Предисловие.

Здравствуй, наш юный ученик! Мы очень рады снова видеть в наших рядах программистов. Программистом быть круто, ты ведь это помнишь??? Если хочешь убедиться еще раз, пройди по ссылке;)
Теперь, когда ты овладел базовыми знаниями о программировании и, в частности, о языке программирования C#, мы начнем более детально рассматривать возможности языка C#.

Семинар1. Вспомнить все.

Итак, наш Юный Программист, давай же вспомним что это за язык C# (читается как «Си-Шарп») и почему мы выбрали именно его.

Как всем известно, если компьютеру приказать сделать что-либо, он ничего не сделает. Это происходит потому, что компьютер - это просто железка, и он не может воспринимать команды человека в явном виде. Поэтому, для взаимодействия компьютера и человека (не простого человека, а «заклинателя компьютеров» - программиста) используют специальные языки, понятные как компьютеру, так и программисту. Эти языки называют языками программирования (сокращенно - ЯП). Языков программирования, так же как и человеческих языков, очень много. Некоторые более доступны компьютеру, некоторые более понятны человеку. Каждый язык имеет свою направленность и свои особенности. Считается, что одним из самых универсальных и удобных языков для программирования является язык C#.

Чтобы нам удобнее было общаться с компьютером, программисты придумали специальные инструменты для облегчения написания кода. Такие инструменты называются IDE (Integrated Development Environment, или Интегрированная Среда Разработки). Но пусть такое название не пугает тебя, IDE по сути это всего лишь смесь блокнота с подсказками и компилятора. Блокнот используется для написания кода, а компилятор - для преобразования написанного кода в команды, понятные процессору.

Как ты скорее всего помнишь, в предыдущем семестре мы использовали среду Visual Studio. В этом семестре мы продолжим использовать VS.

У нас много интересного впереди, но начать нужно с повторения.
Этому мы и посвятим наш сегодняшний семинар. Итак, вперед!

1.1 Синтаксис C#

На языке C# мы делали два типа приложений: Консольные приложения и приложения Windows Forms.

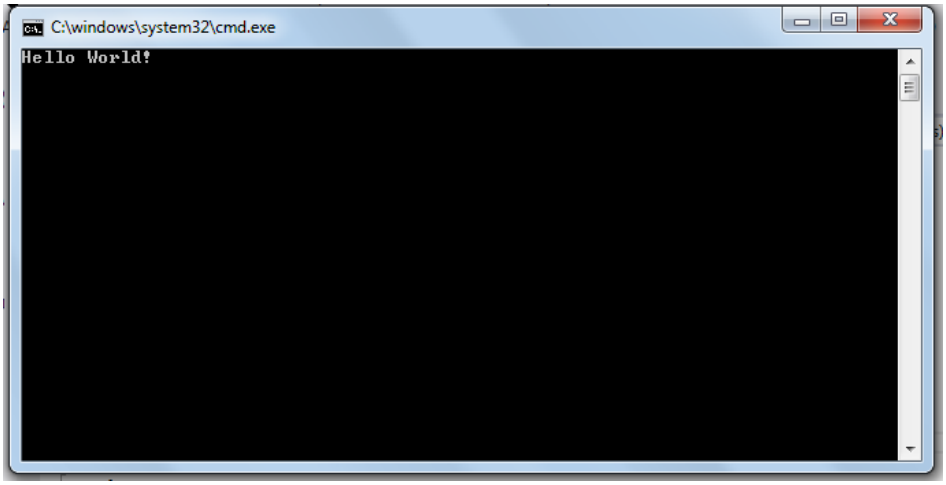

Рис. 1.1. Консольное приложение «Hello World»

Консольные приложения служат для программ, не требующих интерфейса, а требующих производительности и отображения точной информации.

Для обычных пользователей, консольные приложения покажутся неинтересными, т.к. они не имеют кнопок и красивого оформления.

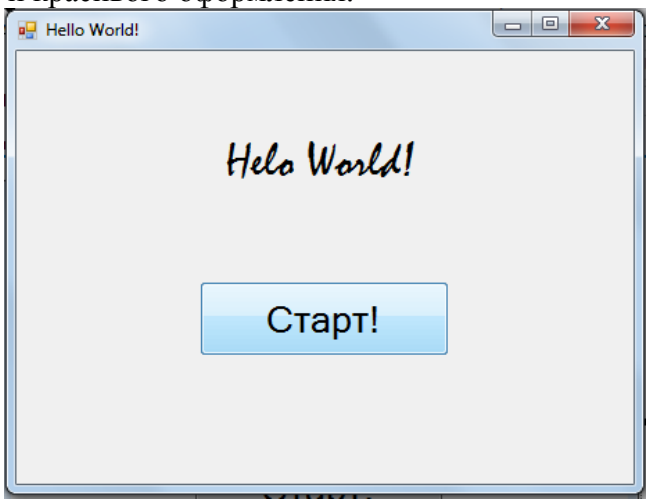

Рис. 1.2. Windows Forms приложение «Hello World. WinForm»

Для этих двух приложений используют разные структуры программ. Однако, синтаксис языка остается одним и тем же. Итак, важные правила языка C#:

- Начало/конец блока обозначается фигурными скобками;
- Фигурные скобки всегда парные(!)
- Каждый оператор заканчивается точкой с запятой;
- Комментарии в коде ставятся с помощью знака «//» - для однострочных комментариев и «/ * */» - для многострочных комментариев.
- Значения функций и методов выводятся, но не сохраняются. Поэтому готовые значения мы должны записывать в переменные.

1.2 Данные в C#. Переменные.

Кстати, о переменных. Вы, надеюсь, помните что такое «переменная»???

Переменная (программирование) — поименованная, либо адресуемая иным способом область памяти, адрес которой можно использовать для осуществления доступа к данным и изменять значение в ходе выполнения программы.

Так гласит Википедиа. Но если говорить проще, то это некое место в памяти компьютера, где можно хранить определенную информацию. Для каждого типа информации (числа, буквы, строки) существуют свои типы переменных.

Тип	*Область значений*	*Размер*
Sbyte	-128 до 127	8-бит целое
Byte	0 до 255	8-бит целое
Char	Символы (буквы, цифры, знаки)	16-битовый символ Unicode
Bool	True или false	1 байт
Short	-32768 до 32767	16-бит целое
Ushort	0 до 65535	Беззнаковое 16-бит целое
Int	-2147483648 до 2147483647	Знаковое 32-бит целое
Uint	0 до 4294967295	Беззнаковое 32-бит целое
Long	-9223372036854775808 до 9223372036854775807	Знаковое 32-бит целое
Ulong	0 до 18446744073709551615	Беззнаковое 32-бит целое
Float	±1,5*10-45 до ±3,4*1033	4 байта
Double	±5*10-324 до ±1,7*10306	8 байт
Decimal	Очень большое число	12 байт

Это простые переменные. Они служат для хранения одного значения. А что, если нам нужно сохранить очень большое количество однотипных переменных? Правильно, нам потребуются массивы!

Массив – это некоторое количество однотипных переменных, называемых одним именем. Для того чтобы создать массив, нужно поставить после указания типа две квадратные скобки:

```
тип[] <название_переменной> = new тип[размер];
```

Пример создания массива состоящего из 10 целых чисел:

```
int[] massiv = new int[10];
```

Массивы – вещь очень полезная. Но бывает, нужно объединить переменные разного типа, например строку и пару чисел. Например, нам нужно написать программу-дневник. В нем, должны содержаться: Имя ученика, класс, оценки по предметам и средняя успеваемость. Для таких задач, мы можем использовать структуры.

Структура – особый способ представления данных. Мы с ними уже сталкивались, применяя структуры Point и Size. Для решения задачи программы-дневника, нам потребуется вот такая структура.

```
public struct Uchenik
 {
    public string Imya;//имя ученика
    public int Vozrast;//возраст
    public int[] Ocenki;//оценки по всем предметам
    public float SrOcenka;//средняя оценка
 }
```

После, мы должны будем создать объект типа Uchenik, который будет содержать эти переменные. Обратить или изменить эти переменные мы можем обращаясь к ним, вызывая объект и после точки указав имя переменной.

В программе это выполняется вот так:

```
Uchenik u1 = new Uchenik();
u1.Imya = "Ilya";
u1.Vozrast = 12;
```

Теперь, вспомнив переменные и структуры, давайте напишем консольную программу-анкету, которая будет спрашивать информацию у ученика и записывать ее в переменные и структуры.

Для этого нам потребуются следующие действия:

1. Программа спрашивает у пользователя имя.

   Если значение пустое, нужно показать сообщение с ошибкой.
   Иначе, программа сохраняет имя в соответствующее поле структуры

2. Программа спрашивает у пользователя возраст и преобразовывает его в числовой формат (вы ведь все помните, в чем разница между “14” и 14?)

Вот пока и весь алгоритм. Мы уверены, что у тебя не возникнет сложностей с реализацией!

Готово! Наша программа записывает данные. А давайте, наша программа будет определять считывать в каком классе учится ученик, зная только его возраст. Теперь, мы вспомним про управляющие структуры.

1.3 Управляющие структуры. if - else.

После написания программы, наша задача усложнилась: нам нужно проверить возраст, и определить класс. Для этого, мы можем воспользоваться двумя способами: оператор if и оператор switch.

Для начала, воспользуемся оператором if, и определим, к какой группе относится ученик: Младшая, Средняя и Старшая. К Младшим мы будем относить тех, кому нет еще 11 лет, к Средним тех, кто старше 11, но нет еще 16 лет. А к Старшим, всех остальных.

Дополните этот фрагмент программы:

```
if (u1.Vozrast < 11)
{
    Console.WriteLine("Младшая группа");// вывод на экран
}
else
{

}
```

1.4 Управляющие структуры. Switch.

Итак, мы смогли определить группу, но теперь, нам нужно точно определить класс. Для этого воспользуйтесь оператором switch (допишите пример):

```
switch(u1.Vozrast)
{
    case 12: Console.WriteLine("5 класс"); break;
    case 13: Console.WriteLine("6 класс"); break;
    // следующие варианты
    default: Console.WriteLine("?"); break;
}
```

1.5. Циклы

Теперь, когда наша программа может заполнить информацию об ученике, сделаем так, чтобы программа могла запомнить информацию не об одном ученике, а о целом классе! Чтобы не копировать программу несколько раз, воспользуемся циклами.

Вы еще с прошлого семестра, должны помнить что в C# есть три основных типа циклов: for, while, do…while.

Цикл из одного действия:

```
for (инициализация; условие; изменение) действие;
```

Цикл, тело которого состоит из нескольких операторов:

```
for (инициализация; условие; изменение) действие
{
    // несколько действий выполняемых подряд
    Оператор1;
    Оператор2;
    Оператор3;
}
```

Для while:

Цикл из одного действия:

```
while (условие) действие;
```

Цикл, тело которого состоит из нескольких операторов:

```
while (условие)
{
      // несколько действий выполняемых подряд
      Оператор1;
      Оператор2;
      Оператор3;
}
```

Теперь самостоятельно реализуй сохранение и заполнение данных целого класса учеников.

Поздравляем! Мы написали простенькую программу и вспомнили основные переменные и операторы.

Итог.

На этом семинаре, мы вспомнили про основные функции языка C#. Написали простенькую программу для заполнения анкет с использованием пройденных тем. На следующем занятии мы окунемся в мир компьютерной графики!