Laboratório de Programação II

Introdução ao C++

Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação

Aula de Hoje

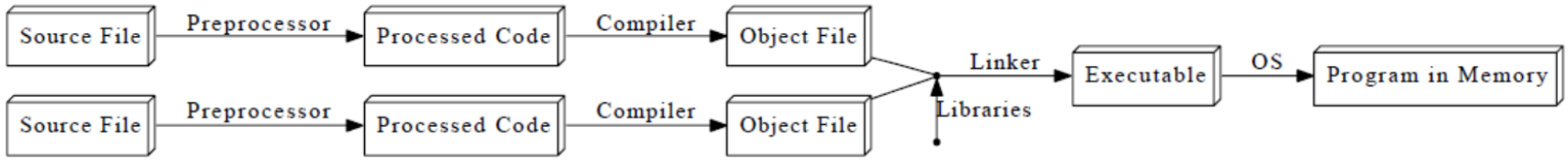
- Objetivo: praticar e se familiarizar com a linguagem C++, sobretudo com os novos recursos e tipos de dados existentes com relação à linguagem C.
- Tipos de dados
  - `bool`
  - `string`
- Entrada e Saída
  - Operador `cout`
  - Operador `cin`
- Strings

Introdução

- A linguagem C++
- Criada em 1979
- Bjarne Stroustrup
- Altamente popular
- Linguagem compilada
- C++ estende a linguagem C: veremos que tudo que já aprendemos em C também é válido em C++
- Programação Orientada a Objetos

Introdução

- Ciclo de desenvolvimento



- Nas aulas de Laboratório de Programação II, o ambiente de desenvolvimento integrado CodeBlocks será utilizado.
- Compilador: GNU GCC e GNU G++

Introdução

- Um primeiro programa em C++

```
#include <iostream>

using namespace std;

int main()
{
  cout << "Hello world!" << endl;
  return 0;
}
```

- Observações:
  - Biblioteca `iostream`
  - Saída em tela com operador `cout`
  - `namespace`

Tipos de Dados

- `char`: Caracter (1 byte)
- `int`: Inteiro (4 bytes)
- `float`: Número real em ponto flutuante (4 bytes)
- `double`: Número real em ponto flutuante (8 bytes)
- `bool`: Valores lógicos: true ou false
- `string`: Cadeia de caracteres

Tipos de Dados

- Exemplo

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

int main()
{
  int x;
  float y;
  x = 4 + 2;
  y = x/4;
  cout << x/3 << " " << x * 2 << endl;
  cout << " y = " << y << endl;
  return 0;
}
```

Tipos de Dados

bool

- Variáveis booleanas podem assumir apenas dois valores
  - `true` (1)
  - `false` (0)
- Para declarar variáveis booleanas usa-se a palavra chave `bool`.

```
int main()
{
  bool b1 = true;
  bool b2 = !b1;
  bool b3 = b1 && b2;

  cout << b1 << endl;
  cout << b2 << endl;
  cout << b3 << endl;

  return 0;
}
```

Tipos de Dados

Declaração de variáveis

- Em C++ pode-se declarar variáveis em qualquer região do código, assim como um outro comando qualquer.
- O escopo da variável começa no ponto onde ela é declarada e vai até o fim do bloco, isto é, até a chave } mais próxima.
- No exemplo abaixo, a variável `i` só existe dentro do `for`.

```
int main()
{
  int x, y, z, soma = 0;
  // ...
  for(int i = 0; i < 10; i++)
  {
    soma = soma + x;
  }
  // ...
}
```

Operadores

Operador	Significado
>	Maior que
>=	Maior que ou igual a
<	Menor que
<=	Menor que ou igual a
==	Igual a
! =	Diferente de
&&	E
||	Ou
!	Não

Entrada e Saída

- Em C++ para escrever algum texto na tela usamos o seguinte comando:

```
cout << "Mensagem";
```

- Podemos concatenar mais strings à saída usando várias vezes o operador <<:

```
cout << "X = " << x << "Y = " << y;
```

- Podemos inserir uma quebra de linha da seguinte forma

```
cout << "X = " << x << "Y = " << y << endl;
```

- Ou usando os formatadores especiais como em C

```
cout << "Hello World! \n";
```

- Note que para escrever texto na tela dessa forma é preciso incluir o seguinte comando no início do programa:

```
using namespace std;
```

Entrada e Saída

- A leitura de dados a partir do teclado é tão fácil quanto a escrita na tela.
- Não é preciso especificar o tipo de dados a ser lido, pois o operador `cin` converte a entrada para o tipo de dados da variável corretamente.
- Basta usar o operador `cin` como no exemplo abaixo:

```
#include <iostream>
using namespace std;
int main ()
{
  int i;
  cout << "Digite um inteiro: ";
  cin >> i;
  cout << "O valor digitado e " << i;
  cout << " e o seu dobro e " << i*2;
  cout << endl;
  return 0;
}
```

Entrada e Saída

```
int x, y;
cout << "Digite X: ";
cin >> x;
cout << "Digite Y: ";
cin >> y;

cout << "x*y = ";
cout << x*y << endl;

// ou .........

int x, y;
cout << "Digite X e Y:";
cin >> x >> y;

cout << "x*y = ";
cout << x*y << endl;
```

```
char sexo;
cout << "Sexo (m/f)?";
cin >> sexo;
if(sexo == 'm')
  cout<<"Masculino"<<endl;
else
  cout<<"Feminino"<<endl;

int i;
float x, soma=0;
for(i = 0; i < 5; i++)
{
  cin >> x;
  soma = soma + x;
}
cout << "Soma = " << soma;
cout << endl;
```

Strings

- Em C++ pode-se trabalhar com *strings* de duas formas:
  - No estilo da linguagem C
  - Com o tipo de dados `string` definido na biblioteca da linguagem
- Lembre-se que em C não existe um tipo de dados específico para isso:

```
char nome1[50];
char nome2[10] = "Fulano";
```

- As cadeias de caracteres em C terminam com um caractere especial: \0

Strings

- É preciso incluir a biblioteca para usar strings.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
  string str1;
  string str2 = "Bob Esponja";
  cout << "Digite um nome: ";
  cin >> str1;
  cout << "Personagem 1: " << str1 << endl;
  cout << "Personagem 2: " << str2 << endl;
  cout << "Terceira letra: " << str1[2];
  cout << endl;
  cout << "Primeira letra: " << str2[0];
  cout << endl;
  return 0;
}
```

Strings

- A grande diferença de trabalhar com o tipo `string` é que você declara uma variável deste tipo como uma simples variável, e não como um vetor (como é o caso da cadeia de caracteres em C).
- Exemplo: `string nome1;`
- Além disso, diversas operações com *strings* são mais simples quando usa-se o tipo de dados `string`.
  - Concatenação
  - Calcular o tamanho
  - Cópia
  - Procurar caractere ou substring
  - Comparação etc.

Strings

```
string str1 = "bom", str2 = " dia";
string str3, str;

str3 = str1 + str2;
str1 += str2;

cout << str1 << endl;
cout << str3 << endl;    // bom dia

str1 += " . boa noite";
cout << str1 << endl;    // bom dia . boa noite

int tam = str2.size();
cout << "Tamanho de str2 = ";
cout << tam << endl;     // Tamanho de str2 = 4

cin >> str;              // The Hobbit
cout << "A string digitada e: ";
cout << str << endl;     // The
```

Strings

- Para ler uma linha inteira, é preciso usar a função `getline(cin,str)`
- O primeiro argumento diz de onde pegar a entrada e o segundo argumento em qual string gravar a entrada.
- Exemplo:

```
string str;
getline(cin,str);

// Entrada: "The Hobbit"

cout << "A string digitada e: " << str;
cout << endl;

// Saida: A string digitada e: The Hobbit
```

Exercícios

1. Faça um programa que leia do teclado um número inteiro `n` e em seguida leia `n` números reais e calcule a sua média.

2. Implemente agora o exercício (1) utilizando uma função que possua o seguinte protótipo:

```
float leCalculaMedia(int n);
```

3. Faça uma função que receba como parâmetros um vetor de números reais e o seu tamanho `n` e que leia do teclado `n` números reais, guarde-os no vetor e calcule a sua média. A função deve retornar a média ao final.

```
float leVetorCalculaMedia(int n, float vet[]);
```

Exercícios

4. Faça uma função que, dados uma string `str` e um caractere `ch`, procure e retorne a posição da primeira ocorrência de `ch` na string `str`. Se `ch` não for encontrada em `str`, retornar o valor -1.

   ```
   int procuraCharNaString(string str, char ch);
   ```

   Dica: para saber o tamanho da string use `str.size()`.

5. Faça uma função que receba um número inteiro `n` > 0 e determine se este é um número primo. A função deve retornar um valor booleano: `true` ou `false`. Protótipo:

   ```
   bool ehPrimo(int n);
   ```

6. Faça um programa que leia os valores lógicos de `X`, `Y` e `Z` e armazene o resultado das seguintes operações lógicas:
   - `(X && Y) || (X && !Z)`
   - `(X || Y) && (!X && Z)`

Exercícios

7. Faça uma função para calcular o fatorial de um número inteiro `n` > 0. Faça um programa que leia um número inteiro do teclado, em seguida utilize a função para calcular o seu fatorial e, por fim, exiba o resultado na tela.

```
int fatorial(int n);
```

8. Faça um programa que leia um número inteiro `n` e um número real `x`. Em seguida calcule a seguinte soma $S = \sum_{i=0}^{n} x^i$ utilizando uma função.