

Программирование под iOS

Лекция 1

Бесараб Иван

Знакомство с роботом, первая программа

Просьба отключить мобильные телефоны

Встречайте робота



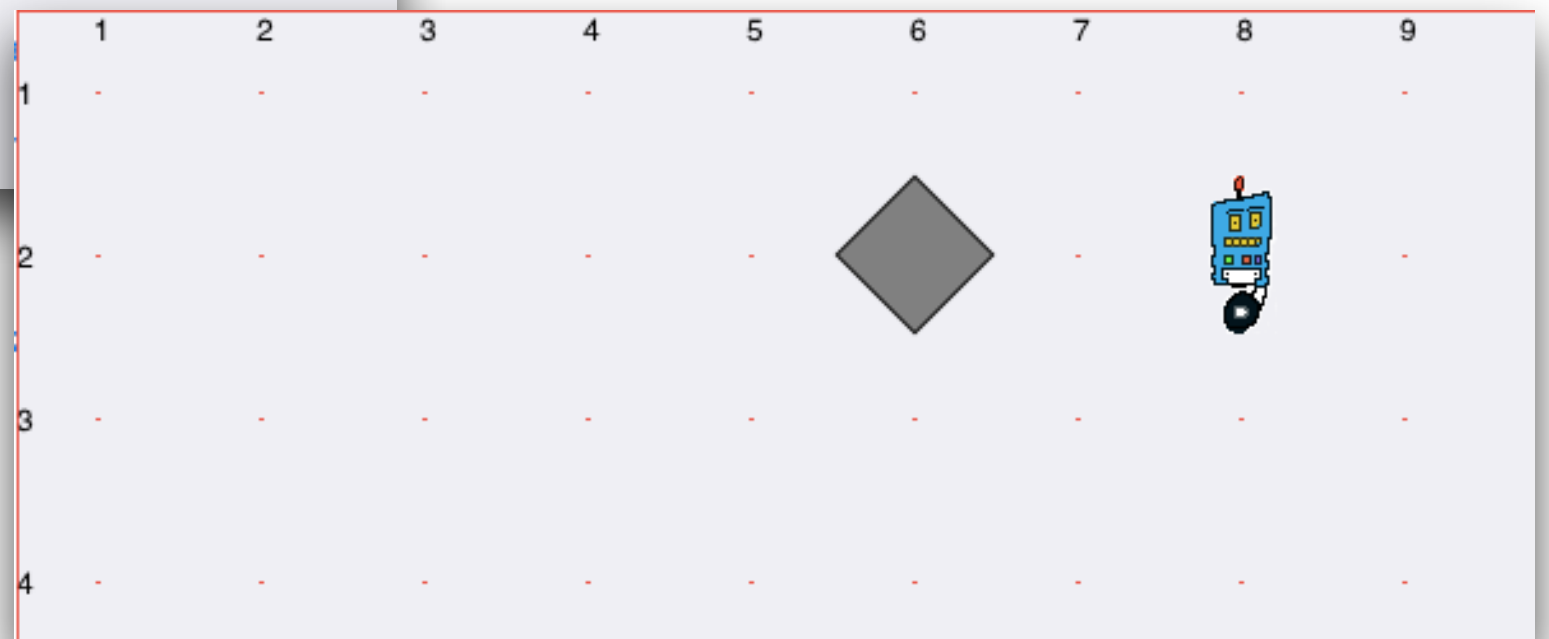
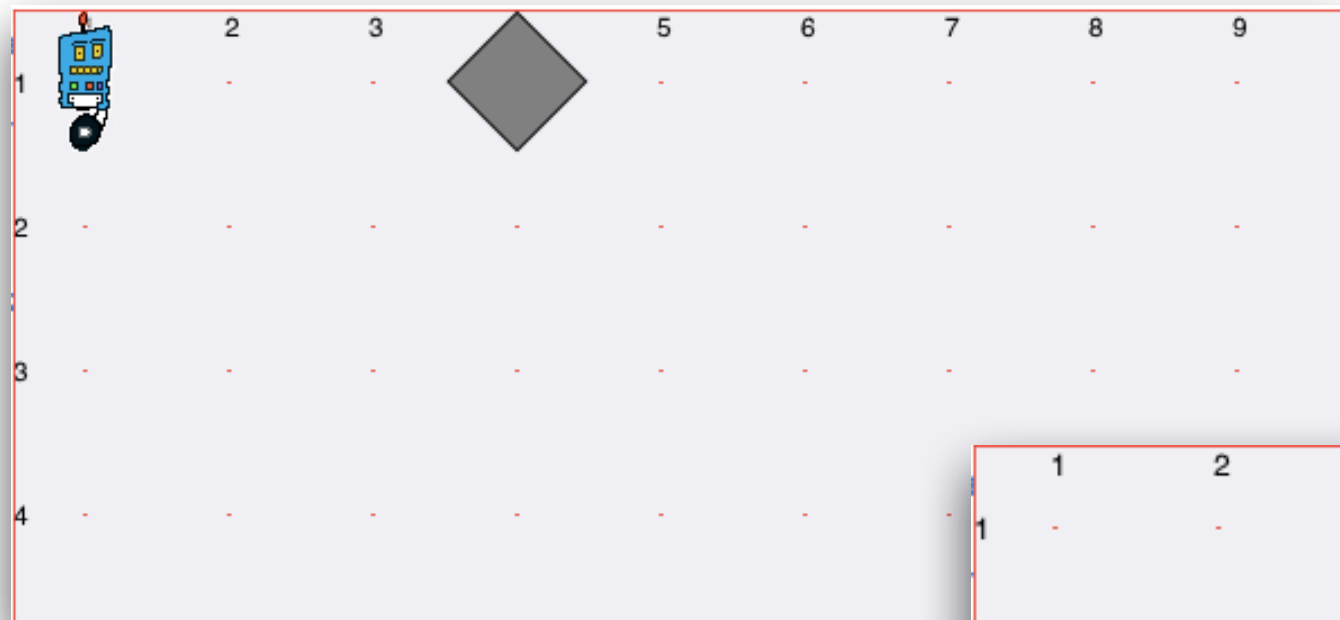
Встречайте робота



Первая программа

Разработаем алгоритм для перехода из начального
состояния в конечное

move()
turnRight()
pick()
put()

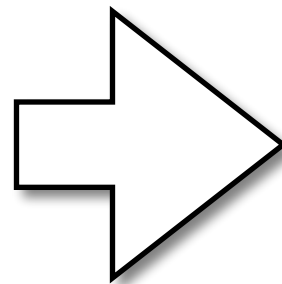


Первая программа

Алгоритм

```
pick  
turnRight  
move  
put
```

Программный код



```
pick()  
turnRight()  
move()  
put()
```

Синтаксис – совокупность правил образования языковых конструкций, или предложений языка программирования

Первая программа

Место откуда начинать - метод viewDidAppear()

- Создание нового метода
- Тело метода - все что будет выполнено при вызове метода
- Метод - набор инструкций для выполнения какого нибудь действия
- Написать новый метод

Методы/Сообщения

- **move** - двигаться вперед на одну клетку
- **turnRight** - повернуться на 90 градусов вправо
- **pick** - взять мешок под собой
- **put** - положить мешок под собой

} **методы/**
сообщения

Методы/сообщения - некоторые инструкции которые мы можем вызывать или использовать и Robot отвечает на эти методы/сообщения

Мы вызываем конкретный метод у Robot и он делает какое нибудь действие в зависимости от того какой метод мы вызываем у него

УСЛОВИЯ

что проверять?

frontIsClear – впереди нет препятствия
leftIsClear – слева нет препятствия
rightIsClear – справа нет препятствия
candyPresent – есть мешок под роботом
candyInBag – мешок есть в работе
facingUp – смотрит вверх
facingDown – смотрит вниз
facingRight – смотрит вправо
facingLeft – смотрит влево

УСЛОВИЯ

что проверять?

frontIsBlocked – впереди препятствие
leftIsBlocked – слева препятствие
rightIsBlocked – справа препятствие
noCandyPresent – нет мешка под роботом
noCandyInBag – мешка нет в роботе
noFacingUp – не смотрит вверх
noFacingDown – не смотрит вниз
noFacingRight – не смотрит вправо
noFacingLeft – не смотрит влево

Операторы условного перехода

If: - позволяет выполнить команду если выполняется условие

```
if условие {
```

```
    move( )
```

```
}
```

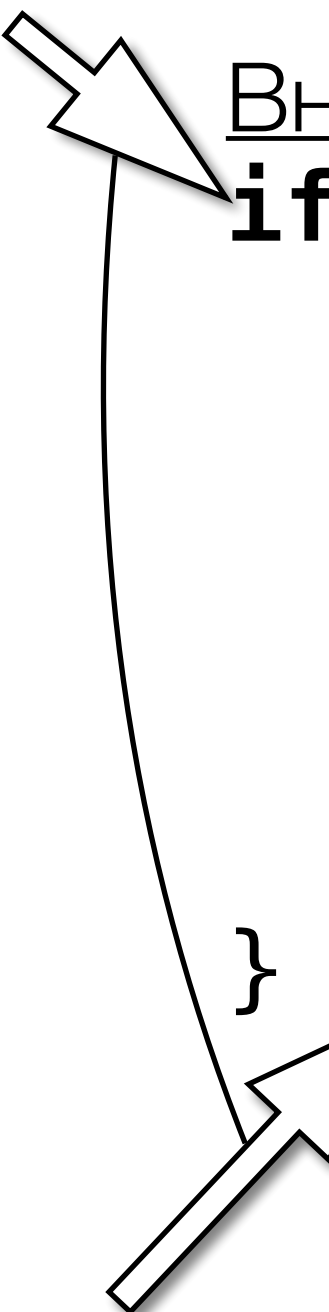
Операторы условного перехода

Вложенный if позволяет проверить два условия

```
if (frontIsBlocked){  
  
    if (leftIsBlocked){  
  
        turnRight()  
  
    }  
}  
else {  
  
    move()  
  
}
```

Операторы условного перехода

Внимание - Ахтунг - Attention



```
if (условие1) {  
  
    if (условие2) {  
  
        move()  
  
    }  
} else {  
    turnRight()  
  
}
```

Циклы

For: - позволяет выполнить команды конкретное количество раз

```
for _ in 0..4 {  
    move()  
}
```

While: - позволяет выполнять команды пока выполняется условие

```
while условие {  
    move()  
}
```

Циклы

break: позволяет оборвать цикл

```
while условие {  
    if (frontIsClear){  
        move()  
    } else {  
        break  
    }  
}  
put( )
```

Комментарии

- Комментарии служат для того чтобы лучше понимать программу и предназначены только для программистов то есть никак не отображаются на поведении программы
- Комментарии могут быть:
 - А. Многострочные
 - В. Однострочные

Комментарии

Многострочный комментарий выглядит так:

```
/* эта программа  
рассчитывает минимальный возраст  
девушки с которой вам можно встречаться  
при наличии у вас N-ного количества  
бабок */
```

Комментарии

Однострочные комментарий выглядит так:

```
// эта программа была написана
```

```
// следующими людьми:
```

```
// Бесараб Иван Васильевич
```

```
// Головач Елена Футурамовна ;)
```

Распространенные ошибки

- Скобки - (попарно, зеркально)
- Синтаксис
- Бесконечные циклы
- Off By One Bug (OBOB)

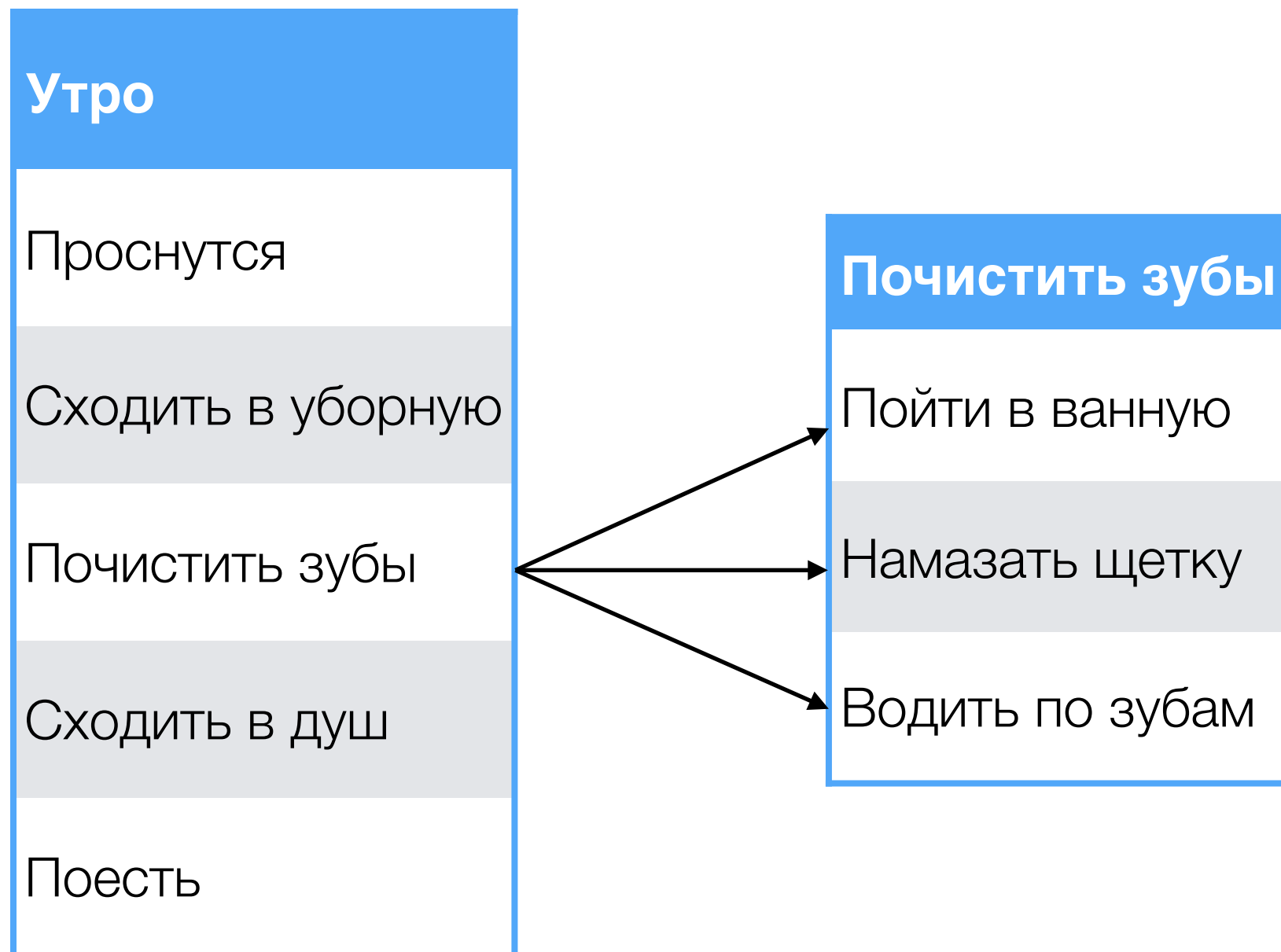
```
while frontIsClear {  
    put()  
    move()  
}
```

Декомпозиция

Декомпозиция - разложение.

Принцип декомпозиции состоит в разложении
большого сложного задания на несколько меньших

Декомпозиция



Декомпозиция

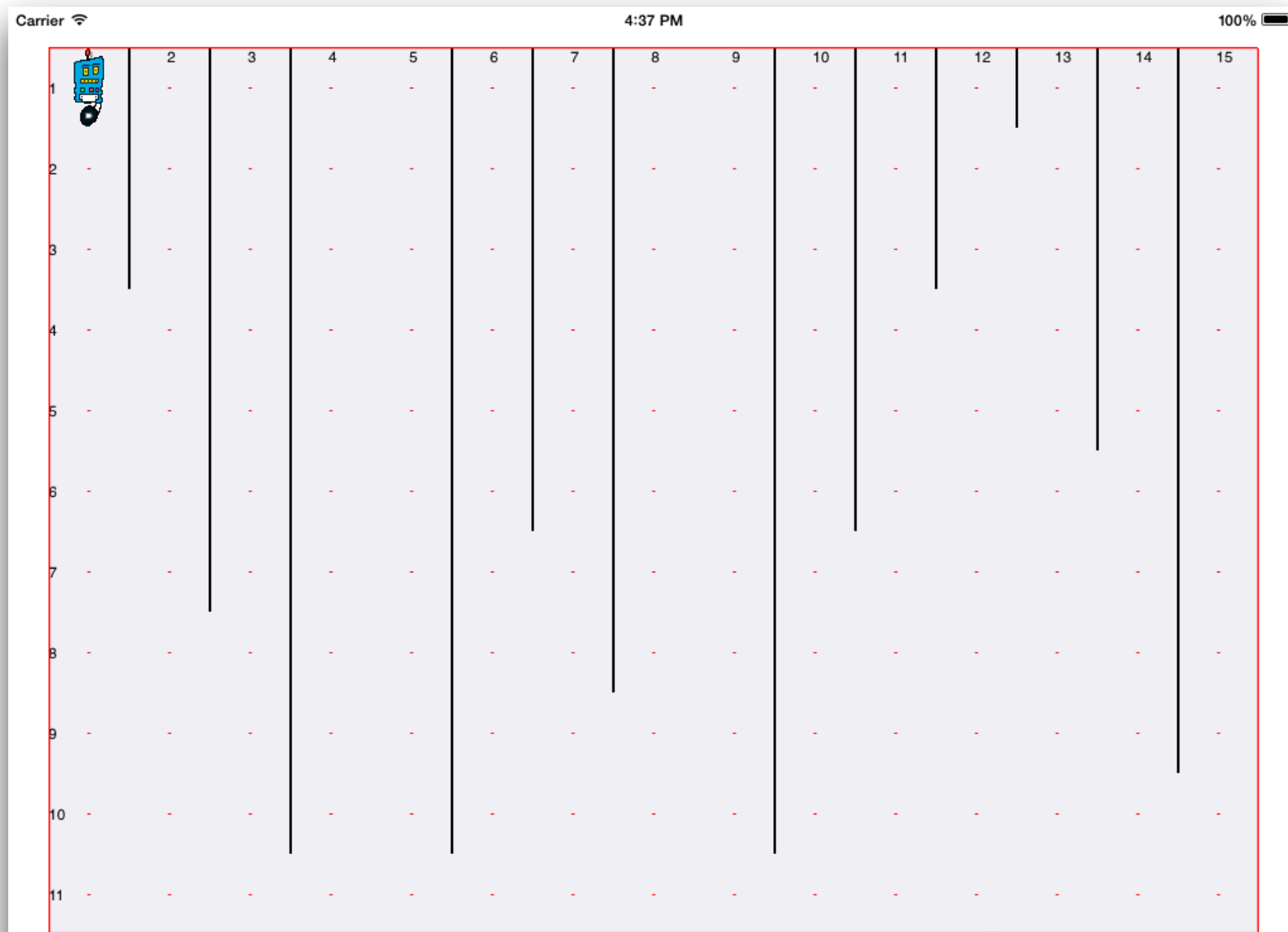
Мы начинаем на высоком уровне и движемся вниз пока не опустимся до таких примитивных конструкций которые будут понятны компьютеру.

Этот процесс называют Top Down Design.

Напротив существует Bottom Up Design

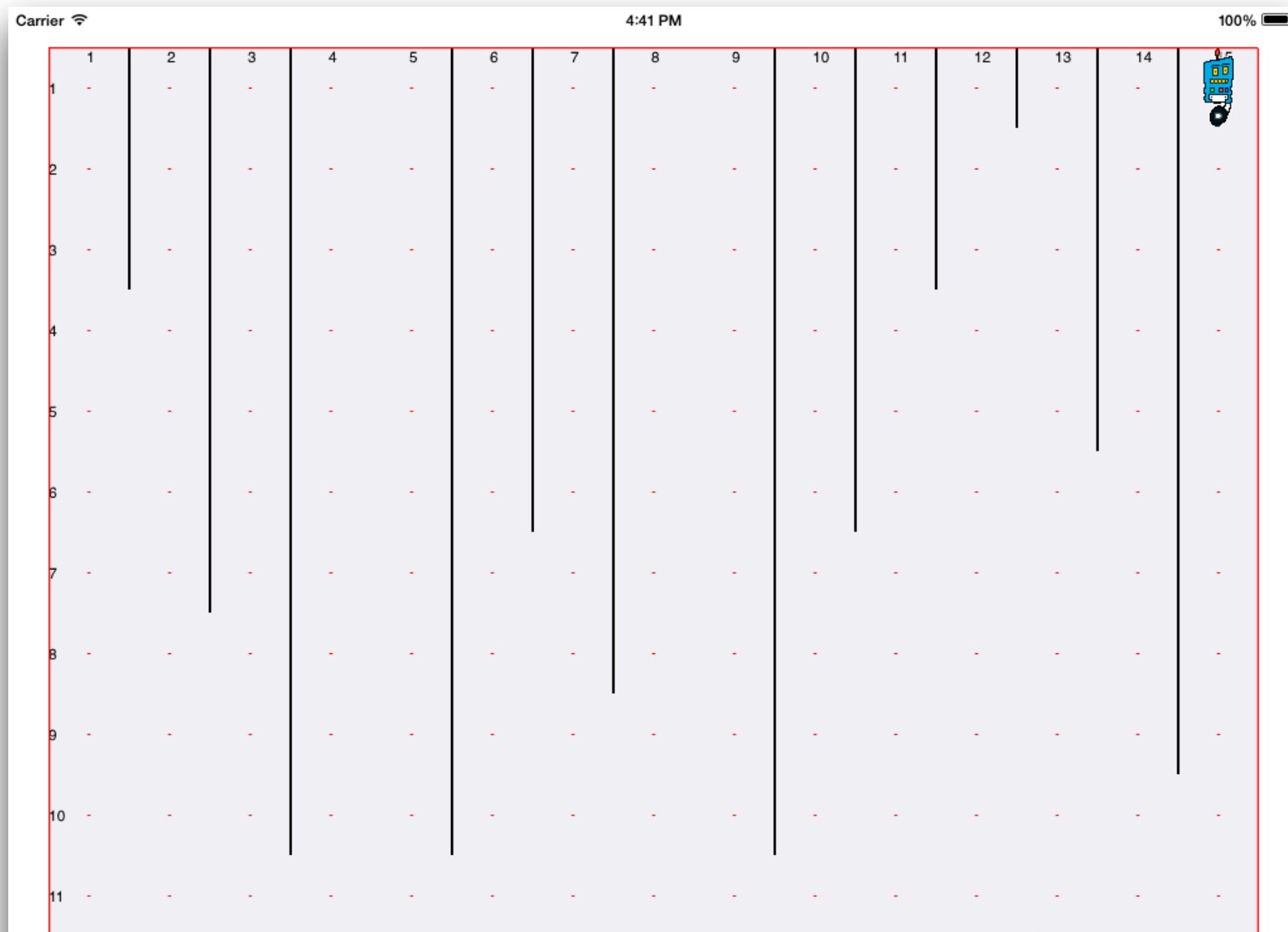
Вторая программа

Используем все что мы узнали чтобы написать более сложную программу: обойти все вершины



Вторая программа

Используем все что мы узнали чтобы написать более сложную программу: обойти все вершины



Вторая программа

Выводы

- программа должна быть как можно более универсальной
- программа должна быть написана так чтобы человеку было удобно понимать как она работает