**Лабораторная работа № 2.**

**Разветвляющиеся алгоритмы. Оператор if**

Цель работы: изучение приемов программирования алгоритмов обработки данных с ветвлениями, выработка умений отладки и тестирования программ с условными операторами перехода, закрепление навыков программирования арифметических выражений.

**Порядок выполнения лабораторных работ**

* Прочитать теоретический материал
* Разобрать, понять определения по теме
* Выполнить задания в соответствии с описанием
* Написать отчет по лабораторной работе
* Защитить отчет

**Определения**

***Оператор*** - это конструкция языка программирования (предложение), предписывающая компьютеру выполнить некоторую строго определенную последовательность действий (операций). В языке C# любое выражение, завершающееся точкой с запятой, рассматривается как оператор.

Примеры операторов: i++; a=b+c; while(true);

***Блок операторов***, или составной оператор – последовательность описаний и операторов, заключенных в фигурные скобки. Блок рассматривается компилятором как один оператор и может использоваться всюду, где синтаксис требует одного оператора, а алгоритм нескольких.

Оператор ***if–else – это…***

Оператор ***switсh – это…***

***Тернарный оператор – это…***

***Логический тип данных – это***

**Теоретический материал: Операторы ветвления**

Операторы ветвления применяются для того, чтобы в зависимости от конкретных значений исходных данных обеспечить выполнение различных последовательностей операторов. В языке C# это операторы ***if, ?*** и ***switch***.

**Оператор if**

Синтаксис оператора if:

1. if (логическое\_выражение) оператор\_1;
2. if (логическое\_выражение) оператор\_1; else оператор\_2;
3. if (логическое\_выражение) оператор\_1; else if оператор\_2; else оператор\_3;

Примеры.

1. if (a>0) b=1;

2. if (a<b && (a>d || a==0))

b++;

else

{b\*=a;

  a=0;}

3. if (a<b)

if (a<c) m=a;

else m=c;

else m=b;

**Тернарный оператор ?**

Синтаксис оператора

Условие **?** Выражение1 : Выражение2

где

* Условие – условное выражение, которое может принимать одно из двух значений: ***true*** или ***false***;
* Выражение1 – некоторое выражение, которое нужно выполнить в случае, если значение Условие = ***true***;
* Выражение2 – выражение, которое нужно выполнить если Условие не выполняется (Условие = ***false***).

Пример: max = x > y ? x : y;

**Логический тип данных (bool)**

Тип ***bool*** – логический тип данных, принимающий значения ***true*** (истина) или ***false*** (ложь). Эти значения могут принимать константы, переменные или выражения.

В C# используются следующие знаки операций отношения и проверки на равенство: **<; <=; >; >=; ==; !=**. Операции **==; !=** имеют более высокий приоритет, чем остальные.

Логические операторы:

& – конъюнкция, или логическое И;

| – дизъюнкция, или логическое ИЛИ;

! – отрицание, или логическое НЕ;

^ – логическое Исключающее ИЛИ.

Используются также операторы &&; ||, которые отличаются от своих односимвольных версий тем, что выполняют ускоренные вычисления. Например, в выражении a && b правый операнд вычисляется лишь в том случае, когда a=true. В выражении a || b, b вычисляется в том случае, если a=false.

Приоритет операторов:

* оператор логического отрицания **!**
* оператор логического **И** **&**
* оператор логического исключающего **ИЛИ ^**
* оператор логического **ИЛИ |**
* условный оператор логического **И &&**
* условный оператор логического **ИЛИ ||**

**Индивидуальное задание**

Для выполнения задания необходимо использовать оператор if-else.

Разработать алгоритм вычисления выражения по заданной формуле для вводимых значений переменной . Алгоритм представить в виде схемы алгоритма и программы для ЭВМ на языке программирования С#.

В программе каждую ветвь необходимо пронумеровать. Перед выводом результата вычислений организовать вывод на экран номера той ветви, по которой производятся вычисления с введенными исходными данными.

Таблица 1 – Варианты задания № 1

|  |  |
| --- | --- |
| Вариант | Формула |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21 |  |
| 22 |  |
| 23 |  |
| 24 |  |
| 25 |  |
| 26 |  |
| 27 |  |
| 28 |  |
| 29 |  |
| 30 |  |
| 31 |  |
| 32 |  |
| 33 |  |
| 34 |  |
| 35 |  |
| 36 |  |
| 37 |  |
| 38 |  |
| 39 |  |