

Краснодар, 2022 г.

Это учебное пособие создавалась с огромным желанием помочь педагогам и начинающим программистам в изучении основ языка программирования С++. А главное, как можно понятней и короче осветить основные темы с использованием практических примеров. Поэтому вы не найдете здесь историю программирования С++. Хотя, об этом тоже нужно знать. Упор сделан на то, чтобы дать больше ценной информации - что, где, как и зачем надо писать, чтобы программа выполняла ту задачу, которая перед вами стоит.

**Введение**

С этого момента наши программы станут намного интересней и немного сложнее. Мы научим нашу программу выбирать при помощи логического выбора. Наконец программы будут думать, анализировать и решать какой путь решения им выбрать.

Начнем с того, что нам нужно повторить логические операции.

Рассмотрим на примере одного кода, чтобы освежить память.

В этой программе сравнивается три переменные **var**, которые изначально имеют свои значения.

C 13 по 30 строки проходит простое сравнение и равенство значений двух переменных.

В 32 – 34 строках мы используем инверсию, т.е. проводим логическое отрицание переменных внутри скобок относительно их оператора сравнения.

С 36 по 41 объединяем результаты сравнения двух скобок. Есть два варианты объединения – **конъюнкция** и **дизъюнкция**.

Конъюнкция – это логическое умножение, т.е. и левое, и правое условие должно быть верно, а в дизъюнкции наоборот – одно или несколько из условий должно быть верно.

Таблица 1 - Логические операции

|  |  |
| --- | --- |
| ***Отрицательная инверсия и логические операции объединения*** | |
| **!** | **НЕ** (инверсия) |
| **&&** | **И** (конъюнкция) |
| **||** | **ИЛИ** (дизъюнкция) |

Рассмотрим пример:



Рисунок 1 - Пример программы

**Логические операции. Оператор ветвления.**

Условные операторы или операторы ветвления подразделяются на две категории: полное и неполное ветвление.

Если мы будем рассматривать структуру полного ветвления, то это будет выглядеть следующим образом:

**if (условие)**

**{**

**блок действий 1;**

**}**

**else**

**{**

**блок действий 2;**

**}**

*if – если (условие) верно, то выполнить блок действий 1;*

*else – иначе выполнить блок действий 2;*

Приступим к нашим операторам выбора (они же условные операторы, они же операторы ветвления). Мы рассмотрим несколько простых примеров, которые помогут вам познакомиться с этими операторами и применять их уже в более сложных программах.

Пример: Пользователь вводит два числа и программа определяет какое их этих двух чисел больше, предусмотреть вариант того, что числа равны.

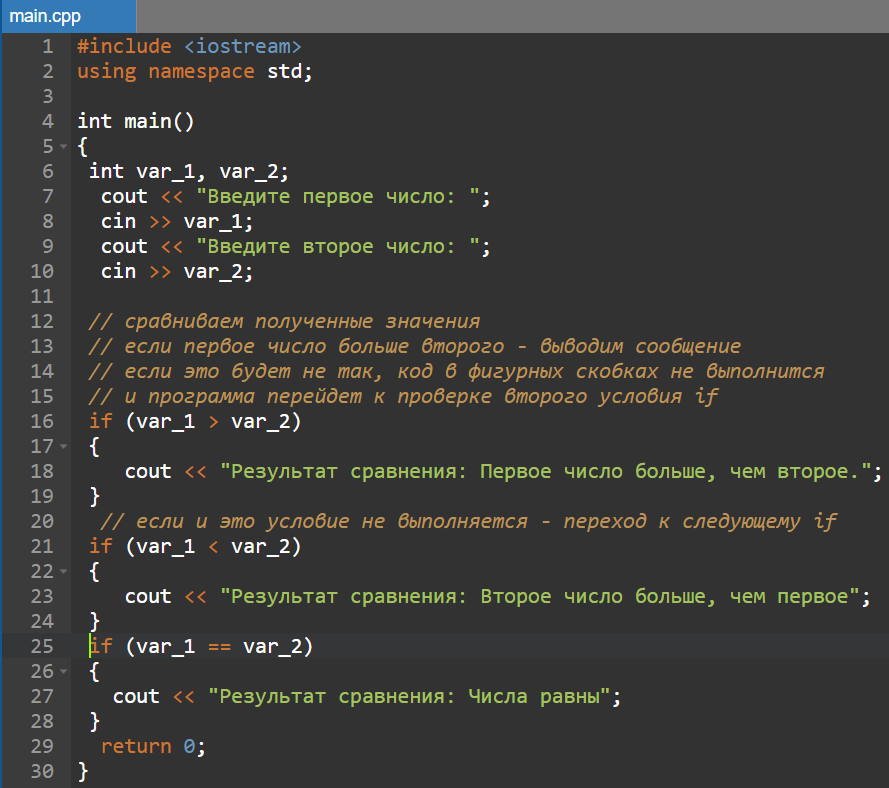


Рисунок 2 - Пример кода программы

Если условие в скобках истинно, то выполнится действие после них, если условие ложно, то следующее условие и т.д. Так будет до тех пор, пока хоть одно условие не окажется верным.

Но этот код можно немного сократить, используя полное ветвление. А также, если блоки **if** или **else** содержат только одну строку кода в фигурных скобках **{}** , то эти фигурные скобки можно не писать. Посмотрим, как изменится наш код:

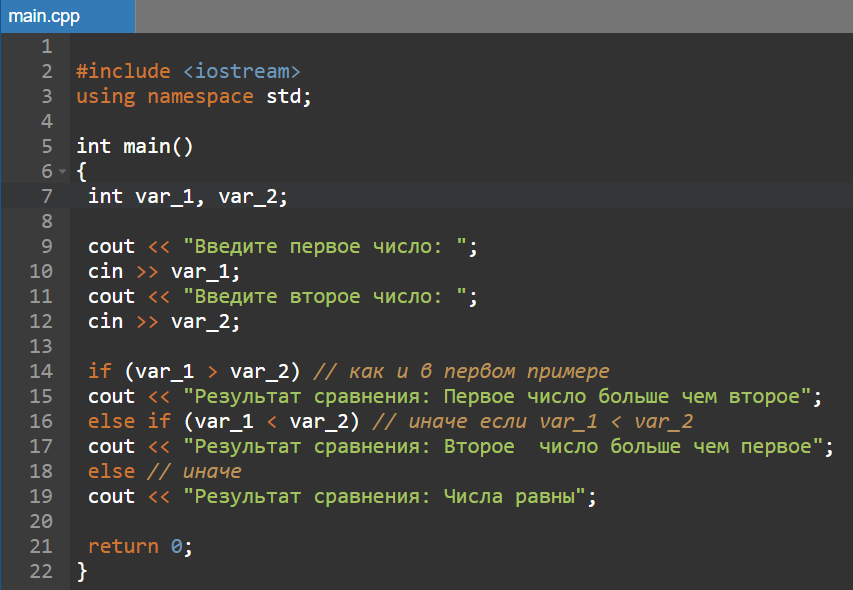


Рисунок 3 - Пример кода программы

Еще один плюс – если первое условие **if** выполняется, то остальные блоки **if else** и **else**, уже не проверяются (т.е. время на выполнение программы сокращается). Ну а если первое условие ложно, программа идет в следующий блок и проверяет его условие. Если и тут ложь, то выполнится последний блок **else**, в котором условие может быть только равенство, т.к. мы до этого перебрали все возможные варианты к этим двум значениям.

Результат работы этих двух кодов программ одинаков. Но во втором - код компактней и работы компилятору меньше.

* **Задачи**

1. Пользователь вводит порядковый номер пальца руки. Необходимо показать его название на экран. Нумерацию начинать с большого пальца тыльной стороны руки (ноги).
2. Как много у тебя ног?! Пользователь вводит количество ног, в соответствии с этим на экран выводится то, кто он: 1 – Пират, 2 Человек, 8 – Осьминог. В случае, если пользователь вводит число, которого нет, вывести ошибку в воде данных.
3. Есть два числа, которые вводит пользователь и знак операции: + - / \*; (int, char, bool f= true;).

**Тернарный оператор**

Давайте познакомимся еще с одним оператором выбора (ветвления) – это **тернарный оператор,** в котором используются следующие команды - **?  :**

Условия его использования очень просты и для простого. Например, если вы хотите написать быстрый блок кода для выхода из программы, то тернарный оператор, это лучший выбор.

Синтаксис:   
**условие ? команда 1 : команда 2;**

Обратите внимание на пример блока кода программы. В этом коде представлен фрагмент пополнения мобильного счета телефона на определённую сумму. Если вас устраивает данная операция вы нажимаете 0, иначе необходимо выбрать другую операцию - клавиша 1.

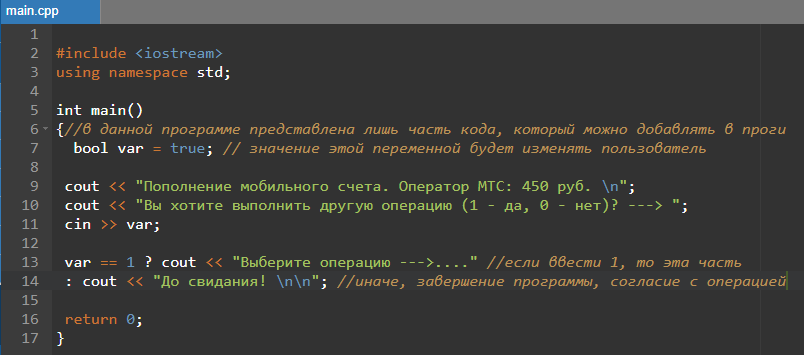


Рисунок 4 - Пример кода программы

Более подробно рассмотрим строку 13. **var==1** – это ваше условие, которое оканчивается знаком **?** , далее указывается **команда 1** – это вывод на экран монитора фразы **«Выберите операцию…»** (она выведется только в том случае, если пользователь ввел **1,** т.е. значение переменной var стало true), потом идет **:** - разграничение двух команд, а далее уже **команда 2 – «До свидания!»** (она выведется только в том случае, если пользователь ввел **0,** т.е. значение переменной var стало false).

В языке C++ применяются несколько видов циклов. Цикл **for**  мы рассмотрим первым, так как его легче понять новичкам.

Для начала, что же такое цикл?! Цикл – это определенный участок кода, в котором выполняется повторение какого-либо элемента определенное количество раз, все это указываете вы. Например, чтобы вывести на экран числа от единицы до пятисот, можно использовать ручной вывод на экран, через команду **cout<<«Number 1.»<<endl;,** но зачем усложнять себе жизнь, если есть цикл **for.**

Синтаксис:

**for (определение управляющей переменной (начальное значение); условие повторения цикла; шаг изменения управляющей переменной)   
{тело цикла;}**

Давайте рассмотри на примере. Нам необходимо вывести на экран числа от 1 до 300. Чтобы не создавать очень длинную программу из обычных cout, используем цикл, который займет всего лишь пару строк.

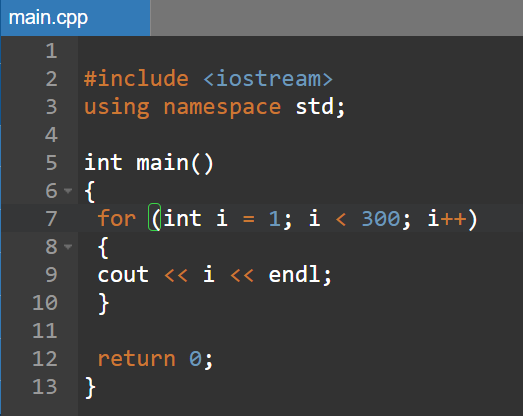


Рисунок 5 - Пример кода программы

Разберем for:

**int i = 1;** - это начальное значение нашей переменной-счетчика;

**i < 300;** - это кол-во повторений в цикле или диапазон i, т.е. максимальное число вывода будет 299. Если я захочу до 300 включительно, то необходимо написать **<=**.

**i++** - шаг изменения переменной-счетчика на 1.

В 9 строке вывод числа. Если бы наше начальное значение **i** было равно нулю, то в выводе нужно написать **i+1,** чтобы вывод был с единицы.

Управляющая переменная может как увеличиваться, так и уменьшаться. В зависимости от того, какая задача поставлена перед программистом. Вот пример уменьшения:

**for (int i = 300; i > 0; i--)**

Помимо постфиксного инкремента (++) или декремента (--) изменение управляющей переменной может быть ++i,  i+= 2,  i += 20,  i -= 15.  
Например:

**for (int i = 755; i > 0; i /= 2) -** управляющая переменная i изменяется от 755 до 0, с делением при каждом шаге на 2.

**for (int i = 1; i >= 144; i\*=12) -** управляющая переменная i изменяется от 1 до 144 включительно, с умножением при каждом шаге на 12.

В дополнение ко всему все три, используемые в круглых скобках, выражения необязательны, достаточно двух выражений, шаг изменения можно объявить в самом теле цикла.

Также можно и вовсе запустить зацикливание. Если сделать такую, к примеру, запись: **for( ; ; )** – это будет воспринято компилятором, как запуск бесконечного цикла. Т.е. цикл будет выполняться бесконечно. В таком случае, чтобы все-таки работать с таким циклом, управляющая переменная может быть определена до цикла, шаг изменения добавить в конец тела цикла **for**, а условие продолжения цикла можно задать, используя нам хорошо известный оператор **if** и оператор **break**. Вот вариант выполнения:

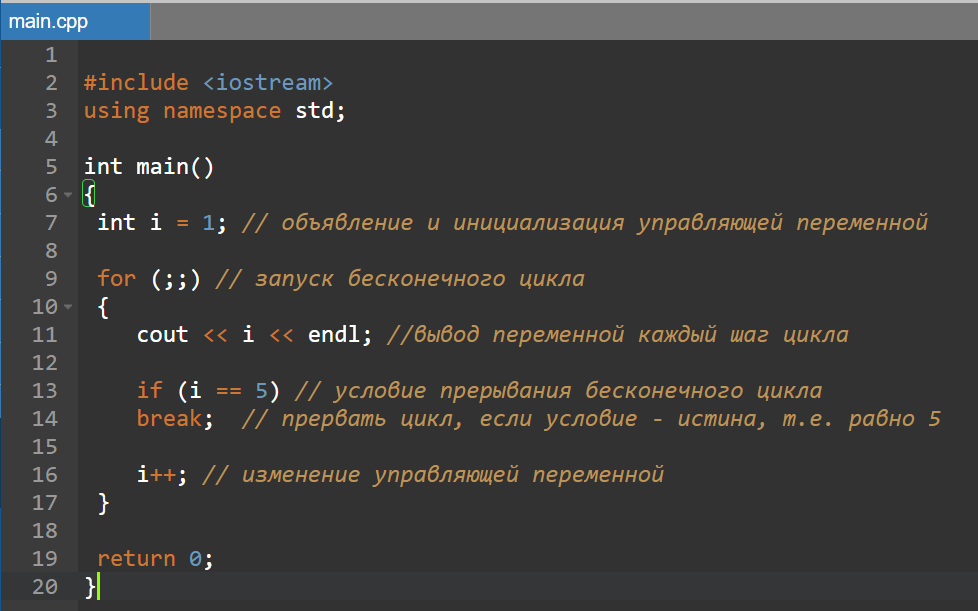
****

Рисунок 6 - Пример кода программы

В этой программе он выведет подряд числа от 1 до 5 и завершит программу, т.е. выйдет из цикла.

Помимо чисел, цикл **for** может также ограничиваться и переменными, который вводит пользователь. Например:

**for (int i = 0; i <= num; i++) – num** это число, которое определяет пользователь.

* **Задачи:**

1. Написать программу, которая выводит квадрат целого числа от 1 до переменной, которую укажет пользователь, шаг изменения – увеличение на 2.
2. Написать программу, выполняющую вывод на экран монитора чисел, **которые при делении на 7 дают остаток 1, 2 или 5 (%). Диапазон чисел от 21 до 87. Шаг изменения – увеличение на 1.**
3. **На складе имеется определённое количество ящиков с фруктами. Когда подъезжает грузовик, попросить пользователя ввести, сколько ящиков загрузить в первую машину, во вторую и так далее, пока не закончатся ящики с фруктами. Предусмотреть тот случай, когда пользователь введёт количество ящиков больше, чем есть на складе. (2 внешние переменные и одна – счетчик).**

**Циклы while и do…while**

Они очень похожи между собой, но разберем каждый из них.

Синтакис **while**:

**while (условие)**

**{**

**тело цикла;**

**}**

Синтакис **do…while**:

**do**

**{**

**тело цикла;**

**} while (условие);**

Разница состоит только в том, что при начале выполнения цикла **while**, сразу происходит проверка условия. Если оно ложно (**false**), то тело цикла вообще не выполнится, а при использовании **do while**, сразу выполнится тело цикла, и только потом будет проверено условие в круглых скобках  
**( )** после **while**. Аналогично, при возврате **false**, цикл завершает работу, иначе – выполняется следующий шаг. Т.е. тело цикла **do while** выполнится хотя бы один раз, при любом раскладе.

Поговорим о них в отдельности. Начнем с **while**. Чтобы выполнилось тело цикла, необходимо чтобы при проверке условия в круглых скобках **( )**, была возвращена истинность (true).

Проверка условия и выполнение тела будет происходить, пока условие не станет ложным. Поэтому для прерывания цикла, в его теле должно быть условие прерывания или условие, которое может повлиять на проверку условия.

Рассмотрим пример с изменением значения управляющей переменной в теле цикла. Пользователю предлагается пополнить мобильный счет через терминал (реализовать примерный внешний интерфейс). Он вводит сумму с клавиатуры. Предусмотреть в программе возможность изменения ошибочно введенной суммы.

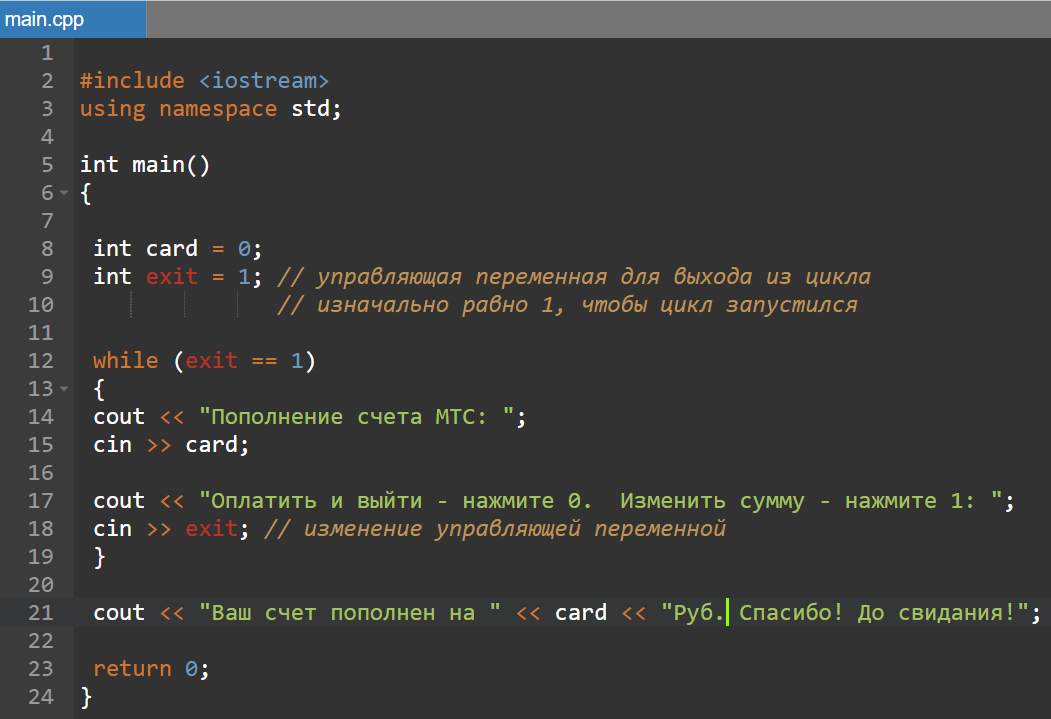


Рисунок 7 - Пример кода программы

Объявим переменную **exit** и зададим ей значение единицы, а также объявим переменную **сard**, которая отвечает за карточный счет. Поэтому при входе в цикл, проверка условия while **(exit == 1)** примет значение **true**и программа начнет выполнять тело цикла. Объясняется это тем, что любое значение переменной **exit**отличное от нуля будет восприниматься компилятором, как **true**. Тело цикла будет выполняться, пока пользователь не изменит значение этой переменной на 0.

В строках 17 – 18 пользователю задается вопрос и предлагается сделать выбор:  **cout << “Оплатить и выйти – нажмите 0. Изменить сумму – нажмите 1: “;** Т.е., если он сделал ошибку при вводе сумм, то у него есть возможность исправить её, нажав 1.

Как только пользователь введет верную сумму и нажмет 0, произойдет выход из цикла и программа продолжит работу. В итоге на экране покажется сообщение о сумме пополнения счет.

Эту же задачу можно решить, применяя цикл **do while**. В этом случае, изначально значение переменной **exit** может быть каким угодно. Цикл в любом случае выполнит тело и предложит сделать выбор.

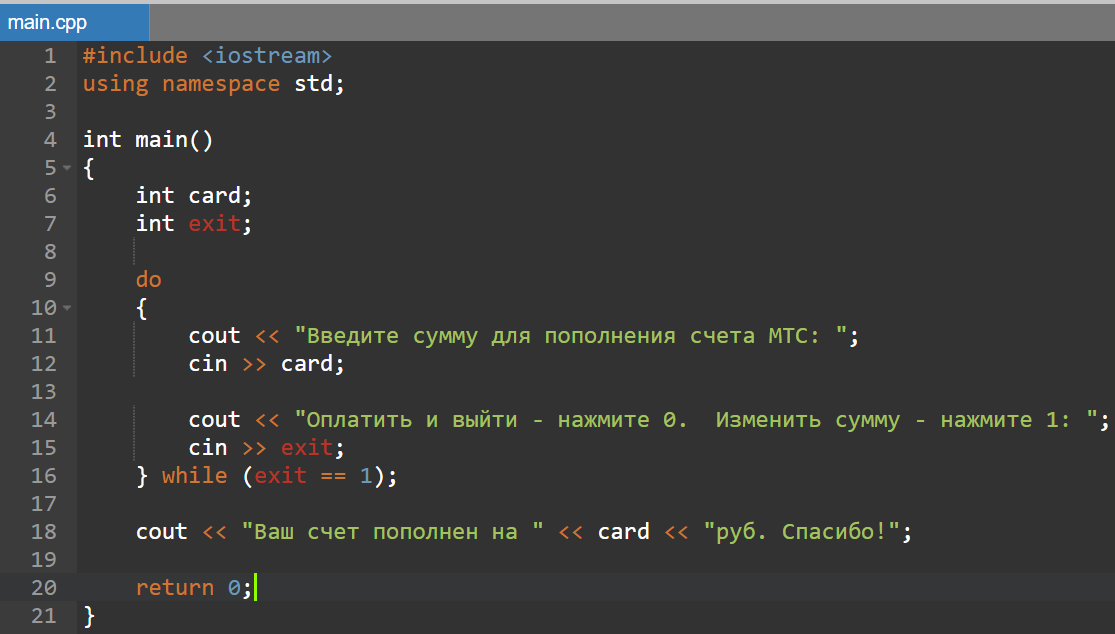


Рисунок 8 - Пример кода программы

Запустив программу, тело цикла **do while** выполнится вне зависимости от значения переменной **exit**, но выведется ли оно на экран определит **while**.  Обратите внимание на точку с запятой **;**  после закрывающей круглой скобки.  Для цикла **do while**она обязательна.

Сравнивая циклы **for**, **while**и **do while**   видно, что они похожи. Только в цикле **for**всё необходимое для его нормальной работы собрано в одной строке. А в циклах **while** и **do while** управляющая переменная объявлена до цикла, а ее изменение происходит ниже в теле.

Какой цикл использовать – решать вам. Имейте в виду, что циклы **while** и **do while** используются тогда, когда не знаете точное количество повторений, а цикл **for** – когда знаете.  Стоит ещё помнить, что в отличии от циклов **while**и **do while** , управляющая переменная цикла **for** не видна за пределами его тела (то есть при выходе из цикла, она исчезает).

В этом пособие мы рассмотрели с вами операторы цикла в С++. Возможно некоторые моменты еще непонятны и трудны в понимании, но это не страшно. Больше практики и будет проще!

В следующем учебном пособие мы с вами продолжим изучать программирование и познакомимся со многими интересными темами и примерами.