Guião do trabalho de Física

Configuração inicial

Foi disponibilizado um projecto Kotlin muito simples para servir de ponto de partida para este trabalho em https://github.com/ULHT-Fisica/trabalho-fisica

Vão precisar de 2 ferramentas:

- Git - Uma ferramenta de controlo de versões que neste caso vos vai permitir descarregar o trabalho que está no github
- Intellij - Uma ferramenta de edição de código, através da qual vão poder alterar o programa base e executar o programa resultante.

Instalação do Git

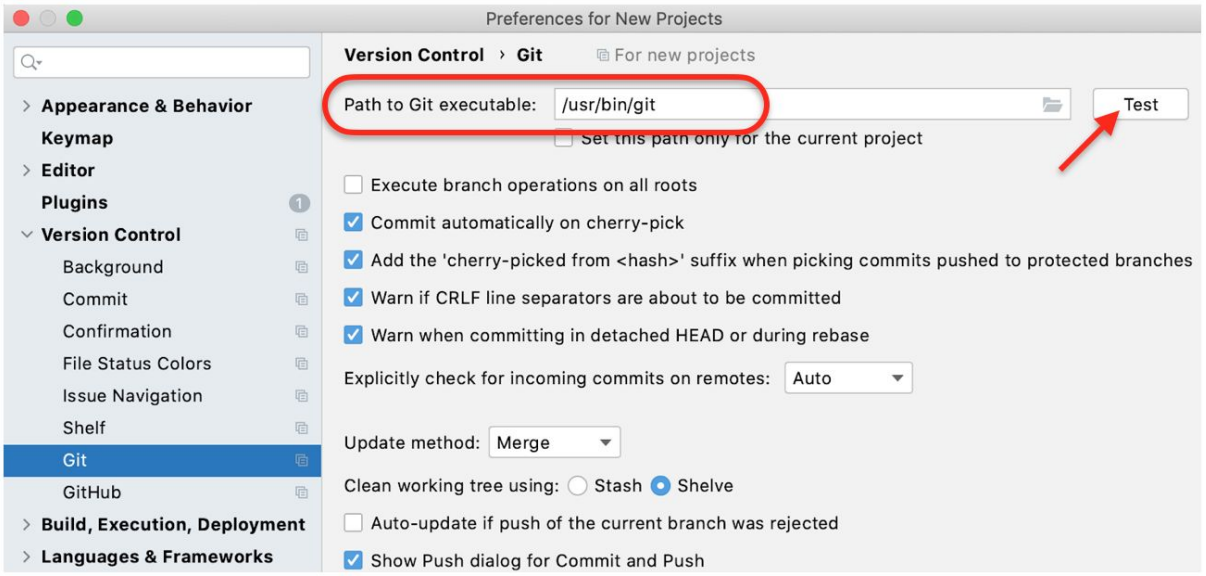
Aceder a https://git-scm.com/downloads, descarregar o instalador apropriado e executá-lo.

Instalação do Intellij

Aceder a https://www.jetbrains.com/idea/download, descarregar e instalar a versão Community que é gratuita e é suficiente para este trabalho.

Configuração do Git no Intellij

O Intellij inclui um plugin para integrar com a ferramenta Git. Após arrancarem o Intellij, acedam às preferências e procurem pelo ecrã de configuração do git, tal como apresentado aqui:



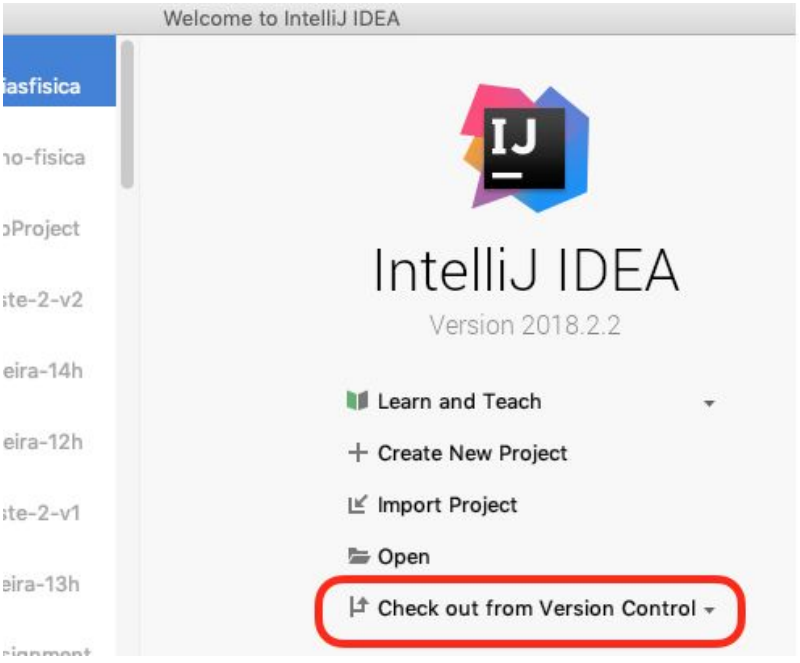
Introduzam a localização do executável do git (na figura está a localização em Mac, em Windows deverá ser qualquer coisa como C:\Program Files\Git\bin\git.exe) e usem o botão "Test" para verificar que ficou bem configurado.

Após estes passos, estão prontos para descarregar o trabalho do github.

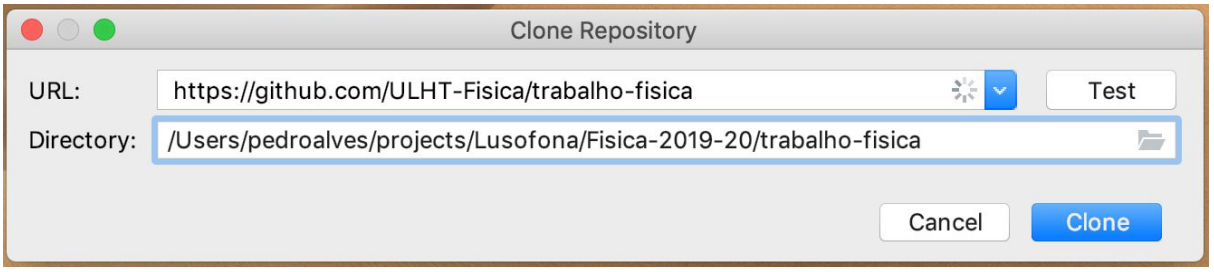
Descarregar o trabalho

Para usarem este projecto no Intellij, devem seguir estes passos:

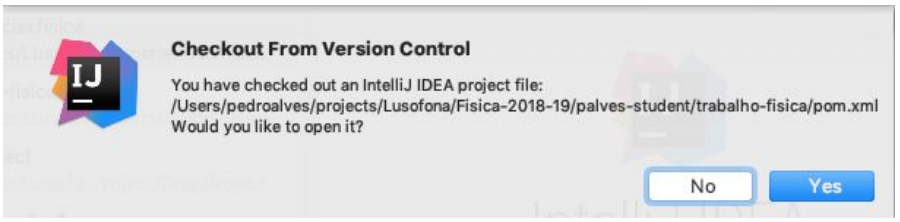
1. Abrir o Intellij e escolher a opção "Check out from Version Control":



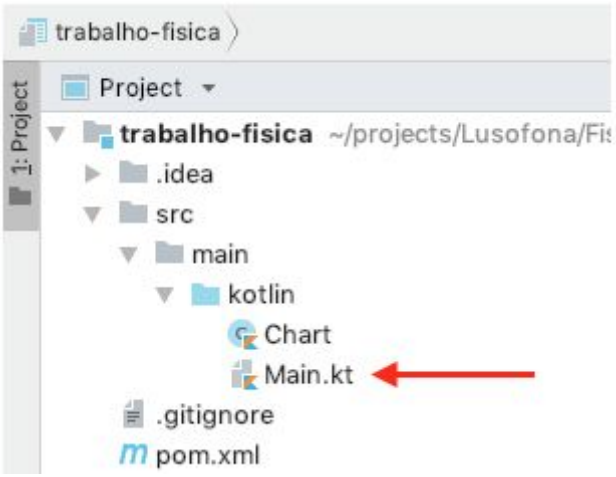
2. Preencher o campo URL com o endereço https://github.com/ULHT-Fisica/trabalho-fisica. No campo Directory devem indicar a pasta onde pretendem guardar o projecto.



3. Caso apareça esta mensagem, **devem responder "Yes"**.



4. Em princípio, o projecto já deverá correctamente configurado e pronto a correr. Podem verificar que têm os seguintes ficheiros no vosso projecto (notem onde ficou o Main.kt):



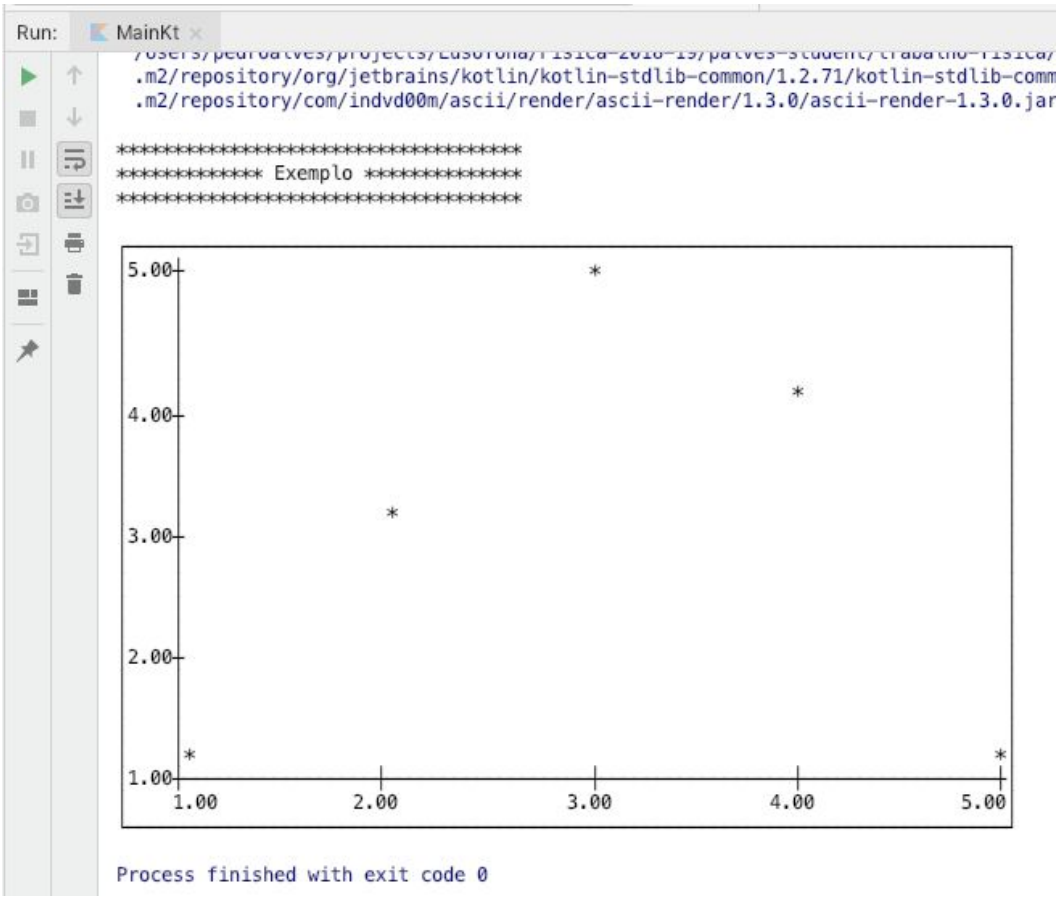
Esta estrutura de pastas não deve ser alterada. Os ficheiros Chart, .gitignore e pom.xml são ficheiros auxiliares do projecto. Não devem alterá-los. O único ficheiro que vão querer alterar é o Main.kt.

5. Se abrirem o Main.kt, verão um programa muito simples mas que faz uso de algumas primitivas que não aprenderam em Fundamentos de Programação. Essas primitivas serão explicadas na secção seguinte.

Main.kt

```kotlin
fun main(args: Array<String>) {

    // vou definir um gráfico com 80 caracteres de largura por 25 caracteres de altura
    val larguraGrafico = 80
    val alturaGrafico = 25
    val grafico = Chart(larguraGrafico, alturaGrafico)

    println()
    println("************************************")
    println("************ Exemplo **************")
    println("************************************")
    println()


    // vou adicionar 5 pontos ao gráfico
    // os pontos são representados por dois Double: o x e o y
    grafico.ponto(1.0, 1.0)
    grafico.ponto(2.0, 3.0)
    grafico.ponto(3.0, 5.0)
    grafico.ponto(4.0, 4.0)
    grafico.ponto(5.0, 1.0)

    // mando desenhar o gráfico com os pontos previamente introduzidos
    grafico.draw()

}
```

6. Podem experimentar correr esse programa, bastando clicar no triângulo verde que está do lado esquerdo da função main. Se tudo correr bem, deverão ver um gráfico similar a este:



Como desenhar gráficos em Kotlin

Os problemas que irão resolver neste trabalho implicam o desenho de pontos num gráfico, como puderam observar na secção anterior.

Para facilitar o vosso trabalho, e graças ao ficheiro Chart que foi incluído no vosso projecto, terão acesso a 3 primitivas que vos vão permitir desenhar gráficos muito facilmente.

1. **Criação do gráfico**
   O gráfico tem que começar por ser criado através das instruções:

   ```
   val larguraGrafico = 80
   val alturaGrafico = 25
   val grafico = Chart(larguraGrafico, alturaGrafico)
   ```

   A variável **grafico** é de um tipo "especial": não é um Int nem uma String nem nenhum dos tipos que estão habituados, é do tipo **Chart**. Por ser desse tipo, poderão usá-la mais à frente para desenhar gráficos.
   Reparem que podem indicar a largura e altura do gráfico. Podem mexer nestes valores se pretenderem ver o gráfico maior ou mais pequeno.

2. **Adicionar pontos ao gráfico**
   Para adicionar um ponto ao gráfico, basta indicarem as suas coordenadas x e y na seguinte primitiva (neste caso, x = 2.0 e y = 3.0):

   ```
   grafico.ponto(2.0, 3.0)
   ```

   Notem que as coordenadas são Doubles. Isto irá dar jeito para certos problemas.
   Podem usar a primitiva anterior várias vezes com coordenadas diferentes, para adicionar vários pontos.

3. **Desenhar o gráfico**
   Finalmente, depois de adicionarem os pontos todos, podem pedir ao programa que desenhe o gráfico respectivo utilizando a primitiva:

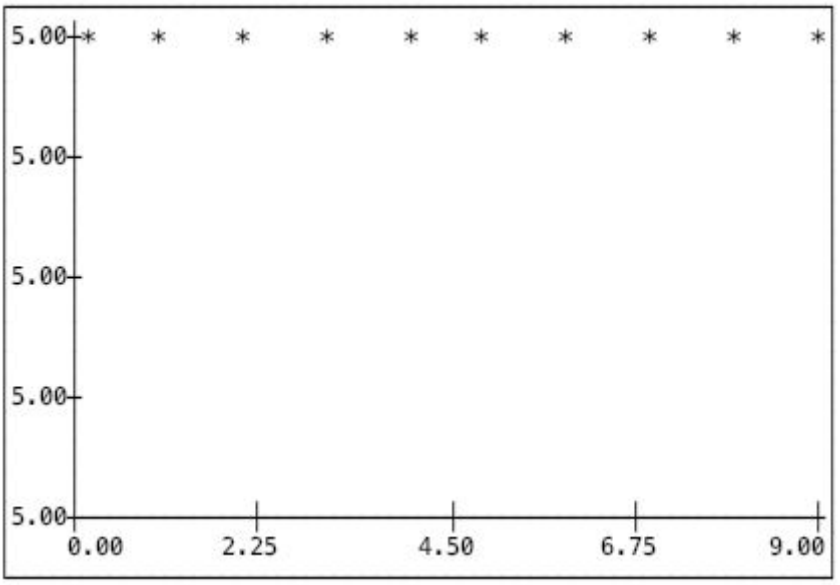
   ```
   grafico.draw()
   ```

Guião de trabalho para a aula prática

Para se familiarizarem com o projecto, deverão fazer as seguintes alterações, por esta ordem:

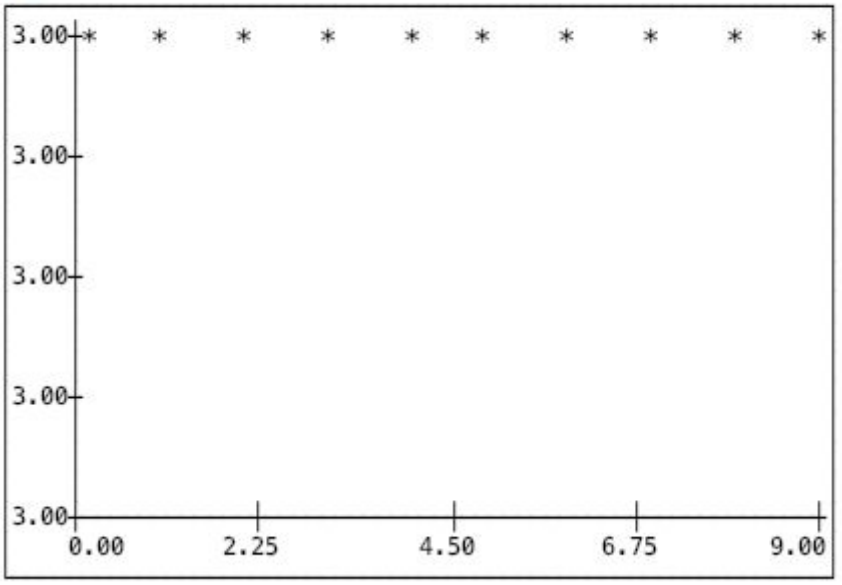
1. Alterem a altura e largura do gráfico para 20 e 60 respetivamente e corram novamente o programa. Notem que, dependendo da altura e largura seleccionados, pode haver "erros de arredondamento" no posicionamento dos pontos relativamente aos valores apresentados nos eixos horizontal e vertical. Não se preocupem com essa situação.

2. Alterem o programa para desenhar 10 pontos que formem um recta horizontal na altura 5.0. Deverão usar um ciclo **while** para o fazer.



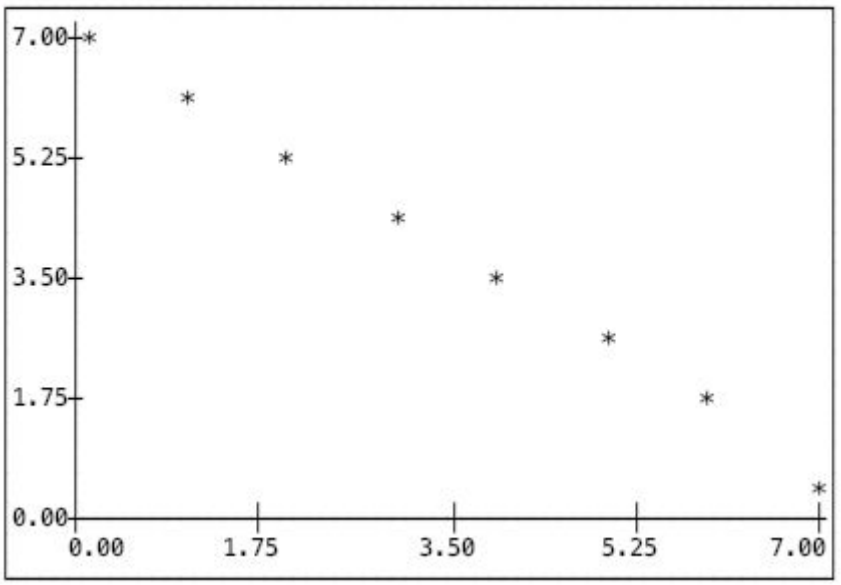
3. Agora vamos tornar o programa mais dinâmico. Em vez da altura estar fixa (no valor 5.0 neste caso), vamos pedir ao utilizador para introduzir a altura e usar o valor introduzido para desenhar o gráfico:

```
Introduza a altura
3.0
```



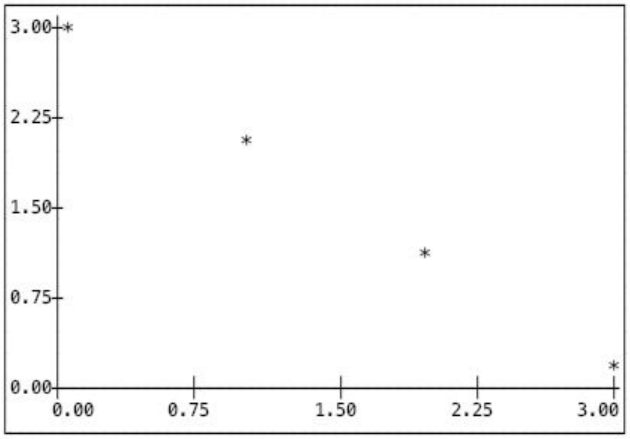
4. Este gráfico é pouco interessante. Vamos alterá-lo para ser uma recta em que o y continua a começar na altura introduzida mas vai diminuindo de forma proporcional ao x e para quando chega ao zero:

```
Introduza a altura
7
```

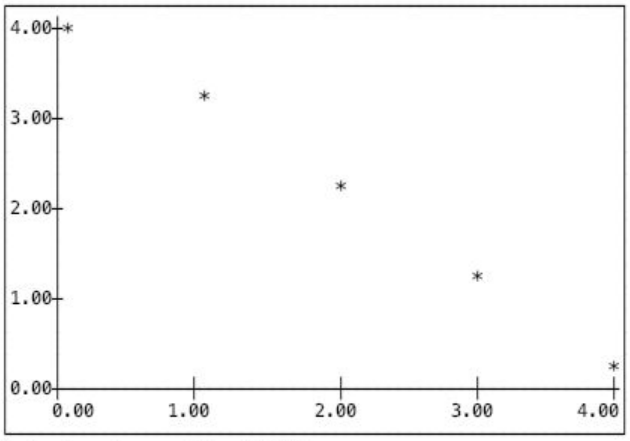


5. Para podermos mais facilmente experimentar diferentes alturas, vamos colocar todo o programa num ciclo, perguntando sempre ao utilizador no final de desenhar um gráfico se pretende desenhar outro gráfico. Caso ele diga que não, o programa termina, caso ele diga que sim, volta a perguntar a altura e a desenhar o gráfico. Deverão usar o ciclo **do..while**.

```
Introduza a altura
3
```




```
Pretende outro grafico (S/N)?
S
Introduza a altura
4
```




```
Pretende outro grafico (S/N)?
```

6. Agora estamos prontos para começar a resolver problemas de física com pequenas adaptações ao nosso programa.
   Ver secção seguinte.

Funções matemáticas em Kotlin

Nas aulas de Fundamentos de Programação, aprenderam apenas os operadores aritméticos básicos mas para resolver problemas de Física, terão que usar operadores mais avançados. Para começar terão que fazer import das funções matemáticas. Coloquem esta linha no ínicio do ficheiro:

```
import kotlin.math.*
```

Apresentam-se de seguida algumas operações:

- Número elevado a uma potência (ex: n^2). Deve-se usar a primitiva **pow**

```
val n : Double = ...

val numQuadrado = n.pow(2)

val numCubo = n.pow(3)
```

- Converter um ângulo em graus para um ângulo em radianos. O Kotlin trabalha sempre com ângulos em radianos por isso é necessário converter todos os ângulos para esta medida. Para isso basta aplicar a fórmula: angRadianos = angGraus * PI / 180

```
val anguloGraus = readLine()!!.toDouble()

val anguloRadianos = anguloGraus * PI / 180
```

- Calcular o seno e o cosseno. O seno e o cosseno devem ser aplicados a um ângulo em radianos, caso contrário vão dar resultados errados:

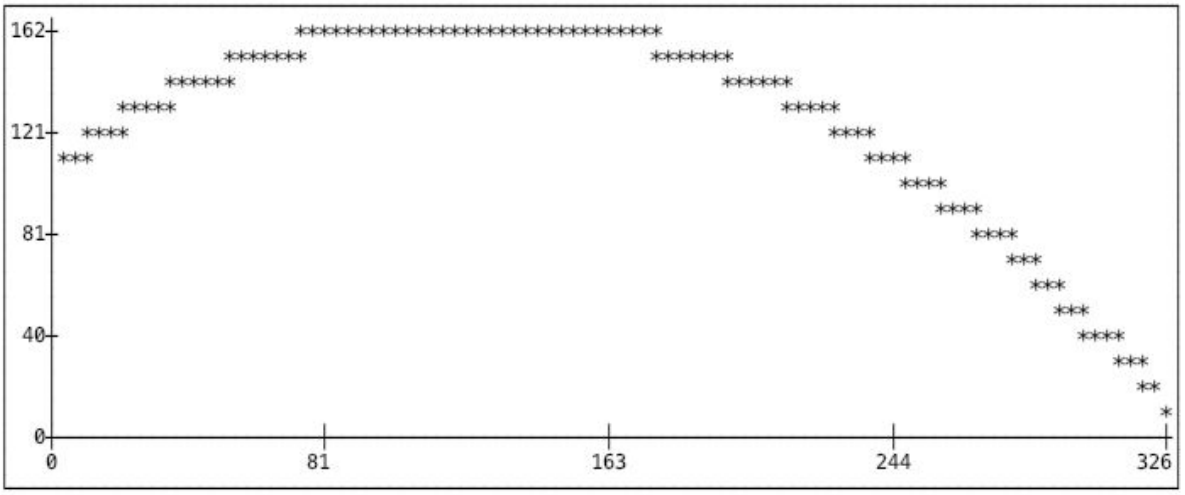
```
val seno = sin(anguloRadianos)

val cosseno = cos(anguloRadianos)
```

Problema da trajetória do projétil

Vamos alterar o programa para “desenhar” a trajetória de um projétil que é lançado de uma certa altura, com uma certa velocidade inicial e num certo ângulo.

Deverão pedir estes dados ao utilizador e guardá-los em variáveis. De seguida deverão desenhar a trajectória do projétil, usando as primitivas de gráficos e de cálculos matemáticos introduzidas anteriormente:

```
Introduza a altura
100
Introduza a velocidade inicial
50
Introduza o ângulo (0-90)
45
```




```
Mais um gráfico? (S/N)
```