Объектно-ориентированное программирование

Лекция 1

Введение в язык программирования C#

Преподаватель: Бобков Владислав Дмитриевич

Ассистент: Теслов Сергей Игоревич

Благодарность за проделанную работу над курсом можно выразить
Макаренко Дмитрию Владимировичу

Курс

- 8 лекций
- 8 практических занятий
- 4 лабораторные работы

- Экзамен

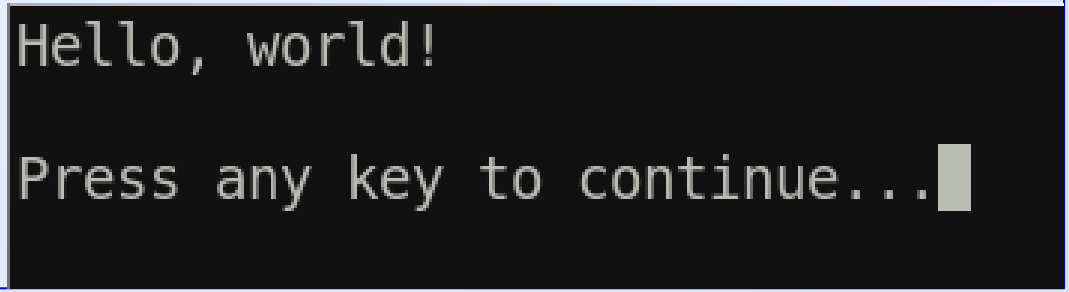
Популярные ООП языки

- Java
- C++
- C#
- Objective C
- Python

План лекций

1. Введение в C# (и немного про гитхаб)
2. Циклы, условия и функции
3. Графические приложения WPF
4. Массивы, перечисления и исключения
5. Классы и объекты
6. Наследование и агрегация
7. Интерфейсы
8. Полиморфизм

```
class MainClass
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Hello, world!");
    }
}
```



```
class MainClass
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Hello, world!");
    }
}
```

Всю работу выполняет эта строка

```
using System;
class Program {
    static void Main(string[] args) {
            Console.Write("Enter your name: ");
            string name = Console.ReadLine();
                Console.WriteLine($"Hello {name}!");
    }
}
```

Эта программа что-то считывает, сохраняет и затем снова выводит на экран

```
using System;

class Program {

    static void Main(string[] args) {

            Console.Write("Enter your name: ");

            string name = Console.ReadLine();

                Console.WriteLine($"Hello {name}!");

    }

}
```

Предлагаем пользователю ввести текст

```
using System;
class Program {
    static void Main(string[] args) {
            Console.Write("Enter your name: ");
            string name = Console.ReadLine();
                Console.WriteLine($"Hello {name}!");
    }
}
```

Считываем его ответ и сохраняем в строковой переменной, которая называется **name**

```
using System;
class Program {
    static void Main(string[] args) {
            Console.Write("Enter your name: ");
            string name = Console.ReadLine();
                Console.WriteLine($"Hello {name}!");
    }
}
```

Это - **переменная**, в которой может храниться какое-то текстовое сообщение

```
using System;
class Program {
    static void Main(string[] args) {
            Console.Write("Enter your name: ");
            string name = Console.ReadLine();
                Console.WriteLine($"Hello {name}!");
    }
}
```

Это - **команда**, которая считывает текст который набирает пользователь

```
using System;
class Program {
    static void Main(string[] args) {
            Console.Write("Enter your name: ");
            string name = Console.ReadLine();
                Console.WriteLine($"Hello {name}!");
    }
}
```

Это - операция **присваивания**, она берет результат команды справа и помещает в переменную слева

```
using System;
class Program {
    static void Main(string[] args) {
            Console.Write("Enter your name: ");
            string name = Console.ReadLine();
            Console.WriteLine($"Hello {name}!");
    }
}
```

Выводим на экран слово **Hello** и текст который хранится в переменной **name**

Как работают программы

- Часть которая выполняет всю работу - это содержимое **метода Main**
- Строки кода выполняются последовательно друг за другом
- После завершения последней команды программа завершается

```
public static void Main(string[] args)
{
    Console.Write("Число А: ");
    string number1Text = Console.ReadLine();
    int number1 = int.Parse(number1Text);
    Console.Write("Число Б: ");
    string number2Text = Console.ReadLine();
    int number2 = int.Parse(number2Text);
    int result = number1 * number2;
    Console.WriteLine("А + Б = {result}");
}
```

```
public static void Main(string[] args)
{
    Console.Write("Число А: ");
    string number1Text = Console.ReadLine();
    int number1 = int.Parse(number1Text);
    Console.Write("Число Б: ");
    string number2Text = Console.ReadLine();
    int number2 = int.Parse(number2Text);
    int result = number1 * number2;
    Console.WriteLine("А + Б = {result}");
}
```

Сохраняем значение введенное пользователем в текстовой переменной

Строки и числа

```
string number1Text = Console.ReadLine();
int number1 = int.Parse(number1Text);
```

- number1Text и number1 - две переменных разного типа.
- Когда пользователь вводит “1”, то это сохраняется в переменной как текст
- “1” не равно 1 в памяти компьютера. Математические операции можно производить только с числами

Строки и числа

```
int number1 = int.Parse(number1Text);
```

- Метод Parse делает определенную магию и преобразует строку в численное представление
- Результат преобразования сохраняется в целочисленной переменной
- Эту переменную уже можно использовать для вычислений

Строки и числа

```
int number1 = int.Parse(number1Text);
```

- Магия метода Parse работает только с теми строками, которые он понимает:
  - “0”, “-12232”, “147”
- Если ему передать строку которую он не поймет:
  - “Один”, “2+2”, “Иннокентий”

  То все сломается и программа “упадет”.

```
public static void Main(string[] args)
{
    Console.Write("Число А: ");
    string number1Text = Console.ReadLine();
    int number1 = int.Parse(number1Text);
    Console.Write("Число Б: ");
    string number2Text = Console.ReadLine();
    int number2 = int.Parse(number2Text);
    int result = number1 * number2;
    Console.WriteLine("А + Б = {result}");
}
```

Преобразуем текстовое представление в числовое.

```
public static void Main(string[] args)
{
    Console.Write("Число А: ");
    string number1Text = Console.ReadLine();
    int number1 = int.Parse(number1Text);
    Console.Write("Число Б: ");
    string number2Text = Console.ReadLine();
    int number2 = int.Parse(number2Text);
    int result = number1 * number2;
    Console.WriteLine("А + Б = {result}");
}
```

Повторяем для второго числа

```
public static void Main(string[] args)
{
    Console.Write("Число А: ");
    string number1Text = Console.ReadLine();
    int number1 = int.Parse(number1Text);
    Console.Write("Число Б: ");
    string number2Text = Console.ReadLine();
    int number2 = int.Parse(number2Text);
    int result = number1 * number2;
    Console.WriteLine("А + Б = {result}");
}
```

Производим вычисление и сохраняем результат в новой переменной

```
public static void Main(string[] args)
{
    Console.Write("Число А: ");
    string number1Text = Console.ReadLine();
    int number1 = int.Parse(number1Text);
    Console.Write("Число Б: ");
    string number2Text = Console.ReadLine();
    int number2 = int.Parse(number2Text);
    int result = number1 * number2;
    Console.WriteLine("А + Б = {result}");
}
```

Выводим результат на экран

Поиск ошибок

- Все программисты независимо от опыта допускают ошибки
- Ошибки не всегда приводят к “падению” программы. И в некоторых случаях программа может работать правильно:
  - 1 + 1; 0 + 0; 2 + 2 === 1 * 1; 0 * 0; 2 * 2
- Поиск ошибок ~50% времени затрачиваемого на написание ПО

Минимизируем вероятность ошибок

- Всегда проверять входные данные на корректность
- Тестировать написанный код используя граничные значения
- Писать комментарии для неочевидных решений

Объявление переменных

Без инициализации:

<тип> <имя>;

С инициализацией:

<тип> <имя> = <значение>;

Правила именования переменных

- Первый символ - буква или знак подчеркивания
- Последующие - любые буквы и цифры

Можно:

name, l33t, Иван_Васильевич

Нельзя:

2ch, #f1singapore, Иван Петрович

Регистр имеет значение, **Name** и **name - это разные имена!**

Целочисленные типы

Знаковые		Беззнаковые	
sbyte	[-128; 127]	**byte**	[0; 255]
short	[-32768; 32767]	**ushort**	[0; 65535]
int	[-2’147’484’648 - 2’147’483’647]	**uint**	[0; 4’294’967’295]
long	[$-9.2 * 10^{18}$; $9.2 * 10^{18}$]	**ulong**	[0; $18.4 * 10^{18}$]

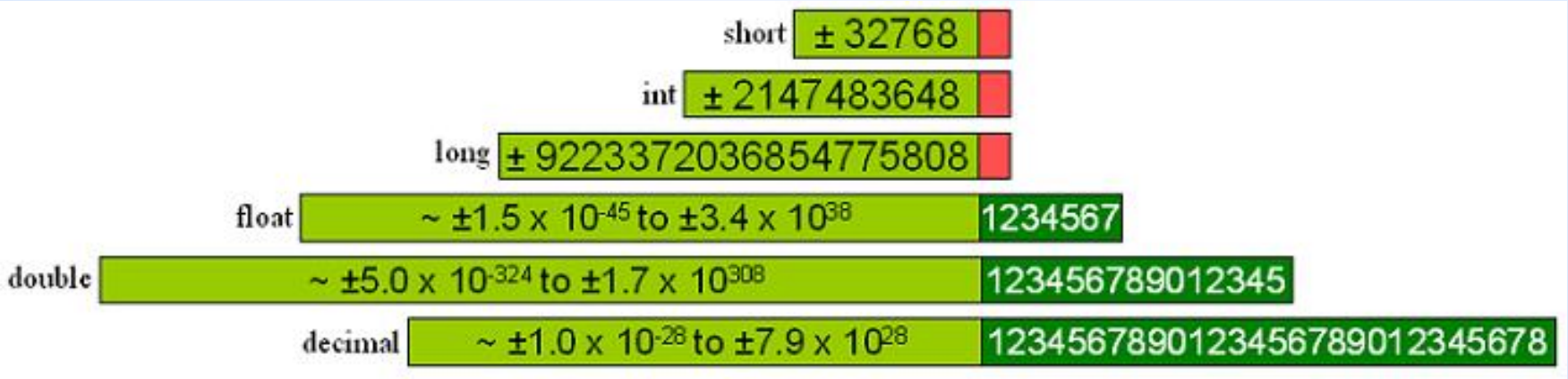
Вещественные типы - IEEE 754
https://evanw.github.io/float-toy/

float	от $\pm 1.5 \times 10^{-45}$ до $\pm 3.4 \times 10^{38}$
double	от $\pm 5.0 \times 10^{-324}$ до $\pm 1.7 \times 10^{308}$
decimal	от $\pm 1.0 \times 10^{-28}$ до $\pm 7.9 \times 10^{28}$

Целое или вещественное

- Целые используем там, где нас интересуют ШТУКИ:
  - Количество студентов
  - Порядковый номер посетителя в очереди
  - Часы и минуты отправления самолета
- Вещественные используем, когда нам необходимо хранить ЧАСТЬ:
  - Средний возраст кошек в питомнике
  - Вес товара в доставке

Сравнительные размеры типов



Примеры объявления переменных и форматов чисел

```
int age;

age = 20;

int year = 2015;

sbyte speed = 130; // не вмещается!

ulong метровДоСолнца = 149597870700;

float height = 1.80f;

double размерМолекулыH2O = 3.0E-10;
```

Приведение одних типов к другим

Можно

char → int, int → long, int → float

Нельзя

int → char, long → int, double → float

Если нельзя, но очень хочется

```
float Pi = 3.1415f;
int roundedPi = (int) Pi; // 3


long ageOfEarth = 4540000000;
int age = (int) ageOfEarth; // 245032704
```

Арифметические операторы

- `+` – сложение
- `-` – вычитание
- `*` – умножение
- `/` – деление
- `%` – остаток от деления

Унарные операторы

- `x--  y++` – постфиксные инкремент, декремент
- `--x  ++y` – префиксные инкремент, декремент
- `-x` – смена знака
- `!x` – логическое отрицание или инверсия
- `~x` – побитовая инверсия

Короткая запись арифметических операторов

- `x += y;` `// x = x + y;`
- `x -= y;` `// x = x - y;`
- `x *= y;` `// x = x * y;`
- `x /= y;` `// x = x / y;`
- `x %= y;` `// x = x % y;`

Целочисленное и вещественное деление

- Деление целого на целое - дает целое
- Деление вещественного и целого - дает вещественное

10 / 3 == 3

10 / 3.0 == 3.33333333333

Базовые типы. Строки

char - один символ

string - много символов

```
char A = 'A';

char arrow = '\x2192'; //
```

```
string name = "Ash Williams";
```

```
using System;


class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
    }
}
```

```
using System;

    class MainClass {

        public static void Main(string[] args)

        {

            string name = "Vasya";

            Console.WriteLine($"Hello, {name}!");

        }

    }
```

```
Hello, Vasya!

Press any key to continue...
```

```
using System;

class Program {

    static void Main(string[] args) {

        Console.Write("Enter your name: ");

        string name = Console.ReadLine();

            Console.WriteLine($"Hello {name}!");

    }

}
```

```
Enter your name: Dima
Hello Dima!

Press any key to continue...
```

Вывод текста и чисел на экран

```
Console.Write("Привет"); //Выводит текст в консоль
```

```
Привет_
```

```
Console.Write("Мир!");
```

```
ПриветМир!_
```

```
Console.Write("Привет\n");
```

```
Привет

_
```

```
Console.Write("Мир!\n");
```

```
Привет
Мир!

_
```

```
Console.WriteLine("Привет, Мир!"); //Выводит текст

                  //и переводит курсор
```

```
Console.Write("2 * 2 = " + 2 * 2);
```

```
2 * 2 = 4_
```

```
int x = 2, y = 10;

Console.Write($"{x} * {y} = {x * y}");
```

```
2 * 10 = 20_
```

Ввод данных с клавиатуры

```
string name = Console.ReadLine();

int age = int.Parse(Console.ReadLine());

double height = double.Parse(Console.ReadLine());



ConsoleKeyInfo key = Console.ReadKey();
```

Условный оператор

```
if (Условие) {
    // выполнить если истина
}
else {
    // выполнить если ложь
}
```

Логический тип (Истина\Ложь)

```
bool everythingIsFine = true;

bool requestFailed = false;


int age = 15;

bool canBuyBeer = age > 18;
```

Операторы сравнения

- `<` — меньше
- `>` — больше
- `<=` — меньше или равно
- `>=` — больше или равно
- `==` — равно
- `!=` — не равно

Оператор равенства и float/double

- Из-за особенностей хранения вещественных чисел в памяти компьютера, использовать оператор равенства с типами float и double опасно

```
if ( average == 1.0f )
        Console.WriteLine(“Среднее равно 1”);
if ( average - 1.0f < epsilon )
        Console.WriteLine(“Среднее ~равно 1”);
```

Оператор равенства и строки

- Строки можно сравнивать так же как и числа
- Сравнение строк учитывает регистр

```
if ( name == “Иннокентий” )
    Console.WriteLine (“Привет, Кеша”);
```

Логические операторы

- `&&` — логическое И, конъюнкция
- `||` — логическое ИЛИ, дизъюнкция

```
Console.Write("Напиши свое имя: ");
string name = Console.ReadLine();
if (name == "Вася") {
    Console.WriteLine("Здравствуй, Василий!");
}
else {
    Console.WriteLine("Вы кто такой? Я вас не звал!
                                    Уходите!");
}
```

Правила хорошего тона

Программа должна хорошо работать

- Программа не должна работать с ошибками: падения, неправильные вычисления - это плохо
- Пользователю должно быть понятно, что программа от него хочет и что делает
- Результат работы должен быть представлен в понятном виде

Правила хорошего тона

Код должен быть “хорошим”

- Он должен быть легкочитаем - названия переменных должны быть осмысленными
- Он должен быть правильно отформатирован - для одинаковых конструкций должен использовать одинаковый вид
- Неочевидные места должны быть прокомментированы

Правила хорошего тона

Комментарии

- Это подсказка читающему код о том, что задумал программист
- Компилятор полностью игнорирует комментарии

```
int Align = 20; //Отступ кнопки
```

```
/*
 * Эта функция вычисляет площадь
 * треугольника по формуле Герона */
```

Правила хорошего тона

Плохие комментарии

- Комментарии не должны повторять информацию очевидную из кода
- Комментарии должны соответствовать тому коду для которого написаны

```
count = count + 1; // увеличиваем значение на 1

int perimeter = 3.14 * R * R; // Вычисляем периметр прямоугольника
```

Среды разработки

- **Visual Studio Community** - Windows

- **Visual Studio для macos -** не подойдет
- **MonoDevelop/Xamarin** для Linux - не подойдет

Если нет возможности установить VS

1. http://vpn.miet.ru/ , установить клиент, подключиться к сети МИЭТ (общежитие уже подключено)
2. Подключение к удаленному рабочему столу к серверу **skylab.miet.ru, galaxy.miet.ru**
3. Логин и пароль от учетной записи МИЭТ

Использование Git

при разработке программного обеспечения

Программа - это вообще что?

- Программа - это множество текстовых файлов
- Программный продукт живет годы или десятки лет
  - Разработка - выполняем начальную задачу
  - Отлаживаем - заставляем задачу выполняться правильно
  - Доработка - оказалось, что задача была в другом
  - Поддержка - обеспечиваем работу на разных машинах и ОС
- В процессе жизни программы одни строки заменяются другими, удаляются и добавляются

Система контроля версий исходного кода

- Яндекс-диск для программистов
- Запоминает состояние каждой строки каждого файла за всё время существования программы

Позволяет:

- помечать определенные версии исходных кодов метками (Версия 1.0, Версия 20.12, Точно работает, Спец версия для Васи)
- “мгновенно” переключать разные версии исходников и работать параллельно
- быстро находить кто, когда и зачем отредактировал любую строку
- создавать резервные копии исходников

Кто использует git или аналоги?

- **Все разработчики ПО - исходный код**
- **Web разработчики - верстка и статика**
- Ученые - ПО, текст статей, данные
- Писатели, журналисты - книги, статьи
- Студенты - ДЗ, курсовые, дипломы

Если это текст - можно использовать git

А если не текст - тоже можно, но без плюшек

Так система контроля версий это какая-то программа?

Тип системы:

- Subversion
- Mercurial
- Git

Программа:

- Visual Studio
- Github desktop
- TortoiseHG

Репозиторий:

- Файлы на локальном диске
- Файлы на сервере

Веб сервис

- Github
- Gitlab
- Sourceforge

Git (brit.) - неприятный в общении человек

А еще - распределенная система контроля версий исходного кода (СКВ)

Основные понятия с которыми работает Git:

- **Репозиторий** - папка в которой хранятся **все** существующие версии исходного кода для определенного проекта (**.git)**.
- **Удаленный репозиторий** - папка на сервере, который хранит и позволяет обмениваться изменениями
- **Коммит** (Фиксация) - некоторый набор изменений исходного кода с поясняющим комментарием. Репозиторий состоит из коммитов.
- **Рабочая директория** - все что вы видите в папке проекта за исключением **.git**

Github - веб сервис для управления репозиториями

- Позволяет создавать **удаленные** репозитории
- Позволяет обмениваться изменениями файлов
- Позволяет просматривать изменения которые были отправлены на сервер

Локальное ПО

- Git - это приложение с интерфейсом командной строки
  - git add .
  - git commit -m “Была исправлена ошибка с обработкой команды по нулевому адресу”
  - git push origin master
- Графический интерфейс по умолчанию Git GUI
- Другие графические интерфейсы:
  - Github Desktop
  - TortoiseGit
  - Sourcetree
  - Fork

Все материалы
https://github.com/MIEE-ACS/OOP-2022

Контакты
Преподаватель: Бобков Владислав Дмитриевич
email: bobkov.miet@gmail.com
telegram: https://t.me/bobkov_vd

Ассистент: Теслов Сергей Игоревич
email: teslov90@mail.ru
ассистент telegram: https://t.me/Medoed_Medvedka

Благодарность за проделанную работу над курсом можно выразить Макаренко Дмитрию Владимировичу: https://t.me/d_makarenko

Все материалы
https://github.com/MIEE-ACS/OOP-2022

Контакты
Преподаватель: Бобков Владислав Дмитриевич
email: bobkov.miet@gmail.com
telegram: https://t.me/bobkov_vd

Ассистент: Теслов Сергей Игоревич
email: teslov90@mail.ru
ассистент telegram: https://t.me/Medoed_Medvedka

Благодарность за проделанную работу над курсом можно выразить Макаренко Дмитрию Владимировичу: https://t.me/d_makarenko