Programação de Computadores

Aula #06-B

Organização do Código

O que é manutenibilidade? Técnicas para aprimorar a manutenibilidade de um código-fonte? Como funciona a quebra de arquivos em C?

Ciência da Computação - BCC e IBM - 2024/01
Prof. Vinícius Fülber Garcia

Organização de Código

À medida que nosso arcabouço de conhecimentos aumenta, as possibilidades de programação também aumentam e, naturalmente, é possível desenvolver programas mais complexos.

Programas complexos, tipicamente (há exceções), se traduzem em programas grandes, **robustos em quantidade de linhas de código**.

Organização de Código

Naturalmente, códigos-fonte de programas complexos exigem um tempo de desenvolvimento maior, que pode ser de dias, semanas, meses, ou anos.

Ainda, códigos-fonte convertidos em programas em produção exigem atualização e correções periódicas.

MANUTENIBILIDADE

A Tal da Manutenibilidade

Existem diversas características desejáveis para garantir a manutenibilidade de um código-fonte. Algumas delas já são conceitos conhecidos para nós:

- Estruturação e indentação
- Nomes intuitivos
- Boa apresentação ao usuário
- Documentação e comentários

A Tal da Manutenibilidade

```c
#include <stdio.h>

            #define a 3.1416

#include <math.h>

int main(){float b; scanf("%f", &b); float c = a
* pow(b, 2);
   printf("Resultado: %f", c);
return 0;
}
```

```c
#include <stdio.h>
#include <math.h>

#define PI 3.1416

/*Programa que recebe o raio de um círculo e
apresenta a área do mesmo como resultado*/
int main(){
    float raio;
    printf("Digite o raio do círculo: ");
    scanf("%f", raio);
    float area = PI * pow(raio, 2);
    printf("Área do círculo: %f\n", area);
    return 0;
}
```

A Tal da Manutenibilidade

A modularização de um código-fonte pode ser realizada em níveis diferentes:

- ➔ Modularização em funções (já estudada)
- ➔ Modularização em arquivos (iremos estudar)

Além de serem tecnicamente diferentes, esses tipos de modularização também apresentam **“semânticas” heterogêneas**.

A Tal da Manutenibilidade

Vou trabalhar com **funções matemáticas** para calcular o **zero de funções de segundo grau**

Funções matemáticas indica o meu escopo de trabalho (math.h)

Zero de funções de segundo grau indica o meu objetivo específico (pow, sqrt)

Modularização em Arquivos em C

Quebrar um programa em arquivos nada mais é do que **definir bibliotecas** para serem utilizadas em um problema específico

Em C, criar bibliotecas consiste em definir dois arquivos específicos:

- .c: arquivo de código, onde existem as implementações das funções pertencentes a uma determinada biblioteca
- .h: arquivo de cabeçalhos, onde são declarados os protótipos das funções pertencentes a um arquivo .c relacionado (interface)

NA VERDADE, É UM POUCO DIFERENTE, MAS VAMOS ASSUMIR ISSO POR ENQUANTO

Modularização em Arquivos em C

Sendo assim, passos genéricos para realizar a quebra de um código fonte em C podem ser:

1. Quebre o programa em funções
2. Separe as funções de mesmo escopo (mesmo tópico) em um novo arquivo .c
3. Crie um arquivo .h (com o mesmo nome e no mesmo diretório do arquivo .c recém criado)
4. Faça a inclusão do arquivo .h nos arquivos de código-fonte (.c) que utilizam as funções presentes no arquivo .h

Modularização em Arquivos em C

E como incluir o .h das nossas bibliotecas em nossos códigos-fonte?

Não utilizaremos o *include <biblioteca.h>*

Utilizaremos o *include “biblioteca.h”*

Modularização em Arquivos em C

NÃO É UMA BOA PRÁTICA (PELO CONTRÁRIO)

Incluir lógica real no arquivo de cabeçalhos (.h)

Fazer a inclusão de arquivos .c

Modularização em Arquivos em C

```c
#include <stdio.h>
#include <math.h>
#define PI 3.1416;

int main(void) {
 float raio;
  printf("Digite o raio de um círculo: ");
 scanf("%f", &raio);
 float diametro = 2 * raio;
 printf("\nO diâmetro do círculo é: %f", diametro);
 float area = pow(raio, 2) * PI;
 printf("\nA área do círculo é: %f", area);
 float comprimento = 2 * raio * PI;
 printf("\nO comprimento do círculo é: %f\n", comprimento);
 return 0;
}
```

Exemplo!

Elementos de uma círculo/circunferência

Fase I

Modularização em Arquivos em C

```c
#include <stdio.h>
#include <math.h>
#define PI 3.1416;

float circ_diametro(float raio){
 return 2 * raio;
}

float circ_area(float raio){
 return pow(raio, 2) * PI;
}

float circ_comprimento(float raio){
 return 2 * raio * PI;
}
```

```c
int main(void) {
 float raio;
  printf("Digite o raio de um círculo: ");
 scanf("%f", &raio);

 float diametro = circ_diametro(raio);
 printf("\nO diâmetro do círculo é: %f",
diametro);
 float area = circ_area(raio);
 printf("\nA área do círculo é: %f", area);
 float comprimento = circ_comprimento(raio);
 printf("\nO comprimento do círculo é:
%f\n", comprimento);

 return 0;
}
```

Exemplo!

Elementos de uma círculo/circunferência

Fase II

Modularização em Arquivos em C

```c
#include <math.h>
#define PI 3.1416;

float circ_diametro(float raio){
 return 2 * raio;
}

float circ_area(float raio){
 return pow(raio, 2) * PI;
}

float circ_comprimento(float raio){
 return 2 * raio * PI;
}
```

circulos.?

```c
float circ_diametro(float raio);
float circ_area(float raio);
float circ_comprimento(float raio);
```

circulos.?

Exemplo!

Elementos de uma círculo/circunferência

Fase III

Modularização em Arquivos em C

```c
#include <stdio.h>
#include "circulos.h"

int main(void) {
 float raio;
  printf("Digite o raio de um círculo: ");
 scanf("%f", &raio);

 float diametro = circ_diametro(raio);
 printf("\nO diâmetro do círculo é: %f", diametro);

 float area = circ_area(raio);
 printf("\nA área do círculo é: %f", area);

 float comprimento = circ_comprimento(raio);
 printf("\nO comprimento do círculo é: %f\n", comprimento);

 return 0;
}
```

Exemplo!

Elementos de uma círculo/circunferência

Fase III

Quebra em Arquivos em C

E se tiverem **múltiplas inclusões** de um arquivo .h em um mesmo arquivo .c?

PODE DAR PROBLEMA!!

Guardas de código:

```
#ifndef ___circulos___
#define ___circulos___
    ...
#endif
```

Modularização em Arquivos em C

```c
#ifndef __circulos__
#define __circulos__

float circ_diametro(float raio);
float circ_area(float raio);
float circ_comprimento(float raio);

#endif
```

Exemplo!

Elementos de uma círculo/circunferência

Fase IV

Quebra em Arquivos em C

Em um projeto de programação modularizado em vários arquivos, o processo de **compilação** deve considerar todos os arquivos

O “tradicional”: `gcc meu_programa.c -o meu_programa`

Quando modularizado: `gcc meu_programa.c circulos.c -o meu_programa`

Exercício #6-B

Considere o seguinte exercício:

O arquivo mapa.txt contém o mapa de um nível do jogo Boulder Dash. Escreva um programa em C que carregue esse mapa em uma matriz de caracteres.

Crie uma biblioteca que contenha funções com as seguintes funcionalidades:

- ➔ Retornar a altura do mapa
- ➔ Retornar a largura do mapa
- ➔ Retornar o número de células do mapa
- ➔ Retornar a matriz do mapa

Obrigado!

Vinícius Fülber Garcia
inf.ufpr.br/vinicius
vinicius@inf.ufpr.br