Основы программирования на C++

Урок 6. Введение в циклы: while, do while, break

Циклы while, do while

**Задача 1. Начальник.**  
  
Напишите программу для робота-начальника. Он спрашивает у пользователя, выполнил ли он задания, которые выдавал вчера, и продолжает это делать до тех пор, пока тот не ответит ему “Да, конечно, сделал”.  
  
Подсказка 1: строки можно сравнивать вот так: (answer == “Да, конечно, сделал”), где answer — переменная с типом std::string  
Подсказка 2: чтобы считать строку с пробелами, воспользуйтесь функцией std::getline(std::cin, line);"  
  
**Задача 2. Купи слона.**  
  
Напишите программу-дразнилку “Купи слона”. Она спрашивает у пользователя, как его зовут, затем отвечает “%username%, купи слона!”. Дальше, что бы он ни говорил, она передразнивает: Все говорят “...”, а ты купи слона!  
Так повторяется бесконечно.  
  
**Пример**  
  
- Хорошо, давай куплю.  
- Все говорят “Хорошо, давай куплю.”, а ты купи слона!  
  
  
**Задача 3. Таблица степеней.**  
  
Напишите программу, которая выводила бы N первых степеней числа К. Посмотрите, что случится, если N или K будут достаточно большими.  
  
**Задача 4. Напоминалка.**  
  
Василий — очень ответственный менеджер, но очень забывчивый. Иногда он пьет воду несколько раз подряд, потому что забыл, что только что пил. Напишите для него программу-напоминалку. В самом начале программа спрашивает, сколько раз ему напомнить, а затем нужное количество раз выводит “Вы хотели не забыть о чем-то”.  
  
Усложнение: добавьте возможность ввести фразу-напоминалку, чтобы можно было напоминать о чем-то конкретном.  
  
**Задача 5. Банкомат 3.0.**  
  
В банке ПлюсПлюсБанк тестируют новую систему пин-кодов на банкоматы. Теперь вместо фиксированного пин-кода пользователь должен ввести 4 числа. Если их сумма оказывается равна 42, пользователь может продолжать работу. Напишите программу, которая спрашивает у пользователя 4 числа и отвечает “Ввод корректный”, если их сумма равна 42, и “Ввод некорректный, попробуйте еще раз” в противном случае. Так повторяется бесконечно.  
  
**Задача 6. Сумма чисел.**  
  
У бухгалтера сломались счеты. Напишите программу, которая считала бы сумму вводимых чисел. В начале у бухгалтера спрашивают, сколько чисел тот хочет сложить, затем нужное количество раз просят ввести число и в конце выводят получившуюся сумму.

Применение цикла while. Инструкция break

**Задача 1. Банкомат 4.0.**  
  
В банке ПлюсПлюсБанк решили улучшить новую систему пин-кодов на банкоматы. Пользователям оказалось неудобно вводить 4 отдельных числа. Перепишите программу так, чтобы пользователь вводил число любой длины, а дальше все то же самое: если сумма цифр числа оказывается равна 42, пользователь может продолжать работу.  
  
Напишите программу, которая спрашивает у пользователя число и отвечает “Ввод корректный”, если сумма его цифр равна 42, и “Ввод некорректный” в противном случае. При этом у пользователя есть только одна попытка ввести корректное число.  
  
**Задача 2. Кадастровые проблемы.**  
  
В столице программистов на квадратные участки дается скидка. А так как работать с целыми числами программисты начинают раньше, то и участки, площадь которых является точным квадратом (например, 25 или 49), являются самыми востребованными. Напишите программу, которая проверяет, является ли введенная площадь участка точным квадратом. Самый большой участок в городе имеет площадь в 1 000 000 м2.  
  
**Задача 3. Кожаный мешок.**  
  
На этот раз нужно написать фрагмент кода для прошивки робота. Как мы знаем, роботы понимают только нули и единицы, других цифр для них не существует. Напишите программу, которая проверяет, состоит ли число только из нулей и единиц.  
  
**Задача 4. Военное время.**  
  
Во времена Второй Мировой в штабах использовались стеклянные доски. С одной их стороны специальные люди писали сводки с фронта, а с другой генералы их читали. Как вы понимаете, для генералов текст выглядел зеркально отраженным, поэтому специальных людей учили писать задом наперед. Напишите программу, упрощающую им жизнь: она должна выводить введенное число задом наперед.

Практическая работа

**В практической работе есть:**  
Обязательные задачи помогут проверить, как вы усвоили материал модуля. Их нужно сдать на проверку куратору.  
Дополнительные задачи подойдут тем, кто хочет ещё немного потренироваться, решая задачки по программированию. Вы также можете отправлять их на проверку и получать обратную связь куратора, хотя это необязательно.  
  
**Цели практической работы**  
• Научиться использовать циклы while, do while и оператор break.  
• Научиться основам работы с циклами.  
  
**Что входит в практическую работу**  
Обязательные задачи:  
1. Кукушка.  
2. Слишком большие числа.  
3. Коллекторы.  
4. Счастливый билетик.  
5. Числа Фибоначчи.  
6. Вклады.  
  
Дополнительные задачи:  
7. Сокращение дроби.  
8. Угадай число\*.

Задание 1. Кукушка

Что нужно сделать  
Напишите программу, которая имитировала бы часы с кукушкой. В начале работы она спрашивает, который час, а затем нужное количество раз пишет «Ку-ку!». При выполнении задания необходимо обеспечить контроль ввода (например, количество часов не может быть отрицательным).  
**Пример выполнения**  
  
Введите, который час: 2  
Ку-ку!  
Ку-ку!  
  
**Рекомендации по выполнению**  
Обратите внимание, если вы захотите использовать 24-часовой формат, в котором часы имеют значения 0–23, то кукушка не сможет вас оповестить о полночи (нулевой час).  
  
**Что оценивается**  
• Есть проверка входных значений.  
• Количество выведенных на консоль фраз равно количеству часов.

Задание 2. Слишком большие числа

**Что нужно сделать**  
У неудачливого бухгалтера всё опять идёт наперекосяк: ему приносят такие большие счета, что числа не помещаются на бумаге. Напишите программу, которая считала бы для него, сколько цифр (знаков) во вводимом числе.  
  
**Пример выполнения**  
  
Введите число: 10  
Цифр в числе: 2  
  
**Рекомендации по выполнению**  
• Тип int вмещает в себя не больше десяти цифр (предположим, что у бухгалтера очень маленькие листы бумаги).  
• Проверьте, как программа будет себя вести на отрицательных числах и нуле.  
  
**Что оценивается**  
• В числе ноль программа определяет одну цифру.  
• Количество цифр определяется правильно.

Задание 3. Коллекторы

**Что нужно сделать**  
Напишите робота для коллекторов. В самом начале он спрашивает имя должника и сумму долга, а затем начинает требовать у него погашения долга до тех пор, пока тот не будет погашен полностью. Сумма долга должна уменьшаться, если пользователь ввёл сумму, меньшую чем сумма долга. Если пользователь внёс большую сумму, чем требуется для погашения, выведите сообщение о том, какой у него остаток на счету после такой операции. Обеспечьте контроль ввода.  
  
**Рекомендации по выполнению**  
При полной выплате долга необходимо учитывать остаток на счету, если он имеется.  
  
**Что оценивается**  
• Правильно считается остаток после полного погашения долга.  
• Программа прекращает выполнение после того, как долг полностью погашен.  
• При задании отрицательного долга программа выводит предупреждение и прекращает выполнение.

Задание 4. Счастливый билетик

**Что нужно сделать**  
В старину, когда даже в столице билеты в общественном транспорте выдавали контролёры, существовало поверье: если на билете сумма первых трёх цифр в номере билета равна сумме трёх последних, то это к удаче. Напишите программу, которая получала бы на вход шестизначный номер билета и выводила, счастливый это билет или нет. К примеру, билеты 666 666 и 252 135 — счастливые, а 123 456 — нет.  
  
**Пример выполнения**  
  
Введите номер билета: 123060  
Билет счастливый!  
Введите номер билета: 123040  
Повезёт в следующий раз!  
  
**Рекомендации по выполнению**  
• Программу необходимо сделать так, чтобы можно было увеличить количество цифр в билете без особых изменений в коде.  
• Для получения младшего разряда числа (самой правой цифры) используйте выражение: one\_digit = digits % 10.  
• Используйте циклы для подсчёта суммы цифр в номере билета.  
  
**Что оценивается**  
• Программа корректно определяет, является ли билет счастливым.  
• Для подсчёта суммы цифр в билете используются циклы.

Задание 5. Числа Фибоначчи

**Что нужно сделать**  
Числами Фибоначчи называется следующая последовательность чисел: 1, 1, 2, 3, 5, 8, 13 и так далее (первое число равно 1, второе число равно 1, а каждое следующее равно сумме двух предыдущих). Напишите программу, которая по данному номеру N выводит N-е число этой последовательности (если оно не превосходит 2 147 483 647, конечно же). Обеспечьте контроль ввода.  
  
**Пример выполнения**  
  
Введите номер последовательности: 1  
Число в последовательности: 1  
Введите номер последовательности: 6  
Число в последовательности: 8  
  
**Рекомендации по выполнению**  
• Заметьте, что последовательность Фибоначчи всегда одинаковая. Пользователь вводит только порядковый номер числа из последовательности.  
• Обратите внимание, что условие типа if(value > 2147483647) не имеет смысла.  
  
**Что оценивается**  
• Результатом запроса является число из последовательности Фибоначчи.  
• Число 1 определяется как первое и второе число последовательности.

Задание 6. Вклады

**Что нужно сделать**  
Вклад в банке составляет X рублей. Ежегодно он увеличивается на P процентов, после чего дробная часть копеек отбрасывается. Определите, через сколько лет вклад составит не менее Y рублей. Напишите программу, которая по данным числам X, Y, P определяет, сколько лет пройдёт, прежде чем сумма достигнет значения Y.

**Пример выполнения**  
  
Введите размер вклада: 100  
Введите процентную ставку: 10  
Введите желаемую сумму: 200  
Придётся подождать: 8 лет  
  
**Пример вычисления:**  
  
1 год: 100 + 10% \* 100 = 110  
2 год: 110 + 10% \* 110 = 121  
3 год: 121 + 10% \* 121 = 133  
  
**Рекомендации по выполнению**  
• Расчёты проводятся в целых числах.  
• У вклада есть капитализация процентов (проценты начисляются на весь остаток на счёте).  
  
**Что оценивается**  
Программа аварийно не завершается и не зависает при подсчёте вклада.

Задание 7. Сокращение дроби\* (дополнительное задание)

**Что нужно сделать**  
Пользователь вводит числа M и N. Напишите программу, которая преобразует дробь M/N к несократимому виду и выдаёт получившийся результат. Обеспечьте контроль ввода.  
  
**Пример выполнения**  
  
Введите числитель: 4  
Введите знаменатель: 6  
Результат: 2 / 3  
Введите числитель: 25  
Введите знаменатель: 40  
Результат: 5 / 8  
Введите числитель: −6  
Введите знаменатель: 12  
Результат: −1 / 2  
(При вводе отрицательных чисел в выводе должен быть только один ведущий минус. Неправильный вывод программы: 1/−2, −1/−2.)  
Введите числитель: 896  
Введите знаменатель: 3584  
Результат: 1 / 4  
  
**Рекомендации по выполнению**  
Не забывайте о правилах деления чисел. Программа должна работать корректно и не должна «падать».  
  
**Что оценивается**  
• Алгоритм оптимальный, не бесконечный и не очень большой. Программа не зависает.  
• Правильно определены критерии начала и конца цикла.

Задание 8. Угадай число\* (дополнительное задание)

**Что нужно сделать**  
Напишите программу, которая угадывает число, задуманное пользователем. Число загадывается в диапазоне от 0 до 63. Программа задаёт вопросы вида «Ваше число больше такого-то?» и на основе ответов пользователя («да» или «нет») угадывает число.  
  
**Рекомендации по выполнению**  
• В этом задании можно использовать любой алгоритм поиска, но полный перебор всех 64 значений — самый нежелательный из алгоритмов.  
• Есть алгоритм, который гарантированно даст ответ за семь вопросов.

### Урок 7. Циклы for и алгоритмы на циклах

## Циклы for

[repl.it/@jebrax/SheepFor](https://repl.it/@jebrax/SheepFor)  
[repl.it/@jebrax/TolyaWhile](https://repl.it/@jebrax/TolyaWhile)  
[repl.it/@jebrax/TolyaFor](https://repl.it/@jebrax/TolyaFor)  
  
**Задача 1. Кукушка (цикл for).**  
Перепишите программу из прошлого модуля, используя цикл for. Напишите программу, которая имитировала бы часы с кукушкой. В начале работы она спрашивает, который час, а затем нужное количество раз пишет “Ку-ку!”. Обеспечьте контроль ввода (например, количество часов не может быть отрицательным).  
  
**Задача 2. Сумма чисел (цикл for).**  
Перепишите программу из прошлого модуля, используя цикл for. У бухгалтера сломались счеты. Напишите программу, которая считала бы сумму вводимых чисел. В начале у бухгалтера спрашивают, сколько чисел тот хочет сложить, затем нужное количество раз просят ввести число и в конце выводят получившуюся сумму.  
  
**Задача 3. Таблица умножения.**  
Первоклассник Вася родился в век высоких технологий и не умеет умножать в уме. Он боится контрольной работы на знание таблицы умножения, поэтому он решил написать на С++ программу-шпаргалку. Вася хочет ввести число, а программа должна выдать таблицу умножения для этого числа. Помогите Васе написать программу.  
  
**Задача 4. Сумма нечётных.**  
Напишите программу, похожую на разобранный нами пример с суммой чисел, но только теперь нужно вычислить сумму всех нечётных чисел, лежащих в диапазоне от единицы до указанного пользователем числа включительно.  
  
**Задача 5. Начальник (цикл for).**  
Перепишите программу из прошлого модуля, используя цикл for. Напишите программу для робота-начальника. Он спрашивает у пользователя, выполнил ли он задание, которое тот выдавал вчера, и продолжает это делать до тех пор, пока пользователь не ответит ему “Да, конечно, сделал”. Для большей реалистичности в конце пусть робот-начальник пожалуется: “Ну почему тебя нужно спрашивать N раз?”, где N — это число попыток, которые были потрачены пользователем, прежде чем он ввёл правильный ответ.  
  
Напоминаем: чтобы считать строку с пробелами, воспользуйтесь функцией std::getline(std::cin, line);

## Варианты цикла for

[repl.it/@jebrax/Folding](https://repl.it/@jebrax/Folding)  
[repl.it/@jebrax/Gollum](https://repl.it/@jebrax/Gollum)  
  
**Задача 1. Пароль (цикл for).**  
Перепишите программу из прошлого модуля, только теперь вместо цикла while используйте цикл for. Программа просит пользователя ввести пароль. Если пароль неверный, программа просит сделать это снова, и так до тех пор, пока пользователь не введёт правильный пароль.  
  
**Задача 2. Кадастровые проблемы (цикл for).**  
Перепишите программу из прошлого модуля, только теперь вместо цикла while используйте цикл for. В столице программистов на квадратные участки дается скидка. А так как работать с целыми числами программисты начинают раньше, то и участки, площадь которых является точным квадратом (например, 25 или 49), являются самыми востребованными. Напишите программу, которая проверяет, является ли введенная площадь участка точным квадратом. Самый большой участок в городе имеет площадь в 1 000 000 м2.  
  
**Задача 3. Крепкий орешек.**  
Мы разрабатываем пошаговую игру по мотивам боевика. Игрок — главный герой и должен обезвредить бомбу, которая взорвётся через 10 итераций цикла. Программа спрашивает пользователя хочет ли он перерезать провод сейчас. Если ответ “нет”, то счетчик бомбы уменьшается. Если он достиг нуля, то программа выдаёт сообщение “Бомба взорвалась”, а если не достиг, то программа вновь переспрашивает, не хочет ли игрок обезвредить бомбу, и сообщает, сколько времени осталось до взрыва. Если ответ “да”, то программа выводит на экран сообщение о том, что бомба обезврежена и сколько шагов оставалось до взрыва. Используйте цикл for.

## Практическая работа

**В практической работе есть:**  
Обязательные задачи помогут проверить, как вы усвоили материал модуля. Их нужно сдать на проверку куратору.  
Дополнительные задачи подойдут тем, кто хочет ещё немного потренироваться, решая задачки по программированию. Вы также можете отправлять их на проверку и получать обратную связь куратора, хотя это необязательно.  
  
**Цели практической работы**  
• Закрепить навыки работы с циклами.  
• Научиться пользоваться циклом for.  
• Использовать на практике рассмотренные алгоритмы на циклах.  
  
**Что входит в практическую работу**  
Обязательные задачи:  
1. Запасы продовольствия.  
2. Кофемашина.  
3. Красный Марс.  
4. Рамка.  
5. Координатные оси.  
6. Важные объявления.  
Дополнительные задачи:  
7. Биолаборатория\*.  
8. Ёлочка\*.

#### Задание 1. Запасы продовольствия

**Что нужно сделать**  
Ваш космический корабль потерпел крушение на пустынной планете. Еда здесь не растёт, но вы спасли из-под обломков 100-килограммовый мешок гречки. Из прошлого опыта вы знаете, что если будете экономно питаться, то у вас будет уходить по четыре килограмма гречки в месяц. Чтобы прикинуть гречневый бюджет, вы решили написать программу, которая выведет информацию о том, сколько килограммов гречки у вас должно быть в запасе через месяц, два и так далее, пока она не закончится.  
**Пример выполнения**  
  
Всего гречки было на начало подсчёта: 100 кг  
После 1 месяца у вас в запасе останется 96 кг гречки  
После 2 месяца у вас в запасе останется 92 кг гречки  
…  
После \_ месяца гречка закончится  
  
**Рекомендации по выполнению**  
Для решения данной задачи рекомендуется использовать цикл for.  
  
**Что оценивается**  
• Программа выводит информацию об остатке гречки за каждый месяц с помощью циклов.  
• Текстовое сообщение выводится в цикле и за каждый месяц.  
• Программа заканчивает выполнение только после того, как в мешке не осталось гречки.

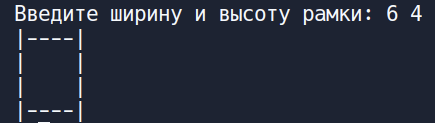
#### Задание 2. Кофемашина

**Что нужно сделать**  
Инженеры компании «Спейс Инжиниринг» не могут продуктивно работать без кофе. Недавно в их кофемашину случайно попал космический луч и повредил программу-прошивку. Вас умоляют помочь. Кофемашина умеет готовить два напитка: американо и латте. Для американо требуется 300 мл воды, а для латте 30 мл воды и 270 мл молока.  
  
Напишите программу, которая спрашивает у пользователя (это действие программа делает один раз в начале работы), сколько всего миллилитров молока и воды залито в кофемашину, после чего начинает обслуживание, спрашивая, какой напиток хочет очередной посетитель. Пользователь выбирает один из двух напитков, программа отвечает одним из трёх вариантов: «Ваш напиток готов», «Не хватает воды» или «Не хватает молока», после чего переходит к обслуживанию следующего посетителя. Если молока и воды не хватает ни на один вид напитка, программа выдаёт отчёт и завершается. В отчёте должно быть написано, что ингредиенты подошли к концу, должен быть указан остаток воды и молока в машине, а также должно быть указано, сколько всего было приготовлено чашек американо и латте за эту смену. Выберите наиболее удобный для написания программы цикл. Обеспечьте контроль ввода.  
  
**Пример выполнения**  
  
Введите количество воды в мл: 30  
Введите количество молока в мл: 270  
Выберите напиток (1 — американо, 2 — латте): 2  
Ваш напиток готов.  
\*\*\*Отчёт\*\*\*  
Ингредиентов осталось:  
Вода: 0 мл  
Молоко: 0 мл  
Кружек американо приготовлено: 0  
Кружек латте приготовлено: 1  
  
**Рекомендации по выполнению**  
• Перед тем как останавливать работу кофемашины, нужно убедиться, что ни один из напитков нельзя приготовить.  
• Количество напитков разных видов следует считать отдельно.  
• Для решения программы рекомендуется использовать цикл while.  
  
**Что оценивается**  
• Правильно определяется количество оставшихся ингредиентов.  
• Количество ингредиентов не становится отрицательным.  
• Программа не выводит отчёт, если количества ингредиентов хватает хотя бы на один из напитков.  
• Если ингредиентов не хватает для приготовления любого из напитков, программа должна вывести отчёт.

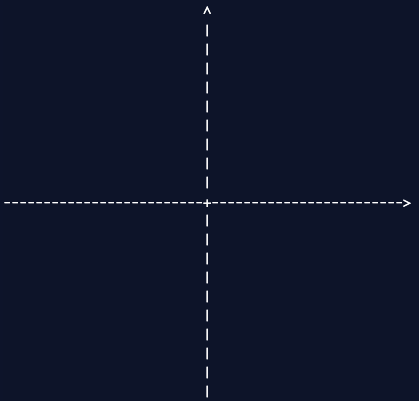
#### Задание 3. Красный Марс

**Что нужно сделать**  
В «Спейс Инжиниринг» были впечатлены тем, как вы справились с кофемашиной и решили отдать вам на аутсорс разработку программы для нового робота-разведчика поверхности Марса. Это первый прототип, поэтому он тестируется в прямоугольном помещении размером 15 на 20 метров. Марсоход высаживается в центре комнаты, после чего управление им передаётся оператору — пользователю вашей программы. Программа спрашивает, в какую сторону оператор хочет направить робота: север (клавиша W), юг (клавиша S), запад (клавиша A) или восток (клавиша D). Оператор делает выбор, марсоход перемещается на 1 метр в эту сторону и программа сообщает новую позицию марсохода. Если марсоход упёрся в стену, то он не должен пытаться перемещаться в сторону стены, в этом случае его позиция не меняется.  
  
**Пример выполнения**  
  
[Программа]: Марсоход находится на позиции 6, 19, введите команду:  
[Оператор]: A  
[Программа]: Марсоход находится на позиции 5, 19, введите команду:  
[Оператор]: W  
[Программа]: Марсоход находится на позиции 5, 20, введите команду:  
[Оператор]: W  
[Программа]: Марсоход находится на позиции 5, 20, введите команду:  
[Оператор]: S  
[Программа]: Марсоход находится на позиции 5, 19, введите команду:  
[Оператор]: S  
[Программа]: Марсоход находится на позиции 5, 18, введите команду:  
…  
  
  
**Рекомендации по выполнению**  
• Необходимо обеспечить контроль ввода пользователя и сообщать об ошибке в случае неправильного ввода.  
• Помните об ограничениях комнаты при перемещении робота.  
  
**Что оценивается**  
• Использованы конструкции else и else if.  
• Координаты марсохода не выходят за границы помещения.  
• Программа не завершает работу если робот упёрся в стенку.

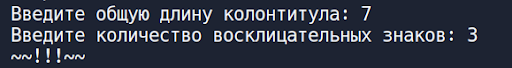
#### Задание 4. Рамка

**Что нужно сделать**  
Напишите программу, которая рисует с помощью символьной графики прямоугольную рамку. Для вертикальных линий используйте символ вертикального штриха «|», а для горизонтальных — дефис «-». Пусть пользователь вводит ширину и высоту рамки.  
  
**Пример выполнения**  
6 — ширина, 4 — высота:  
  
**Рекомендации по выполнению**  
Ширину и высоту удобно вводить с помощью одной операции std::cin:  
  
int width, height;  
std::cin >> width >> height;  
  
**Что оценивается**  
Рамка рисуется в соответствии с заданными размерами и с помощью условных конструкций и циклов.

#### Задание 5. Координатные оси

**Что нужно сделать**  
Модифицируйте пример с координатными осями так, чтобы в точке их пересечения рисовался знак «+», на верхнем конце вертикальной оси была стрелка вверх «^», а на правом конце горизонтальной оси — стрелка вправо «>». Это сделает рисунок более красивым и точным.  
  
**Пример выполнения**  
  
**Рекомендации по выполнению**  
Заметьте, что в центре координатных осей есть пересечение.  
**Что оценивается**  
Координатные оси рисуются с помощью условных конструкций и циклов.

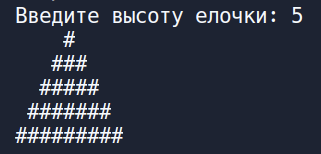
#### Задание 6. Важные объявления

**Что нужно сделать**  
Напишите программу для печати важных объявлений. Сверху объявления должен располагаться вот такой колонтитул:  
  
Восклицательные знаки всегда располагаются по центру строки, причём в зависимости от важности объявления количество восклицательных знаков может быть разным. Напишите программу, которая спрашивает у пользователя сначала общую длину колонтитула в символах, потом желаемое количество восклицательных знаков, после чего выводит на экран готовую строку. Обеспечьте контроль ввода.  
**Пример выполнения**  
  
**Рекомендации по выполнению**  
Если невозможно разместить восклицательные знаки ровно по центру, допускается сдвиг на один символ.  
  
**Что оценивается**  
• Символов всегда ровно столько, сколько требует пользователь.  
• Вывод восклицательных знаков находится посередине, когда это возможно, и со сдвигом на один символ в остальных случаях.

#### Задание 7. Биолаборатория\* (дополнительное задание)

**Что нужно сделать**  
В борьбе со всемирной эпидемией коринебактерий исследователи компании «Терран Биотех» открыли новый антибиотик. Напишите программу, помогающую протестировать его.  
В чашку Петри исследователи кладут N бактерий и добавляют X капель антибиотика. Известно, что число коринебактерий в чашке Петри увеличивается в два раза каждый час, а каждая капля антибиотика в первый час убивает 10 бактерий, во второй час — 9 бактерий, в следующий — 8 и так далее, пока антибиотик не перестанет действовать. Заметьте, что сначала число коринебактерий увеличивается, а затем действует антибиотик. Пользователь вашей программы вводит N и X, а программа печатает на экране, сколько бактерий останется в чашке Петри в конце каждого часа, до тех пор, пока не закончатся бактерии или антибиотик не перестанет действовать. Судьба человечества — в ваших руках. Обеспечьте контроль ввода.  
  
**Пример выполнения**  
  
Введите количество бактерий: 12  
Введите количество антибиотика: 1  
После 1 часа бактерий осталось 14  
После 2 часа бактерий осталось 19  
После 3 часа бактерий осталось 30  
После 4 часа бактерий осталось 53  
…  
  
**Что оценивается**  
• Цикл не бесконечный.  
• Количество бактерий и антибиотика не становится отрицательным.  
• После того как количество антибиотики или бактерий становиться равным нулю выполнение программы должно быть завершено.

#### Задание 8. Ёлочка\* (дополнительное задание)

**Что нужно сделать**  
Напишите программу, которая выводит на экран равнобедренный треугольник, заполненный символами решётки «#». Пусть высота треугольника вводится пользователем. Обеспечьте контроль ввода.  
  
**Пример выполнения**  
  
**Рекомендации по выполнению**  
• Решение можно организовать с помощью двух циклов, один для вывода пробельных символов, второй для вывода символа решётки.

### Урок 8. Числа с плавающей запятой. Математические вычисления

## Тип данных float

[Дорожные камеры](https://repl.it/@jebrax/GIBDD)  
  
**Цель задания**  
Научиться работать с вещественными числами, используя тип данных float.  
  
**Задача 1. Орехи**  
Араик Вачаганович торгует на рынке орехами и сухофруктами. Товары продаются на развес, у каждого есть цена за 100 грамм. Араик пользуется механическими весами с гирями, поэтому каждый раз рассчитывает итоговую стоимость продажи на бумаге или в уме. Это неудобно: например, если покупатель хочет 265 грамм грецких орехов, а 100 грамм стоят 600 рублей, приходится умножать и делить в столбик. Напишите простую программу в помощь Араику: он вводит цену товара за 100 грамм и итоговый вес покупки, а программа выдаёт стоимость покупки указанного веса.  
  
**Задача 2. ЗОЖ**  
Лена решила следить за питанием и вести учёт потребляемых белков и углеводов. На этикетках продуктов указано содержание белков и углеводов на 100 грамм продукта, но обычно упаковка весит меньше или больше 100 грамм. Напишите программу, которая принимает на вход информацию с упаковки: количество белков и углеводов на 100 грамм и массу нетто продукта. На выходе программа сообщает Лене, сколько белков и углеводов содержится в упаковке с продуктом этого веса.  
  
**Задача 3. Постоянный покупатель**  
Иногда к Араику Вачагановичу приходят постоянные покупатели, для которых он делает скидку. Для каждого такого клиента скидка своя. Модифицируйте программу из задачи «Орехи»: Араик будет вводить цену товара за 100 грамм, вес покупки и размер скидки в процентах, а программа должна выводить итоговую стоимость.

## Приведение типов между int и float

[Аттракцион](https://repl.it/@jebrax/PunchMeter)  
[Дорожные камеры](https://repl.it/@jebrax/GIBDD)  
  
**Цель задания**  
Освоить приёмы работы с целыми и дробными числами и механизмы приведения типов.  
  
**Задача 1. Ку!**  
Общеизвестно, что на планете Плюк можно купить гравицапу за пол-КЦ или, что то же самое, 2 200 чатлов. Причем чатлы неделимы и всегда являются целым числом. Напишите простую программу-конвертер валют. В программу вводится количество чатлов, а она сообщает, сколько это КЦ и сколько гравицап можно купить на эту сумму. Обеспечьте контроль ввода.  
  
**Задача 2. Компьютерное зрение**  
Вы участвуете в разработке искусственного интеллекта, который играет в шахматы на доске размером 0,8 х 0,8 метра. Робот смотрит камерой на доску и видит расположение фигур в вещественных числах с высокой точностью. Напишите программу: после ввода вещественных координат шахматной фигуры программа должна определить, в какой клетке доски находится эта фигура. Каждая клетка доски имеет размер 10х10 сантиметров и целочисленные координаты, состоящие из двух чисел. Например, левая верхняя клетка имеет целые координаты (0, 0), справа от неё клетка (1, 0), а снизу (0, 1). Обеспечьте контроль ввода.  
Пример  
Ввод:  
  
Введите местоположение фигуры:  
0.731;  
0.149.  
  
Вывод:  
Фигура находится в клетке (7, 1).  
  
**Задача 3. Метеостанция**  
Для удобства работы сотрудников международной метеостанции каждый день нужно распечатывать различные таблицы соответствия градусов по шкалам Цельсия и Фаренгейта. Напишите программу, которая принимает на вход три целых числа в градусах Цельсия: нижняя граница температуры, верхняя граница температуры и шаг. Программа выводит на экран таблицу соответствия градусов Цельсия градусам Фаренгейта от нижней до верхней границы с указанным шагом. Обеспечьте контроль ввода. Верхняя граница должна печататься, даже если последний шаг «перепрыгнул» её. Известно, что 0 °С соответствует 32 °F, а каждый градус Цельсия эквивалентен 1,8 градусам Фаренгейта.  
Пример  
Ввод:  
  
Нижняя граница: 0.  
Верхняя граница: 40.  
Шаг: 20.  
  
Вывод:  
  
C F  
0 32  
20 68  
40 104  
  
  
**Задача 4. Аккуратность**  
Робот из задачи «Компьютерное зрение» верно определяет клетку, на которой стоят фигуры. Но соперник-человек может поставить фигуру не по центру клетки, а со смещением. Если во время игры такое смещение станет велико, робот не поймёт, в какой клетке стоит фигура. Чтобы избежать этого, робот должен поправлять фигуры на доске, выставляя их по центру клетки. Модифицируйте программу «Компьютерное зрение», чтобы она выдавала не только номера клетки с фигурой, но и две вещественные поправки: показывала, как передвинуть фигуру по горизонтали и вертикали, чтобы она встала по центру своей клетки. Обеспечьте контроль ввода.  
  
Пример  
Ввод:  
  
Введите местоположение фигуры:  
0.731;  
0.167.  
  
Вывод:  
  
Фигура находится в клетке (7, 1).  
Поправьте положение фигуры на (0.019, -0.017).

## Стандартные математические функции

[GPS-навигатор](https://repl.it/@jebrax/FindDistance)  
[Радар](https://repl.it/@jebrax/Radar)  
[Список математических функций](https://en.cppreference.com/w/cpp/header/cmath)  
  
**Цель задания**  
Научиться использовать математические функции из стандартной библиотеки языка С++ в задачах с вещественными числами.  
  
**Задача 1. Ход конём**  
В рамках разработки шахматного искусственного интеллекта стоит новая задача. По заданным вещественным координатам коня и второй точки программа должна определить, может ли конь ходить в эту точку. Используйте как можно меньше конструкций if и логических операторов. Обеспечьте контроль ввода.  
  
**Пример**  
Ввод:  
  
Введите местоположение коня:  
0.071;  
0.118.  
Введите местоположение точки на доске:  
0.213;  
0.068.  
  
Вывод:  
  
Конь в клетке (0, 1). Точка в клетке (2, 0).  
Да, конь может ходить в эту точку.  
  
**Задача 2. Автопилот**  
Одна из базовых частей программы автопилота пассажирских самолётов — контроль угла тангажа. Угол тангажа — это угол между продольной осью самолёта и горизонтом. Если тангаж положительный, нос самолёта смотрит вверх, если отрицательный, нос смотрит вниз и самолёт пикирует. При больших и малых углах тангажа можно потерять управление.  
У самолёта SKB-1 допустимый угол тангажа лежит в пределах от -0,28 до 0,28 радиан. Вам поручено написать код, который проверяет, лежит ли введённый пользователем угол в этих пределах. Пользователь вводит угол в градусах. Используйте как можно меньше конструкций if. Обеспечьте контроль ввода.  
  
**Примеры**  
Ввод:  
Введите угол тангажа в градусах: -28.  
Вывод:  
Угол небезопасен!  
Ввод:  
Введите угол тангажа в градусах: 724.  
Вывод:  
Угол безопасен.  
  
**Задача 3. Лифт**  
В Объединённых Арабских Эмиратах находится самый высокий небоскрёб Земли — Burj Khalifa. Он оборудован скоростными пассажирскими лифтами, но вот беда: иногда они дают сбой и застревают между этажами. В этом случае известно число H — высота пола остановившейся кабины лифта над уровнем земли. Чтобы команда лифтёров знала, куда ей ехать, шейх поручил вам написать программу, которая по известному H определяет, на каком этаже находится большая часть кабины. Известно, что расстояние между этажами Burj Khalifa равно 3,975 м, а этаж на уровне земли имеет номер 1. На ввод программа принимает вещественное число H, на экран выводит целое число — номер этажа. Не используйте при решении условный оператор, кроме как для обеспечения контроля ввода.  
  
**Задача 4. Убийца Photoshop**  
Вы разрабатываете новый крутой графический редактор. Один из инструментов редактора позволяет делать градиент — плавный переход между двумя точками разной яркости. Яркость точки определяется целым числом от 0 до 255, где 0 — чёрный, 255 — белый, а всё, что между ними — оттенки серого.  
Напишите код для расчёта яркости точки градиента между двумя заданными точками. Пользователь вводит яркость левой границы градиента, потом яркость правой границы и параметр: число от 0 до 1, которое означает интересующую нас точку. Например, если параметр равен 0.5, то искомая точка находится ровно посередине между левой и правой границами, если параметр равен 0, то искомая точка совпадает с левой границей, и, наконец, если параметр равен 1, то искомая точка совпадает с правой границей. Программа должна вывести число — яркость искомой точки. Обеспечьте контроль ввода.  
  
**Пример**  
Ввод:  
  
Введите яркость левой границы градиента: 10.  
Введите правую границу градиента: 255.  
Введите положение точки между границами: 0.7.  
  
Вывод:  
Яркость точки: 181.5.

## Практическая работа

**В практической работе есть:**  
Обязательные задачи помогут проверить, как вы усвоили материал модуля. Их нужно сдать на проверку куратору.  
Дополнительные задачи подойдут тем, кто хочет ещё немного потренироваться, решая задачки по программированию. Вы также можете отправлять их на проверку и получать обратную связь куратора, хотя это необязательно.  
  
**Цели практической работы**  
• Познакомиться с вещественным типом данных float.  
• Попробовать на практике использовать математические функции из стандартной библиотеки.  
• Реализовать алгоритмы на числах с плавающей точкой.  
  
**Что входит в практическую работу**  
**Обязательные задачи:**  
1. Космический симулятор.  
2. Immolate Improved!  
3. Игрушечная история.  
4. Убийца Steam.  
5. Кенийский бегун.  
6. Маятник.  
**Дополнительные задачи:**  
7. Друзья Оушена\*.  
8. Биолаборатория 2.0\*.  
9. Яйца\*.  
10. Тяжело в учении, легко в бою\*.

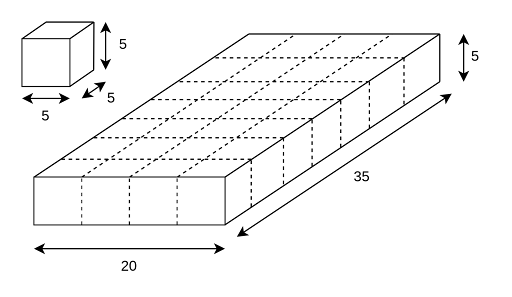
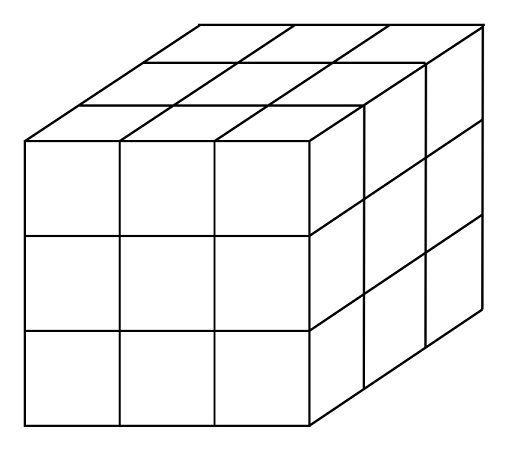
#### Задание 1. Космический симулятор

Что нужно сделать  
Вы пишете симулятор космических полётов. Ваш звездолёт массой m килограмм включает двигатель с силой тяги F ньютонов на t секунд. Напишите программу, которая по заданным F, m и t покажет, на каком расстоянии от первоначального положения окажется космический корабль через t секунд. Обеспечьте контроль ввода.  
Примечание: космический корабль находится в открытом космосе.  
Напоминаем, что расстояние можно рассчитать по формуле:  
  
**Рекомендации по выполнению**  
Для операции возведения в степень удобно использовать функцию std::pow.  
  
**Что оценивается**  
• Правильно использована формула.  
• В расчётах используются переменные типа float.

#### Задание 2. Immolate Improved!

**Что нужно сделать**  
Мы придумываем фэнтезийную онлайн-игру в жанре RPG. В орка стреляют огненными шарами разной степени мощности, но у него есть очки здоровья и показатель сопротивляемости магии. Очки здоровья, сопротивляемость магии и мощность огненного шара — это вещественные числа от нуля до единицы включительно. Чем больше мощность, тем больше урон здоровью, но чем больше сопротивляемость магии, тем урон меньше. Например, если сопротивляемость равна нулю (0%), то будет нанесён урон, равный мощности огненного шара. Если сопротивляемость равна единице (100%), урон будет равен нулю, даже если мощность заклинания максимальна.  
  
Напишите программу, в которой пользователь сначала задаёт количество здоровья и сопротивляемость магии орка, а после стреляет в него огненными шарами разной мощности, пока орк не погибнет. Мощность удара вводится с клавиатуры с каждым выстрелом. После каждого заклинания программа выводит на экран итоговый урон от удара и оставшиеся очки здоровья орка. Показатель сопротивляемости магии не изменяется при попадании. Обеспечьте контроль ввода.  
  
Обратите внимание, что сопротивляемость обозначает только долю от повреждения, которая приходится на орка. Пример:  
Урон 0.1, сопротивляемость 0.5, по орку будет нанесено 0.05 урона;  
Урон 0.2, сопротивляемость 0.5, по орку будет нанесено 0.1 урона;  
Урон 1, сопротивляемость 0.2, по орку будет нанесено 0.8 урона.  
  
**Рекомендации по выполнению**  
Для решения использовать цикл while.  
  
**Что оценивается**  
Все величины должны измеряться в одной шкале — от 0 до 1.  
Мощность удара считается с учётом сопротивляемости.

#### Задание 3. Игрушечная история

**Что нужно сделать**  
Вы решили открыть бизнес по производству игровых деревянных кубиков для детей. Вы узнали, что лучше всего продаются кубики со стороной 5 см в наборах по несколько штук, причём кубиков в наборе должно быть достаточно, чтобы сложить из них один большой куб. Для изготовления кубиков к вам в мастерскую поступают деревянные бруски в форме прямоугольных параллелепипедов любых размеров.  
  
Для оптимизации бизнес-процессов напишите программу, которая по заданным размерам исходного бруска определяет, сколько кубиков из него можно изготовить, можно ли из них составить набор для продажи и если можно, то максимальное число кубиков в этом наборе. Все кубики должны быть из цельного дерева без использования клея. Размеры бруска — вещественные числа. Обеспечьте контроль ввода.  
  
Например:  
Из бруска 20x35x5 можно изготовить 28 кубиков.  
  
Из 28 кубиков можно собрать набор из 27 кубиков.  
  
**Пример выполнения**  
Введите размеры бруска:  
X: 5  
Y: 35,76  
Z: 35,05  
Вывод:  
Из этого бруска можно изготовить 49 кубиков.  
Из них можно составить набор из 27 кубиков.  
  
**Рекомендации по выполнению**  
• Кубики изготавливаются только цельными, не из нескольких частей бруска. Минимальный размер бруска — 5x5x5 см. Подумайте как проще разбить брусок на такие кубики.  
• Минимальный размер набора — 2x2x2 = 8 кубиков.  
  
**Что оценивается**  
• Правильно считается количество кубиков, которое можно получить из бруска.  
• Количество кубиков в наборе не превышает количество кубиков, полученное из бруска.

#### Задание 4. Убийца Steam

**Что нужно сделать**  
Вы пишете программу-инсталлятор для компьютерной игры. Пока обновление скачивается, пользователю нужно показать прогресс загрузки, чтобы он мог решить, подождать ему у экрана компьютера или заварить чай.  
Напишите программу, принимающую на вход размер файла обновления в мегабайтах и скорость интернет-соединения в мегабайтах в секунду. В каждую секунду программа показывает, сколько процентов от общего объёма обновления уже скачано, до тех пор, пока файл не загрузится. После установки обновления программа выводит время в секундах, которое заняло скачивание. Обеспечьте контроль ввода.  
**Пример выполнения**  
Ввод:  
  
Укажите размер файла для скачивания: 123  
Какова скорость вашего соединения? 27  
  
Вывод:  
  
Прошло 1 сек. Скачано 27 из 123 МБ (21%).  
Прошло 2 сек. Скачано 54 из 123 МБ (43%).  
Прошло 3 сек. Скачано 81 из 123 МБ (65%).  
Прошло 4 сек. Скачано 108 из 123 МБ (87%).  
Прошло 5 сек. Скачано 123 из 123 МБ (100%).  
  
**Рекомендации по выполнению**  
Обратите внимание, что проценты это целые числа.  
  
**Что оценивается**  
• Программа работает корректно с любыми заданными значениями.  
• Количество процентов не вводит в заблуждение пользователя. Например, не должно выводиться 100%, если файл ещё не загрузился.

#### Задание 5. Кенийский бегун

**Что нужно сделать**  
Самуэль родился в Кении, и когда он немного подрос, то решил сделать карьеру профессионального бегуна. Во время тренировок на стадионе он стал запоминать свой темп на каждом километре с помощью простых наручных часов с секундной стрелкой. После тренировки он хотел бы узнать свой средний темп бега, но не может этого сделать: в Кении дети, решившие стать бегунами, обычно не ходят в школу. Помогите Самуэлю.  
Напишите программу для расчёта среднего темпа бега. Темп измеряется в секундах на километр и означает количество времени, которое бегун потратит на преодоление одного километра. Программа должна получить на вход количество километров, покрытое за тренировку, и темп в секундах на каждый из этих километров. На выходе программа выводит на экран средний темп в минутах и секундах за всю тренировку. Обеспечьте контроль ввода.  
  
**Пример выполнения**  
Ввод:  
  
Привет, Сэм! Сколько километров ты сегодня пробежал? 6  
Какой у тебя был темп на километре 1? 305  
Какой у тебя был темп на километре 2? 291  
Какой у тебя был темп на километре 3? 343  
Какой у тебя был темп на километре 4? 255  
Какой у тебя был темп на километре 5? 279  
Какой у тебя был темп на километре 6? 312  
  
Вывод:  
Твой средний темп за тренировку: 4 минуты 58 секунд.  
  
**Рекомендации по выполнению**  
Чтобы добиться требуемого результата, необходимо воспользоваться функцией округления.  
  
**Что оценивается**  
Важно, чтобы время среднего темпа было выведено в формате «минуты и секунды», как в примере.

#### Задание 6. Маятник

**Что нужно сделать**  
Известно, что амплитуда качающегося маятника с каждым разом затухает на 8,4% от амплитуды прошлого колебания. Если качнуть маятник, он, строго говоря, никогда не остановится: его амплитуда будет уменьшаться до тех пор, пока мы не сочтём такой маятник остановившимся. Напишите программу, определяющую, сколько раз качнётся маятник, прежде чем он, по нашему мнению, остановится. Программа получает на вход начальную амплитуду колебания в сантиметрах и конечную амплитуду его колебаний, которая считается остановкой маятника. Обеспечьте контроль ввода.  
Рекомендации по выполнению  
Считаем, что затухание происходит на каждое качание.  
  
**Что оценивается**  
• Программа правильно считает количество затуханий.  
• Цикл не бесконечный.

#### Задание 7. Друзья Оушена\* (дополнительное задание)

Что нужно сделать  
Банковский сейф оборудован механическим кодовым замком, состоящим из N крутящихся ручек, каждая из которых имеет K положений. Существует только одна комбинация положений всех N ручек, которая открывает сейф. Чтобы его открыть, достаточно перебрать все существующие комбинации. На проверку одной комбинации уходит в среднем T секунд, где T — вещественное число. Напишите программу, которая по заданным N, K и T определяет время, которое потребуется для перебора всех комбинаций и гарантированного открытия сейфа. Не используйте циклы. Обеспечьте контроль ввода.  
  
**Рекомендации по выполнению**  
Для решения этого задания нужно вспомнить комбинаторику.  
  
**Что оценивается**  
Правильно определяется количество комбинаций.

#### Задание 8. Биолаборатория 2.0\* (дополнительное задание)

**Что нужно сделать**  
Благодаря вашей помощи компания «Терран Биотех» вышла на финишную прямую в борьбе со всемирной эпидемией коринебактерий. Однако прошлая симуляция была недостаточно точной, да и новая версия антибиотика работает по-другому.  
В чашку Петри добавляется W капель воды, D капель чистого антибиотика и подселяется N коринебактерий. Известно, что каждую минуту количество бактерий изменяется прямо пропорционально концентрации антибиотика в растворе по формуле Nnew = 2 × Nold − Nold × D / W, где Nold — количество коринебактерий минуту назад, Nnew — их новое количество, а D / W — концентрация действующего антибиотика. При этом каждую минуту 20% от всего количества антибиотика разлагается и больше не действует.  
  
Напишите программу, которая получает на вход значения W, D и Nold и симулирует процесс в течение 20 минут, для каждой минуты выводя на экран текущую концентрацию антибиотика и количество живых коринебактерий в чашке Петри. Обеспечьте контроль ввода.  
  
**Что оценивается**  
• Количество измеряется в целых числах.  
• Ничто не уходит в отрицательные значения.  
• Цикл не бесконечный.

#### Задание 9. Яйца\* (дополнительное задание)

**Что нужно сделать**  
В рамках программы колонизации Марса компания «Спейс Инжиниринг» вывела особую породу черепах, которые должны размножаться, откладывая яйца в марсианском грунте. Откладывать яйца слишком близко к поверхности опасно из-за радиации, а слишком глубоко опасно из-за давления грунта и недостатка кислорода. Вообще, факторов много, но специалисты проделали большую работу и предположили, что уровень опасности для черепашьих яиц рассчитывается по формуле D = x^3 − 3x^2 − 12x + 10, где x — глубина кладки в метрах, а D — уровень опасности в условных единицах. Для тестирования гипотезы нужно взять пробу грунта на безопасной, согласно формуле, глубине.  
Напишите программу, находящую такое значение глубины х, при котором уровень опасности наиболее близок к нулю, учитывая допустимое отклонение. На вход программе подаётся максимально допустимое отклонение уровня опасности от нуля, а программа должна рассчитать приблизительное значение х, удовлетворяющее этому отклонению. Известно, что глубина точно больше нуля и меньше четырёх метров.  
Обеспечьте контроль ввода.  
  
**Пример выполнения**  
Введите максимально допустимый уровень опасности: 0,01  
Вывод:  
Приблизительная глубина безопасной кладки: 0,732422 метра  
  
**Рекомендации по выполнению**  
• Отрицательный уровень опасности — это тоже плохо. Нужно найти наиболее близкий к нулю уровень опасности.  
• Здесь не обязательно считать корни кубического уравнения. Достаточно найти решение на отрезке от 0 до 4.  
  
**Что оценивается**  
Для поиска решения не используется полный перебор.

#### Задание 10. Тяжело в учении, легко в бою\* (дополнительное задание)

**Что нужно сделать**  
Вы встретились со своим старым другом, который тоже изучает программирование, правда, в другом учебном заведении. За чашкой кофе он пожаловался, что сумасбродный препод дал задание написать программу, которая из двух введённых чисел определяет наибольшее, не используя при этом условных операторов и циклов. Радуясь, что на вашем курсе такого не требуют, вы всё-таки решаете помочь своему другу. Напишите для него эту программу.  
  
**Рекомендации по выполнению**  
Попробуйте решить это задание с помощью арифметических операций и простых математических функций.  
  
**Что оценивается**  
Решение не является просто функцией std::max.  
  
**Критерии оценки работы**  
1. Программы запускаются и корректно работают, нет синтаксических ошибок.  
2. Имена переменных корректные, отражают суть хранимых данных.  
3. Создан текстовый интерфейс для всех программ.  
4. Для всех заданий обеспечен контроль ввода, который обусловлен логикой работы программы. Допускается отсутствие контроля на переполнение типа.  
5. Выполнено шесть и более заданий.

### Урок 11. Функции в программировании

**Цели практической работы**  
Приобрести практическое понимание работы с функциями.  
Научиться:  
• объявлять свои функции с заданными типами возвращаемого значения и аргументами,  
• задавать собственное тело функции,  
• вызывать функции в программе и в других функциях.  
  
**Что входит в работу**  
1. Реализовать шифр Цезаря.  
2. Разработать программу проверки корректности email-адреса.  
3. Разработать программу валидации IP-адреса.  
4. Разработать программу оценки результатов игры в крестики-нолики.  
  
**Общие условия**  
Разработка выполняется в среде [VS Code](https://code.visualstudio.com/download).

#### Задание 1. Шифр Цезаря

**Что нужно сделать**  
В древности для шифрования сообщений использовался такой способ: все буквы в сообщении сдвигались на одно и то же число позиций в алфавите. Число позиций могло быть как положительным, так и отрицательным и являлось параметром шифра — его ключом. Если для сдвига на данное число позиций алфавита не хватает, то он зацикливается (то есть буква с номером 27 — это снова буква a, буква с номером 28 — это буква b и так далее).  
Например, слово abracadabra при сдвиге на десять позиций превратится в klbkmknklbk. Этот простейший шифр называется шифром Цезаря. Регистр букв (заглавные и строчные) менять не нужно. Напишите функцию encrypt\_caesar от строки и числа, которая возвращает исходную строку, зашифрованную шифром Цезаря с параметром, равным данному числу. Также напишите функцию decrypt\_caesar, выполняющую обратное преобразование. Считаем, что входные строки состоят лишь из английских букв. Если там содержатся и другие символы, то их надо игнорировать.  
**Пример 1**  
The quick brown fox jumps over the lazy dog  
**Результат**  
Ymj vznhp gwtbs ktc ozrux tajw ymj qfed itl  
**Пример 2**  
Lorem ipsum dolor sit amet consectetur adipiscing elit sed do eiusmod tempor incididunt ut labore et dolore magna aliqua Ut enim ad minim veniam quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur Excepteur sint occaecat cupidatat non proident sunt in culpa qui officia deserunt mollit anim id est laborum  
**Результат**  
Knqdl hortl cnknq rhs zlds bnmrdbsdstq zchohrbhmf dkhs rdc cn dhtrlnc sdlonq hmbhchctms ts kzanqd ds cnknqd lzfmz zkhptz Ts dmhl zc lhmhl udmhzl pthr mnrsqtc dwdqbhszshnm tkkzlbn kzanqhr mhrh ts zkhptho dw dz bnllncn bnmrdptzs Cthr ztsd hqtqd cnknq hm qdoqdgdmcdqhs hm unktoszsd udkhs drrd bhkktl cnknqd dt etfhzs mtkkz ozqhzstq Dwbdosdtq rhms nbbzdbzs btohczszs mnm oqnhcdms rtms hm btkoz pth neehbhz cdrdqtms lnkkhs zmhl hc drs kzanqtl

#### Задание 2. Проверка корректности email-адреса

**Что нужно сделать**  
Напишите программу, которая проверяет корректность email-адреса, введённого пользователем.  
Согласно международным требованиям, корректный email-адрес должен состоять из двух частей, разделённых знаком @. Первая часть должна иметь длину не менее одного и не более 64 символов, вторая часть — не менее одного и не более 63 символов. Из символов допускаются только английские буквы, цифры и знак «-» (дефис), а также точка. Точка не может быть первым или последним символом, а кроме того, две точки не могут идти подряд. В первой части (которая предшествует символу @), кроме вышеперечисленных, разрешены ещё следующие символы:  
!#$%&'\*+-/=?^\_{|}~`  
Пользователь вводит строку, задающую email-адрес. Программа должна вывести слово Yes, если адрес корректен, а в противном случае — слово No.  
**Примеры**  
Корректные адреса:  
[simple@example.com](mailto:simple@example.com)  
[very.common@example.com](mailto:very.common@example.com)  
[disposable.style.email.with+symbol@example.com](mailto:disposable.style.email.with+symbol@example.com)  
[other.email-with-hyphen@example.com](mailto:other.email-with-hyphen@example.com)  
[fully-qualified-domain@example.com](mailto:fully-qualified-domain@example.com)  
[user.name+tag+sorting@example.com](mailto:user.name+tag+sorting@example.com) (может на самом деле быть перенаправлен на [user.name@example.com](mailto:user.name@example.com) в зависимости от почтового сервиса)  
[x@example.com](mailto:x@example.com) (однобуквенная локальная часть)  
[example-indeed@strange-example.com](mailto:example-indeed@strange-example.com)  
admin@mailserver1  
[example@s.example](mailto:example@s.example)  
mailhost![username@example.org](mailto:username@example.org)  
user%[example.com@example.org](mailto:example.com@example.org)  
Некорректные адреса:  
[John..Doe@example.com](mailto:John..Doe@example.com) (две точки не могут идти подряд)  
Abc.example.com (нет символа @)  
A@b@[c@example.com](mailto:c@example.com) (символ @ должен быть только один)  
a"b(c)d,e:f;g<h>i[j\k][l@example.com](mailto:l@example.com) (есть недопустимые символы)  
[1234567890123456789012345678901234567890123456789012345678901234+x@example.com](mailto:1234567890123456789012345678901234567890123456789012345678901234+x@example.com) (первая часть длиннее 64 символов)  
i\_like\_underscore@but\_its\_not\_allow\_in this\_part.example.com (во второй части есть недопустимый символ «»)

#### Задание 3. Валидация IP-адреса

**Что нужно сделать**  
Проверьте, задаёт ли введённая пользователем строка корректный IP-адрес четвёртой версии (IPv4). IP-адрес должен представлять из себя четыре числа, разделённые точками без пробелов, причём каждое число должно быть в диапазоне от 0 до 255 включительно. Числа не должны содержать ведущих нулей. Других символов в строке, кроме вышеописанных, быть не должно.  
Пользователь вводит строку, задающую IP-адрес через стандартный ввод. В результате программа должна вывести слово Valid, если адрес корректен, и слово Invalid, если это не так.  
  
**Примеры**  
Корректные:  
127.0.0.1  
255.255.255.255  
1.2.3.4  
55.77.213.101  
Некорректные:  
255.256.257.258 (есть числа больше 255)  
0.55.33.22. (лишняя точка в конце)  
10.00.000.0 (лишние нули)  
23.055.255.033 (опять лишние нули)  
65.123..9 (две точки подряд)  
a.b.c.d (посторонние символы вместо чисел и точек)

#### Задание 4. Оценка результатов игры в крестики-нолики

**Что нужно сделать**  
Петя и Вася играли в крестики-нолики и не стёрли результаты своей игры. Теперь перед вами табличка 3 × 3, которая осталась после их поединка. Вы хотите понять, выиграл ли кто-то из них или они не доиграли.  
На вход программе даётся три строки, кодирующие верхнюю, среднюю и нижнюю строчки игрового поля. Крестик обозначается символом X (английская заглавная буква X), нолик — символом O (заглавная английская буква O), пустая клетка — точкой.  
Если ввод некорректен, то есть такая таблица не могла получиться в ходе настоящей игры, то надо написать в ответ слово Incorrect.  
Если таблица корректна и выиграли крестики, то нужно вывести в ответ слова Petya won, если нолики — Vanya won, иначе (то есть они не доиграли или сыграли вничью) надо написать Nobody.  
  
**Примеры**  
X..  
OXO  
OOO  
Ответ: Incorrect (потому что ноликов в конце игры не может быть больше, чем крестиков).  
X..  
.X.  
OO.  
Ответ: Nobody (потому что они не доиграли).  
XXO  
OOX  
XOX  
Ответ: Nobody (потому что это ничья).  
XO.  
XO.  
X.O  
Ответ: Incorrect (крестики уже победили, и ноликов не может быть столько же).  
OX.  
XOX  
X.O  
Ответ: Incorrect (потому что нолики выиграли за три хода, и на поле за это время не могло успеть появиться четыре крестика).  
..X  
OX.  
X.O  
Ответ: Petya won.  
0..  
...  
...  
Ответ: Incorrect (потому что допускаются только символы X, O и точка, а цифра 0 не является допустимым символом).