Introdução à Linguagem C++

Roland Teodorowitsch

Programação Orientada a Objetos - ECo - Curso de Engenharia de Computação - PUCRS

14 de agosto de 2023

Visão Geral

Extensão e Compilação

- Arquivos escritos em linguagem C++ são salvos com a extensão .cpp
- O compilador para C++ é o g++ (funciona como o gcc)
- Uso:
  `g++ -std=c++11 meu_programa.cpp -o meu_programa`
- Se não tiver sido criado nenhum `Makefile` (ainda), pode-se usar:
  `make meu_programa`

Estrutura Básica de um Programa

1. Inclusão de arquivos
2. Definição do espaço de nomes (opcional)
3. Declaração de macros, tipos e variáveis globais (quando houver)
4. Métodos, incluindo o método `main()`

```
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <string>

using namespace std;

int main(){
  // Declaracao de variaveis
  // Comandos da funcao main
  return 0;
}
```

Inclusões

- Arquivos de cabeçalho das bibliotecas não possuem a extensão `.h`
- Principais arquivos de cabeçalho:
  - `iostream`: funções de entrada e saída
  - `iomanip`: formatação de entrada e saída
  - `cmath`: funções matemáticas
  - `cstdlib`: funções de uso geral
  - `ctime`: funções para manipulação de tempo e data
  - `string`: funções para uso de *strings*

Espaço de Nomes do Compilador

- Permite a definição de estruturas, classes, funções, etc.
- Permite duplicidade de identificadores, desde que eles estejam em espaços de nomes diferentes
- Por padrão, a linguagem utiliza o *namespace* `std`
  - Por exemplo, em `std::cout`, `cout` pertence ao *namespace* `std`
- Pode-se simplificar isto usando:

```
using std::cout;
// ou
using namespace std;
```

Espaço de Nomes – Exemplo

```
#include <iostream>

// using namespaces std; ==> evitaria o uso de std:: no resto do programa
using std::cout ; // agora basta usar cout, mas ainda eh preciso usar std::endl

void funcao() { cout << "funcao()" << std::endl; }

namespace escopo1 {
  void funcao() { cout << "funcao() do escopo1" << std::endl; }
}

namespace escopo2 {
  void funcao() { cout << "funcao() do escopo2" << std::endl; }
  namespace escopo3 {
    void funcao() { cout << "funcao() do escopo3 [interno de escopo2]" << std::endl; }
  }
}

int main() {
  funcao();
  escopo1::funcao();
  escopo2::funcao();
  escopo2::escopo3::funcao();
  return 0;
}
```

Função principal

- É a mesma usada em C
- É formada por declarações de variáveis (não precisam estar no início dos blocos) e comandos

```
int main()
// ou recebendo parametros da linha de comando:
int main(int argc, char *argv[])
```

Novos Tipos

Novos Tipos

- C++ trouxe dois novos tipos
  - `bool`
    - Serve para valores lógicos
    - Pode receber `true` ou `false`
  - `string`
    - É uma forma mais prática de armazenar e trabalhar com cadeias de caracteres
    - É uma classe e está definida no arquivo de cabeçalho `string`
    - Não é preciso definir o tamanho
    - Pode ser usado com operadores como, por exemplo, == ou = (evitando ter que usar métodos como `strcmp()` ou `strcpy()`)
    - Possui uma série de métodos definidos e prontos para serem usados (veja `https://cplusplus.com/reference/string/string/`)

bool e string

```
#include <iostream>
#include <string>

using namespace std;

int main() {
  bool a = true, b = false;
  cout << "OR:  " << (a || b) << endl;
  cout << "AND: " << (a && b) << endl;

  string str = "Esta eh uma string em C++";
  cout << "Tamanho:                          " << str.length() << endl;
  cout << "Letra de indice 2:                " << str.at(2) << endl;
  cout << "Substring (inicio=5, tamanho=2): " << str.substr(5,2) << endl;
  str = "FIM";  // str agora tem outro conteudo
  if (str == "FIM")
     cout << "> Encerrando o programa..." << endl;
  return 0;
}
```

Entrada e Saída

Saída

- std::cout ou cout é o fluxo de saída padrão
- O operador << (de inserção) é usado para enviar informações para a saída

```
#include <iostream>

using namespace std;

int main() {
  cout << "Ola turma!" << endl;
  return 0;
}
```

Saída

- A quebra de linha é definida pelo manipulador `endl`
- Este manipulador também esvazia o *buffer* de saída
- O operador << pode ser encadeado, permitindo uma combinação de constantes, variáveis e expressões

```
#include <iostream>

using namespace std;

int main() {
  int x=10, y=20;
  cout << "Soma de x e y: " << x+y << endl;
  return 0;
}
```

Erro

- `std::cerr` ou `cerr` é o fluxo de saída de erro padrão
- O operador << (de inserção) também é usado para enviar informações para a saída de erro

```
#include <iostream>

int main() {
  std::cout << "DADOS" << std::endl;
  std::cerr << "ERROS" << std::endl;
  return 0;
}
```

- Compile (por exemplo: `g++ -std=c++11 output.cpp -o output`) e execute:

```
./output > arquivo.txt   # ou:  ./output 1>  arquivo.txt
./output 2> arquivo.txt
./output &> arquivo.txt
```

Entrada

- std::cin ou cin é o fluxo de entrada padrão
- O operador >> (de extração) é usado para obter informações da entrada

```
#include <iostream>

int main() {
  int x;
  std::cout << "Lendo um inteiro: ";
  std::cin >> x;
  std::cout << "Valor lido: " << x << std::endl;
  return 0;
}
```

Verificando a Consistência da Entrada

```
#include <iostream>
using namespace std;
int main() {
  int x;
  cout << "Digite um inteiro: ";
  cin >> x;
  if (cin.bad()) {
     cerr << "Houve uma falha na leitura!" << endl;
     return 1;
  }
  if (cin.fail()) {
     cerr << "Dado digitado incompativel com o dado esperado!" << endl;
     return 1;
  }
  cout << endl;
  cout << "Valor lido: " << x << endl;
  return 0;
}
```

Entrada de *Strings*

- O operador de extração ignora carateres como tabulações, espaços em branco e novas linhas
- Para ler *strings* que englobem essas situações, usar `getline`

```
#include <iostream>

using namespace std;

int main() {
  char frase[30];
  cout << "Digite uma frase: " << endl;
  cin.getline(frase, 30);
  cout << "Frase digitada: " << frase << endl;
  // OU
  string texto;
  cout << "Digite algum texto: " << endl;
  getline(cin, texto);
  cout << "Texto  digitado: " << texto << endl;
  return 0;
}
```

Manipuladores de Fluxos

- Em `iomanip` estão definidos diversos manipuladores, como, por exemplo:
  - Para formatos de valores inteiros: `hex`, `oct`, `setbase`
  - Para ponto flutuante: `fixed`, `setprecision`
  - Definição de tamanho: `setw(int size)`
  - Alinhamento: `left`, `right`
  - Preenchimento: `setfill(char c)`

Manipuladores de Fluxos (Exemplo)

```
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
  int x = 10;
  cout << hex << x << endl;
  cout << oct << x << endl;
  cout << setbase(10) << x << endl;
  cout << "Numero = " << setw(10) << right << x << endl;
  cout << "Numero = " << setw(10) << left << x << endl;
  double pi = 3.14159265;
  cout << fixed << setprecision(5);
  cout << pi << endl;
  return 0;
}
```

Similaridades entre C e C++

Similaridades entre C e C++

- Todas as **estruturas de controle de fluxo** estudadas para a linguagem C podem ser utilizadas na linguagem C++
- Todas as **estruturas de dados** estudadas para a linguagem C podem ser utilizadas na linguagem C++

Estruturas de Seleção: if

```
#include <iostream>

using namespace std;

int main() {
  int x, y;
  cin >> x >> y;
  if (x == y)
     cout << "x = y" << endl;
  else if (x > y)
     cout << "x > y" << endl;
  else
     cout << "x < y" << endl;
  return 0;
}
```

Estruturas de Seleção: switch

```
#include <iostream>

using namespace std;

int main() {
  int dia;
  cin >> dia;
  switch (dia){
     case 1:  cout << "domingo"       << endl; break;
     case 2:  cout << "segunda-feira" << endl; break;
     case 3:  cout << "terca-feira"   << endl; break;
     case 4:  cout << "quarta-feira"  << endl; break;
     case 5:  cout << "quinta-feira"  << endl; break;
     case 6:  cout << "sexta-feira"   << endl; break;
     case 7:  cout << "sabado"        << endl; break;
     default: cout << "INVALIDO"      << endl;
  }
  return 0;
}
```

Estruturas de Repetição: while

```
#include <iostream>

using namespace std;

int main() {
  int a = -1;
  while (a < 0) {
    cout << "Valor positivo: ";
    cin >> a;
  }
  cout << a << endl;
  return 0;
}
```

Estruturas de Repetição: do/while

```
#include <iostream>

using namespace std;

int main() {
  int a;
  do {
    cout << "Valor positivo: ";
    cin >> a;
  } while(a < 0);
  cout << a << endl;
  return 0;
}
```

Estruturas de Repetição: for

```
#include <iostream>

using namespace std;

int main() {
  int x, y;
  for (x=0, y=10; x<10; x++, y--)
      cout << "x = " << x << " y = " << y << endl;
  cout << "Valor de x e y apos o laco: " << x << ", " << y << endl;
  return 0;
}
```

Estruturas de Dados: Vetores e Matrizes

```
#include <iostream>

using namespace std;

int main() {
  int i, j, v[10], m[4][4];
  for (i=0; i<10; i++)
      v[i] = i + 1;
  for (i=0; i<4; i++)
      for (j=0; j<4; j++)
          m[i][j] = i + j;
  // ...
  return 0;
}
```

Estruturas de Dados: Registros

```
#include <iostream>
#include <string.h>

using namespace std;

typedef struct {
  char nome[41];
  int altura;
  double peso;
} pessoa_t;

int main() {
  pessoa_t joao;
  strcpy(joao.nome,"Joao Paulo");
  joao.altura = 181;
  joao.peso = 85.4;
  cout << joao.nome << " (" << joao.altura << "," << joao.peso << ")" << endl;
  return 0;
}
```

Estruturas de Dados: Registros (com string)

```
#include <iostream>
#include <string>

using namespace std;

typedef struct {
  string nome;
  int altura;
  double peso;
} pessoa_t;

int main() {
  pessoa_t joao;
  joao.nome = "Joao Paulo";
  joao.altura = 181;
  joao.peso = 85.4;
  cout << joao.nome << " (" << joao.altura << "," << joao.peso << ")" << endl;
  return 0;
}
```

Exercícios

Exercício 1

1. Usando a linguagem C++, escreva um programa que permita armazenar o nome, a altura e a data de nascimento de até 10 pessoas. Cada pessoa deve ser representada por uma `struct` dentro de um vetor. A data de nascimento também deve ser uma `struct` que contém os campos dia, mês e ano. O programa deve iniciar perguntando se o usuário deseja inserir uma pessoa no sistema (respeitando o limite de 10 pessoas), lendo as suas informações em caso positivo. O nome, a altura e a data de nascimento de cada pessoa devem ser informados pelo teclado. Caso o usuário não queira mais inserir pessoas ou caso o limite de 10 pessoas tenha sido atingido, o programa deverá ler a data do dia de hoje e mostrar a lista das pessoas (ordenada pelo nome de forma crescente) com o seu nome, altura e idade (considerando a data fornecida como referência para o dia de hoje).

Exercícios 2 e 3

2. Usando a linguagem C++, escreva um programa capaz de ler os dados de uma matriz quadrada de inteiros. Ao final da leitura o programa deverá imprimir o número da linha que contém o menor dentre todos os números lidos.
3. Usando a linguagem C++, escreva um programa que apresenta a seguinte saída, perguntando ao usuário o número máximo (no exemplo, 9). Este número deve ser sempre ímpar.

```
1 2 3 4 5 6 7 8 9
  2 3 4 5 6 7 8
    3 4 5 6 7
      4 5 6
        5
```

Exercícios 4, 5 e 6

4. Usando a linguagem C++, escreva um programa capaz de ler dois nomes de pessoas e imprimi-los em ordem alfabética. Dica: `http://www.cplusplus.com/reference/string/string/compare/`.
5. Usando a linguagem C++, escreva uma função capaz de substituir todos os números negativos de uma matriz por seu módulo.
6. Usando a linguagem C++, escreva uma rotina que remova um caracter de uma *string* do tipo `char str[100]`, dada a posição do caracter.

Créditos

Créditos

- Estas lâminas contêm trechos de materiais disponibilizados pelos professores Rafael Garibotti e Matheus Trevisan.