Funções

Parte 1

Conceitos
Motivação

Sumário

- Motivação
  - Por que usar funções?
- Funções simples
  - Definição e chamada
  - Valor de retorno
  - Mecanismo geral

Motivação

- Reaproveitamento de código já feito
- Evitar repetição de um trecho de código no mesmo programa
- Legibilidade de código
  - Código do programa principal muito extenso
    - Difícil de ler e entender
  - Facilidade em visualizar as diferentes partes do programa
- Lidar com complexidade de programa
  - **Modularização:**
    - Dividir problema em subproblemas
    - Combinação das soluções dos subproblemas leva à solução do problema como um todo

Motivação

Ler **n** até digitar **0**.
Para cada **n**, obter um código de operação para efetuar um cálculo com **n** e exibir o resultado do cálculo.
Só são permitidos códigos positivos e só executa o cálculo após um código de operação válido.

O subproblema de obter um código válido pode ser tão complexo conforme seja a forma com que se deseje a validação.

```
/* Programa 'menuCodigo' */
#include <iostream>
#include <cmath>
using namespace std;

int main()  {
    int cod;  float n, x;

    cin >> n;
    while ( n != 0 ) {

        cin >> cod;
        while ( cod <= 0 )
            cin >> cod;

        if ( cod == 1 )          x = sqrt(n);
        else if ( cod == 2 )     x = cbrt(n);
        else if ( cod == 3 )     x = exp(n);
        cout << x  << endl;

        cin >> n;
    }

    return 0;
}
```

Duas repetições aninhadas baseadas em entrada do usuário.

// enquanto código
// **não é válido**
// solicita novo código

Motivação

```
/* Programa 'menuOpcoes' */
#include <iostream>
#include <cmath>
using namespace std;

int lerCodigo ( ) {
        int code;

        cin >> code;
        while ( code <= 0 )
            cin >> code;

        return code;
}

int main( ) {
    int op;  float n, x;

    cin >> n;
    while ( n != 0 ) {

        op = lerCodigo ( );

        if ( op == 1 )          x = sqrt (n);
        else if ( op == 2 )     x = cbrt (n);
        else if ( op == 3 )     x = exp (n);
        cout << x << endl;

        cin >> n;
    }

    return 0;
}
```

Definir uma **FUNÇÃO** responsável por **APENAS** obter um código válido do usuário

O resultado gerado pela execução da função é DEVOLVIDO para quem USA a função

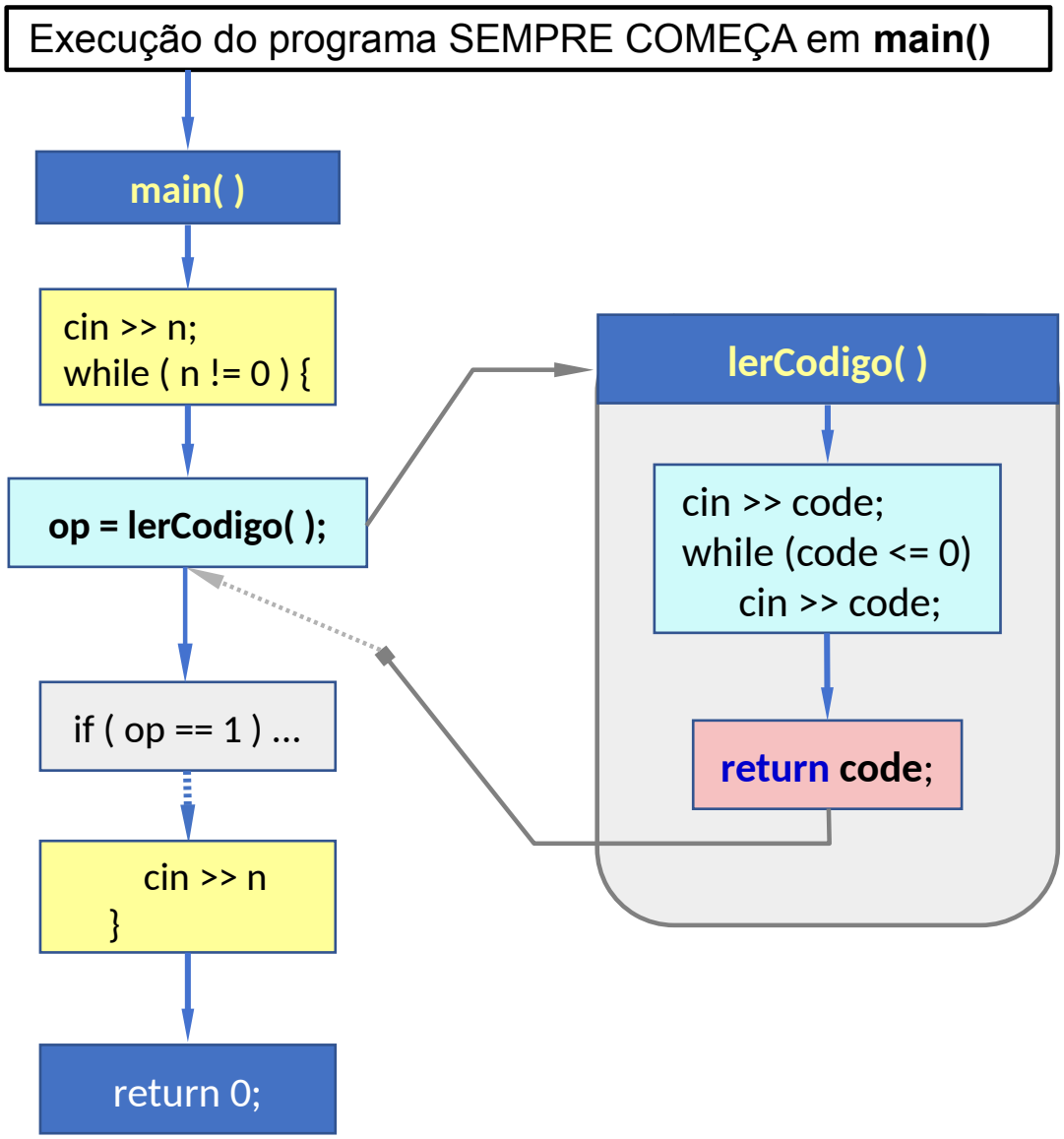
A EXECUÇÃO do programa SEMPRE começa em **main().**

→ A execução é desviada para a função.
→ Terminada a execução da função
→ resultado é devolvido à atribuição
→ Só então a atribuição é executada

Se escreve o programa principal sem se preocupar com detalhes do código da função. Basta saber o que ela faz e o que devolve como resultado.

Como acontece a execução?

```
/* Programa 'menuOpcoes' */
#include <iostream>
#include <cmath>
using namespace std;

int lerCodigo ( ) {
        int code;

        cin >> code;
        while ( code <= 0 )
            cin >> code;

        return code;
}

int main( ) {
    int op;  float n, x;

    cin >> n;
    while ( n != 0 ) {

        op = lerCodigo ( );

        if ( op == 1 )          x = sqrt (n);
        else if ( op == 2 )     x = cbrt (n);
        else if ( op == 3 )     x = exp (n);
        cout << x << endl;

        cin >> n;
    }

    return 0;
}
```



Definição de função simples

- Forma geral de **definição**:

```
cabeçalho
{
    declarações de variáveis;
    comandos ;
    return valor_retorno;
}
```

  - Bloco de comandos (entre **{ }**) → código que é executado pela função

- Para funções simples, que apenas devolvem um resultado:
  - **cabeçalho: tipo_retorno** `nomeFuncao ( )`
    - **tipo_retorno:** indica o tipo de **valor_retorno**
      - tipo do resultado que a função deve retornar ao final de sua execução
        - comando **`return`** indica final da execução da função e o valor a ser devolvido por ela → **valor de retorno**

    - Uma função definida nesta forma simples não recebe dados:
      - É o que indicam os **( )** sem nada entre eles.

Onde definir uma função no código?

- Função pode ser **definida** antes ou depois de **main()**:
  - Se definida **antes**, não há passo adicional;
  - Se definida **depois**, é preciso **declarar** o **protótipo da função** antes de **main()**:
    - O **protótipo** consiste em escrever o mesmo cabeçalho de definição da função seguido apenas de **ponto-e-vírgula**.
    - O importante é que ou a **definição** ou **protótipo** de uma função estejam escritas no código-fonte ANTES da PRIMEIRA VEZ que ela é **referenciada** no código-fonte de todo o programa.

Definição e protótipo de função

```
/* Programa 'menuOpcoes' */
#include <iostream>
#include <cmath>
using namespace std;

// Definição da função lerCodigo()
int lerCodigo ( ) {
        int code;

        cin >> code;
        while ( code <= 0 )
            cin >> code;
        return code;
}

int main( ) {
    int op;  float n, x;

    cin >> n;
    while ( n != 0 ) {

      op = lerCodigo ( );

        if ( op == 1 )          x = sqrt (n);
        else if ( op == 2 )   x = cbrt (n);
        else if ( op == 3 )   x = exp (n);
        cout << x << endl;
        cin >> n;
    }
    return 0;
}
```

```
/* Programa 'menuOpcoes' */
#include <iostream>
#include <cmath>
using namespace std;

// Protótipo da função lerCodigo()
int lerCodigo ( );

int main( ) {
    int op;  float n, x;

    cin >> n;
    while ( n != 0 ) {
      op = lerCodigo ( );
        if ( op == 1 )          x = sqrt (n);
        else if ( op == 2 )   x = cbrt (n);
        else if ( op == 3 )   x = exp (n);
        cout << x << endl;
        cin >> n;
    }
    return 0;
}

// Definição da função lerCodigo()
int lerCodigo ( ) {
        int code;

        cin >> code;
        while ( code <= 0 )
            cin >> code;
        return code;
}
```

O comando **return**

- O comando **return** tem um duplo objetivo:
  - Sinalizar que a execução da função terminará:
    - Nenhum comando escrito após um **return** será executado
    - A execução do **programa** continua no ponto onde a função foi **usada**.
  - Devolver um valor a quem chamou a função
    - valor deve ter o mesmo **tipo_retorno** definido no **cabeçalho** da função
    - Somente é possível devolver **UM ÚNICO VALOR**
- **Não se retorna valores com cout**
  - **cout** serve para saída de dados para usuário
  - **return** é usado para devolver um valor de função para o programa

Uso (Chamada) de uma função

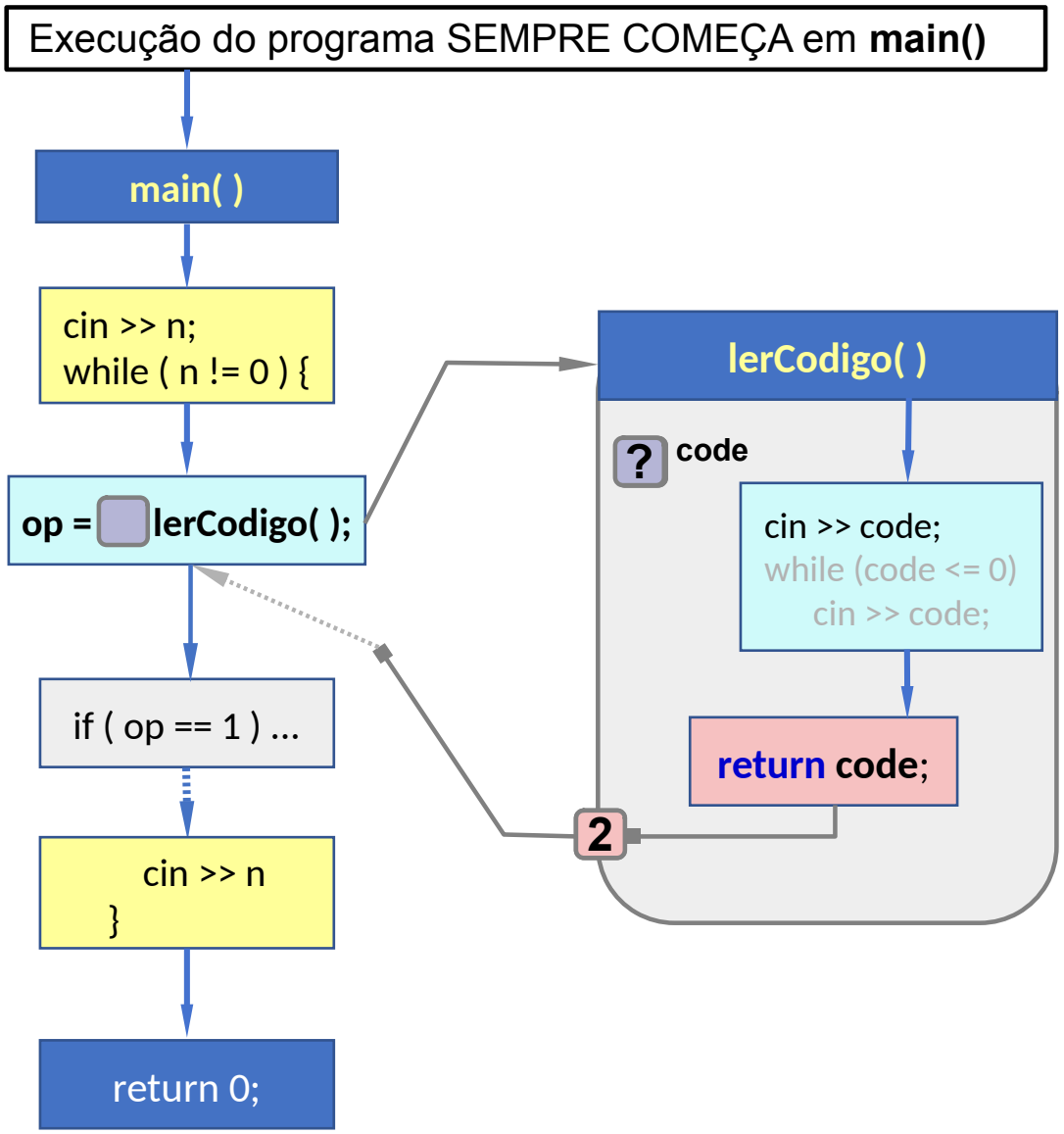
- Como o programa usa ou invoca a execução de uma função?
  - Através da **chamada**
    - Para uma funçao simples:

```
nomeFuncao ( )
```

  - A chamada da função pode ser feita em qualquer ponto de um **programa** onde seu valor de retorno deva ser usado:
    - op = **codigoOp( )**;
      → O **valor retornado** pela função será atribuído à variável **op**
    - if (**codigoOp( )** == 1) x = x + 3;
      → Se o **valor retornado** pela função é igual a 1, somar 3 a **x**
    - y = **codigoOp( )** + x;
      → Soma com **x** o **valor retornado** pela função e atribui resultado a **y**

Como acontece a execução?

```
/* Programa 'menuOpcoes' */
#include <iostream>
#include <cmath>
using namespace std;

int lerCodigo ( ) {
      int code;

      cin >> code;
      while ( code <= 0 )
          cin >> code;

      return code;
}

int main( ) {
    int op;  float n, x;

    cin >> n;
    while ( n != 0 ) {

      op = lerCodigo ( );

      if ( op == 1 )          x = sqrt (n);
      else if ( op == 2 )   x = cbrt (n);
      else if ( op == 3 )   x = exp (n);
      cout << x << endl;

      cin >> n;
    }

    return 0;
}
```



Funções e variáveis

- Variáveis de funções diferentes estão em locais diferentes na memória
  - mesmo que tenham o mesmo nome
- Variáveis de uma função são **locais** a esta função
  - São válida APENAS dento do bloco de comandos da função
  - Não são visíveis em outra função

```
/* Programa 'menuOpcoes' */
#include <iostream>
using namespace std;

int func ( ) {

        int code;
        .......
        code = op;  // ERRO: op não foi declarada aqui
                    // e NÃO SE REFERE à variável
                    // declarada em main()
        .......
        return code;
}

int main( ) {

    int  op;
    .......
    op = func ();
    return 0;
}
```

```
/* Programa 'menuOpcoes' */
#include <iostream>
using namespace std;

int func ( ) {

        int code, op;
        .....
        op = 23;  // op NÃO SE REFERE à variável
                  // declarada em main()
        cout  << op << endl;   // imprime 23
        return code;
}

int main( ) {

    int x, op; // op NÃO SE REFERE à variável declarada em func()

    op = 37;
    x = func ();
    cout << op << endl;   // imprime 37
    return 0;
}
```

A função **main()**

- O programa principal **main()** é uma função especial
  - possui um tipo de retorno fixo → **int**
  - é invocada automaticamente quando se inicia a execução do programa
- O comando **return** que se coloca ao final do bloco do **main()** informa ao Sistema Operacional (SO) se o programa terminou corretamente ou não.
  - Por convenção:
    - **return 0** (valor zero) indica que tudo correu bem,
    - qualquer valor diferente de zero indica ao SO que houve erro no programa.

Como passar dados para funções?

PRÓXIMO SEGMENTO

Leitura complementar

Acesse as **Leituras complementares** do Tópico **Funções**, na sala virtual da disciplina da UFPR Virtual.

Elas são importantes e auxiliam na compreensão dos temas abordados até aqui.

Créditos: O conteúdo original deste documento é de autoria do Prof. Armando Luiz N. Delgado, para uso na disciplina *Programação de Computadores* (CI208, CI180, CI183). Agradecemos à Profª Myriam Regattieri (UTFPR), ao Prof. Ricardo Lüders (UTFPR) e ao Prof. Luciano Silva (UFPR) pelas sugestões.