Задание РК: описать работу с обобщенным списком на языке программирования C#.

Первый этап: создание объекта списка:

```
List<int> mylist = new List<int>();
```

Добавление элементов в список:

```
mylist.Add(1);
 mylist.Add(2);
mylist.Add(3);
```

В версии языка C# 6 появился более удобный синтаксис для инициализации списков:

```
List<int> mylist = new List<int> {
 1,2,3
};//Для данных типа int
 List<string> mylist = new List<string> {
"строка 1",
"строка 2",
"строка 3"
};
```

Вывод элементов списка:

```
foreach (int i in mylist)
{
Console.WriteLine(i);
}
```

В данном примере в качестве класса-обобщения используется int, поэтому создается список целых чисел. Преимущества применения обобщенной коллекции состоят в том, что тип элемента известен как при добавлении, так и при выводе. При добавлении Visual Studio подсказывает тип добавляемого элемента с помощью механизма IntelliSense. Тип элемента уже известен, так как он был ранее указан при создании списка.

При выводе с использование цикла foreach тип переменной i также заранее известен. Для обобщенного списка перегружен индексатор, который позволяет получить элемент по его номеру в коллекции, элементы нумеруются с нуля. Пример: int item2 = mylist[1];

Свойство Count возвращает количество элементов в коллекции: int count = mylist.Count;

Метод Contains позволяет проверить существование элемента в коллекции. В данном примере проверяется существование в коллекции числа «3»: bool is3 = mylist.Contains(3);

Метод Remove позволяет удалить элемент из коллекции, здесь из списка удаляется число «2»: mylist.Remove(2);

Основное преимущество стандартных коллекций языка C# состоит в том, что для их обработки может быть использован интегрированный язык запросов LINQ. Это чрезвычайно облегчает обработку сложных данных в C#. Ограничение рассмотренного примера заключается в том, что в такой список можно поместить только целые числа и нельзя – элементы других типов. Если в список необходимо сохранять элементы различных типов, то можно использовать необобщенный список.