

Linguagem C - Introdução

Programação e Desenvolvimento de Software I

Gleison S. D. Mendonça, Luigi D. C. Soares
`{gleison.mendonca, luigi.domenico}@dcc.ufmg.br`



Linguagem C

- ▶ Criada em 1972
- ▶ Linguagem imperativa
 - ▶ Uso de comandos: Se **x** Faça! **y**
 - ▶ Comandos alteram o estado do programa



Linguagem C

- ▶ Criada em 1972
- ▶ Linguagem imperativa
 - ▶ Uso de comandos: Se x Faça! y
 - ▶ Comandos alteram o estado do programa
- ▶ Linguagem procedural
 - ▶ Decomposição de um problema em diferentes funções e módulos
 - ▶ Reutilização de código

Linguagem C

- ▶ Criada em 1972
- ▶ Linguagem imperativa
 - ▶ Uso de comandos: Se x Faça! y
 - ▶ Comandos alteram o estado do programa
- ▶ Linguagem procedural
 - ▶ Decomposição de um problema em diferentes funções e módulos
 - ▶ Reutilização de código
- ▶ Acesso de baixo nível à memória
- ▶ Utilização de instruções Assembly (código de máquina)
- ▶ Gera códigos geralmente mais velozes

Linguagem C

- ▶ Criada em 1972
- ▶ Linguagem imperativa
 - ▶ Uso de comandos: Se x Faça! y
 - ▶ Comandos alteram o estado do programa
- ▶ Linguagem procedural
 - ▶ Decomposição de um problema em diferentes funções e módulos
 - ▶ Reutilização de código
- ▶ Acesso de baixo nível à memória
- ▶ Utilização de instruções Assembly (código de máquina)
- ▶ Gera códigos geralmente mais velozes
- ▶ Sair de C para outras linguagens é mais fácil que o sentido oposto

Primeiro programa em C

```
1  #include <stdio.h> // Inclusão do módulo de entrada/saída
2
3  // Ponto de entrada do programa
4  int main(int argc, char *argv[]) {
5      // TODO
6      return 0;
7  }
```

Primeiro programa em C

```
1  #include <stdio.h> // Inclusão do módulo de entrada/saída
2
3  // Ponto de entrada do programa
4  int main(int argc, char *argv[]) {
5      // Escreve uma mensagem na tela de saída
6      printf("Olá mundo\n");
7      return 0;
8  }
```



Indentação

- ▶ Espaçamento colocado antes de começar a escrever o código na linha

Indentação

- ▶ Espaçamento colocado antes de começar a escrever o código na linha
- ▶ É importante!

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]) { printf("Olá mundo\n"); return 0; }
```

Primeiro problema

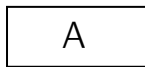
Suponha que soma (+) e subtração (-) são as únicas operações disponíveis.

Dados dois números inteiros positivos **A** e **B**, determine o **quociente** e o **resto** da divisão de **A** por **B**.

Primeiro problema

Ideia de solução:

- ▶ Representar os números **A** e **B** por retângulos de **larguras** proporcionais aos seus valores
- ▶ Verificar quantas vezes **B** cabe em **A**



- ▶ $A = 7$
- ▶ $B = 3$



Algoritmo

- ▶ Para resolver um problema, primeiro precisamos de uma descrição clara da solução
- ▶ Um algoritmo é uma sequência de instruções (receita)
 - ▶ Finita
 - ▶ Não pode ser ambígua

Algoritmo

- ▶ Para resolver um problema, primeiro precisamos de uma descrição clara da solução
- ▶ Um algoritmo é uma sequência de instruções (receita)
 - ▶ Finita
 - ▶ Não pode ser ambígua

Voltando ao nosso problema:

Algoritmo

- ▶ Para resolver um problema, primeiro precisamos de uma descrição clara da solução
- ▶ Um algoritmo é uma sequência de instruções (receita)
 - ▶ Finita
 - ▶ Não pode ser ambígua

Voltando ao nosso problema:

- ▶ Sejam A e B os valores dados
- ▶ Atribuir o valor 0 ao quociente (q)

Algoritmo

- ▶ Para resolver um problema, primeiro precisamos de uma descrição clara da solução
- ▶ Um algoritmo é uma sequência de instruções (receita)
 - ▶ Finita
 - ▶ Não pode ser ambígua

Voltando ao nosso problema:

- ▶ Sejam A e B os valores dados
- ▶ Atribuir o valor 0 ao quociente (q)
- ▶ Enquanto $B \leq A$:

Algoritmo

- ▶ Para resolver um problema, primeiro precisamos de uma descrição clara da solução
- ▶ Um algoritmo é uma sequência de instruções (receita)
 - ▶ Finita
 - ▶ Não pode ser ambígua

Voltando ao nosso problema:

- ▶ Sejam A e B os valores dados
- ▶ Atribuir o valor 0 ao quociente (q)
- ▶ Enquanto $B \leq A$:
 - ▶ Somar 1 ao valor de q
 - ▶ Subtrair B do valor de A

Algoritmo

- ▶ Para resolver um problema, primeiro precisamos de uma descrição clara da solução
- ▶ Um algoritmo é uma sequência de instruções (receita)
 - ▶ Finita
 - ▶ Não pode ser ambígua

Voltando ao nosso problema:

- ▶ Sejam A e B os valores dados
- ▶ Atribuir o valor 0 ao quociente (q)
- ▶ Enquanto $B \leq A$:
 - ▶ Somar 1 ao valor de q
 - ▶ Subtrair B do valor de A
- ▶ Atribuir o valor final de A ao resto (r)

Implementando o algoritmo

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]) {
3      // TODO
4  }
```

Implementando o algoritmo

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]) {
3      int a = 7;
4      int b = 3;
5      int q = 0; // Inicializando quociente
6      while (b <= a) {
7          q = q + 1; // Somar 1 ao valor de q
8          a = a - b; // Subtrair B do valor de A
9      }
10     int r = a; // resto = valor final de A
11     printf("Quociente: %d\n", q);
12     printf("Resto: %d\n", r);
13     return 0;
14 }
```

Algoritmo vs Implementação

- ▶ Note que o algoritmo (pseudo-código) que escrevemos anteriormente é mais alto nível, sem muitas regras
- ▶ A implementação em alguma linguagem (nosso caso = C), por outro lado, obedece **regras de sintaxe**

Erro de Sintaxe

- ▶ Em C, todo comando deve ser terminado por ponto e vírgula

Erro de Sintaxe

- ▶ Em C, todo comando deve ser terminado por ponto e vírgula

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]) {
3      int a = 7;
4      return 0 // expected ';' before '}' token
5  }
```

Erro de Lógica

- ▶ **Atenção!** Não basta obter um programa executável!! Será que ele está correto?
- ▶ Considere o seguinte problema: determinar e exibir o valor de $y = \text{seno}(1,5)$

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main(int argc, char *argv[]) {
5      // TODO
6      return 0;
7  }
```

Erro de Lógica

- ▶ **Atenção!** Não basta obter um programa executável!! Será que ele está correto?
- ▶ Considere o seguinte problema: determinar e exibir o valor de $y = \text{seno}(1,5)$

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main(int argc, char *argv[]) {
5      float y = sin(1.5);
6      printf("seno de 1,5 = %f\n", y);
7      return 0;
8  }
```

Erro de Lógica

- ▶ Se ao invés de `y = sin(1.5);` tivéssemos escrito `y = sin(2.5);`, o programa seria produzido e executaria normalmente

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main(int argc, char *argv[]) {
5      float y = sin(2.5);
6      printf("seno de 1,5 = %f\n", y);
7      return 0;
8  }
```

Erro de Lógica

- ▶ Se ao invés de `y = sin(1.5);` tivéssemos escrito `y = sin(2.5);`, o programa seria produzido e executaria normalmente
- ▶ Embora um resultado tenha sido obtido, ele **não é correto**
- ▶ Se um programa executável não produz os resultados corretos, é porque ele contém **erros de lógica** ou **bugs**
- ▶ O processo de identificação e correção de erros de lógica é denominado **depuração** (**debug**).

Como o computador entende o que escrevemos?

- ▶ Nosso programa em C é um texto
- ▶ O computador entende textos?

Como o computador entende o que escrevemos?

- ▶ Nosso programa em C é um texto
- ▶ O computador entende textos?
- ▶ **Não!** Computadores executam instruções escritas em formato binário (0's e 1's)
- ▶ Precisamos traduzir nosso texto para **linguagem de máquina**

Como o computador entende o que escrevemos?

- ▶ Nosso programa em C é um texto
- ▶ O computador entende textos?
- ▶ **Não!** Computadores executam instruções escritas em formato binário (0's e 1's)
- ▶ Precisamos traduzir nosso texto para **linguagem de máquina**



Como o computador entende o que escrevemos?

- ▶ Nosso programa em C é um texto
- ▶ O computador entende textos?
- ▶ **Não!** Computadores executam instruções escritas em formato binário (0's e 1's)
- ▶ Precisamos traduzir nosso texto para **linguagem de máquina**



Como o computador entende o que escrevemos?

- ▶ Nosso programa em C é um texto
- ▶ O computador entende textos?
- ▶ **Não!** Computadores executam instruções escritas em formato binário (0's e 1's)
- ▶ Precisamos traduzir nosso texto para **linguagem de máquina**



- ▶ O processo de “tradução” do código é chamado de **compilação**
- ▶ O **compilador** recebe o **código fonte** e produz um **programa executável**
- ▶ `gcc main.c -o main`

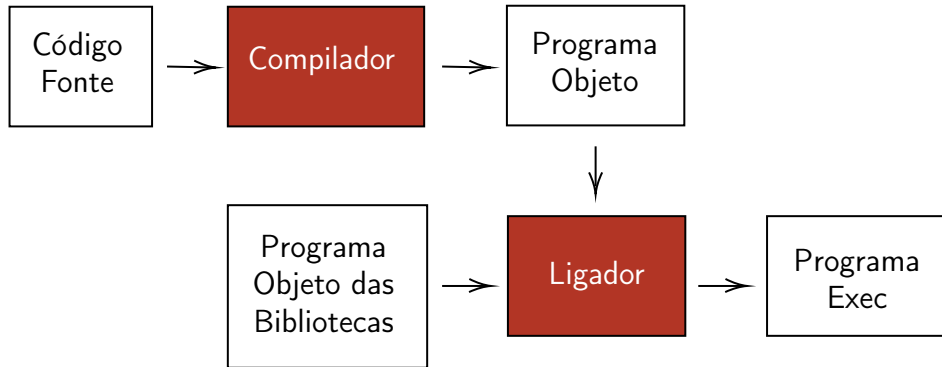


Compilação

- ▶ É um pouquinho mais complexo que isso

Compilação

- ▶ É um pouquinho mais complexo que isso



Ambiente de Programação

- ▶ O que vamos precisar?

Ambiente de Programação

- ▶ O que vamos precisar?
- ▶ Compilador de C: gcc
 - ▶ Linux: já vem instalado
 - ▶ Windows: MinGW (tutorial de instalação no Moodle)

Ambiente de Programação

- ▶ O que vamos precisar?
- ▶ Compilador de C: gcc
 - ▶ Linux: já vem instalado
 - ▶ Windows: MinGW (tutorial de instalação no Moodle)
- ▶ Editor de texto
 - ▶ Bloco de notas
 - ▶ Notepad++: <https://notepad-plus-plus.org>
 - ▶ Visual Studio Code (VS code): <https://code.visualstudio.com/>
 - ▶ Atom: <https://atom.io>
 - ▶ Sublime: <https://www.sublimetext.com/>

Ambiente de Programação

- ▶ O que vamos precisar?
- ▶ Compilador de C: gcc
 - ▶ Linux: já vem instalado
 - ▶ Windows: MinGW (tutorial de instalação no Moodle)
- ▶ Editor de texto
 - ▶ Bloco de notas
 - ▶ Notepad++: <https://notepad-plus-plus.org>
 - ▶ Visual Studio Code (VS code): <https://code.visualstudio.com/>
 - ▶ Atom: <https://atom.io>
 - ▶ Sublime: <https://www.sublimetext.com/>
- ▶ Ambiente integrado: Code::Blocks (tutorial no Moodle)

Ambiente de Programação

- ▶ O que vamos precisar?
- ▶ Compilador de C: gcc
 - ▶ Linux: já vem instalado
 - ▶ Windows: MinGW (tutorial de instalação no Moodle)
- ▶ Editor de texto
 - ▶ Bloco de notas
 - ▶ Notepad++: <https://notepad-plus-plus.org>
 - ▶ Visual Studio Code (VS code): <https://code.visualstudio.com/>
 - ▶ Atom: <https://atom.io>
 - ▶ Sublime: <https://www.sublimetext.com/>
- ▶ Ambiente integrado: Code::Blocks (tutorial no Moodle)
- ▶ Ambiente online: <https://repl.it>