Técnicas de Programação

Anotações referentes a linguagem c++

Autor: Arthur Emanuell de Sousa Firmino
Data: 2026

Prefácio

Nesta anotação será falado um pouco sobre C++.

Sumário

Capítulo 1 - Criação de um Código

Para o que ocorra o processo de criação de um código em c++, é necessário que haja a seguinte estrutura:

- 1° Código Fonte em C++
- 2° Compilador C++
- 3° Código-Objeto
- 4° O linker
- 5° Programa executável

A seguir será exemplificado o passo a passo do proceso de criação de um programa em C++.

1.1 Código Fonte em C++

O código fonte é a parte inicial de todo o projeto, na qual é necessário para que ocorra todo o decorrer da criação de um programa. Dentro dele é onde haverá a maior ênfase dentro deste PDF.

- Exemplo de um código fonte em C++

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

int main(){
    cout << "Hello Word";
return 0;
}
```

1.2 Compilador C++

Para que ocorra o processo de compilação do código fonte necessitamos de um compilador chamado **g++**, com ele podemos criar o código objeto, apartir do seguinte comando:

```
g++ -c codigo.cpp
```

> **Observação**: Onde está escrito codigo é o nome do arquivo.

Pórem, desta maneira não criamos o código executável por conseguinte não fazemos a linkedeção, apenas criamos o código objeto.

1.3 Código Objeto

Desta maneira já possui-se uma compilação para a criação de um arquivo **.o** que seria o arquivo que o processador do computador leria, no entando, não existiria como modifica-lo.

1.4 Linker

O linker é o responsável pela linkedição, ou seja, é ele que irá compilar o código .o para o .exe, tornando assim o arquivo executável.

- É possivel compilar o arquivo direto de .cpp para .exe, utilizando o seguinte comando.

```
//No caso o linux
g++ -o hello.cpp hello


//No caso do Windows
g++ hello.cpp -o hello
```

Capítulo 2 - Funções

Uma função é um bloco de código nomeado que executa uma tarefa específica. Ela pode receber valores de entrada (parâmetros), processar esses valores e retornar um resultado. O qual seu objetivo principal é:

- Organizar o Código.
- Evitar repetição.
- Facilitar a manutenção e reutilização.

2.1 Sintaxe

Para a contrução de uma função é necessário que siga a sintaxe de escrita de uma função, na qual será estudada subdividida.

- Exemplo de uma função:

```
int somatorio(int a, int b){
    return a+b;
}
```

2.1.1 - Tipo de Retorno

Toda função deve ser especificado o tipo de retorno, pois é com ela que será baseada a saída da função. Dentre os tipos de retono, temos eles: **int**, **float**, **char**, **void**, etc.

A seguir será demonstrado a declaração dos tipos de retorno e suas saídas:

- Int:

```
int somatorio(int a, int b){
    return a+b;
}
```

> A saída da função int, será um falor inteiro.

- Float:

```
float soma_medidas(float a, float b){
    return a+b;
}
```

> A saída da função float, será um valor decimal.

- Char:

```
char maiuscula(char letra) {
    if (letra >= 'a' && letra <= 'z') {
        return letra - 32; // converte para maiúscula
    }
return letra;
}
```

> A saída da função char, será uma variável char.

- Void

```
void hello(){
    cout << "Hello Word";
}
```

> Pelo fato do void não retornar nada, não há a necessidade de colocar "return".

2.1.2 - Nome da função

O nome da função nada mais é que o indentificador da função para auxiliar o programador.

2.1.3 - Lista de Parâmetros

Dentro da lista de parâmetros é onde iremos indicar a quantidade e o tipo das variáveis de entrada, por exemplo:

```
int somatorio(int a, int b){
    return a+b;
}
```

> Note que foi especificado quantas varáveis e quais os tipos de variáveis entraram na função somatorio.

2.2 - Funções de Biblioteca

Além das funções definidas pelo próprio programador, o C++ disponibiliza um amplo conjunto de funções de biblioteca, prontas para uso.

A linguagem C++ possui sua biblioteca padrão, que oferece recursos para:

- Entrada e saída de dados
- Manipulação de strings
- Operações matemáticas
- Controle de erros
- Manipulação de tempo, caracteres, memória, entre outros

Além disso, o C++ mantém compatibilidade com a biblioteca padrão da linguagem C, disponibilizando 18 cabeçalhos herdados.

2.2.1 - Biblioteca < cmath >

- Função pow()

Na Função pow() temos a implementação de uma função que elevava o numero dependendo do que se colocar na condição, o qual possui a seguinte sitaxe:

```
float pow(float base, float exp);
```

Exemplo de código usando a função.

```
#include <iostream>
#include <cmath>

int main(){

    // typical usage
    std::cout << "pow(2, 10) = " << std::pow(2,10) << '\n'
    << "pow(2, 0.5) = " << std::pow(2,0.5) << '\n'
    << "pow(-2, -3) = " << std::pow(-2,-3) << '\n';

    // special values
    std::cout << "pow(-1, NAN) = " << std::pow(-1,NAN) << '\n'
    << "pow(+1, NAN) = " << std::pow(+1,NAN) << '\n'
    << "pow(INFINITY, 2) = " << std::pow(INFINITY,2) << '\n'
    << "pow(2,-INFINITY) = " << std::pow(2,-INFINITY) << '\n'
    << "pow(2,INFINITY) = " << std::pow(2,INFINITY) << '\n'
    << "pow(-INFINITY,1) = " << std::pow(-INFINITY,1) << '\n'
    << "pow(INFINITY, -1) = " << std::pow(INFINITY, -1) << '\n';

    return 0;
}
```

- Função sqrt()

Na função sqrt(), temos a implementação de uma função que realiza a raiz quadrada de um número, a partir da seguinte sintaxe:

```
    float sqrt(float arg);
```

Exemplo de código usando a função.

```
#include <iostream>
#include <cmath>
    int main(){

    // normal use
    std::cout << "sqrt(100) = " << sqrt(100) << '\n'
    << "sqrt(2) = " << sqrt(2) << '\n'
    << "golden ratio = " << (1+sqrt(5))/2 << '\n';

    // special values
    std::cout << "sqrt(-0) = " << sqrt(-0.0) << '\n';

return 0;
}
```

- Função cbrt()

Na função cbrt(), temos a implementação de uma função que realiza a raiz cúbica de um número, a partir da seguinte sintaxe:

```
    float cbrt(float arg);
```

Exemplo de código usando a função.

```
    #include <iostream>
#include <cmath>
int main(){

    // normal use
    std::cout << "cbrt(729) = " << cbrt(729) << '\n'
    << "cbrt(-0.125) = " << cbrt(-0.125) << '\n';

    // special values
    std::cout << "cbrt(-0) = " << cbrt(-0.0) << '\n'
    << "cbrt(+inf) = " << cbrt(INFINITY) << '\n';
return 0;

}
```

- Função hypot()

Na função hypot(), temos a implementação de uma função que realiza o cálculo da hypotenusa, a partir da seguinte sitaxe:

```
float hypot(float a,float b);
```

Exemplo de utilização da função:

```
#include <iostream>
#include <cmath>
int main(){
    float x,y,h;
    x = 3;
    y = 4;
    h = hypot(x,y);
    std::cout << "Hipotenusa igual a " << h << '\n';
return 0;
}
```

- Funções Trigonométricas

A bilbioteca cmath tabém possui funções que realizam o cálculo de funções trignométricas, como por exemplo:

```
float sin(float theta);
float cos(float theta);
float tan(float theta);
```

Também é possível designar funções inversas, ou seja, o arco seno, o arco cosseno e o arco tangente.

```
float asin(float theta);
float acos(float theta);
float atan(float theta);
```

Capítulo 3 - Espaço de Nomes

É uma região declarativa que vincula um identificador a um grupo de variáveis, funções, subespaços, estruturas, constantes, etc.

- Exemplo:

```
namespace maria{
    int idade = 12;
    int cra = 8.3;
}

namespace jose{
    int idade = 21;
    int cra = 4.3;
}
```

No exemplo, idade e cra fazem parte do namespace **maria** e **jose**, ou seja, podemos criar variáveis de nomes iguais dentro de namespaces diferentes que mesmo assim não existirá confito entre as variáveis.

- Exemplo 2: Qual seria a saída deste código?

```
#include <iostream>
namespace maria {
int idade = 18;
float cre = 8.3;
}
namespace jose {
int idade = 21;
float cre = 4.3;
}
int main(){
    int idade = 12;
    std::cout << idade << " " << maria::idade << std::endl;
    std::cout << jose::idade << std::endl;
return 0;
}
```

A saída deste código será:

```
12 18
21
```