



Cálculo Numérico Computacional
programação e raízes
T. Fraciano-Pereira

Lista 04
tarcisio@member.ams.org
Dep. de Computação

alun@:

Univ. Estadual Vale do Acaraú	26 de agosto de 2009
página da disciplina	www.calculo-numerico.sobralmatematica.org/
Documento produzido com L ^A T _E X	sis. op. Debian/Gnu/Linux

0.2 Informações

Por favor, se você usar o método medieval para entrega desta lista, em papel, prenda esta *folha de rosto* na solução, preenchendo com os seus dados, ela será usada na correção. Se você quiser entregar o trabalho eletronicamente, envie o arquivo para o meu e-mail ou entregue em CD na secretária do Curso de Computação. Por favor, siga as instruções sobre nomes de arquivos, leia as intruções na página da disciplina. Data da entrega da lista: dia 04 de Maio, segunda-feira. Se o trabalho for feito em equipe, basta um único trabalho ser entregue e neste caso, no cabeçalho, devem estar os nomes completos de tod@s @s alun@s junto com os seus respectivos e-mails. O número de membros de uma equipe não devem ultrapassar três.

0.2.1 Objetivo

Palavras chave raízes de funções polinomiais, método da tangente, método da secante, método da busca binária, recursividade.

0.2.2 Avaliação do trabalho

Leia na página da disciplina a este respeito.

0.3 Exercícios

1. Os primeiros programas

- (a) $(V)[](F)[]$ `cout` é um objeto que representa a saída de dados em C++, quer dizer, o comando
- ```
cout << "Estou aprendendo a programa";
```
- imprime na tela a frase que está entre aspas.
- (b)  $(V)[ ](F)[ ]$  `cout` é um objeto que representa a saída de dados em C++, quer dizer, o comando
- ```
cout << "Estou aprendendo a programa";
```
- imprime na tela a frase que está entre aspas e produz uma troca de linha.

- (c) $(V)[](F)[]$ `cout` é um objeto que representa a saída de dados em C++, quer dizer, o comando
- ```
cout << "Estou aprendendo a programa" << endl;
```
- imprime na tela a frase que está entre aspas e produz uma troca de linha.
- (d)  $(V)[ ](F)[ ]$  Todo programa em C++, obrigatoriamente, tem uma função `main()` que é a gerente do programa. Ela é que chama as demais funções.
- (e)  $(V)[ ](F)[ ]$  O programa `segundo01.cc` tem um erro, falta o *direcionador* `<<` na frente de uma das mensagens.
- (f)  $(V)[ ](F)[ ]$  O programa `segundo01.cc` usa duas bibliotecas, elas são incluídas no programa com o comando `include`.
- (g)  $(V)[ ](F)[ ]$  O programa `segundo01.cc` define uma variável do tipo **Ambiente**, chamada `Tela` que na verdade é um *objeto* usado no programa no comando
- ```
Tela.entrada_int()
```
- em que é feita uma entrada de dados do programa.
- (h) $(V)[](F)[]$ O programa `segundo01.cc` tem duas funções definidas no mesmo arquivo em que está `main()`, são as funções `rotulo()` e `adicao()`.

2. O programa `raizes.cc`

Leia o programa que se encontra no arquivo `raizes.cc` no link "programas" da página. Leia também o arquivo `raizes.h` que se encontra no mesmo link.

- (a) $(V)[](F)[]$ Este programa tem uma única função, `main()` que chama as demais funções do projeto. Nesta função estão definidas cinco variáveis: `malha`, `data`, `quantidade`, `cout` e `noticia()`.
- (b) $(V)[](F)[]$ Este programa tem uma única função, `main()` que chama as demais funções do projeto. Nesta função estão definidas três variáveis: `mal`, do tipo `malha`, `data`, do tipo `resultado` e `quantidade` do tipo `int`. Os tipos, `malha` e `resultado` estão definidos em `raizes.h`.
- (c) $(V)[](F)[]$ A função `main()`, no programa `raizes.cc` chama apenas as seguintes funções
- ```
Tela.apeteco2(), varredura() e relatorio_final()
```
- (d)  $(V)[ ](F)[ ]$  A função `main()`, no programa `raizes.cc` chama as seguintes funções
- ```
noticia(), entrada_dados(), escreve_a_malha(),
Tela.apeteco2(), varredura() e relatorio_final()
```
- (e) $(V)[](F)[]$ Algumas das funções que `main()` chama, se encontram definidas no mesmo arquivo `tt raizes.cc` em que `main()` está definida.

- (f) (V)[](F)[] Todas as funções que `main()` chama, se encontram definidas na biblioteca `tt raizes.h`.
- (g) (V)[](F)[] A função das bibliotecas, em programação, é a de *esconder do usuário* a forma como o programa foi estruturado.
- (h) (V)[](F)[] A função de uma biblioteca é a de separar o planejamento do programa, representado pela função `main()` da implementação das rotinas que ficam em algumas bibliotecas.
- (i) (V)[](F)[] O programa `raizes.cc` inclui as três bibliotecas, `iostream.h`, `Ambiente.h`, `raizes.h` - embora na chamada à biblioteca `iostream` não apareça a extensão “.h” como aparece nas outras duas.
- (j) (V)[](F)[] O programa `raizes.cc` precisa de uma única biblioteca, `raizes.h`, e, indiretamente, precisa de mais algumas outras bibliotecas chamadas pela biblioteca `raizes.h`.
- (k) (V)[](F)[] O papel da função `noticia()`, definida em `raizes.h`, é de apresentar o programa para o usuário.
3. A biblioteca `raizes.h` Leia a biblioteca se encontra no arquivo `raizes.h`, no link “programas” da página.
- (a) (V)[](F)[] Na biblioteca `raizes.h` estão definidas 13 funções que são chamadas, diretamente ou indiretamente por `main()`.
- (b) (V)[](F)[] Na biblioteca `raizes.h` estão definidas 10 funções que são chamadas, diretamente ou indiretamente por `main()`.
- (c) (V)[](F)[] Numa biblioteca pode haver funções que **não** serão chamada pela função `main()` - quer dizer, *uma biblioteca pode ter funções que serão utilizadas por outro programa*. Elas são de “multi-uso”! Mas este não é o caso de `raizes.h`.
- (d) (V)[](F)[] No início da biblioteca `raizes` estão definidos dois tipos de dados denominados `malha`, `resultado`.
- (e) (V)[](F)[] No início da biblioteca `raizes` estão definidos dois tipos de dados denominados `malha`, `resultado`. Também é declarada uma variável “Tela” do tipo *Ambiente*.
- (f) (V)[](F)[] A biblioteca `raizes.h` inclui 4 bibliotecas que serão usadas direta ou indiretamente pelo programa `raizes.cc`, a saber `iostream`, `fstream`, `math`, `Ambiente` em alguns casos parece que é necessário aparecer a extensão “.h” e noutros não.
- (g) (V)[](F)[] No início da biblioteca `raizes.h` há um comentário que *sugere* que no futuro o programa `raizes.cc` vai ser refeito com *orientação a objeto*, mas neste momento ele não tem *esta metodologia*.
- (a) (V)[](F)[] A função `escreve_a_malha()` é do tipo `void`, quer dizer que ela não devolve nada - não tem no final a função `return()`. Ela recebe um parâmetro do tipo `malha` e descreve esta malha, quando chamada.
- (b) (V)[](F)[] A função `escreve_a_malha()` é do tipo `void`, quer dizer que ela não devolve nada - mas tem no final a função `return()`.
- (c) (V)[](F)[] A função `escreve_intervalo()` é do tipo `void`, quer dizer que ela não devolve nada - e não tem, ao final, a função `return()`. Ela recebe dois parâmetros do tipo `float` e escreve o um intervalo tendo estes dois números “reais” como extremos.
- (d) (V)[](F)[] A função `escreve_intervalo()` é do tipo `void`, quer dizer que ela vai devolver uma tabela com dois campos, um deles (dos campos) é do tipo `float` e se chama `raiz` e outro é do tipo `texto - string` e se chama `metodo` - Observe, sem acentuação.
- (e) (V)[](F)[] A função `seleciona_metodo()` é do tipo `resultado`, quer dizer que ela vai devolver uma tabela com dois campos, um deles (dos campos) é do tipo `float` e se chama `raiz` e outro é do tipo `texto - string` e se chama `metodo` - Observe, sem acentuação.
- (f) (V)[](F)[] As funções `seleciona_metodo()`, `raiz_tangente()`, `raiz_secante()`, `resultado_raiz_binaria()` são do tipo `resultado`, quer dizer devolvem uma tabela com os campos `raiz`, `metodo` - Observe, sem acentuação.
- (g) (V)[](F)[] As duas funções `raiz_exata()`, `relatorio_final()` são do tipo `void`, quer dizer que elas não tem a função `return()` ao final.
- (h) (V)[](F)[] A função `verifica_raiz_tangente()` é do tipo `int` quer dizer que, quando termina o seu trabalho, devolve um dado tipo `int`.
4. As funções da biblioteca `raizes.h` Leia esta biblioteca.