Знакомство с языками программирования

Лекция 1. Язык C# (сишарп)

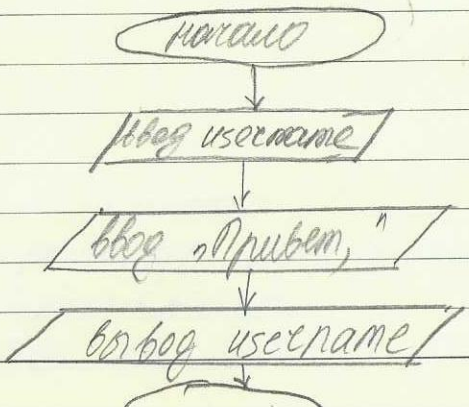
Создал MsFt как альтернативу Java

В VSC расширение C# extension - для более сложных проектов

// - это комментарии к коду, не влияют ни на что, это напоминание

Console - позволяет работать с окном терминала
WriteLine - команда для вывода чего-то в окно терминала (то, что выводится - пишется в круглых скобках ("Hello world!")
dotnet run - запустить и посмотреть работоспособности приложения ("Hello world!")

dotnet new console - создать новый проект

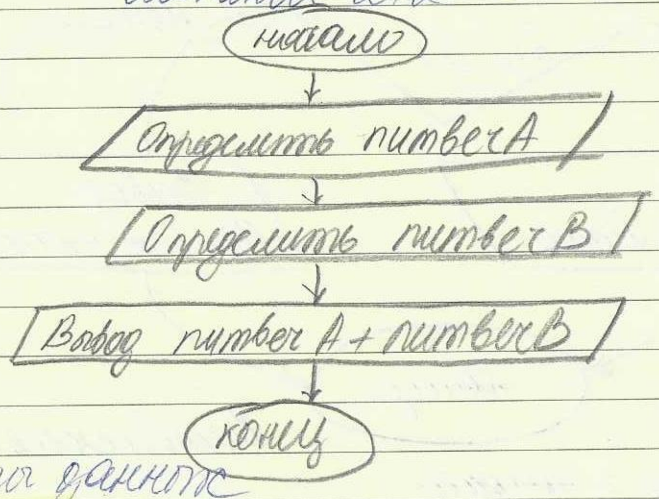


Console.ReadLine() - происходит считывание данных

string - строка
= - возьми то, что справа и положи влево

Write(); - вывод в 1 строку
WriteLine(); - в конце перейди на новую строку
ReadLine(); - считать строку из терминала

Сложение 2+2

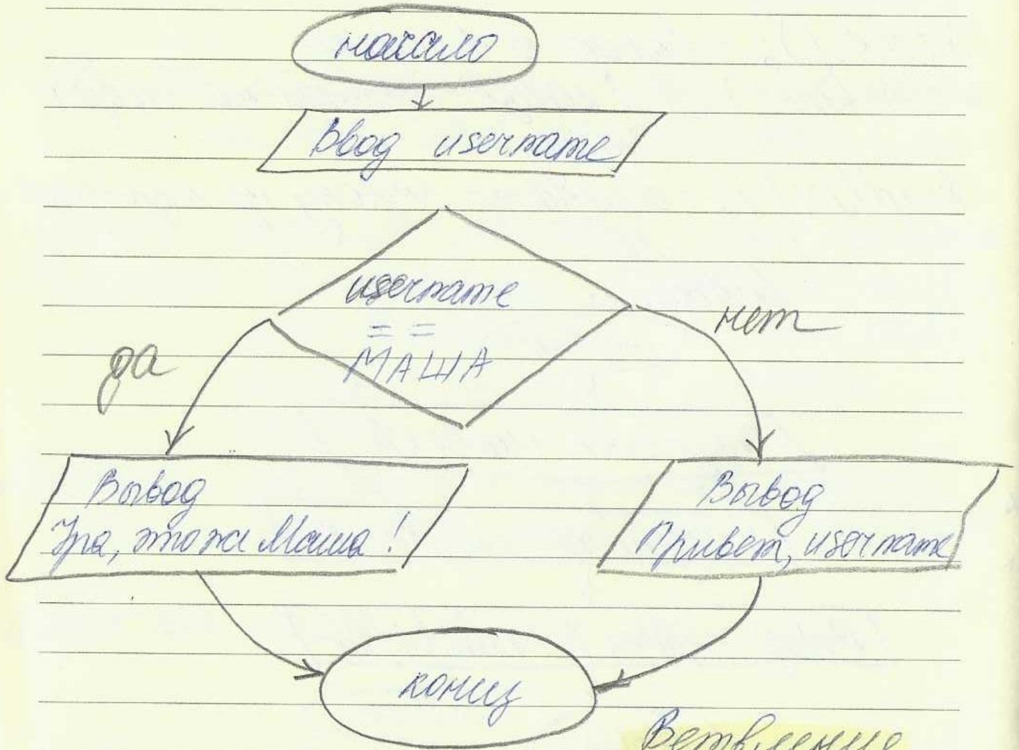


Типы данных
int - целое число
~~bo~~ double - вещественные числа
string - символы
bool - логический тип данных (ложь и истина)

% - нахождение остатка от деления

Случайности

new Random().Next(min, max) —
дает случайное целое число
от min до max-1 [min, max-1]
или [min, max)



Ветвление

Синтаксис: начало

```
if (условие)
{
 Набор действий 1
}
else
{
 набор действий2
```

Циклы

Синтаксис: начало

```
while (условие продолжения)
{
 набор действий
}


int count = 0;

while(count < 100)
{
 набор действий
 count = count+1
}
```

Лекция 2. Знакомство с ф-циями и массивами

Функции (метод) в программировании — это часть программного кода, к-рый создает разработчик

- ф-ция имеет идентификатор (имя)
- может иметь входные аргументы
- может возвращать значение.

Имя - латинские маленькие и большие, цифры, но не на 1 месте.

```
ВозвращаемыйТипДанных ИмяМетода([ТипДанных1 ИмяАргумента1, ...])
{
    Тело метода
    return ЗначениеСоотв-щееВозвращаемомуТипуДанных;
}
```

$f(x) = x^2+1$

Тип аргумента

```
double f(double x)
{
    double result = x*x+1;
    return result
}
```

имя аргумента

возвращаемый тип

тело метода

возвращаемое значение

index = index+1 <=> index++

Массив

== - эквивалентно равенству правой и левой части

int array = {9, 28, 1, 32, 1990}

Массив array из n элементов → найти элемент массива find:

1) Установить счетчик index в позицию 0.
2) Если array[index] = find → алгоритм завершил работу успешно
3) Увеличить index на 1
4) Если index < n, то перейти к шагу 2. Иначе алгоритм завершил работу безуспешно.

int new [10] - создай новый массив, в к-ом будет 10 элементов. По умолчанию он будет заполнен нулями.

void - метод, который ничего не возвращает, войд метод.

Для нахождения случайного числа (генерация случайного числа)

```
Random random = new Random( );
int randomNumber = random.Next(10, 100);
Console.WriteLine(randomNumber);
```

рандомное число от 10 до 99].

В 3-х значном числе найти 2 последних:

~~[illegible]~~

int lastNumber = randomNumber % 100

найти 3 последних:

% на 1000

notes Языки
Лекция 3. Функции

Пн Вт Ср Чт Пт Сб Вс

Функции в пр-ме:
Вид 1. - ничего не принимают и ничего не возвращают (пример 012)

Вид 2 - принимают какие-то аргументы, но ничего не возвращают (пример 012)

Вид 1 и вид 2 - это войд методы (void), к-ые ничего не возвращают

Вид 3 - могут что-то возвращать, но не принимают никаких аргументов (служат для генерации случайных данных)

Вид 4 - (самой распространеной) что-то принимают и что-то возвращают для дальнейшей работы

Пн Вт Ср Чт Пт Сб Вс notes

Циклы и не только while

count ++
увеличение счетчика на единицу
- инкремент,
уменьшение - декремент

при зацикливании программы в VSC - ctrl+c.

```
for (int i=0; i<10; i++)
{
  Console.WriteLine(i)
}
```

Массив
Алгоритм сортировки методом выбора (или минимакса, или выбором минимального, или выбором max-ного)

Лекция 4. Двумерные массивы и рекурсия.

Двумерный массив

, - указание на то, что у нас 2 размерности массива

```
string[,] table = new string[2, 3];
```

название массива

строки

столбцы

строки matrix.GetLength(0)

кол-во строк

```
int[,] matrix = new int[5,8]
```

столбцы

matrix.GetLength(1)

возвращает кол-во столбцов в массиве

Рекурсия - это ф-ция, которая вызывает сама себя

Факториал: $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$

$5 \cdot 4!$

$4 \cdot 3!$

$3 \cdot 2!$

2

double в int преобразовать

```
doubl tmp = 1,3;
int tmp2 = (int)tmp;
```

TryParse - ~~т~~ try - попытается
Parse - из строки в число преобразует

Лекция 5. Как не нужно писать код

Код читается больше, чем пишется, не нужно экономить на понятности кода ради скорости.

Стараться не использовать сокращения кроме общепринятых в разрабатываемом продукте.

Когда придумываете названия для общедоступных единиц, старайтесь не использовать имена, потенциально или явно конфликтующие со стандартными идентификаторами

Использовать имена с простым написанием, Избегать слов с двойными буквами, сложным чередованием согласных

Никогда не использовать матерные (запрещенные) слова в коде (в т.ч. в комментариях

Не использовать знаки подчеркивания, дефисы и др. символы, не являющиеся буквенно-цифровыми.

Избегать использование идентифика-торов, совпадающих с ключевыми словами широко используемых языков программирования
string class = „1В";

Стараться не использовать сокращения (GetWindow лучше, чем GetWin)

Не использовать акронимы, к-ые не являются широко понятными

Используйте универсальные имена платформы, не относящиеся к конкретному языку.

Используйте общие, не привязанные к контексту, имена, когда это нужно

Имена методов, аргументов, переменных и других системных единиц отличаются

Нотации:

Pascal – указание этого стиля оформления идентификатора обозначает, что первая буква заглавная и все последующие первые буквы слов тоже заглавные.
НП: BackColor, LastModified

Camel – указание этого стиля обозначает, что первая буква строчная, а остальные первые буквы слов заглавные
НП: backColor.

Избегать длинных строк, переносите инструкцию на новую строку.
Не размещать несколько инструкций на 1 строке, каждая инстр-ция р.д. с новой строки
При разбивке длинных выражений на неск-ко строк, оставляйте оператор на новой строке?

Использовать автоформатирование кода вашей IDE после модификации кода.

Не используйте проверки вида b==false. Вместо этого: !b

C# C# C# C#

Для метода использовать Pascal

Если асинхронный метод - заканчивать на „Async"

Для переменных - Camel нотацию

Использовать не явную типизацию (var) для локальных переменных в случаях, когда тип переменной понятен из правой части назначения или когда точный тип не важен.

Объявляйте переменные непосредственно перед их использованием

Счетчики в циклах i, j, k, l, m, n, если этого не требует условие

Инициализируйте переменные при объявлении, если есть такая возм-ть

Неиспользуемый код не комментируйте, а удаляйте

Лекция 6. Как не нужно писать код

Красивый код: - гибкий
- расширяемый
- модульный
- поддерживаемый
- документированный

Don't repeat yourself. DRY - это принцип разработки ПО, нацеленный на снижение повторения информации различного рода (избегать копипаста)

YAGNI („You aren't gonna need it") - с англ. „вам это не понадобится" - процесс или принцип проектирования ПО, при к-ом в качестве основной цели и/или ценности декларируется отказ от избыточной функциональности, - то есть отказ добавления функ-сти, в к-ой нет непосредственной надобности.

KISS (акроним для „Keep it simple, stupid") Принцип, запрещающий использование более сложных средств, чем необходимо. Декларирует простоту системы в кач-ве осн-ой цели и/или ценности

Нотации:
- Hungarian
- CamelCase
- PascalCase

и ещё:
- придерживаться SOLID
- не использовать антипаттерны
- использовать паттерны
- декомпозировать
- писать тесты
- придерживаться Code Convention

Лекция 7. Рекурсия

- функция, к-ая вызывает сама себе

При описании рекурсии:
1. Описать условие выхода