**Лабораторная работа № 3.**

**Разветвляющиеся алгоритмы. Оператор switch**

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;

**Оператор switch**

Оператор предназначен для реализации множества альтернатив.

Синтаксис оператора **switch (переключатель)**:

***switch*** (выражение)

{

***сase*** <константное\_выражение\_1>: [<список\_операторов\_1>]

***сase*** <константное\_выражение\_2>: [<список\_операторов\_2>]

…

***сase*** <константное\_выражение\_n>: [<список\_операторов\_n>]

[***default***: <операторы>]

}

Выполнение оператора начинается с вычисления выражения. Тип выражения может быть различным, чаще всего либо целочисленный, либо символьный, либо строковый. Затем управление передается первому оператору из списка, помеченное константным выражением, значение которого совпадает с вычисленным. Каждая ветвь переключателя должна заканчиваться оператором перехода: ***break***; ***goto***; ***return***.

Оператор перехода ***break*** выполняет выход из самого внутреннего из объемлющих его операторов переключателя и операторов цикла.

Оператор перехода ***goto*** выполняет переход на указанную после него метку, обычно это метка ветви case одной из низлежащих ветвей переключателя. ***Этот оператор не рекомендуется использовать!!!***

Оператор перехода ***return*** выполняет выход из метода, в теле которого он записан.

Ветвь ***default*** реализуется тогда, когда значение выражения не совпадает ни с одним из константных выражений. Ее рекомендуется использовать в целях выявления возможных ошибок.

**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**

Выполнить индивидуальные задания в соответствии с таблицей. Задания взять из прилагаемого Сборника задач по программированию Д. М. Златопольского (файл «Златопольский Д. М. Сборник задач по программированию 2011.pdf»).

Для выполнения задания использовать оператор switch (и, если требуется, if).

Таблица 2 – Варианты задания № 2

|  |  |
| --- | --- |
| Вариант | Номер задачи из Сборника Д. М. Златопольского |
| 1 | 4.107 (2) |
| 2 | 4.111 (а) |
| 3 | 4.106 |
| 4 | 4.107 (1) |
| 5 | 4.109 |
| 6 | 4.112 |
| 7 | 4.110 |
| 8 | 4.107 (2) |
| 9 | 4.111 (а) |
| 10 | 4.106 |
| 11 | 4.107 (1) |
| 12 | 4.109 |
| 13 | 4.112 |
| 14 | 4.110 |
| 15 | 4.107 (2) |
| 16 | 4.111 (а) |
| 17 | 4.106 |
| 18 | 4.107 (1) |
| 19 | 4.109 |
| 20 | 4.112 |
| 21 | 4.110 |
| 22 | 4.107 (2) |
| 23 | 4.111 (а) |
| 24 | 4.106 |
| 25 | 4.107 (1) |
| 26 | 4.109 |
| 27 | 4.112 |
| 28 | 4.110 |
| 29 | 4.107 (2) |
| 30 | 4.111 (а) |
| 31 | 4.106 |
| 32 | 4.107 (1) |
| 33 | 4.109 |
| 34 | 4.112 |
| 35 | 4.110 |
| 36 | 4.107 (2) |
| 37 | 4.111 (а) |
| 38 | 4.106 |
| 39 | 4.107 (1) |