Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебный предмет
«Конструирование программ и языки программирования»

Инструкция

по выполнению лабораторной работы №1
«Разработка, отладка и испытание разветвляющихся и циклических алгоритмов и программ»

Минск 2024 г.

Лабораторная работа № 1

Тема работы: «Разработка, отладка и испытание разветвляющихся и циклических алгоритмов и программ»

1 Цель работы

Сформировать умения разрабатывать программы с использованием операторов выбора, цикла, передачи управления.

2 Задание

Лабораторная работа состоит из 4 частей. Первые три части обязательны для выполнения, четвертая часть дополнительная. Задачи третей части нужно решить двумя способами, с помощью цикла **while** и **do..while**. В каждой части номер задачи соответствует вашему номеру по списку.

Часть 1: ветвление

1. Прямоугольник задан координатами двух противоположных своих вершин. Определите, принадлежит ли точка с заданными координатами области прямоугольника, если стороны прямоугольника параллельны осям координат.
2. Прямоугольник задан координатами своих вершин. Определите, принадлежит ли окружность с заданным радиусом и координатами центра области прямоугольника.
3. Даны длины сторон треугольника. Определите, является ли данный треугольник прямоугольным.
4. Даны радиус круга и сторона квадрата. Проверьте, пройдет ли квадрат в круг?
5. Даны радиус круга и сторона квадрата. Проверьте, пройдет ли круг в квадрат?
6. Даны координаты точки $M(x, y)$. Определите, принадлежит ли данная точка замкнутому множеству D, заданному системой ограничений:
$$\begin{cases} x + y \le 1, \\ 2x - y \le 1, \\ y \ge 0. \end{cases}$$
7. Введите два положительных числа и покажите, что среднее арифметическое этих чисел не меньше их среднего геометрического.
8. Введите два положительных числа и покажите, что среднее геометрическое этих чисел не меньше их среднего гармонического.
9. Даны длины трех отрезков. Определите, можно ли из этих отрезков сложить треугольник.
10. Даны длины сторон треугольника. Определите, является ли данный треугольник равнобедренным.
11. Даны длины сторон треугольника. Определите, является ли данный треугольник равносторонним.
12. Даны длины сторон треугольника. Определите, является ли данный

треугольник разносторонним.

13. Даны в градусах величины двух углов треугольника. Определите, является ли данный треугольник равнобедренным.
14. Найдите наибольшее значение из трех $f(1)$, $f(2)$ и $f(3)$, где $f(x) = \sin(5x)$.
15. Даны в градусах величины двух углов треугольника. Определите, является ли данный треугольник остроугольным.
16. Вывести какой-либо вопрос и несколько возможных ответов на него, один из которых верный. После ввода пользователем номера ответа, который он считает верным, выдать сообщение о правильности его выбора.
17. Вывести какой-либо вопрос и несколько возможных ответов на него, некоторые из которых верные. После ввода пользователем номеров ответов, которые он считает верными, выдать сообщение о правильности его выбора.
18. Треугольник задан координатами вершин. Определите, принадлежит ли точка с заданными координатами области треугольника?
19. Треугольник задан координатами вершин. Определите находится ли отрезок внутри данного треугольника.
20. Треугольник задан координатами своих вершин. Определите длину большей стороны треугольника.
21. Треугольник задан координатами своих вершин. Определите длину большей медианы треугольника.
22. Треугольник задан координатами своих вершин. Определите длину меньшей стороны треугольника.
23. Четырехугольник задан координатами своих вершин. Определите, является ли данный четырехугольник прямоугольником.
24. Четырехугольник задан координатами своих вершин. Определите, является ли данный четырехугольник квадратом.
25. Дано трехзначное натуральное число. Определите, является ли сумма цифр данного числа четной.
26. Дано трехзначное натуральное число. Определите, является ли вторая цифра числа наибольшей.
27. Дано трехзначное натуральное число. Получите наименьшее число, составленное из цифр данного числа.
28. Даны два натуральных числа. Определите, является ли одно из них делителем другого.
29. В каждой больничной палате четыре койки. Введите количество палат, количество больных мужчин и количество больных женщин. Определите, сколько всего свободных мест.
30. В каждой больничной палате четыре койки. Введите количество палат, количество больных мужчин и количество больных женщин. Сколько больных мужского пола можно еще положить в больницу?

Часть 2: цикл for

1. Напечатайте таблицу стоимости порций сыра весом 50, 100, 150, ..., 1000 г (цена 1 кг вводится с клавиатуры)
2. Вычислите сумму $1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n}$.
3. Пусть n – натуральное число и пусть n!! означает $1 \cdot 3 \cdot \cdot ... \cdot n$ для нечетного n и $2 \cdot 4 \cdot ... \cdot n$ для четного n. Для заданного натурального n вычислите n!!
4. Вычислите сумму $\sum_{i=1}^{n}\frac{1}{i!}$.
5. Найдите n-й член ряда Фибоначчи, элементы которого вычисляются по формулам: $a_1 = a_2 = 1$; $a_i = a_{i-1} + a_{i-2}$, ($i > 2$).
   П р и м е ч а н и е. Для нахождения членов ряда используйте только две переменные a и b.
6. Найдите сумму n-членов ряда Фибоначчи, используя также только две переменные для нахождения суммы ряда.
7. Выведите на экран таблицу значений: 1, 1 + h, 1 + 2h, … 1 + nh.
8. Вычислите сумму $\sum_{i=1}^{n}\frac{1}{2^i}$.
9. Вычислите число сочетаний из n по m $C_n^m = \frac{n!}{m!(n-m)!}$.
10. Вычислите число размещений из n по m
    $A_n^m = \frac{n!}{(n-m)!} = n(n-1)...(n-m+1)$.
11. Выведите на экран таблицу значений функции $y=a^x$ для x, изменяющегося от a до b с шагом h.
12. Вычислите сумму $\sum_{i=1}^{n}\frac{1}{i^3}$.
13. Вычислите сумму $\sum_{i=1}^{n}(-1)^i 2^i$.
14. Вычислите сумму $\sum_{i=1}^{n}\frac{(-1)^i}{2i}$.
15. Вычислите произведение $P = 1 \cdot 3 \cdot 5 \cdot 7 \cdot \ldots \cdot (2n + 1)$ для заданного n.
16. Вычислите произведение $P = 2 \cdot 4 \cdot 6 \cdot 8 \cdot \ldots \cdot 2n$ для заданного n.
17. Вычислите сумму $S = 1 \cdot 3 + 3 \cdot 5 + 5 \cdot 7 + \ldots + (2n - 1)(2n + 1)$ для заданного n.
18. Вычислите произведение $S = (1 + 3) \cdot (3 + 5) \cdot \ldots \cdot ((2n - - 1) + (2n + 1))$ для заданного n.
19. Выведите на экран элементы последовательности $a_n = = a_{n-1} + nd$ для n, изменяющегося от 1 до k, $a_0 = 0$; k и d заданные натуральные числа.
20. Вычислите сумму $A = 1 + (1 + 2) + (2 + 3) + (3 + 4) + \ldots + ((n - 1) + n)$
21. Составьте таблицу умножения для заданного числа N, которая содержит результаты умножения $1 \cdot N$, $2 \cdot N$, ..., $N \cdot N$.
22. Покажите, что при всех натуральных n число $n^3 + 2n$ кратно 3.
23. Вычислите сумму $A = \frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \frac{1}{3 \cdot 4} + \ldots + \frac{1}{(n-1) \cdot n}$.

24. Вычислите сумму

$$A = \frac{1}{1\cdot 4} + \frac{1}{4\cdot 7} + \frac{1}{7\cdot 10} + \ldots + \frac{1}{(3\cdot(n-1)+1)\cdot(3\cdot n+1)}.$$

25. Вычислите сумму

$$A = \frac{1}{1\cdot 3\cdot 5} + \frac{1}{3\cdot 5\cdot 7} + \ldots + \frac{1}{(2n-1)\cdot(2n+1)\cdot(2n+3)}.$$

26. Вычислите сумму

$$A = \frac{1}{1\cdot 2\cdot 3} + \frac{1}{2\cdot 3\cdot 4} + \ldots + \frac{1}{n\cdot(n+1)\cdot(n+2)}.$$

27. Вычислите произведение $\left(1-\frac{1}{2}\right)\left(1-\frac{1}{3}\right)\ldots\left(1-\frac{1}{n+1}\right)$.

28. Вычислите сумму

$$A = \frac{1}{1\cdot 2\cdot 3\cdot 4} + \frac{1}{2\cdot 3\cdot 4\cdot 5} + \ldots + \frac{1}{n\cdot(n+1)\cdot(n+2)\cdot(n+3)}.$$

29. Покажите, что при всех натуральных n число $n^3 + 2n$ кратно 3.
30. Покажите, что при всех натуральных n число $3^{3n+2} + 5\cdot 2^{3n+1}$ кратно 19.
31. Покажите, что при всех натуральных n число $2^{n+5}\cdot 3^{4n} + 5^{3n+1}$ кратно 37.

Часть 3: цикл while и do..while

1. Вычислите сумму ряда с заданной степенью точности α:
$$\sum_{n=0}^{\infty}(-1)^n \times \frac{n}{\left(1+n^3\right)^2}, \qquad \alpha = 0{,}001$$
2. Вычислите сумму ряда с заданной степенью точности α:
$$\sum_{n=1}^{\infty}(-1)^n \times \frac{1}{3n^2}, \qquad \alpha = 0{,}0001$$
3. Вычислите сумму ряда с заданной степенью точности α:
$$\sum_{n=1}^{\infty}(-1)^{n+1} \times \frac{1}{n}, \qquad \alpha = 0{,}0001$$
4. Вычислите сумму ряда с заданной степенью точности α:
$$\sum_{n=1}^{\infty}(-1)^{n+1} \times \frac{1}{(2n)^3}, \qquad \alpha = 0{,}001$$
5. Вычислите сумму ряда с заданной степенью точности
α: $\sum_{n=0}^{\infty}(-1)^n \times \frac{1}{n(2n+1)}, \qquad \alpha = 0{,}001$
6. Вычислите сумму ряда с заданной степенью точности
α: $\sum_{n=1}^{\infty}(-1)^n \times \frac{1}{(2n+1)}, \qquad \alpha = 0{,}0001$
7. Вычислите сумму ряда с заданной степенью точности α:
$$\sum_{n=1}^{\infty}(-1)^n \times \frac{n}{2^n}, \qquad \alpha = 0{,}001$$
8. Вычислите сумму ряда с заданной степенью точности α:

$$\sum_{n=1}^{\infty}(-1)^n \times \frac{n^2}{3^n}, \qquad \alpha = 0{,}001$$

9. Вычислите сумму ряда с заданной степенью точности
α: $$\sum_{n=1}^{\infty}(-1)^n \times \frac{n}{(2n-1)^2(2n+1)^3}, \qquad \alpha = 0{,}001$$
10. Вычислите сумму ряда с заданной степенью точности
α: $$\sum_{n=1}^{\infty}(-1)^n \times \frac{1}{(2n-1)n}, \qquad \alpha = 0{,}0001$$
11. Вычислите сумму ряда с заданной степенью точности
α: $$\sum_{n=0}^{\infty}(-1)^{n+1} \times \left(\frac{-2}{3}\right)^n, \qquad \alpha = 0{,}001$$
12. Вычислите сумму ряда с заданной степенью точности α: $$\sum_{n=1}^{\infty}(-1)^n \times \frac{n}{7^n}, \qquad \alpha = 0{,}0001$$
13. Вычислите сумму ряда с заданной степенью точности
α: $$\sum_{n=1}^{\infty}(-1)^{n+1} \times \left(\frac{-2}{3}\right)^{n+1}, \qquad \alpha = 0{,}01$$
14. Вычислите сумму ряда с заданной степенью точности α: $$\sum_{n=1}^{\infty}(-1)^n \times \frac{1}{2n}, \qquad \alpha = 0{,}001$$
15. Вычислите сумму ряда с заданной степенью точности α: $$\sum_{n=0}^{\infty}(-1)^n \times \frac{1}{3n+1}, \qquad \alpha = 0{,}01$$
16. Вычислите сумму ряда с заданной степенью точности
α: $$\sum_{n=1}^{\infty}(-1)^n \times \frac{1}{(2n)^2}, \qquad \alpha = 0{,}00001$$
17. Вычислите сумму ряда с заданной степенью точности
α: $$\sum_{n=1}^{\infty}(-1)^n \times \frac{(2n+1)}{2n^2}, \qquad \alpha = 0{,}001$$
18. Вычислите сумму ряда с заданной степенью точности α: $$\sum_{n=1}^{\infty}(-1)^n \times \frac{1}{2^n \times n}, \qquad \alpha = 0{,}001$$
19. Вычислите сумму ряда с заданной степенью точности α: $$\sum_{n=0}^{\infty}(-1)^n \times \frac{1}{(3^n+1) \times n}, \qquad \alpha = 0{,}001$$
20. Вычислите сумму ряда с заданной степенью точности α: $$\sum_{n=1}^{\infty}(-1)^n \times \frac{1}{2n^3}, \qquad \alpha = 0{,}0001$$
21. Вычислите сумму ряда с заданной степенью точности α: $$\sum_{n=1}^{\infty}\frac{\cos \pi n}{3^n(n+1)}, \qquad \alpha = 0{,}001$$
22. Вычислите сумму ряда с заданной степенью точности α:

$$\sum_{n=0}^{\infty}(-1)^n \times \frac{\cos \pi n}{4^n(2n+1)}, \qquad \alpha = 0{,}001$$

23. Вычислите сумму ряда с заданной степенью точности α:
$$\sum_{n=1}^{\infty}\frac{\sin(\pi/2+\pi n)}{n^3}, \qquad \alpha = 0{,}01$$
24. Вычислите сумму ряда с заданной степенью точности α:
$$\sum_{n=0}^{\infty}(-1)^n \times \frac{2^n}{(n+1)^n}, \qquad \alpha = 0{,}001$$
25. Вычислите сумму ряда с заданной степенью точности α:
$$\sum_{n=0}^{\infty}(-1)^n \times \frac{1}{(n+1)^n}, \qquad \alpha = 0{,}001$$
26. Вычислите сумму ряда с заданной степенью точности α:
$$\sum_{n=1}^{\infty}\frac{\sin(\pi/2+\pi n)}{n^3+1}, \qquad \alpha = 0{,}01$$
27. Вычислите сумму ряда с заданной степенью точности α:
$$\sum_{n=0}^{\infty}(-1)^n \times \frac{1}{n^3(n+3)}, \qquad \alpha = 0{,}01$$
28. Вычислите сумму ряда с заданной степенью точности α:
$$\sum_{n=0}^{\infty}\frac{\cos \pi n}{(n^3+1)^2}, \qquad \alpha = 0{,}001$$
29. Вычислите сумму ряда с заданной степенью точности α:
$$\sum_{n=0}^{\infty}(-1)^n \times \frac{1}{1+n^3}, \qquad \alpha = 0{,}001$$
30. Вычислите сумму ряда с заданной степенью точности α:
$$\sum_{n=1}^{\infty}(-1)^n \times \frac{2n+1}{n^3(n+1)}, \qquad \alpha = 0{,}01$$

Часть 4: дополнительные задачи

1. Значение функции $\sin^2(x)$ можно вычислить с помощью разложения ее в ряд Маклорена
$$\sin^2(x) = x^2 - \frac{x^4}{3} + \frac{2x^6}{45} - \ldots + (-1)^{n-1}2^{2n-1}\frac{x^{2n}}{(2n)!} + \ldots$$
Вычислите $\sin^2(x)$ с точностью EPS, т.е. вычисление суммы ряда нужно продолжать до тех пор, пока абсолютная величина очередного члена ряда не станет меньше EPS. Определите количество членов ряда, которое для этого понадобилось.
2. Напишите программу сложения двух рациональных дробей. Если полученный результат является сократимой дробью, то сократите эту дробь.
3. Напишите программу умножения двух рациональных дробей. Если полученный результат является сократимой дробью, то сократите эту дробь.

4. Вычислите значение корня n-ой степени $y = \sqrt[n]{x}$ с точностью EPS с использованием итерационной формулы Ньютона:
   $Y_0 = 1$
   $Y_i = 1/n\ ((n-1)Y_{i-1} + X/Y^{n-1}_{i-1})$ (i = 1, 2, 3, ...).
   Вычисления производить пока $|Y_i - Y_{i-1}|$ не станет меньше EPS. Определите количество итераций, за которое достигается эта точность.
5. Введите натуральное число n. Определите количество цифр в этом числе.
6. Вычислите значение числа π с заданной точностью, используя формулу $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots$ Выведите количество слагаемых, которое понадобилось для вычислений.
7. Вычислите значение числа π с заданной точностью, используя формулу $\frac{\pi}{8} = \frac{1}{1 \cdot 3} + \frac{1}{5 \cdot 7} + \frac{1}{9 \cdot 11} + \ldots$
8. Вычислите значение квадратного корня $y = \sqrt{x}$ с точностью EPS с использованием итерационной формулы Ньютона:
   $Y_0 = 1$
   $Y_i = 1/2\ (Y_{i-1} + X/Y_{i-1})$ (i = 1, 2, 3, ...).
   Вычисления производить пока $|Y_i - Y_{i-1}|$ не станет меньше EPS. Определите количество итераций, за которое достигается эта точность.
9. Вычислите значение кубического корня $y = \sqrt[3]{x}$ с точностью EPS с использованием итерационной формулы Ньютона:
   $Y_0 = 1$
   $Y_i = 1/3\ (2Y_{i-1} + X/Y^2_{i-1})$ (i = 1, 2, 3, ...).
   Вычисления производить пока $|Y_i - Y_{i-1}|$ не станет меньше EPS. Определите количество итераций, за которое достигается эта точность.
10. Алгоритм Евклида нахождения НОД(m, n) основан на следующих свойствах этой величины: пусть m и n – два натуральных числа и пусть $m \geq n$. Тогда для чисел m, n и r, где r – остаток от деления m на n, выполняется равенство НОД(m, n) = НОД(n, r). Используя алгоритм Евклида, найдите наибольший общий делитель m и n.

3 Оснащение работы

ПК, среда Visual Studio 2019, MSword.

4 Основные теоретические сведения

По мере развития вычислительной техники возникали различные методики программирования. На каждом этапе создавался новый подход, который позволял программистам справляться с растущим усложнением программ.

Объектно-ориентированное программирование (ООП) аккумулирует лучшие идеи, воплощенные в структурном программировании и сочетает их с новыми мощными концепциями, которые позволяют оптимально организовывать программы.

ООП позволяет разложить проблему на составные части, причем каждая составная становится самостоятельным объектом, содержащим свои коды и данные, которые относятся к этому объекту.

В этом случае вся процедура в целом упрощена, и программист получает возможность оперировать большими по объему программами.

Все языки ООП, включая C#, основаны на трех основополагающих принципах:

- инкапсуляция;
- полиморфизм;
- наследование.

Принципы ООП проще всего понять на примере программ моделирования. В реальном мире каждый предмет или процесс обладает набором статических и динамических характеристик, иными словами, свойствами и поведением. Поведение объекта зависит от его состояния и внешних воздействий. Например, объект «автомобиль» никуда не поедет, если в баке нет бензина, а если повернуть руль, изменится положение колес.

Понятие объекта в программе совпадает с обыденным смыслом этого слова: объект представляется как совокупность данных, характеризующих его состояние, и функций их обработки, моделирующих его поведение. Вызов функции на выполнение часто называют посылкой сообщения объекту.

При создании объектно-ориентированной программы предметная область представляется в виде совокупности объектов. Выполнение программы состоит в том, что объекты обмениваются сообщениями. Это позволяет использовать при программировании понятия, более адекватно отражающие предметную область.

При представлении реального объекта с помощью программного необходимо выделить в первом его существенные особенности. Их список зависит от цели моделирования. Например, объект «крыса» с точки зрения биолога, изучающего миграции, ветеринара или, скажем, повара будет иметь совершенно разные характеристики.

Выделение существенных с той или иной точки зрения свойств называется абстрагированием. Таким образом, программный объект — это абстракция.

Важным свойством объекта является его обособленность. Детали реализации объекта, то есть внутренние структуры данных и алгоритмы их обработки, скрыты от пользователя объекта и недоступны для непреднамеренных изменений. Объект используется через его интерфейс — совокупность правил доступа.

Скрытие деталей реализации называется инкапсуляцией (от слова «капсула»). Ничего сложного в этом понятии нет: ведь и в обычной жизни мы пользуемся объектами через их интерфейсы. Сколько информации пришлось бы держать в голове, если бы для просмотра новостей надо было знать устройство телевизора!

Таким образом, объект является «черным ящиком», замкнутым по отношению к внешнему миру. Это позволяет представить программу в

укрупненном виде — на уровне объектов и их взаимосвязей, а, следовательно, управлять большим объемом информации и успешно отлаживать сложные программы.

Сказанное можно сформулировать более кратко и строго: объект — это инкапсулированная абстракция с четко определенным интерфейсом.

Инкапсуляция позволяет изменить реализацию объекта без модификации основной части программы, если его интерфейс остался прежним. Простота модификации является очень важным критерием качества программы, ведь любой программный продукт в течение своего жизненного цикла претерпевает множество изменений и дополнений.

Кроме того, инкапсуляция позволяет использовать объект в другом окружении и быть уверенным, что он не испортит не принадлежащие ему области памяти, а также создавать библиотеки объектов для применения во многих программах.

Каждый год в мире пишется огромное количество новых программ, и важнейшее значение приобретает возможность многократного использования кода. Преимущество объектно-ориентированного программирования состоит в том, что для объекта можно определить наследников, корректирующих или дополняющих его поведение. При этом нет необходимости не только повторять исходный код родительского объекта, но даже иметь к нему доступ.

Наследование является мощнейшим инструментом ООП и применяется для следующих взаимосвязанных целей:

- исключения из программы повторяющихся фрагментов кода;
- упрощения модификации программы;
- упрощения создания новых программ на основе существующих.

Кроме того, только благодаря наследованию появляется возможность использовать объекты, исходный код которых недоступен, но в которые требуется внести изменения.

Наследование позволяет создавать иерархии объектов. Иерархия представляется в виде дерева, в котором более общие объекты располагаются ближе к корню, а более специализированные — на ветвях и листьях. Наследование облегчает использование библиотек объектов, поскольку программист может взять за основу объекты, разработанные кем-то другим, и создать наследников с требуемыми свойствами.

Объект, на основании которого строится новый объект, называется родительским объектом, объектом-предком, базовым классом, или суперклассом, а унаследованный от него объект — потомком, подклассом, или производным классом.

ООП позволяет писать гибкие, расширяемые и читабельные программы. Во многом это обеспечивается благодаря полиморфизму, под которым понимается возможность во время выполнения программы с помощью одного и того же имени выполнять разные действия или обращаться к объектам разного типа.

Таким образом, основные достоинства ООП заключаются в следующем:

- использование при программировании понятий, близких к предметной области;
- возможность успешно управлять большими объемами исходного кода благодаря инкапсуляции, то есть скрытию деталей реализации объектов и упрощению структуры программы;
- возможность многократного использования кода за счет наследования;
- сравнительно простая возможность модификации программ;
- возможность создания и использования библиотек объектов.

Эти преимущества особенно явно проявляются при разработке программ большого объема и классов программ. Однако ничто не дается даром: создание объектно-ориентированной программы представляет собой весьма непростую задачу, поскольку требует разработки иерархии объектов, а плохо спроектированная иерархия может свести к нулю все преимущества объектно-ориентированного подхода.

Кроме того, идеи ООП не просты для понимания и в особенности для практического применения. Чтобы эффективно использовать готовые объекты из библиотек, необходимо освоить большой объем достаточно сложной информации.

Неграмотное же применение ООП способно привести к созданию излишне сложных программ, которые невозможно отлаживать и усовершенствовать.

Конструкция if/else

Конструкция if/else проверяет истинность некоторого условия и в зависимости от результатов проверки выполняет определенный код:

```
1       int num1 = 8;
2       int num2 = 6;
3       if(num1 > num2)
4       {
5           Console.WriteLine($"Число {num1} больше числа {num2}");
6       }
```

После ключевого слова if ставится условие. И если это условие выполняется, то срабатывает код, который помещен далее в блоке if после фигурных скобок. В качестве условий выступают ранее рассмотренные операции сравнения.

В данном случае у нас первое число больше второго, поэтому выражение num1 > num2 истинно и возвращает true, следовательно, управление переходит к строке Console.WriteLine("Число {num1} больше числа {num2}");

Но что, если мы захотим, чтобы при несоблюдении условия также выполнялись какие-либо действия? В этом случае мы можем добавить блок else:

```
int num1 = 8;
int num2 = 6;
if(num1 > num2)
{
    Console.WriteLine($"Число {num1} больше числа {num2}");
}
else
{
    Console.WriteLine($"Число {num1} меньше числа {num2}");
}
```

Но при сравнении чисел мы можем насчитать три состояния: первое число больше второго, первое число меньше второго и числа равны. Используя конструкцию else if, мы можем обрабатывать дополнительные условия:

```
int num1 = 8;
int num2 = 6;
if(num1 > num2)
{
    Console.WriteLine($"Число {num1} больше числа {num2}");
}
else if (num1 < num2)
{
    Console.WriteLine($"Число {num1} меньше числа {num2}");
}
else
{
    Console.WriteLine("Число num1 равно числу num2");
}
```

Также мы можем соединить сразу несколько условий, используя логические операторы:

```
int num1 = 8;
int num2 = 6;
if(num1 > num2 && num1==8)
{
    Console.WriteLine($"Число {num1} больше числа {num2}");
}
```

В данном случае блок if будет выполняться, если num1 > num2 равно true и num1==8 равно true.

Конструкция switch

Конструкция **switch/case** аналогична конструкции if/else, так как позволяет обработать сразу несколько условий:

```
Console.WriteLine("Нажмите Y или N");
string selection = Console.ReadLine();
switch (selection)
{
    case "Y":
        Console.WriteLine("Вы нажали букву Y");
        break;
    case "N":
        Console.WriteLine("Вы нажали букву N");
        break;
    default:
        Console.WriteLine("Вы нажали неизвестную букву");
        break;
}
```

После ключевого слова **switch** в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора **case**. И если совпадение будет найдено, то будет выполняться определенный блок **case**.

В конце каждого блока case должен ставиться один из операторов перехода: **break**, **goto case**, **return** или **throw**. Как правило, используется оператор **break**. При его применении другие блоки case выполняться не будут.

Однако если мы хотим, чтобы, наоборот, после выполнения текущего блока case выполнялся другой блок case, то мы можем использовать вместо break оператор goto case:

```
int number = 1;
switch (number)
{
    case 1:
        Console.WriteLine("case 1");
        goto case 5; // переход к case 5
    case 3:
        Console.WriteLine("case 3");
        break;
    case 5:
        Console.WriteLine("case 5");
        break;
    default:
        Console.WriteLine("default");
        break;
}
```

Если мы хотим также обработать ситуацию, когда совпадения не будет найдено, то можно добавить блок **default**, как в примере выше.

Применение оператора return позволит выйти не только из блока case, но и из вызывающего метода. То есть, если в методе Main после конструкции switch..case, в которой используется оператор return, идут какие-либо операторы и выражения, то они выполняться не будут, а метод Main завершит работу.

Оператор throw применяется для выброса ошибок и будет рассмотрен в одной из следующим тем.

Тернарная операция

Тернарную операция имеет следующий синтаксис: [первый операнд - условие] ? [второй операнд] : [третий операнд]. Здесь сразу три операнда. В зависимости от условия тернарная операция возвращает второй или третий операнд: если условие равно true, то возвращается второй операнд; если условие равно false, то третий. Например:

```
1	int x=3;
2	int y=2;
3	Console.WriteLine("Нажмите + или -");
4	string selection = Console.ReadLine();
5
6	int z = selection=="+"? (x+y) : (x-y);
7	Console.WriteLine(z);
```

Здесь результатом тернарной операции является переменная z. Если мы выше вводим "+", то z будет равно второму операнду - (x+y). Иначе z будет равно третьему операнду.

Циклы являются управляющими конструкциями, позволяя в зависимости от определенных условий выполнять некоторое действие множество раз. В C# имеются следующие виды циклов:

- for
- foreach
- while
- do...while

Цикл for

Цикл for имеет следующее формальное определение:

```
1	for ([инициализация счетчика]; [условие]; [изменение счетчика])
2	{
3	    // действия
4	}
```

Рассмотрим стандартный цикл for:

```
1	for (int i = 0; i < 9; i++)
2	{
3	    Console.WriteLine($"Квадрат числа {i} равен {i*i}");
4	}
```

Первая часть объявления цикла - int i = 0 - создает и инициализирует счетчик i. Счетчик необязательно должен представлять тип **int**. Это может быть и другой числовой тип, например, float. И перед выполнением цикла его значение будет равно 0. В данном случае это то же самое, что и объявление переменной.

Вторая часть - условие, при котором будет выполняться цикл. Пока условное выражение возвращает true, будет выполняться цикл. В данном случае цикл будет выполняться, пока счетчик i не достигнет 9.

И третья часть - приращение счетчика на единицу. Опять же нам необязательно увеличивать на единицу. Можно уменьшать: i--.

В итоге блок цикла сработает 9 раз, пока значение i не станет равным 9. И каждый раз это значение будет увеличиваться на 1.

Нам необязательно указывать все условия при объявлении цикла. Например, мы можем написать так:

```
1       int i = 0;
2       for (; ;)
3       {
4           Console.WriteLine($"Квадрат числа {++i} равен {i * i}");
5       }
```

Формально определение цикла осталось тем же, только теперь блоки в определении у нас пустые: for (; i <;). У нас нет инициализированной переменной-счетчика, нет условия, поэтому цикл будет работать вечно - бесконечный цикл.

Мы также можем опустить ряд блоков:

```
1       int i = 0;
2       for (; i<9;)
3       {
4           Console.WriteLine($"Квадрат числа {++i} равен {i * i}");
5       }
```

Этот пример по сути эквивалентен первому примеру: у нас также есть счетчик, только создан он вне цикла. У нас есть условие выполнения цикла. И есть приращение счетчика уже в самом блоке for.

Цикл do

В цикле do сначала выполняется код цикла, а потом происходит проверка условия в инструкции while. И пока это условие истинно, цикл повторяется. Например:

```
1       int i = 6;
2       do
3       {
4           Console.WriteLine(i);
5           i--;
6       }
7       while (i > 0);
```

Здесь код цикла сработает 6 раз, пока i не станет равным нулю. Но важно отметить, что цикл do гарантирует хотя бы единократное выполнение действий, даже если условие в инструкции while не будет истинно. То есть мы можем написать:

```
1       int i = -1;
2       do
3       {
4           Console.WriteLine(i);
5           i--;
6       }
7       while (i > 0);
```

Хотя у нас переменная i меньше 0, цикл все равно один раз выполнится.

Цикл while

В отличие от цикла do цикл **while** сразу проверяет истинность некоторого условия, и если условие истинно, то код цикла выполняется:

```
1       int i = 6;
2       while (i > 0)
3       {
4           Console.WriteLine(i);
5           i--;
6       }
```

Операторы continue и break

Иногда возникает ситуация, когда требуется выйти из цикла, не дожидаясь его завершения. В этом случае мы можем воспользоваться оператором **break**.

Например:

```
1       for (int i = 0; i < 9; i++)
2       {
3           if (i == 5)
4               break;
5           Console.WriteLine(i);
6       }
```

Хотя в условии цикла сказано, что цикл будет выполняться, пока счетчик i не достигнет значения 9, в реальности цикл сработает 5 раз. Так как при достижении счетчиком i значения 5, сработает оператор break, и цикл завершится.

Теперь поставим себе другую задачу. А что если мы хотим, чтобы при проверке цикл не завершался, а просто пропускал текущую итерацию. Для этого мы можем воспользоваться оператором **continue**:

```
1       for (int i = 0; i < 9; i++)
2       {
3           if (i == 5)
4               continue;
```

```
5           Console.WriteLine(i);
6       }
```

В этом случае цикл, когда дойдет до числа 5, которое не удовлетворяет условию проверки, просто пропустит это число и перейдет к следующей итерации.

Цикл foreach

Цикл foreach перебирает коллекции, например, массивы, и будет рассмотрен далее в теме массивов.

5. Порядок выполнения работы

1. Выделить ключевые моменты задачи.
2. Построить алгоритм и теоретическую объектную модель решения задачи.
3. Запрограммировать полученные алгоритмы и объектную модель.

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы: _____

Результат выполнения работы: _____

_____

_____

7. Контрольные вопросы и задания

1. Принципы ООП?
2. Что такое класс?
3. Что такое «Инкапсуляция»?
4. Основные достоинства ООП?

8. Рекомендуемая литература

1. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. СПб. : Изд-во Питер, 2021. 896 с.

2. Прайс, М. Дж. C# 10 и .NET 6. Современная кросс-платформенная разработка / М. Дж. Прайс. СПб : Изд-во Питер, 2023. 848 с.

3. Васильев, А.Н. Программирование на C# для начинающих. Особенности языка / А.Н. Васильев. М. : Эксмо, 2022. 528 с.

4. Фримен, А. ASP.NET Core 3 с примерами на C# для профессионалов / А. Фримен. СПб. : Изд-во Вильямс, 2021. 1184 с.