Лекция 7. Процедуры и функции

Козьминых Н.М.

дисциплина «Программирование»

Определение

- Процедуры и функции позволяют:
  - задать определенную функциональность,
  - многократно выполнять один и тот же параметризированный программный код при различных значениях параметров.
- В C# процедуры и функции существуют только как методы некоторого класса, они не существуют вне класса.

Функции

- Функция отличается от процедуры двумя особенностями:
  - Она всегда вычисляет некоторое значение, возвращаемое в качестве результата функции;
  - И вызывается в выражениях.

Процедуры

- Процедура C# имеет свои особенности:
  - Она возвращает формальный результат **void**, указывающий на отсутствие результата;
  - Вызов процедуры является оператором языка;
  - И она имеет входные и выходные аргументы, причем выходных аргументов - ее результатов - может быть достаточно много.

Функции и процедуры (методы)

- Обычно метод предпочитают реализовать в виде функции тогда, когда он имеет один выходной аргумент, рассматриваемый как результат вычисления значения функции.
- Возможность вызова функций в выражениях также влияет на выбор в пользу реализации метода в виде функции.
- В других случаях метод реализуют в виде процедуры.

Метод

- Синтаксически в описании метода различают две части:
  - **заголовок_метода**
  - **тело_метода**

Заголовок метода

- Синтаксис:

[атрибуты][модификаторы]{void | тип_рез_функции}

имя_метода

([список_формальных_аргументов])

- Имя метода и список формальных аргументов составляют сигнатуру метода.
- В сигнатуру не входят имена формальных аргументов - здесь важны типы аргументов.
- В сигнатуру не входит тип возвращаемого результата.

Модификатор

- Модификатор доступа может иметь четыре возможных значения, рассмотрим только два - **public** и **private.**
- Модификатор **public** показывает, что метод открыт и доступен для вызова клиентами и потомками класса.
- Модификатор **private** говорит, что метод предназначен для внутреннего использования в классе и доступен для вызова только в теле методов самого класса.
- Если модификатор доступа опущен, то по умолчанию предполагается, что он имеет значение **private.**

Тип результата

- Тип результата метода указывается всегда, но значение **void** однозначно определяет, что метод реализуется процедурой.
- Тип результата, отличный от **void**, указывает на функцию.

```
void A() {...};
int B(){...};
public void C(){...};
```

- Методы **A** и **B** являются закрытыми, а метод **C** - открыт.
- Методы **A** и **C** реализованы процедурами, а метод B - функцией, возвращающей целое значение.

Список формальных аргументов

- Список формальных аргументов метода может быть пустым.
- Список может содержать фиксированное число аргументов, разделяемых символом запятой.
- Синтаксис:

[ref | out | params] тип_аргумента имя_аргумента

- Обязательным является указание типа и имени аргумента.

Список формальных аргументов

- **params -** ключевое слово, позволяющее передавать методу произвольное число фактических аргументов.
- Аргументы метода разделяются на три группы:
  - входные,
  - выходные (**out**),
  - обновляемые (**ref**).

Список формальных аргументов

- **Входные** аргументы передают информацию методу, их значения в теле метода только читаются.
- **Выходные** (**out**) представляют собой результаты метода, они получают значения в ходе работы метода.
- **Обновляемые** (**ref**) выполняют обе функции:
  - их значения используются в ходе вычислений;
  - обновляются в результате работы метода.

Тело метода

- Тело метода является блоком, который представляет собой последовательность операторов и описаний переменных, заключенную в фигурные скобки.
- Если это тело функции, то в блоке должен быть хотя бы один оператор перехода, возвращающий значение функции в форме **return (выражение).**

Пример


```
// Процедура с входным и выходным параметрами
private void A(out double p2, double p1) {
     p2 = Math.Pow(p1, 3);
}
// Процедура с обновляемым параметром
private void B(ref int a) {
     a++;
}
// Функция
private int C(int a) {
     return ++a;
}
```

Вызов метода

- Синтаксис:

```
имя_метода( [список_фактических_аргументов] )
```

- Формальный аргумент задается при определении метода - это всегда имя аргумента (идентификатор).
- Фактический аргумент – это выражение, используемое при вызове метода.

Вызов метода

- Между списком формальных и списком фактических аргументов должно выполняться определенное соответствие по:
  - числу,
  - порядку следования,
  - типу,
  - статусу аргументов.
- Если формальный аргумент объявлен с ключевым словом **ref** или **out**, то фактический аргумент должен сопровождаться таким же ключевым словом при вызове метода.

Пример


```
// Определение метода
private void A(out double p2, double p1){
     p2 = Math.Pow(p1, 3);
}


// Вызов метода
 private void btn_Click(object sender, EventArgs
  e){
       double p, s = 2;
       A(out p, s);
       btn.Text = p.ToString();
 }
// Ответ: 8
```

Пример


```
// Определение метода
private void B(ref int a){
     a++;
}
// Вызов метода
private void btn_Click(object sender, EventArgs e){
     int p = 10;
     B(ref p);
     btn.Text = p.ToString();
}
// Ответ: 11
```

Пример


```
// Определение метода
private int C(int a){
    return ++a;
}
// Вызов метода
private void btn_Click(object sender, EventArgs
  e){
    int p = 4;
    btn.Text = C(p).ToString();
}
// Ответ: 5
```

Спасибо за внимание

Вопросы…