ГБПОУ г. Москвы
«Первый московский образовательный комплекс»

Объектно-ориентированное программирование

Лекция 6. Типы данных, выражения, операторы

План лекции

- Выражения
- Основные операторы языка
  - Условные
  - Цикла
- Обработка исключений
- Отладка и трассировка работы программы

3. Выражения

Выражения

- В языках программирования под *выражением* понимается комбинация констант, переменных, вызовов методов (возвращающих значения), объединенных знаками операций и скобками, которая в результате вычисления, возвращает некоторый результат.
- Например:

```
(a+1)/2.55 // арифметическое выражение
a == 5       // логическое выражение
```

- При вычислении *выражения* определяется его значение и тип.
- Тип результата выражения зависит от типов переменных и констант участвующих в выражении.

Тип результата выражения

- Тип результата выражения зависит от типов переменных и констант участвующих в выражении и выполняемых операций.
- Типом результата выражения определяется типом результата операций, входящих в выражение.
- Например, результатом выполнения следующего выражения:

```
int a = 5;
float f;
f = a/4;        // значение f = 1.0, так как a/4 = 1 (int)
f = a/4f;       // значение f = 1.25 (float)
```

- Промежуточный тип операций может отличаться от типа выражения:

```
double aaa = 2 / 4 * 0.5;     // = 0.0, так как 2/4 = 0 (int)
double bbb = 0.5 * 2 / 4;     // 0.25, так как 0.5 * 2 = 1.0 (double)
```

4. Операторы языка программирования

Операторы языка программирования

- ***Операторы*** являются ***инструкциями*** языка программирования, которые представляет собой законченные фразы и определяют некоторый вполне законченные этапы обработки данных.
- В состав операторов входят ключевые слова, переменные, константы, операции и выражения.
- Каждый оператор заканчивается символом “;”.
- В одно строке программы может быть записано несколько операторов и один

Оператор присваивания

- В языке C# *присваивание* считается операцией.
- Вместе с тем запись вида: x = expr; можно считать настоящим *оператором присваивания*, так же, как и одновременное *присваивание* со списком переменных в левой части:

  x1 = x2 = ... = xk = expr;
- В качестве выражения expr может выступать просто переменная или константа.
- Присваивание возможно только в том случае, если есть соответствие типов:
  - Совпадают
  - Существует неявное преобразование
  - Задано явное преобразование (кастинг)

Результат присваивания

- Для значащих типов создается копия данных.
- Например:

```
int x=5;
int z;
z = x; // данные копируются (5)
```

- Для ссылочных типов создается копия ссылки на тот самый объект (новый объект не создается).
- Например:

```
Box p1;
p1 = new Box (4,5);  // создаем объект в куче
Box p2;
p2 = p1; // копируется ссылка на ранее созданный объект в куче
```

Операторы выбора

- Для выбора одной из нескольких возможностей используются два оператора *if* и *switch*.
  - *if* - альтернативный выбор,
  - *switch* – выбор из набора вариантов.

Оператор if

- Синтаксис *оператора if*:

  if(выражение_1) оператор_1
  else if(выражение_2) оператор_2
  …
  else if(выражение_K) оператор_K
  else оператор_N

- Выражения *if* должны заключаться в круглые скобки и быть булевого типа. Точнее, выражения должны давать значения true или false.
- Арифметический тип не имеет явных или неявных преобразований к булевому типу.
- Ветви else и *if*, позволяющие организовать выбор из многих возможностей, могут отсутствовать. Может быть опущена и заключительная else-ветвь. В этом случае краткая форма *оператора if* задает альтернативный выбор – делать или не делать – выполнять или не выполнять then-оператор.
- Выражения в *if* проверяются в порядке их написания. Как только получено значение true, проверка прекращается и выполняется оператор (это может быть *блок*), который следует за выражением, получившим значение true. С завершением этого оператора завершается и *оператор if*.

Оператор switch

- Частным, но важным случаем выбора из нескольких вариантов является ситуация, при которой выбор варианта определяется значениями некоторого выражения.
- Соответствующий оператор C# называется *оператором switch*:

```
switch(выражение)
{
  case константное_выражение_1:
   [операторы_1 оператор_перехода_1]
  ...
  case константное_выражение_K:
   [операторы_K оператор_перехода_K]
  [default: операторы_N оператор_перехода_N]
}
```

- *Оператор switch* работает следующим образом.
  - Вначале вычисляется значение switch-выражения.
  - Затем оно поочередно в порядке следования case сравнивается на совпадение с константными выражениями. Как только достигнуто совпадение, выполняется соответствующая последовательность операторов case-ветви.
  - Поскольку последний оператор этой последовательности является *оператором перехода* (чаще всего это оператор break), то обычно он завершает выполнение *оператора switch*.
  - Если значение switch-выражения не совпадает ни с одним константным выражением, то выполняется последовательность операторов ветви default, если же таковой ветви нет, то *оператор switch* эквивалентен *пустому оператору*.
- Ветвь default может отсутствовать.
- Синтаксически допустимо, чтобы после двоеточия следовала пустая последовательность операторов, а не последовательность, заканчивающаяся *оператором перехода*.
- case-выражения могут быть только константным выражениями. Константные выражения в case должны иметь тот же тип, что и *switch*-выражение.

Пример использования оператора switch

```
public void Starosta(string group)
{
  string stud;
  switch (group)
  {
    case "8551":
      stud = "Иванов С.П."; break;
    case "8552":
      stud = "Сидоров А.И."; break;
    case "8553":
      stud = "Петров В.Т."; break;
    default :
      stud = "не определен"; break;
  }

  Console.WriteLine ("Староста группы {0} – {1}",  group, stud);
}
```

Выбор диапазона значений

- Когда требуется проверить попадание в некоторый диапазон значений, приходится прибегать к *оператору if* для определения значения специальной переменной.
- Например:

```
string period ;
if ((age > 0)&& (age <7))period=“ребенок”;
else if ((age >= 7)&& (age <17))period=“подросток”;
else if ((age >= 17)&& (age <22))period=“юноша”;
else period =“взрослый”;
```

Операторы перехода

- *Операторы перехода*, позволяют прервать естественный порядок выполнения операторов *блока*.

Оператор goto

- Оператор goto имеет следующий вид:

  goto [метка|case_выражение|default];

- Все операторы языка C# могут иметь метку – уникальный идентификатор, предшествующий оператору и отделенный от него символом двоеточия:

  <метка> : <оператор>;

- Передача управления помеченному оператору – это классическое использование *оператора goto*.
- Два других способа использования *goto* (передача управления в case или default-ветвь) используются в *операторе switch*.

Операторы break и continue

- В структурном программировании признаются полезными "переходы вперед" (но не назад), позволяющие при выполнении некоторого условия выйти из цикла, из оператора выбора, из блока.
- Для этой цели можно использовать оператор goto, но лучше применять специально предназначенные для этих целей операторы break и continue.

- Оператор break может стоять в теле цикла или завершать case-ветвь в *операторе switch*.
- При выполнении оператора break в теле цикла завершается выполнение самого внутреннего цикла.
- В теле цикла, чаще всего, оператор break помещается в одну из ветвей *оператора if*, проверяющего условие преждевременного завершения цикла.
- Например:

```
int i = 1, j=1;
for(i =1; i<100; i++){
  for(j = 1; j<10; j++) {if (j>=3) break;}
  Console.WriteLine(
      "Выход из цикла j при j = {0}", j);
  if (i>=3) break;
}
Console.WriteLine("Выход из цикла i при i= {0}", i);
```

- Оператор continue используется только в теле цикла.
- В отличие от оператора, break завершающего внутренний цикл, continue осуществляет переход к следующей итерации этого цикла.
- Например, сложить все четные суммы чисел от 1 до 100:

```
int s=0;
for(i =1; i<100; i++){
  for(j = 1; j<100; j++) {if ((i+j)%2 == 1) continue; s += (i+j);}
}
Console.WriteLine(“Сумма четных величин s= {0}", s);
```

Операторы цикла

Операторы цикла позволяют выполнить участок программы требуемое число раз

- ***Цикл for***
- ***Циклы while***
- ***Цикл foreach***

Цикл for

- *Оператор цикла for* имеет следующий вид:

```
for(инициализации; условие; изменение) оператор;
```

- Оператор, стоящий после закрывающей скобки, задает тело цикла. В большинстве случаев телом цикла является *блок*. Сколько раз будет выполняться тело цикла, зависит от трех управляющих элементов, заданных в скобках.
  - ***Инициализация*** задает начальное значение одной или нескольких переменных, часто называемых переменными цикла. В большинстве случаев цикл *for* имеет одну переменную.
  - ***Условие*** задает условие окончания цикла, соответствующее выражение при вычислении должно получать значение true или false.
  - ***Изменение*** описывает, как меняется переменная цикла в каждой итерации выполнения. Если условие цикла истинно, то выполняется тело цикла, затем изменяются значения переменной цикла и снова проверяется условие. Как только условие становится ложным, цикл завершает свою работу.

- Например, для вычисления значений целых чисел от 1 до 10 можно использовать следующий цикл:

```
int s=0;
for (int i = 1; i <= 10; i++) s += i;
```

- Переменная цикла часто объявляется непосредственно в инициализации и соответственно являются локальной в цикле переменной, так что после завершения цикла она перестанет существовать.
- В тех случаях, когда предусматривается возможность преждевременного завершения цикла с помощью одного из *операторов перехода*, переменные цикла объявляются до цикла, что позволяет анализировать их значения при выходе из цикла.

Циклы while

- ***Цикл*** `while(условие)` является универсальным видом цикла, включаемым во все языки программирования.
- Тело цикла выполняется до тех пор, пока остается истинным условие оператора while.
- В языке C# у этого вида цикла есть два варианта – с проверкой условия в начале и в конце цикла.
- Первый вариант имеет следующий вид:

  ```
  while(логическое выражение) оператор;
  ```

- Этот оператор сначала проверяет условие, а затем выполняет тело цикла. Если в результате проверки условие не выполнится, то тело может быть ни разу не выполнено.
- В нормальной ситуации каждое выполнение тела цикла – это очередной шаг к завершению цикла.

Циклы do-while

- Вариант цикла do-while, проверяет условие завершения в конце очередной итерации. Тело такого цикла выполняется, в крайнем случае, мере, один раз. Такой цикл записывается следующим образом:

```
do
    оператор
while(выражение);
```

Пример цикла do-while

```
string answer;
double x, y;
do
{
  Console.Write("Введите значение:");
  x = (Convert.ToDouble(Console.ReadLine()));
  Console.Write("Возвести в:\n1. 2 степень\n2. 3 степень\n");
  int i = Convert.ToInt32(Console.ReadLine());
  y = x;
  switch (i)
  {
    case 1:
      y = Math.Pow(x, 2.0);
      break;
    case 2:
      y = Math.Pow(x, 3.0);
      break;
  }
  Console.WriteLine("Результат: {0}", y);
  Console.WriteLine("Продолжить? (да/нет)");
  answer = Console.ReadLine();
}
while (answer == "да");
```