Estruturas de Dados - Revisão de Fundamentos de Programação 02

Aula passada

- O que vimos:
  - Constantes, variáveis e tipos de variáveis
  - Dados estruturados
  - Condicional
  - Loops
  - Pseudocódigo

Funções

- Funções:
  - criadas para executar alguma tarefa, resolver algum problema
  - executam uma sequência de passos
  - são chamadas pelo programa principal ou por outras funções
  - podem ou não retornar alguma informação

Protótipo e assinatura

- Como declarar o protótipo uma função?
  - tipo: define o tipo de informação que a função retorna (saída)
    - ex: **int**
  - nome: é o nome utilizado na chamada da função
    - ex: Fatorial
  - lista de parâmetros: conjunto de parâmetros/valores passados para a função (entrada)
    - ex: **int** n
  - Protótipo da função: **int** Fatorial (**int** n);
    - assinatura da função: **int** Fatorial (**int** n)

Definindo uma função

- Como definir uma função?
  - tipo
  - nome
  - lista de parâmetros
  - bloco de comandos: linhas de comando a serem executadas quando a função for chamada

Definindo uma função

- **int** Fatorial (**int** n) { [bloco de comandos] }
- no início de uma função, é importante declarar as variáveis usadas dentro do bloco de comandos dela
  - não é necessário declarar as variáveis dos parâmetros da função
- C: definição das funções devem ser feitas antes da main()
  - para criarmos uma função F que chama a outra função G, devemos primeiro criar a função G, e só então criamos a função F
  - pode-se colocar apenas o protótipo da função no começo, e definir a função posteriormente

Definindo uma função

Algoritmo: Fatorial(n)

Entrada: natural n

Saída: valor de $n!$

```
1 nn = n
2 para i = n − 1 até 1 faça
3 |   nn = nn · i
4 retorne nn
```

Definindo uma função

Algoritmo: Fatorial(n)

Entrada: natural n

Saída: valor de $n!$

```
1 inteiro i          //não é necessário!
2 nn = n
3 para i = n − 1 até 1 faça
4 |   nn = nn · i
5 retorne nn
```

Sobrecarga

- Sobrecarga de funções:
  - funções que tem o mesmo nome mas:
    - tem listas de parâmetros diferentes
    - tem um tipo de retorno diferente
  - compilador utiliza a função correta a partir dessas duas informações
  - C não permite, C++ permite
- Utilidade:
  - podemos ter uma função para somar inteiros, outro para vetores e outra para matrizes, todas com o mesmo nome
  - mesma coisa para outras operações, como produto
- Boa prática: não sobrecarregar

Chamando uma função

- Como utilizar uma função?
  - basta inserir uma chamada de função no código
    - $x =$ Fatorial(3)
    - $y =$ Fatorial(nn)
  - x e y devem ser do mesmo tipo de retorno da função Fatorial
    - a função pode não retornar nada (o tipo de retorno da função pode ser **void**
    - neste caso, não há necessidade de armazenar o valor retornado
  - os parâmetros passados devem ser do mesmo tipo que os parâmetros definidos no protótipo da função

Escopo de uma variável

- Representa a região (ou bloco) onde a variável pode ser utilizada
  - se uma variável x for declarada fora de todas as funções (inclusive a main()), dizemos que x é uma variável global
    - variáveis globais podem ser acessadas a qualquer momento por qualquer função
  - se uma variável y for declarada dentro de alguma função (inclusive a main()), dizemos que y é uma variável local
    - variáveis locais só podem ser utilizadas dentro da própria função
    - ex: uma variável declarada no main() não pode ser utilizada dentro de Fatorial
    - isso permite que nós declaremos variáveis de mesmo nome dentro de funções diferentes
    - ex: podemos declarar uma variável *nn* na função main()

Passagem de parâmetros

- Passagem de parâmetros:
  - os parâmetros passados devem ser do mesmo tipo que os parâmetros definidos no protótipo da função
  - x e y devem ser do mesmo tipo de retorno da função Fatorial
    - a função pode não retornar nada (o tipo de retorno da função pode ser **void**
    - neste caso, não há necessidade de armazenar o valor retornado

Passagem de parâmetros

- Passagem de parâmetros por valor:
  - ao passarmos uma variável como parâmetro para uma função, passamos somente o valor dela
    - permitimos o acesso ao valor da variável, mas não permitimos sua alteração
    - se $n = 3$, então $x = \text{Fatorial}(n)$ é semelhante a fazer $x = \text{Fatorial}(3)$
- o valor de n não será alterado dentro do bloco de comandos da função

Passagem de parâmetros

- Passagem de parâmetros por referência:
  - ao passarmos uma variável como parâmetro para uma função, permitimos que ela seja alterada
    - **int** Fatorial (**int** $^*n$)
    - $x =$ Fatorial($\&n$)
- o valor de n pode ser alterado dentro do bloco de comandos da função
- útil para reaproveitar memória
  - ex: quadrado de um número imaginário
  - ex: ordenar vetor
- não é necessário que a função retorne algo