Modularisation

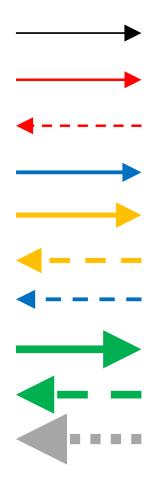
ISEP / SWitCH_QA / DESOFT I

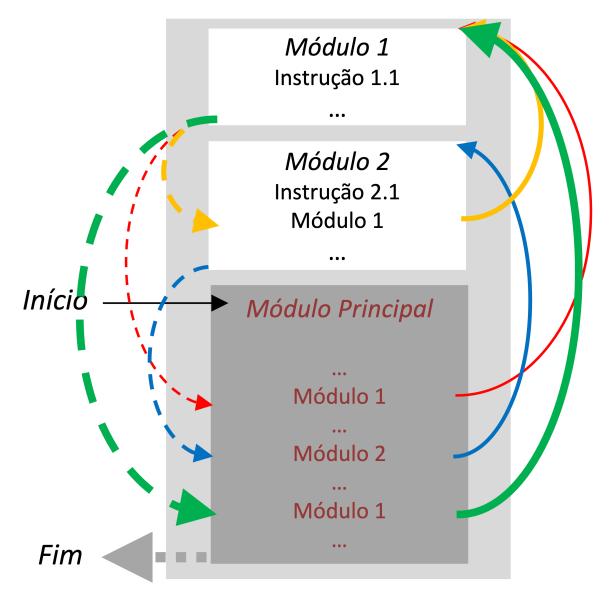
Modularisation

- A program can be composed of a Main Module and other modules
- Main Module
 - It is always the first module to be executed
 - Controls a program's execution
- When calling a module, control temporarily passes to the called module
- After the execution of a module, control returns to the module that called it and to the instruction following the one that originated the call

Control Flow

Sequência de execução:





Advantages

Decomposition

- Smaller subprograms
- Reduce development complexity
- Make development faster
- Independent modules can be developed in parallel

Reutilisation

- Reuse of modules
- Make development faster (avoid redundancy)
- Make the program smaller and simpler to read/understand

Abstraction

- Abstraction of task implementation
- Facilitate development
- A module's developer abstracts itself from implementation details of other modules
 - E.g, squareRoot(x)

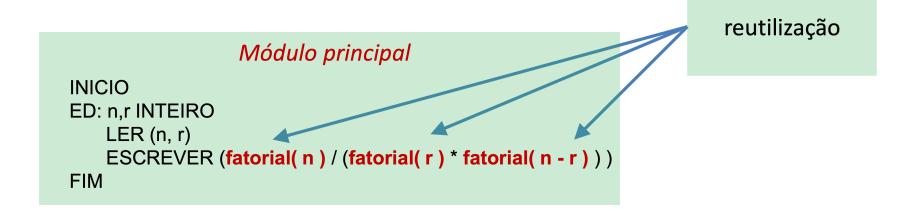
Example: demo-one.ts

Exemplo: Calcular combinações $C_{(n,r)} = \frac{11}{r! (n-r)!}$

```
INICIO
ED: n, r, resultado, fn, fr, fnr, i INTEIRO
                                                                       redundância
   LER (n, r)
//n!
   resultado ← 1
    REPETIR PARA i ← 2 ATE n PASSO 1
               resultado ← resultado * i
    FIMREPETIR
   fn ← resultado
//r!
   resultado ← 1
    REPETIR PARA i ← 2 ATE r PASSO 1
               resultado ← resultado * i
    FIMREPETIR
   fr ← resultado
//(n-r)!
   resultado ← 1
    REPETIR PARA i ← 2 ATE (n-r) PASSO 1
               resultado ← resultado * i
    FIMREPETIR
   fnr ← resultado
    ESCREVER (fn / (fr * fnr))
FIM
```

Example: demo-two.ts

Exemplo: Calcular combinações $C_{(n,r)} = \frac{n!}{r! (n-r)!}$



```
Módulo fatorial

DEFINIR INTEIRO fatorial (numero)

ED: resultado, i INTEIRO

resultado ← 1

REPETIR PARA i ← 2 ATE numero PASSO 1

resultado ← resultado * i

FIMREPETIR

RETORNAR resultado

FIM
```

Summary

So far,

• We've understood how to modularise your software, splitting it in smaller sized functions.

- You should try and apply this to your exercises.
 - Use your existing tests as regression testing.

Bibliography

- Adapted from LEI/APROG's, LEI/ESOFT's & SWITCH/DESOFT's slide decks
- "Programação, algoritmos e estruturas de dados"; João Pedro Neto;
 Escolar Editora