Funções
Estrutura de Dados

Prof. Msc. Felipe Leivas Teixeira

Versão 1.0

Subprograma

- Um **subprograma** (às vezes chamado de sub-rotina) consiste de um trecho de código com estrutura semelhante à de um programa, que é executado somente quando acionado por outro trecho de código

Subprograma

- Um **subprograma** (às vezes chamado de sub-rotina) consiste de um trecho de código com estrutura semelhante à de um programa, que é executado somente quando acionado por outro trecho de código
- Um subprograma deve executar uma única tarefa, claramente definida

Subprograma

- Um **subprograma** (às vezes chamado de sub-rotina) consiste de um trecho de código com estrutura semelhante à de um programa, que é executado somente quando acionado por outro trecho de código
- Um subprograma deve executar uma única tarefa, claramente definida
- Um programa, ao utilizar um subprograma para executar uma tarefa, não deve se preocupar em como essa tarefa será executada

Subprograma

- Um **subprograma** (às vezes chamado de sub-rotina) consiste de um trecho de código com estrutura semelhante à de um programa, que é executado somente quando acionado por outro trecho de código
- Um subprograma deve executar uma única tarefa, claramente definida
- Um programa, ao utilizar um subprograma para executar uma tarefa, não deve se preocupar em como essa tarefa será executada
- A execução correta de um subprograma deve ser assegurada sempre que esse seja adequadamente chamado

Subprograma

- A utilização de um subprograma é uma técnica de programação que visa:

Subprograma

- A utilização de um subprograma é uma técnica de programação que visa:
  - a definição de trechos de códigos menores, mais fáceis de serem construídos e testados;

Subprograma

- A utilização de um subprograma é uma técnica de programação que visa:
  - a definição de trechos de códigos menores, mais fáceis de serem construídos e testados;
  - a diminuição do tamanho dos programas, pela eliminação de redundâncias, ao evitar que códigos semelhantes sejam repetidos dentro de um programa;

Subprograma

- A utilização de um subprograma é uma técnica de programação que visa:
  - a definição de trechos de códigos menores, mais fáceis de serem construídos e testados;
  - a diminuição do tamanho dos programas, pela eliminação de redundâncias, ao evitar que códigos semelhantes sejam repetidos dentro de um programa;
  - a construção mais segura de programas complexos, pela utilização de unidades menores (os subprogramas) já construídas e testadas;

Subprograma

- A utilização de um subprograma é uma técnica de programação que visa:
  - a definição de trechos de códigos menores, mais fáceis de serem construídos e testados;
  - a diminuição do tamanho dos programas, pela eliminação de redundâncias, ao evitar que códigos semelhantes sejam repetidos dentro de um programa;
  - a construção mais segura de programas complexos, pela utilização de unidades menores (os subprogramas) já construídas e testadas;
  - a reutilização de código em um programa ou em programas diferentes.

Funções

- Na linguagem C um subprograma é chamados de função

Funções

- Na linguagem C um subprograma é chamados de função
- Uma **função** é um subprograma que devolve um valor, resultante de um cálculo ou da execução de uma determinada tarefa, ao programa que o chamou por meio de seu nome

Funções

- Na linguagem C um subprograma é chamados de função
- Uma **função** é um subprograma que devolve um valor, resultante de um cálculo ou da execução de uma determinada tarefa, ao programa que o chamou por meio de seu nome
- Uma função pode ou não receber parâmetros. Parâmetros são valores que são passados para uma função

Funções

- Na linguagem C um subprograma é chamados de função
- Uma **função** é um subprograma que devolve um valor, resultante de um cálculo ou da execução de uma determinada tarefa, ao programa que o chamou por meio de seu nome
- Uma função pode ou não receber parâmetros. Parâmetros são valores que são passados para uma função
- Assim como uma variável a função deve ser declarada antes de ser usada

Funções

- Na linguagem C um subprograma é chamados de função
- Uma **função** é um subprograma que devolve um valor, resultante de um cálculo ou da execução de uma determinada tarefa, ao programa que o chamou por meio de seu nome
- Uma função pode ou não receber parâmetros. Parâmetros são valores que são passados para uma função
- Assim como uma variável a função deve ser declarada antes de ser usada
- O cabeçalho de declaração de uma função compreende seu tipo e nome, e, após, entre parênteses, se existirem, o tipo e o nome de cada um de seus parâmetros formais, separados por vírgula:

<tipo da função> <nome da função> ([<lista de parâmetros formais>|void])

Funções

- Na linguagem C um subprograma é chamados de função
- Uma **função** é um subprograma que devolve um valor, resultante de um cálculo ou da execução de uma determinada tarefa, ao programa que o chamou por meio de seu nome
- Uma função pode ou não receber parâmetros. Parâmetros são valores que são passados para uma função
- Assim como uma variável a função deve ser declarada antes de ser usada
- O cabeçalho de declaração de uma função compreende seu tipo e nome, e, após, entre parênteses, se existirem, o tipo e o nome de cada um de seus parâmetros formais, separados por vírgula:

  <tipo da função> <nome da função> ([<lista de parâmetros formais>|void])

- **<tipo da função>** especifica o tipo de dado que irá retornar de uma função, caso a função não tenha retorno o seu tipo será void

Funções

- Alguns exemplos de cabeçalhos de funções:

Funções

- Alguns exemplos de cabeçalhos de funções:
  - Uma função que não devolve valor e que não recebe parâmetros

    void foo()

Funções

- Alguns exemplos de cabeçalhos de funções:
  - Uma função que não devolve valor e que não recebe parâmetros

    void foo()
  - Uma função que devolve algum valor e que não recebe parâmetros

    float sorteio()

Funções

- Alguns exemplos de cabeçalhos de funções:
  - Uma função que não devolve valor e que não recebe parâmetros

    void foo()
  - Uma função que devolve algum valor e que não recebe parâmetros

    float sorteio()
  - Uma função que não devolve valor mas recebe valores como parâmetros

    void variasletras(int vezes, char carac)

Funções

- Alguns exemplos de cabeçalhos de funções:
  - Uma função que não devolve valor e que não recebe parâmetros

    void foo()
  - Uma função que devolve algum valor e que não recebe parâmetros

    float sorteio()
  - Uma função que não devolve valor mas recebe valores como parâmetros

    void variasletras(int vezes, char carac)
  - Uma função que devolve algum valor e recebe valores como parâmetros

    int incrementa(int num)

Funções

- Alguns exemplos de cabeçalhos de funções:
  - Uma função que não devolve valor e que não recebe parâmetros

    void foo()
  - Uma função que devolve algum valor e que não recebe parâmetros

    float sorteio()
  - Uma função que não devolve valor mas recebe valores como parâmetros

    void variasletras(int vezes, char carac)
  - Uma função que devolve algum valor e recebe valores como parâmetros

    int incrementa(int num)
- Toda a função que tiver um retorno definido, dentro de sua implementação deve utilizar o comando **return** para devolver o valor, e em sua chamada deve ter uma variável que receba o retorno da função

Funções

EXEMPLO FUNÇÕES

Passagens de Parâmetros

- Uma coisa importante nas funções são as passagens de parâmetros

Passagens de Parâmetros

- Uma coisa importante nas funções são as passagens de parâmetros
- Existem dois tipos de passagem de parâmetros:

Passagens de Parâmetros

- Uma coisa importante nas funções são as passagens de parâmetros
- Existem dois tipos de passagem de parâmetros:
  - **Passagem por Valor**: o parâmetro é copiado para a função e qualquer alteração feita neste parâmetro dentro da função não é vista fora da função

Passagens de Parâmetros

- Uma coisa importante nas funções são as passagens de parâmetros
- Existem dois tipos de passagem de parâmetros:
  - **Passagem por Valor**: o parâmetro é copiado para a função e qualquer alteração feita neste parâmetro dentro da função não é vista fora da função
  - **Passagem po Referência**: é passado pra função uma referência de um valor e qualquer alteração feita neste parâmetro dentro da função é vista fora dela

Passagens de Parâmetros

- Uma coisa importante nas funções são as passagens de parâmetros
- Existem dois tipos de passagem de parâmetros:
  - **Passagem por Valor**: o parâmetro é copiado para a função e qualquer alteração feita neste parâmetro dentro da função não é vista fora da função
  - **Passagem po Referência**: é passado pra função uma referência de um valor e qualquer alteração feita neste parâmetro dentro da função é vista fora dela
- Em C não existe passagem de parâmetros por referência

Passagens de Parâmetros

- Uma coisa importante nas funções são as passagens de parâmetros
- Existem dois tipos de passagem de parâmetros:
  - **Passagem por Valor**: o parâmetro é copiado para a função e qualquer alteração feita neste parâmetro dentro da função não é vista fora da função
  - **Passagem po Referência**: é passado pra função uma referência de um valor e qualquer alteração feita neste parâmetro dentro da função é vista fora dela
- Em C não existe passagem de parâmetros por referência
- Mas a passagem de parâmetros por referência pode ser "simulado"por meio de ponteiros

Passagens de Parâmetros

- Uma coisa importante nas funções são as passagens de parâmetros
- Existem dois tipos de passagem de parâmetros:
  - **Passagem por Valor**: o parâmetro é copiado para a função e qualquer alteração feita neste parâmetro dentro da função não é vista fora da função
  - **Passagem po Referência**: é passado pra função uma referência de um valor e qualquer alteração feita neste parâmetro dentro da função é vista fora dela
- Em C não existe passagem de parâmetros por referência
- Mas a passagem de parâmetros por referência pode ser "simulado"por meio de ponteiros
- Isso também permite uma função retornar mais de um valor

EXEMPLO PASSAGEM DE PARÂMETROS

Recursividade

- **Recursão** é um método que resolve um problema dividindo o mesmo em subproblemas mais simples

Recursividade

- **Recursão** é um método que resolve um problema dividindo o mesmo em subproblemas mais simples
- Quando os subproblemas se tornam *triviais*, eles são resolvidos diretamente

Recursividade

- **Recursão** é um método que resolve um problema dividindo o mesmo em subproblemas mais simples
- Quando os subproblemas se tornam *triviais*, eles são resolvidos diretamente
- Um exemplo clássico de problema que pode ser resolvido com recursão é o calculo fatorial

Recursividade

- **Recursão** é um método que resolve um problema dividindo o mesmo em subproblemas mais simples
- Quando os subproblemas se tornam *triviais*, eles são resolvidos diretamente
- Um exemplo clássico de problema que pode ser resolvido com recursão é o calculo fatorial
- O fatorial(N) funciona da seguinte forma:

Recursividade

- **Recursão** é um método que resolve um problema dividindo o mesmo em subproblemas mais simples
- Quando os subproblemas se tornam *triviais*, eles são resolvidos diretamente
- Um exemplo clássico de problema que pode ser resolvido com recursão é o calculo fatorial
- O fatorial(N) funciona da seguinte forma:
  - **se N = 0 então fatorial(N) = 1**
  - **se N > 0 então fatorial(N) = N * fatorial(N-1)**

Recursividade

- **Recursão** é um método que resolve um problema dividindo o mesmo em subproblemas mais simples
- Quando os subproblemas se tornam *triviais*, eles são resolvidos diretamente
- Um exemplo clássico de problema que pode ser resolvido com recursão é o calculo fatorial
- O fatorial(N) funciona da seguinte forma:
  - **se N = 0 então fatorial(N) = 1**
  - **se N > 0 então fatorial(N) = N * fatorial(N-1)**
- A recursividade se dá neste caso na situação que o resultado do fatorial(N) depende do fatorial(N-1).

Recursividade

EXEMPLO RECURSIVIDADE

Exercícios

1. Implemente uma função que receba quatro números inteiros e retorne a soma dos três maiores números, dentre os quatro recebidos.
2. Implemente uma função que receba três números reais, 'a', 'b' e 'c', que são os coeficientes de uma equação do segundo grau e retorne o valor do delta, que é dado por '$b^2 - 4ac$'.
3. Implemente uma função que recebe como parâmetros dois valores inteiros, por meio de passgem de parâmetros, e troque o conteúdo deles.
4. Implemente uma função recursiva que recebe um número 'n' e retorna n-ésimo numero da sequencia de fibonacci. A sequencia de fibonacci funciona da seguinte forma:
   se n = 1 então fibonacci(n) = 0
   se n = 2 então fibonacci(n) = 1
   se n > 2 então fibonacci(n) = fibonacci(n-1)+fibonacci(n-2)

Funções

Estrutura de Dados

Prof. Msc. Felipe Leivas Teixeira

Versão 1.0