Estrutura de Dados

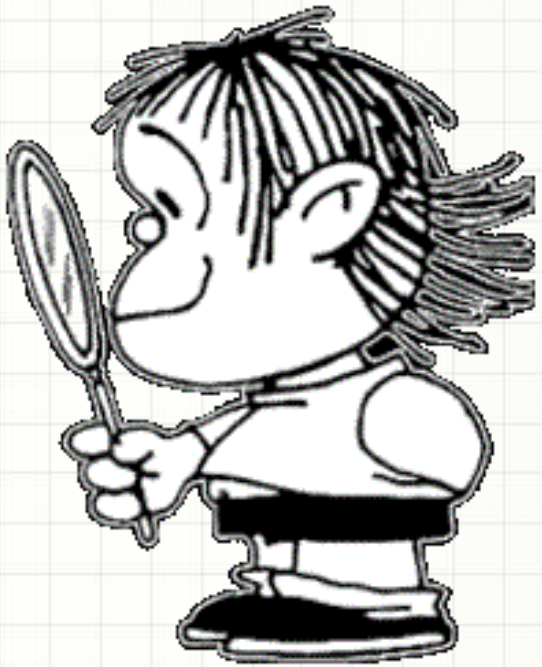
Introdução às Funções

Prof. Dr. Daniel Caetano

2014 - 2

Objetivos

- Compreender o que são as funções
- Implementar funções
- Compreender os escopos de variáveis



Momento Lúdico

Momento Lúdico

- Como fazer um misto quente?

- Como fazer um sanduiche com um “recheio” genérico?

Funções Simples

Funções Simples

- Queremos uma função que imprima a assinatura de nosso e-mail:

  Atenciosamente,

  Prof. Daniel Caetano

  prof@caetano.eng.br

```
cout << “Atenciosamente” << endl;
cout << “Prof. Daniel Caetano” << endl;
cout << “prof@caetano.eng.br” << endl;
```

  - Mas toda função precisa de um nome...!

Nome e Delimitação da Função

- Chamemos a função de “assina”:

```
assina()
{
    cout << “Atenciosamente” << endl;
    cout << “Prof. Daniel Caetano” << endl;
    cout << “prof@caetano.eng.br” << endl;
}
```

Os parênteses são importantes!

- Assim como uma receita de bolo...
  - Uma função precisa ser usada para ter resultado!
  - Vamos inserir a função em um programa!

Uso da Função

```
#include <iostream>
using namespace std;

assina()
{
    cout << “Atenciosamente,” << endl;
    cout << “Prof. Daniel Caetano” << endl;
    cout << “prof@caetano.eng.br” << endl;
}

main()
{
    assina();
}
```

Erro?

Funções têm retorno!

```
#include <iostream>
using namespace std;

void assina()
{
    cout << "Atenciosamente," << endl;
    cout << "Prof. Daniel Caetano" << endl;
    cout << "prof@caetano.eng.br" << endl;
}

main()
{
    assina();
}
```

void : indica função sem retorno!

main não precisa ter retorno?!?

Funções com Retorno

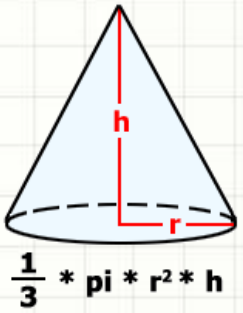
Funções com Retorno

```
#include <iostream>
using namespace std;

float pi()
{
    return 3.14159;
}

main()
{
    cout << pi();
}
```

E se mudar o **float** por **int**?

Funções com Parâmetros

Funções com Parâmetros



- Calcular o volume de um cone

```
#include <iostream>
using namespace std;

float volumeCone(float r, float h)
{
    return (1.0/3.0)*3.14*r*r*h;
}

main()
{
    cout << volumeCone(10,2);
}
```

Mude esses valores e veja o que ocorre!

Funções com Parâmetros por Referência

Parâmetros por Referência

- Função que atualiza salário (+10%)

```
#include <iostream>
using namespace std;

void atualizaSalario(float &sal) {
  sal = sal * 1.10;
}

main() {
   float s=100.00 ;
   atualizaSalario(s);
   cout << s;
}
```

E se colocar um **número** aqui?

Escopo de Variáveis e Variáveis Globais

Escopo de Variáveis

- Variáveis das funções: **locais**
  - Valem apenas dentro da função

```
#include <iostream>
using namespace std;

float volumeCone(float r, float h) {
    float x;
    x = (1.0/3.0)*3.14*r*r*h;
    return x;
}
main() {
    volumeCone(10,2);
    cout << x;
}
```

ERRO!

Escopo de Variáveis

- Variáveis **globais**
  - Valem no programa todo

Cuidado!

```
#include <iostream>
using namespace std;
float x;

void volumeCone(float r, float h) {
    x = (1.0/3.0)*3.14*r*r*h;
}
main() {
    volumeCone(10,2);
    cout << x;
}
```

Fixando os Conceitos:

Refazendo Funções "Pré-Definidas"

Funções

- Funções pré-definidas
  - abs(x)

```
int x;
x = -4;
cout << “O valor absoluto de ”;
cout << x << “ = ”;
cout << abs(x) << endl;
```

  - O que fazer se a função abs(x) não existisse?

Funções

- abs “na raça”:

```
int x;
x = -4;
cout << “O valor absoluto de ”;
cout << x << “ = ”;
if (x >= 0) cout << x << endl;
else cout << -x << endl;
```

- Se tivéssemos de usar isso toda hora...?
  - O que fazer?
  - Copiar e colar?

Funções

- abs “na raça”:

```
int x;
x =
cout <<
cout
if
else
```

Que tal criar nosso próprio **abs**?

- Se tivésse … to… ra…?
  - O que fazer?
  - Copiar e colar?

Criando Funções

- **Passo 1**: criar um programa que calcule o perímetro de um círculo de raio 2
  - $P = 2 \cdot \prod \cdot R$
  - $\prod = 3,141592$

Criando Funções

- **Passo 2**: transformar o cálculo em uma função chamada **calcula**

Criando Funções

- **Passo 2**: transformar o cálculo em uma função chamada **calcula**
  - As variáveis criadas dentro da função só existem dentro desta função
  - Elas são chamadas **variáveis locais**
  - Não é possível acessar uma variável local a não ser de dentro da própria função
  - Os valores das variáveis locais **são destruídos** quando a função finaliza

Criando Funções

- **Passo 3**: modificar a função **calcula** para que ela para que ela retorne o resultado, ao invés de imprimi-lo

Criando Funções

- **Passo 3**: modificar a função **calcula** para que ela para que ela retorne o resultado, ao invés de imprimi-lo
  - **return** deve sempre retornar um valor do tipo correto
  - **return** pode ser usado sem nenhum valor em funções cujo retorno é do tipo “void”

Criando Funções

- **Passo 4**: modificar a função **calcula** para que ela receba o raio do círculo como parâmetro

Criando Funções

- **Passo 4**: modificar a função **calcula** para que ela receba o raio do círculo como parâmetro
  - Parâmetros funcionam como “variáveis” locais
  - O valor fornecido como parâmetro (o raio) é **copiado** para essa “variável local”

Criando Funções

- **Passo 5**: modificar a função **calcula** para que ela retorne, além do perímetro da circunferência, também a área do círculo e o volume da esfera
  - $A = \prod \cdot R^2$
  - $V = (4/3) \cdot \prod \cdot R^3$

Criando Funções

- **Passo 5**: modificar a função **calcula** para que ela retorne, além do perímetro da circunferência, também a área do círculo e o volume da esfera
  - $A = \prod \cdot R^2$
  - $V = (4/3) \cdot \prod \cdot R^3$
  - Parâmetros cujo nome é precedido por & são passados “por referência”, isto é, podem ser modificados na função

Criando Funções

- **Passo 6**: mova a função para o fim do arquivo

Criando Funções

- **Passo 6**: mova a função para o fim do arquivo
  - A declaração inicial é chamada “protótipo de função”

“Resumão” de Funções

Funções: Descrição Básica

- **Função**: um algoritmo que pode ser executado por outro algoritmo, por meio de seu nome
- **Protótipo**: indica como função deve ser usada, no início do programa:

```
tipo_ret nome([tipo1 parm1][, tipon parmn]);
```

- **Declaração**: implementação da função:

```
tipo_ret nome([tipo1 parm1][, tipon parmn]) {
    // código da função
    return valor;
}
```

Funções: Uso

- **Chamada Sem Parâmetro**:

  nome();

- **Chamada com um Parâmetro**:

  nome(x);

- **Chamada com vários (3) Parâmetros**:

  nome(1, 7, x);

- **Chamada com Retorno (armazenado)**:

  x = nome();

Funções: Escopo

- **Variáveis Globais:**
  - Declaradas no topo do programa, valem em todo lugar
- **Variáveis Locais**:
  - Declaradas na função, só valem dentro dessa função
- **Parâmetros por Valor**:
  - Podem ser passados como números
  - Se forem variáveis, as modificações são perdidas quando a função finaliza
- **Parâmetros por Referência**:
  - Só variáveis podem ser passadas
  - As modificações se mantém ao fim da função

EXERCÍCIOS DE FIXAÇÃO

Exercícios de Fixação

1) Crie uma função que imprima o texto

Eu estou funcionando!

Exercícios de Fixação

2) Crie uma função que some dois valores inteiros e retorne o resultado.

Exercícios de Fixação

3) Crie uma função que receba 3 parâmetros: dividendo, divisor e quociente. Essa função deve retornar 1 se a divisão ocorreu com sucesso, e 0 se a divisão não pode ser realizada (quando o divisor é zero).

Exercícios de Fixação

4) Faça uma função em C/C++ que calcule a área de um retângulo e tenha o seguinte protótipo:

double calculaArea1(double base, double altura);

Depois construa a **main** que usa essa função

Exercícios de Fixação

5) Faça uma função em C/C++ que calcule a área de um retângulo e tenha o seguinte protótipo:

```
void calculaArea2(double base, double alt, double &area);
```

Depois construa a **main** que usa essa função

Formação de Grupos de Trabalho

Formação de Grupos

- Por que formar grupos?
- Quantos alunos?
  - No **mínimo** 4 alunos
  - No **máximo** 8 alunos
- Entregar, **agora**, lista de NOMES de cada aluno, indicando o NOME DA EQUIPE.
- Atenção:
  - Elejam UM responsável por subir os dados no SIA, que deve fornecer o e-mail para o professor!

PERGUNTAS?

Conclusões

Resumo

- Definição de funções
- Implementação de funções
- Tipos diferentes de passagem de parâmetros
- Escopo de validade de variáveis

- Vetores...
  - O que são?
  - Como funcionam?
  - Por que são úteis?