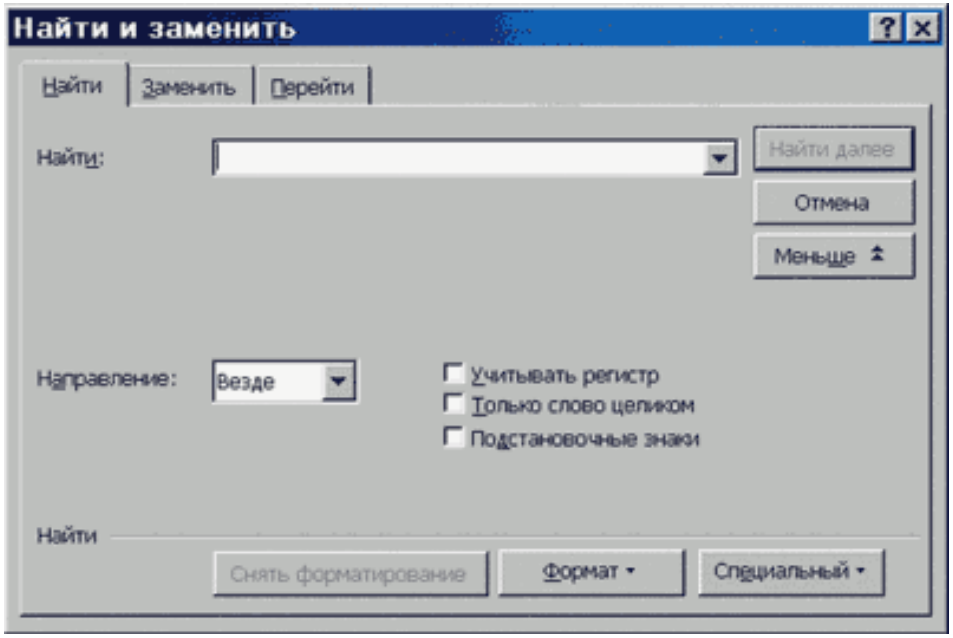
Механизм обратного вызова

При обратном вызове программист задает действия, которые должны выполняться всякий раз, когда происходит некоторое событие.

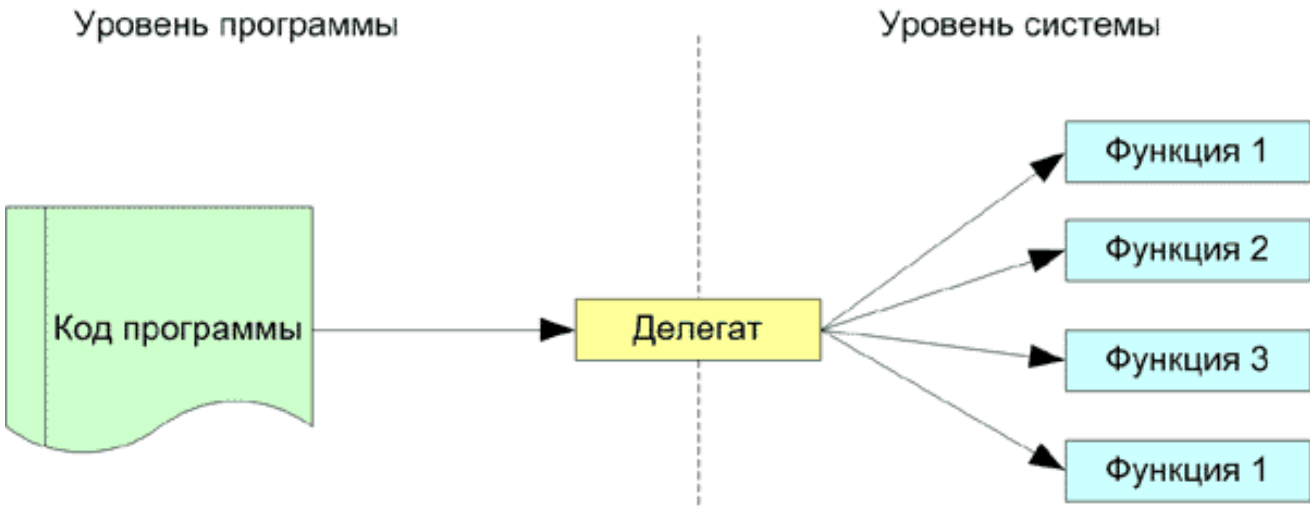


Мы часто задаем действие, которое должно быть выполнено после щелчка на некоторой кнопке или при выборе определенного пункта меню.

Callback (**функция обратного вызова**) — передача исполняемого кода в качестве одного из параметров другого кода.

Обратный вызов позволяет в функции исполнять код, который задаётся в аргументах при её вызове.

Делегат представляет собой объект, который может вызвать метод. Следовательно, при существовании делегата, будет возможность получить ссылку на метод.



Сигнатура метода — это «шапка» какого-л. метода или методов. Представляет собой сочетание названия типа, который метод возвращает, плюс название типов входящих параметров (по порядку! порядок очень важен., так же как и слова ref и out, которые также входят в понятие сигнатуры).

Например,

метод *int NewMethod(int x, char y)* будет иметь сигнатуру **int (int, char)**

метод *void NewMethod()* — **void (void)**.

Тип делегата объявляется с помощью ключевого слова delegate:

delegate *возвращаемый_тип* **имя** (*список_параметров*) ;

,где

возвращаемый_тип - обозначает тип значения, возвращаемого методами, которые будут вызываться делегатом;

имя — конкретное имя делегата;

список_параметров — параметры, необходимые для методов, вызываемых делегатом. Как только будет создан экземпляр делегата, он может вызывать и ссылаться на те методы, возвращаемый тип и параметры которых соответствуют указанным в объявлении делегата.

```
static void Notify (string name)
{ Console.WriteLine("Notification received for: {0}", name); }
```

```
delegate void Del (string);
```

Экземпляр делегата — это объект, реализующий класс типа делегата.

```
Del refFunc = new Del(Notify);
```

```
delegate string StrMod(string); // Объявили тип делегата

class DelegateTest
   {
static string ReplaceSpaces(string s) // Заменить пробелы дефисами.
      {
         Console.WriteLine("Замена пробелов дефисами."); return s.Replace(' ', '-');
      }
static string RemoveSpaces(string s) // Удалить пробелы.
      {
         string temp = ""; int i;
         Console.WriteLine("Удаление пробелов.");
         for (i = 0; i < s.Length; i++)
            if (s[i] != ' ') temp += s[i];
         return temp;
      }
static string Reverse(string s) // Обратить строку.
      {
         string temp = ""; int i, j;
 Console.WriteLine("Обращение строки. ");
for (j = 0, i = s.Length - 1; i >= 0; i--, j++) temp += s[i];
         return temp;
      }
```

Групповое преобразование методов

```
static void Main(string[] args)
 {
            string str = "Мама мыла раму";
// Сконструировали делегат, используя групповое преобразование методов.
   StrMod strOp = ReplaceSpaces;
// Вызвали метод ReplaceSpaces с помощью делегата
   Console.WriteLine("Результирующая строка: " + strOp(str) + "\n");

// перенастроили ссылку, теперь делегат указывает на RemoveSpaces
        strOp = RemoveSpaces;
        Console.WriteLine("Результирующая строка: " + strOp(str) + "\n");

 // перенастроили ссылку, теперь делегат указывает на Reverse
            strOp = Reverse;
           Console.WriteLine("Результирующая строка: " + strOp(str) );
  }
```

```
Замена пробелов дефисами.
Результирующая строка: Мама-мыла-раму

Удаление пробелов.
Результирующая строка: Мамамылараму

Обращение строки.
Результирующая строка: умар алым амаМ
```

Групповая адресация

delegate void StrMod(ref string);

```
 class DelegateTest
{
   public static void ReplaceSpaces(ref string s)
   {
Console.WriteLine("Замена пробелов дефисами."); s = s.Replace(' ', '-');
   }
   public static void RemoveSpaces(ref string s)
   {
      string temp = ""; int i;
Console.WriteLine("Удаление пробелов."); for (i = 0; i < s.Length; i++) if (s[i] != ' ')
temp += s[i];
      s = temp;
   }
  public static void Reverse(ref string s)
   {
      string temp = ""; int i, j;
Console.WriteLine("Обращение строки."); for (j = 0, i = s.Length - 1; i >= 0; i--, j++)
temp += s[i];
      s = temp;
   }  }
```

```
static void Main(string[] args)            {
                        StrMod strOp; // Объявление экземпляра делегата

string str = "Аргентина манит негра"; Console.WriteLine("Исходная строка: " + str);
      strOp = Reverse;// Настроили делегат на метод переворачивания
            strOp(ref str); // Обратиться к делегату с групповой адресацией
            Console.WriteLine("Результиру

     strOp += RemoveSpaces;    // добавит
            strOp(ref str); // Обратиться к
            Console.WriteLine("Результиру

    str = "Аргентина манит негра";
    strOp += ReplaceSpaces; // добавить м
            strOp(ref str);  // Обратиться к делегату с групповой адресацией.
            Console.WriteLine("Результирующая строка: " + str);

   strOp -= RemoveSpaces; // Удалить метод удаления пробелов.
   strOp -= Reverse; // Удалить метод метод переворачивания
         str = "Аргентина манит не гра";
            strOp(ref str);  // Обратиться к делегату с групповой адресацией.
            Console.WriteLine("Результирующая строка: " + str);
```

```
Исходная строка: Аргентина манит негра
                            Обращение строки.
Результирующая строка: арген тинам анитнегрА
                            Обращение строки.
                            Удаление пробелов.
Результирующая строка: Аргентинаманитнегра
Восстановленная исходная строка: Аргентина манит негра
                            Обращение строки.
                            Удаление пробелов.
                            Замена пробелов дефисами.
Результирующая строка: аргентинаманитнегрА
Восстановленная исходная строка: Аргентина манит негра
                            Замена пробелов дефисами.
Результирующая строка: Аргентина-манит-негра
```

Назначение делегатов.
События

Событие — это экземпляр делегата, при вызове которого, будут запущены все методы, подписавшиеся на момент вызова этого события .

События являются членами класса и объявляются с помощью ключевого слова event.

event *делегат_события* ***имя_события****;*

, где *делегат_события* обозначает имя делегата, используемого для поддержки события

, а *имя_события* — конкретный объект объявляемого события.

Назначение: ***Делегаты поддерживают события, а события запускают механизм обратного вызова***

Алгоритм обработки собственных событий:

1. Определите *условие возникновения* события и методы, которые должны сработать.

2. Определите сигнатуру этих методов и создайте *делегат* на основе этой сигнатуры.

3. Создайте *общедоступное событие* на основе этого делегата и вызовите его, когда условие возникновения сработает.

4. Обязательно (где-угодно в доступном пространстве) *подпишитесь на это событие* теми методами, которые должны сработать и сигнатуры которых подходят к делегату.