



Seminário de Linguagens de Programação

Tema: C++

Disciplina: Orientação à Objetos II.

Professor: Hugo Perlin.

Alunos: George Lincon Veloso Cruz, Guilherme Arthur e Jefferson Rodrigo da Silva Chaves.

Introdução

O C++ foi inicialmente desenvolvido por Bjarne Stroustrup, durante a década de 1980 com o objetivo de melhorar a linguagem de programação C, mantendo a compatibilidade com esta linguagem.

As linguagens que também serviram de inspiração para o cientista da computação foram ALGOL 68, Ada, CLU e ML.



Bjarne Stroustrup
Idealizador da Linguagem C++

Histórico do C++

- **Início da Década de 1980:** Bjarne Stroustrup inicia o desenvolvimento do C++ enquanto trabalhava nos laboratórios Bell da AT&T. Ele estava procurando uma maneira de estender a linguagem C com recursos de programação orientada a objetos, mantendo a eficiência do C.
- **1983:** O nome "C++" é adotado. A escolha do nome sugere uma evolução do C, indicado pelo operador "++" que incrementa o valor de uma variável.
- **1985:** A primeira edição do "The C++ Programming Language," livro escrito por Stroustrup, é publicada. Isso marca o início da divulgação pública do C++. Neste mesmo ano, é lançada uma versão preliminar do compilador C++.
- **1989:** O C++ 2.0 é lançado, introduzindo melhorias significativas. Nessa versão, Stroustrup e sua equipe resolveram muitas questões e desafios encontrados nas primeiras implementações.
- **1990:** O Comitê Padrão Americano de Linguagem de Programação C++ (ANSI) formaliza o primeiro padrão para a linguagem C++. Este é um passo crucial para a padronização da linguagem.
- **1998:** A Organização Internacional de Normalização (ISO) publica o padrão C++98, estabelecendo as bases para a linguagem. Esta padronização ajudou a garantir a portabilidade do código entre diferentes compiladores.

Histórico do C++

- **2003:** É lançado o padrão C++03, uma revisão menor do C++98, corrigindo alguns problemas e ambiguidades.
- **2011:** O C++11 é lançado, introduzindo uma série de novos recursos, como suporte a programação multithread, inferência de tipo, ponteiros inteligentes, lambdas e melhorias na sintaxe.
- **2014:** O C++14 é lançado como uma atualização do C++11, adicionando algumas melhorias e correções.
- **2017:** O C++17 é lançado, trazendo mais recursos, incluindo melhorias na linguagem, bibliotecas e suporte a paralelismo.
- **2020:** O C++20 é lançado, introduzindo várias melhorias, como conceitos, ranges, e melhorias em paralelismo.

Características do C++

- * C++ é desenvolvido para ser o quanto mais compatível com C possível, fornecendo transições simples para o código;
- * C++ é desenvolvido para suportar múltiplos paradigmas de programação, principalmente a programação estruturada e a programação orientada a objetos, possibilitando múltiplas maneiras de resolver um mesmo problema;
- * C++ é desenvolvido para fornecer ao programador múltiplas escolhas, mesmo que seja possível ao programador escolher a opção errada.

Exemplos de Aplicações Escritas em C++

- * Grande parte dos programas da Microsoft, incluindo Windows XP, Windows NT, Windows 9x, Pacote Office, Internet Explorer, Visual Studio e outros;
- * Sistemas Operacionais como o já citado Windows, Apple OS X, BeOS, Solaris e Symbian (sistema operacional para celulares);
- * Aplicações gráficas como os programas da Adobe (Photoshop, Illustrator), Maya e AutoCAD;
- * Aplicações Web, como a máquina de busca Google e o sistema de comércio virtual da Amazon.

Vantagens do C++

- * Possibilidade em programação de alto e baixo nível;
- * Alta flexibilidade, portabilidade e consistência;
- * Compatibilidade com C, resultando em vasta base de códigos;
- * Ampla disponibilidade e suporte, devido principalmente à grande base de desenvolvedores;
- * Adequado para grandes projetos.

Desvantagens do C++

- * Compatibilidade com o C, herdou os problemas de entendimento de sintaxe do mesmo;
- * Os compiladores atuais nem sempre produzem o código mais otimizado, tanto em velocidade quanto tamanho do código;
- * Devido à grande flexibilidade no desenvolvimento, é recomendado o uso de padrões de programação mais amplamente que em outras linguagens;
- * Grande período para o aprendizado.

Paradigmas da Programação C++

* A linguagem C++ é uma das linguagens que suportam vários paradigmas. Inicialmente, sendo uma “evolução” de C, ela suporta inteiramente o paradigma da programação estruturada. Além disso, ela suporta outros paradigmas como a programação procedural, a programação genérica, abstração de dados e a programação orientada a objetos. Dentre estes paradigmas, o mais utilizado atualmente é a Programação Orientada a Objetos (POO) que apesar de ter sido criada nos anos 60, este paradigma só começou a ganhar aceitação maior após os anos 90 com a explosão das linguagens C++, Java e Visual Basic.

Compiladores para C++

Existem muitos compiladores de C++ no mercado. Os mais famosos são os softwares da Borland e da Microsoft, que oferecem muitos recursos. O problema é que estes compiladores são caros e voltados principalmente para programadores experientes, que podem fazer uso dos recursos avançados destes programas.

O Dev-C++ é um compilador freeware das linguagens C, C++ e C#. É uma opção muito interessante, pois é de fácil utilização e aprendizado para usuários novos e possui muitos recursos avançados para usuários experientes. Além de, claro, seu download ser gratuito.

Onde baixar Dev-C++?

<http://www.bloodshed.net/devcpp.html>

Diferenças do C++ para o Java

1. Paradigma de Programação:

- **C++:** Suporta programação procedural, orientada a objetos e genérica. Oferece controle de baixo nível sobre a memória.
- **Java:** Foca principalmente na programação orientada a objetos e segue um modelo de execução gerenciado com coleta de lixo.

2. Gerenciamento de Memória:

- **C++:** Requer que o programador gerencie manualmente a alocação e liberação de memória. Isso oferece mais controle, mas também aumenta o risco de erros de alocação/desalocação.
- **Java:** Possui um sistema de coleta de lixo automático, o que simplifica o gerenciamento de memória. Programadores não precisam se preocupar explicitamente com alocação/desalocação de memória.

3. Ponteiro e Referências:

- **C++:** Permite o uso de ponteiros, o que oferece grande flexibilidade, mas também pode levar a erros de acesso à memória.
- **Java:** Não possui ponteiros e utiliza referências para objetos. A manipulação direta de memória é mais restrita, o que contribui para a segurança.

Diferenças do C++ para o Java

4. Sistema de Tipos:

- **C++:** Oferece tipagem forte e estática. O programador precisa declarar o tipo das variáveis e o tipo é verificado em tempo de compilação.
- **Java:** Também possui tipagem forte, mas é tipada dinamicamente. O tipo de uma variável é verificado em tempo de execução.

5. Herança Múltipla:

- **C++:** Suporta herança múltipla, permitindo que uma classe herde de várias classes.
- **Java:** Não suporta herança múltipla direta de classes, mas permite a implementação de múltiplas interfaces.

6. Compilação e Interpretação:

- **C++:** Linguagem compilada, onde o código-fonte é traduzido para código de máquina antes da execução.
- **Java:** Usa uma abordagem de compilação para bytecode, que é interpretado pela Máquina Virtual Java (JVM) durante a execução.

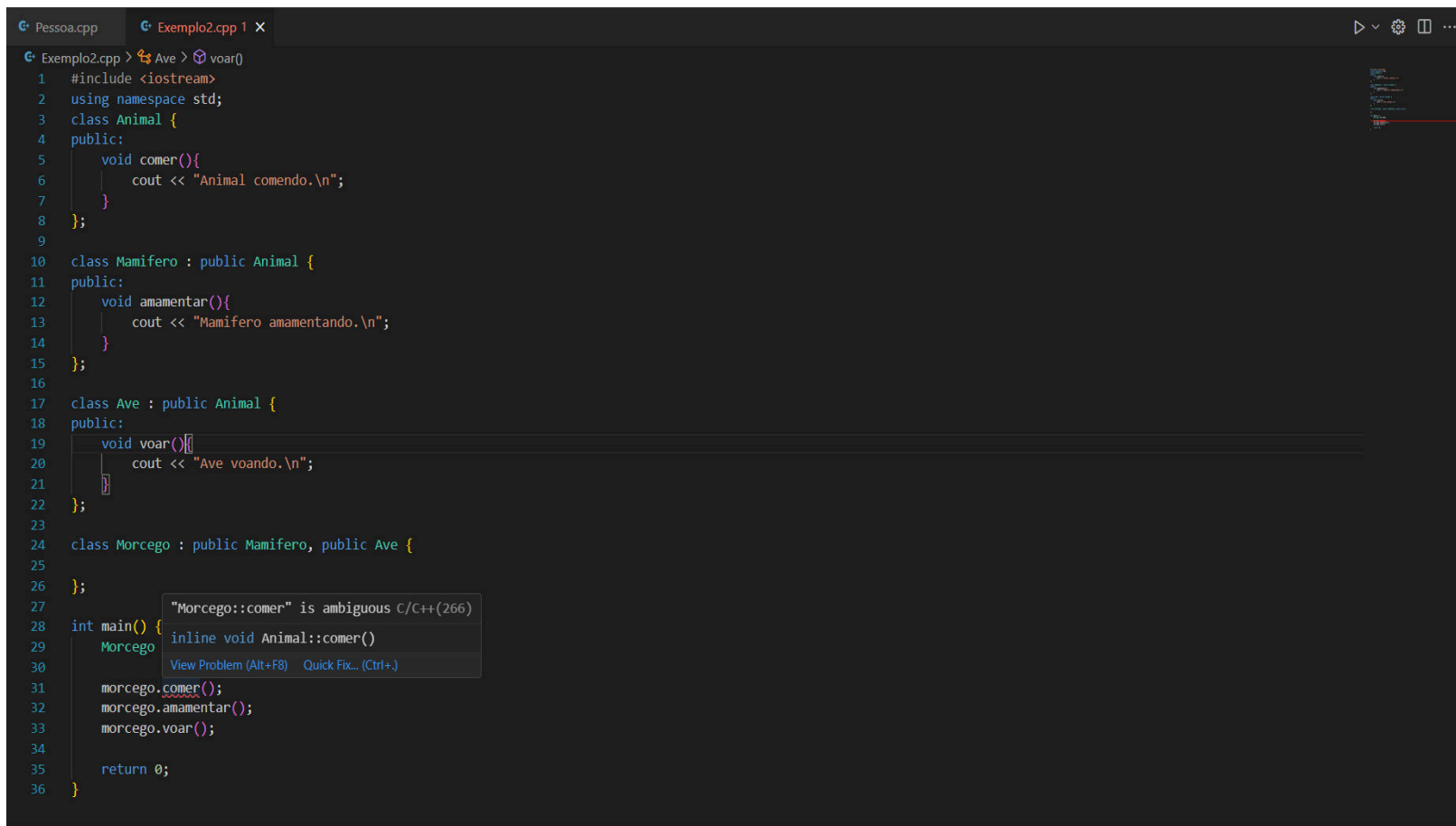
Classe, Objeto, Construtor e Encapsulamento

```
g Pessoa.cpp > main()
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  class Pessoa
7  {
8  public:
9      Pessoa() : nome(""), email(""), telefone(""), idade(0) {}
10
11      Pessoa(string n, string e, string t, int i) : nome(n), email(e), telefone(t), idade(i) {}
12
13 >  string getName(){...
16 >  string getEmail(){...
19 >  string getTelefone(){...
22 >  int getIdade(){...
25 >  void setName(string n){...
28 >  void setEmail(string e){...
31 >  void setTelefone(string t){...
34 >  void setIdade(int i){...
37 >  string info2(){...
43 private:
44     string nome;
45     string email;
46     string telefone;
47     int idade;
48 };
49 int main()
50 {
51     Pessoa p;
52     Pessoa p2("pedro", "p@gmail.com", "1234", 50);
53
54     cout<<p.info2();
55     cout<<p2.info2();
56
57     return 0;
58 }
```

```
Nome:
E-mail:
Telefone:
Idade: 0

Nome: pedro
E-mail: p@gmail.com
Telefone: 1234
Idade: 50
```

Herança Múltipla e o Problema do Diamante



```
Exemplo2.cpp > Ave > voar()
1  #include <iostream>
2  using namespace std;
3  class Animal {
4  public:
5      void comer(){
6          cout << "Animal comendo.\n";
7      }
8  };
9
10 class Mamifero : public Animal {
11 public:
12     void amamentar(){
13         cout << "Mamifero amamentando.\n";
14     }
15 };
16
17 class Ave : public Animal {
18 public:
19     void voar(){
20         cout << "Ave voando.\n";
21     }
22 };
23
24 class Morcego : public Mamifero, public Ave {
25
26 };
27
28 int main() {
29     Morcego morcego;
30     morcego.comer();
31     morcego.amamentar();
32     morcego.voar();
33
34     return 0;
35 }

"Morcego::comer" is ambiguous C/C++(266)
inline void Animal::comer()
View Problem (Alt+F8) Quick Fix... (Ctrl+.)
```

Um jeito de resolver esse problema

```
Exemplo2.cpp > main()
1  #include <iostream>
2  using namespace std;
3  class Animal {
4  public:
5      void comer(){
6          cout << "Animal comendo.\n";
7      }
8  };
9
10 class Mamifero : public Animal {
11 public:
12     void amamentar(){
13         cout << "Mamifero amamentando.\n";
14     }
15 };
16
17 class Ave : public Animal {
18 public:
19     void voar(){
20         cout << "Ave voando.\n";
21     }
22 };
23
24 class Morcego : public Mamifero, public Ave {
25
26 };
27
28 int main() {
29     Morcego morcego;
30
31     morcego.Mamifero::comer();
32     morcego.Ave::comer();
33     morcego.amamentar();
34     morcego.voar();
35
36     return 0;
37 }
```

```
Animal comendo.
Animal comendo.
Mamifero amamentando.
Ave voando.
```

Polimorfismo

Exemplo3.cpp

Exemplo3.cpp > Carro > mover()

```
1  #include <iostream>
2
3  using namespace std;
4
5  class Veiculo
6  {
7  public:
8      virtual void mover(){
9          cout<<"O Veiculo esta se movendo"<<endl;
10     }
11 };
12
13 class Carro : public Veiculo
14 {
15 public:
16     void mover() override{
17         cout<<"O Carro esta se movendo!"<<endl;
18     }
19 };
20
21 class Moto : public Veiculo
22 {
23 public:
24     void mover() override{
25         cout<<"A Moto esta se movendo!"<<endl;
26     }
27 };
28
29 int main() {
30     Carro carro;
31     Moto moto;
32
33     carro.mover();
34     moto.mover();
35     return 0;
36 }
37
```

O Carro esta se movendo!
A Moto esta se movendo!

Exemplo de Algoritmo em C++

```
// AloMundo
#include <iostream.h>
#include <stdlib.h>
Int main()
{
cout << "Alo, Mundo!\n";
system("pause");
return 0;
} // Fim de main()
```

Bibliografia

DEITEL, Harvey, DEITEL, Paul. C++: Como programar. 5ª ed., Editora Pearson. 2006.

MALIK, D. S. C++ Programming: From Problem Analysis to Program Design, 5ª ed., Cengage Learning. 2010.