

Recapitulando

- Já aprendemos:
 - Valores
 - Como nos organizamos como turma
 - Valores em trabalho em equipe e desenvolvimento de software
 - Conteúdo
 - O que é um algoritmo
 - O que é um programa
 - Qual ferramenta utilizaremos
 - Representações de algoritmos
 - Operações de entrada e saída
 - O que são variáveis e constantes
 - Desvios condicionais (se e senão);
 - Operadores lógicos (E, OU ...).
 - Laços de repetição (enquanto);

Próximos Passos

- Subrotinas (Funções);
 - Recursividade;
 - Bibliotecas.
- Estruturas de dados (Vetores, Matrizes, Filas e Pilhas);

Lembra do código que verificava se já podíamos sair de casa?

```
programa
{
    funcao inicio () {
        logico acabou_coronavirus = falso
        enquanto (acabou_coronavirus == falso){
            acabou_coronavirus = verifica_pandemia()
            espera(1 dia)
        }// fim enquanto
        escreva("Podemos sair!!")
    }// fim inicio
}// fim programa
```

Podemos escrever a execução da subrotina (ou função, ou método) abaixo do programa início. A lógica é semelhante à função início

```
programa
{
    funcao inicio () {
        logico acabou_coronavirus = falso
        inteiro dias_parados = 0
        enquanto (acabou_coronavirus == falso){
            acabou_coronavirus = verifica_pandemia(dias_parados)
            dias_parados ++
        }
        escreva("Podemos sair!!")
    }
    funcao logico verifica_pandemia(inteiro dias_parados){
        se(dias_parados>15){
            retorne verdadeiro
        }
        retorne falso
    }
}
```

Podemos escrever a execução da subrotina (ou função, ou método) abaixo do programa início. A lógica é semelhante à função início

- Definição : Sequência de instruções executadas somente quando chamadas por um programa em execução
 - Devem executar **uma tarefa** específica
 - Um programa **pode conter diversas funções**, além da função principal **início()** , que é **obrigatória**
 - As funções executam **somente** quando chamadas à partir da função início()
 - Após a execução, o fluxo retorna ao ponto **imediatamente após** o da chamada da função
 - Uma função pode (ou não) **retornar um valor** ao bloco que a chamou
 - Uma função pode (ou não) **necessitar de um ou mais argumentos** ao ser chamada

Mais alguns exemplos – Repetição de código

```
programa {  
    funcao inicio(){  
        inteiro i  
        para(i=0; i<20; i++)  
            escreva("**")  
        escreva("\n")  
        escreva("Numeros entre 1 e 5\n")  
        para(i=0; i<10; i++)  
            escreva("**")  
        escreva("\n")  
        para(i=1; i<=5; i++)  
            escreva(i, "\n")  
        para(i=0; i<20; i++)  
            escreva("**")  
        escreva("\n")  
    }  
}
```

Note o código repetido. Se
tivermos que consertar,
teremos que fazer o mesmo
ajuste várias vezes



Saída:

Números entre 1 e 5

1
2
3
4
5

Mais alguns exemplos – Repetição de código

```
programa {  
    funcao inicio(){  
        inteiro i  
        escreve_linha()  
        escreva("Numeros entre 1 e 5\n")  
        escreve_linha()  
        para(i=1; i<=5; i++)  
            escreva(i, "\n")  
        escreve_linha()  
    }  
    funcao escreve_linha(){  
        para(i=0; i<20; i++)  
            escreva("x")  
        escreva("\n")  
    }  
}
```

Observe a diferença ao
encapsularmos esse código
repetido em uma função =D



Saída:

Números entre 1 e 5

1

2

3

4

5

Podemos também fazer a função chamar ela mesma para resolvermos problemas chamados recursivos.

Mas o que é recursão?

Podemos também fazer a função chamar ela mesma para resolvermos problemas chamados recursivos.

Mas o que é recursão?

- Em Matemática e Ciência da Computação, uma classe de métodos tem comportamento recursivo quando eles podem ser definidos por duas propriedades:
 - Um caso base simples (ou vários casos)
 - Um conjunto de regras que reduz todos os outros casos para o caso base

Exemplo : Fatorial de um número inteiro positivo!

$5 * (\text{fatorial de } 4)$

$5! = 5 * 4 * 3 * 2 * 1$

Recursividade

Fatorial Recursivo

```
programa {  
    funcao inteiro fatorial(inteiro  
        n){  
        se(n == 0){  
            retorne 1  
        } senao {  
            retorne n * fatorial( n - 1  
        )  
        }  
    }  
}
```

Execução : 4 fatorial

fatorial(4) -> 4 * 3 * 2 * 1

n = 4

retorne 4 * fatorial(3)

n = 3

retorne 3 * fatorial(2)

n = 2

retorne 2 * fatorial(1)

n = 1

retorne 1 *

fatorial(0)

retorne 1

Passos para escrever uma função recursiva

1. Escreva um protótipo da função recursiva
2. Escreva um comentário que descreve o que a função deve fazer
3. Determine o caso base (pode haver mais de um) e a solução desse caso
4. Determine qual é o problema menor do que o atual a ser resolvido
5. Use a solução do problema menor para resolver o problema maior.

Voltando ao coronavírus parte 3 ...

```
programa
{
    funcao inicio () {
        logico acabou_coronavirus = falso
        inteiro dias_parados = 0
        enquanto (acabou_coronavirus == falso){
            acabou_coronavirus = verifica_pandemia(dias_parados)
            dias_parados ++
        }
        escreva("Podemos sair!!")
    }
    funcao logico verifica_pandemia(inteiro dias_parados){
        se(dias_parados > 15){
            retorne verdadeiro
        }
        retorne falso
    }
}
```

Além do método/função/subrotina `verifica_pandemia`, temos mais alguma outra função?

Funções de bibliotecas

- Nós vimos várias funções como **escreva()**, **leia()**, **limpa()**.
- Estas funções são métodos padrões já disponíveis em qualquer programa do PortugolStudio. Além dessas funções, podemos adicionar outras funções através da importação de bibliotecas.

```
programa
{
    inclua biblioteca Matematica --> mat
    funcao inicio()
    {
        real numero = 4.0
        real raiz = mat.raiz(numero, 2.0) // Obtém a raiz quadrada do
        número
        escreva("A raiz quadrada de ", numero, " é: ", raiz, "\n")
    }
}
```

Recapitulando

- Já aprendemos:
 - Valores
 - Como nos organizamos como turma
 - Valores em trabalho em equipe e desenvolvimento de software
 - Conteúdo
 - O que é um algoritmo
 - O que é um programa
 - Qual ferramenta utilizaremos
 - Representações de algoritmos
 - Operações de entrada e saída
 - O que são variáveis e constantes
 - Desvios condicionais (se e senão);
 - Operadores lógicos (E, OU ...).
 - Laços de repetição (enquanto);
 - Subrotinas (Funções);

Próximos Passos

- Estruturas de dados (Vetores, Matrizes, Filas e Pilhas);