element. |
| 3 | Разделить элементы массива на половину максимального элемента.  Подсказка: используйте std::max\_element |
| 4 | Умножить каждый отрицательный элемент массива на произведение максимального и минимального элементов исходного массива  Подсказка: используйте std::minmax\_element, чтобы найти минимальный и максимальный элементы массива |
| 5 | Умножить каждый элемент массива на максимальный элемент исходного массива и разделить на минимальный элемент исходного массива  Подсказка: используйте std::minmax\_element, чтобы найти минимальный и максимальный элементы |
| 6 | Прибавить к каждому элементу массива сумму трех минимальных элементов массива. Если массив содержит меньше трех элементов, надо прибавить сумму имеющихся.  Подсказка: используйте std::partial\_sort\_copy, чтобы найти три минимальных элемента. |
| 7 | Элементы, стоящие на четных позициях массива умножить на 2, а из элементов, стоящих на нечетных позициях, вычесть сумму всех неотрицательных элементов.  Подсказка: используйте std::accumulate, чтобы найти сумму неотрицательных элементов. |
| 8 | Вычесть из каждого элемента исходного массива значение его медианы.  Подсказка: используйте алгоритм std::nth\_element, чтобы найти медианный элемент массива. Учтите, что nth\_element изменяет порядок элементов порядок элементов контейнера. |

##### Бонус 10 баллов за сортировку элементов массива в порядке возрастания

Бонус начисляется за вывод элементов массива в порядке возрастания их значений. При этом функция обработки вектора, разработанная в основной части задания, не должна выполнять сортировку элементов.

#### Задание 2 – работа с контейнером string

Ознакомьтесь с возможностями класса [string](https://en.cppreference.com/w/cpp/string/basic_string) (точнее, шаблона basic\_string) библиотеки STL и выполните задание, соответствующее номеру Вашего варианта.

**Для тестирования разрабатываемых функций должны быть разработаны тесты, проверяющие корректность их работы на некотором разумном наборе входных параметров.**

##### Вариант 1 – TrimBlanks – 20 баллов

Разработайте функцию  
**std::string TrimBlanks(std::string const& str)**,  
выполняющую отрезание пробелов в начале и в конце строки str, и возвращающую результирующую строку. Если строка состоит из одних пробелов, должна вернуться пустая строка.

Разработайте на ее основе программу, выполняющую отрезание пробелов в начале и конце **каждой строки**, поступающей со стандартного потока ввода, и выводящую результат в стандартный поток вывода.

**Внимание, реализация данной функции должна иметь сложность не более O(N).**

##### Вариант 2 – RemoveExtraSpaces – 30 баллов

Разработайте функцию   
**std::string RemoveExtraSpaces(std::string const& arg)**,  
удаляющую из строки, переданной в параметре arg, лишние пробелы. Лишними считаются все пробелы в начале и конце строки, а также дополнительные пробелы между словами.

Разработайте с ее использованием программу, выполняющую удаление лишних пробелов из каждой строки входного потока символов и вывод результирующих строк в выходной поток

**Внимание, реализация данной функции должна иметь сложность не более O(N)**

##### Вариант 3 – FindAndReplace – 40 баллов

Разработайте функцию  
**std::string FindAndReplace(std::string const& subject, std::string const& search, std::string const& replace)**,  
возвращающую результат замены всех вхождений подстроки **search** в строке **subject** на строку **replace**. Если искомая строка пустая, замены строк производиться не должно.

Вычислительная сложность алгоритма, лежащего в основе FindAndReplace, должно линейно зависеть от длины строки Subject

Разработайте на ее основе программу, заменяющую все вхождения искомой строки в стандартном потоке ввода на строку-заменитель и выводящую результат в стандартный поток вывода. Чтение и замена вхождений подстроки должны проводиться строка за строкой. Ввод продолжается до появления символа конца файла в стандартном потоке ввода.

Синтаксис командной строки:

**replace.exe** <search-string> <replace-string>

**Внимание, реализация данной функции должна иметь сложность близкую к O(N)**

##### Вариант 4 – HTML Encode – 40 баллов

Разработайте функцию  
**std::string HtmlEncode(std::string const& text)**,  
выполняющую кодирование специальных символов строки text соответствующими сущностями HTML:

* “ (двойная кавычка) заменяется на **&quot;**
* ‘ (апостроф) заменяется на **&apos;**
* < (знак меньше) заменяется на **&lt;**
* > (знак больше) заменяется на **&gt;**
* & (амперсанд) заменяется на **&amp;**

Например, строка Cat <says> "Meow". M&M’s должна быть преобразована в строку  
Cat &lt;says&gt; &quot;Meow&quot;. M&amp;M&apos;s

Разработайте на ее основе программу, выполняющую построчное html-кодирование текста, поступающего со стандартного потока ввода, и выводящую результат в стандартный поток вывода.

Символы &, полученные в процессе кодирования, не должны повторно кодироваться.

**Внимание, реализация данной функции должна иметь сложность близкую к O(N)**

##### Вариант 5 – HTML Decode – 60 баллов

Разработайте функцию  
**std::string HtmlDecode(std::string const& html)**,  
выполняющую декодирование HTML-сущностей строки **html**, перечисленных в варианте 4, обратно в их символьное представление.

Разработайте на ее основе программу, выполняющую построчное декодирование html-сущностей текста, поступающего со стандартного потока ввода, и выводящую результат в стандартный поток вывода.

**Внимание, реализация данной функции должна иметь сложность близкую к O(N)**

#### Задание 3 – работа с контейнером map

Ознакомьтесь с возможностями контейнера [**std::map**](https://en.cppreference.com/w/cpp/container/map)библиотеки STL и выполните задание, соответствующее номеру Вашего варианта.

**Для тестирования разрабатываемых функций должны быть разработаны тесты, проверяющие корректность их работы на некотором разумном наборе входных параметров.**

##### Вариант 1- Подсчет частоты встречаемости слов – 40 баллов

Разработайте программу, выполняющую подсчет частоты встречаемости слов, поступающих со стандартного потока ввода и выводящую слова их частоты их встречаемости в стандартный поток вывода.

Под словом понимается последовательность из одного и более символов, не являющихся пробелами, табуляциями, символами конца строки.

Частота встречаемости слов должна выводиться после того, как входной поток данных закончится (eof).

Для подсчета частоты встречаемости символов используйте отображение **слово**→**частота встречаемости**.

###### Бонус 10 баллов за подсчет слов, записанных в case-insensitive формате

**Дополнительно можно получить до 10 баллов,** если программа будет способна обнаруживать русские и английские слова, записанные в разном регистре символов, т.е. считать слова **HeLLo** и **heLLO** одинаковыми.

##### Вариант 2 – мини-словарь – 90 баллов

Разработайте программу-словарь, осуществляющую перевод **слов и словосочетаний**, поступающих со стандартного потока ввода, **с английского языка на русский** с использованием заданного файла словаря и выводящую результат перевода в стандартный поток вывода.

Если вводимое слово или словосочетание, отсутствует в словаре, программа должна попросить пользователя ввести перевод и запомнить его, в случае, если пользователь ввел непустую строку.

**Для выхода из диалога с программой** пользователь должен ввести строку, **состоящую из трех точек**. Перед выходом программа спрашивает о необходимости сохранить изменения в файле словаря, в том случае, **если в словарь были добавлены фразы** во время текущей сессии работы с программой.

Имя файла словаря передается программе с помощью параметра командной строки. Если файл словаря отсутствует, то программа должна считать его пустым и предложить сохранить словарь по окончании работы, если туда были добавлены фразы.

Пример диалога пользователя с программой:

>cat

кот, кошка

>ball

мяч

>meat

Неизвестное слово “meat”. Введите перевод или пустую строку для отказа.

>мясо

Слово “meat” сохранено в словаре как “мясо”.

>meat

мясо

>The Red Square

Неизвестное слово “The Red Square”. Введите перевод или пустую строку для отказа.

>Красная Площадь

Слово “The Red Square” сохранено в словаре как “Красная Площадь”.

>lkkvksmdv

Неизвестное слово “lkkvksmdv”. Введите перевод или пустую строку для отказа.

>

Слово “lkkvksmdv”проигнорировано.

>...

В словарь были внесены изменения. Введите Y или y для сохранения перед выходом.

>y

Изменения сохранены. До свидания.

###### Бонус 10 баллов за возможность распознавания слов, записанных в разном регистре

**Дополнительно можно получить до 10 баллов,** если программа будет способна осуществлять перевод английских слов, вводимых пользователем в произвольном регистре символов. Например, слова **CaT**, при известном переводе для слова **cat**. Регистр перевода теряться не должен.

###### Бонус 20 баллов за реализацию двунаправленного перевода

**Дополнительно можно получить до 20 баллов,** если программа сможет осуществлять и обратный перевод словосочетаний. При этом необходимо поддержать возможность существования **нескольких вариантов** перевода одного и того же слова.

Например, после добавления слов «кошка»«cat» и «кот»«cat», программа должна иметь возможность перевести слово cat, выдав 2 возможных перевода.

#### Задание 4. Работа с контейнером set

Ознакомьтесь с возможностями контейнера [std::set](https://en.cppreference.com/w/cpp/container/set) и выполните задание, соответствующее номеру Вашего варианта.

**Для тестирования разрабатываемых функций должны быть разработаны тесты, проверяющие корректность их работы на некотором разумном наборе входных параметров.**

##### Вариант 1 – Cross Sets - 30 баллов

Разработайте функцию  
**std::set<int> CrossSet(std::set<int> const& set1, std::set<int> const& set2)**,  
возвращающую результат пересечения[[2]](#footnote-2) двух множеств целых чисел.

С ее использованием разработайте программу, выводящую в стандартный поток вывода элементы двух множеств целых чисел и результат их пересечения.

**Первое множество** – множество чисел от 1 до N, делящихся без остатка на сумму своих цифр.

**Второе множество** – множество целых чисел то 1 до N, сумма цифр которых является четной.

Параметр N передается пользователем в виде аргумента командной строки.

**Подсказка**: используйте алгоритм **set\_intersection**.

##### Вариант 2 – Спортшкола – 40 баллов

В спортивной школе обучается некоторое количество учеников. Имена учеников – уникальны. Каждый ученик посещает одну или более спортивных секций школы.

У тренера каждой секции есть список учеников его секции, заданный в виде текстового файла, в котором в каждой строке записано ФИО ученика.

**Задача**: на основе файлов со списками учеников каждого тренера построить список всех учеников школы.

Программа принимает в качестве параметров командной строки имена файлов со списками учеников каждой секции (количество секций в школе, а, следовательно, параметров командной строки – произвольно) и выводит в output список всех учащихся школы.

##### Вариант 3 – Фильтр нецензурных слов – 60 баллов

Для повышения культуры общения в чате необходимо написать программу-фильтр, удаляющую из сообщений участников чата недопустимые слова.

Список недопустимых слов задается в текстовом файле (слова разделены последовательностью из одного и более пробелов, табуляций или символов конца строки), имя которого передается программе при помощи аргумента командной строки.

Программа из каждой вводимой из стандартного потока ввода строки должна удалять присутствующие в ней недопустимые слова и выводить обработанный результат в стандартный поток ввода.

Под словом понимается последовательность из одного и более символов, разделенных последовательностью из одного и более символов-разделителей (пробелы, табуляции, символы конца строки, знаки препинания, знаки арифметических операций, скобки).

**Указания**: считайте недопустимые слова во множество строк и при обработке текста проверяйте вхождение каждого слова текста в данное множество.

###### Бонус в 10 баллов за распознавание нецензурных слов, записанных в разном регистре

**Дополнительно можно получить 10 баллов**, если фильтрация слов будет производиться **с игнорированием регистра символов**, в котором они записаны. Т.е. если недопустимым словом является слово «дурак», то должны фильтроваться слова «ДуРак», «дурак», «ДУРАК» и подобные.

##### Вариант 4 – Генератор простых чисел – 100 баллов

Разработайте функцию **std::set<int> GeneratePrimeNumbersSet(int upperBound)**, возвращающую множество всех простых чисел, **не превышающих** значения upperBound.

С ее использованием разработайте программу, выводящую в стандартный поток вывода элементы множества простых чисел, не превышающие значения, переданного приложению через обязательный параметр командной строки.

Максимальное значение верхней границы множества принять равным 100 миллионам.

Время построения множества простых чисел в указанном диапазоне на компьютере с 2GHz-процессором не должно превышать 10-12 секунд (программа будет запускаться в Release-конфигурации).

**Примечание**: наивный поиск простых чисел не позволит добиться указанной производительности. Используйте «Решето Эратосфена». Для предварительного просеивания воспользуйтесь vector<bool> (для хранения каждого элемента он использует 1 бит информации, т.к. на хранение признака простоты 100 миллионов чисел потребуется всего 12,5 мегабайт памяти)

Для проверки программы используйте тот факт, что в диапазоне от 1 до 100000000 содержится 5761455 простых чисел.

### Дополнительные задания

#### Задание 5. – регулярные выражения

Ознакомьтесь с возможностями класса [std::regex](http://www.cplusplus.com/reference/regex/) и функций для работы с регулярными выражениями библиотеки STL и выполните задание, соответствующее номеру Вашего варианта.

**Для тестирования разрабатываемых функций должны быть разработаны тесты, проверяющие корректность их работы на некотором разумном наборе входных параметров.**

##### Вариант 1 – парсер URL-ов – 80 баллов

Разработайте функцию  
**bool ParseURL(string const& url, Protocol & protocol, int & port, string & host, string & document)**,  
выполняющую разбор строки url, и извлечение из нее информации об используемом протоколе, номере порта, имени хоста и имени документа.

В случае успеха функция должна возвращать true, в случае ошибки – false.

Protocol – это перечислимый тип, задающий один из известных программе протоколов:

enum class Protocol

{

HTTP,

HTTPS,

FTP

};

Валидным (допустимым) URL-ом программа должна считать строку в следующем формате:

***протокол***://***хост***[:***порт***][/***документ***], где  
протокол – http, https или ftp (**в любом регистре**),  
порт – положительное число от 1 до 65535   
хост – **непустая строка**

(в квадратных скобках указаны опциональные элементы URL-а)

Если порт не указан, то он должен считаться равным номеру порта по умолчанию для данного протокола (для HTTP – это 80, для HTTPS – 443, для FTP – 21).

Разработайте на ее основе программу, распознающую допустимые URL-строки в каждой вводимой пользователем строке ввода и выводящую в стандартный поток вывода информацию **о каждом URL-е** в следующем формате:

**<*Исходный URL*>**

**HOST: <*хост*>**

**PORT: <*порт*>**

**DOC: <*документ*>**

Например, для URL-а

[http](http://www.mysite.com/docs/document1.html?page=30&lang=en#title)://www.mysite.com/docs/document1.html?page=30&lang=en#title

должно быть выведено:

http://www.mysite.com/docs/document1.html?page=30&lang=en#title

HOST: www.mysite.com

PORT: 80

DOC: docs/document1.html?page=30&lang=en#title1

#### ***Задание 6***

##### **Вариант 1 – Expand Template - 100 баллов**

Разработайте функцию   
**std::string ExpandTemplate(std::string const& tpl, std::map<std::string, std::string> const& params)**, возвращающую результат подстановки параметров, заданных при помощи аргумента params (шаблонный параметр=>значение) в строку tpl. Пустые строки, выступающие в роли ключа в params, должны игнорироваться. Любой шаблонный параметр может встречаться в строке tpl произвольное количество раз. При наличии нескольких возможных вариантов подстановки должен выбираться параметр, имеющий наибольшую длину. Та часть строки tpl, в которую уже была выполнена подстановка, не должна повторно модифицироваться.

**Для тестирования функции ExpandTemplate должны быть разработаны тесты, проверяющие корректность их работы на некотором разумном наборе входных параметров.**

Пример использования данной функции и ожидаемые результаты ее работы[[3]](#footnote-3):

using namespace std;

int main()

{

{

string const tpl = "Hello, %USER\_NAME%. Today is {WEEK\_DAY}.";

map<string, string> params;

params["%USER\_NAME%"] = "Ivan Petrov";

params["{WEEK\_DAY}"] = "Friday";

assert(ExpandTemplate(tpl, params) == "Hello, Ivan Petrov. Today is Friday.");

}

{

string const tpl = "Hello, %USER\_NAME%. Today is {WEEK\_DAY}.";

map<string, string> params;

params["%USER\_NAME%"] = "Super %USER\_NAME% {WEEK\_DAY}";

params["{WEEK\_DAY}"] = "Friday. {WEEK\_DAY}";

assert(ExpandTemplate(tpl, params) ==

"Hello, Super %USER\_NAME% {WEEK\_DAY}. Today is Friday. {WEEK\_DAY}.");

}

{

string const tpl = "-AABBCCCCCABC+";

map<string, string> params;

params["A"] = "[a]";

params["AA"] = "[aa]";

params["B"] = "[b]";

params["BB"] = "[bb]";

params["C"] = "[c]";

params["CC"] = "[cc]";

assert(ExpandTemplate(tpl, params) ==

"-[aa][bb][cc][cc][c][a][b][c]+");

}

}

С использованием данной функции **разработайте приложение expand\_template**, позволяющее заменить вхождения определенных строк в одном файле на определяемые пользователем и записать результат в выходной файл. Синтаксис командной строки приложения:

**expand\_template.exe <input-file> <output-file> [<param> <value> [<param> <value> …]]**

Пример использования:

**expand\_template.exe input.txt output.txt "тепло" "холодно" "соленый" "сладкий" "день" "ночь"**  
записывает в выходной файл результат замены всех вхождений подстроки «тепло» на «холодно», «соленый» на «сладкий», «день» на «ночь», найденных во входном файле.

###### Бонус в 150 баллов за эффективное решение данной задачи с использованием алгоритма «Ахо-Корасик»

Рекомендации по выполнению: для эффективной реализации данного задачи ознакомьтесь с алгоритмом «Ахо-Корасик» ([Википедия](https://ru.wikipedia.org/wiki/Алгоритм_Ахо_—_Корасик), [Хабрахабр](https://habrahabr.ru/post/198682/))

#### ***Задание 7***

##### Вариант 1 — разбор выражения — 100 баллов

Программе в каждой строке стандартного ввода поступает арифметическое выражение в небинарной скобочной [польской нотации](https://ru.wikipedia.org/wiki/Польская_запись).

Программа должна вывести результат каждого из этих выражений выражения в виде числа.

Выражение в небинарной скобочной польской нотации имеет вид:

* (операция аргумент1 … аргументN)
* Операция имеет не менее одного аргумента
* Операция: либо сложение (символ +), либо умножение (символ \*).
* Каждый аргумент – либо целое (положительное либо отрицательное) число, либо выражение в скобочной польской нотации.

Результат выражения – операция, применяемая ко всем аргументам.

Примеры ввода-вывода:

|  |  |  |
| --- | --- | --- |
| Ввод | Вывод | Примечание |
| (+ 7) | 7 | Если всего один аргумент, то результат выражения – этот самый аргумент |
| (\* 8) | 8 | Аналогично и тут |
| (+ 2 3) | 5 | 2 + 3 = 5 |
| (+ 2 3 4) | 9 | 2 + 3 + 4 = 9 |
| (\* 2 4) | 8 | 2\* 4 = 8 |
| (\* 2 3 4) | 24 | 2 \* 3 \* 4 = 24 |
| (+ (\* 2 3) (\* 3 4)) | 18 | (2 \* 3) + (3 \* 4) = 18 |
| (\* (+ 1 2) (+ 3 1)) | 12 | (1 + 2) \* (3 + 1) = 12 |
| (+ 5 (\* 2 3 2) (+ 5 (+ 2 5) (\* 2 2) )) | 33 | 5 + (2 \* 3 \* 2) + (5 + (2 + 5) + (2 \* 2)) = 5 + 12 + 16 = 33 |
| (+ -2 3) | 1 | -2 + 3 = 1 |

Гарантируется, что программе поступают на вход синтаксически верные выражения.

Для работы над ней вам может понадобиться:

1) Уметь считывать символы из стандартного ввода:

  char ch;

  std::cin >> ch;

Прочитает в переменную из cin первый непробельный символ.

2) Возвращать последний считанный символ обратно в поток cin:

  std::cin.*unget*();

Если после чтения символа вы решили вернуть его обратно в поток cin, то вызовите у потока метод unget.

Например

char ch1, ch2;

std::cin >> ch1;

std::cin.unget(); // вернули считанный символ в поток cin

std::cin >> ch2;

// В ch2 будет лежать тот же символ, что и в ch

Справочная информация

Нотация называется **скобочной**, так как выражение обязательно заключено в круглые скобки.

Нотация называется **небинарной**, так как количество аргументов может быть отлично от двух.

Нотация называется **польской**, так как её изобрёл польский математик Ян Лукасевич.

**В программе должна быть выделена функция, вычисляющая значение выражения, записанного в обратной скобочной нотации. Для этой функции должны быть разработаны модульные тесты.**

###### Бонус в 50 баллов за обработку ошибок

Если в строке содержится невалидное выражение, программа должна вывести строку **ERROR** вместо результата вычисления выражения. Вы можете использовать для обработки ошибок коды возврата из функции, либо механизм исключений.

###### Бонус в 80 баллов за нерекурсивное решение задачи

Бонус начисляется, если для разбора выражения не используется рекурсия.

1. Функция обработки вектора значений должна изменять содержимое переданного массива. [↑](#footnote-ref-1)
2. Пересечением двух множеств является множество, содержащее элементы, присутствующие одновременно и в первом и во втором множестве [↑](#footnote-ref-2)
3. Приведенный пример проверяет лишь некоторые наиболее сложные из возможных ситуаций. Вместо обычных макросов assert следует использовать один из популярных C++-фреймворков для написания тестов [↑](#footnote-ref-3)