

Algoritmos e Programação

Aula 12

Introdução à Linguagem C



Conteúdo

- Linguagens de Programação
- Declarações de variáveis
- Atribuição e expressões aritméticas
- Entrada e saída de dados

Linguagem de Programação

- Uma **linguagem de programação** é uma linguagem formal utilizada para **escrever programas de computador**.

Linguagem de Programação

- Uma **linguagem de programação** é uma linguagem formal utilizada para **escrever programas de computador**.
 - **Linguagem formal**: com sintaxe (como escrever) e semântica (o que significa) precisamente definidas.

Linguagem de Programação

- Uma **linguagem de programação** é uma linguagem formal utilizada para **escrever programas de computador**.
 - **Linguagem formal**: com sintaxe (como escrever) e semântica (o que significa) precisamente definidas.
- Apesar de serem mais restritas que linguagens naturais (português, inglês, vietnamita..), são bem **mais legíveis** que **linguagem de máquina** (binário).

Linguagem de Programação

- Uma **linguagem de programação** é uma linguagem formal utilizada para **escrever programas de computador**.
 - **Linguagem formal**: com sintaxe (como escrever) e semântica (o que significa) precisamente definidas.
- Apesar de serem mais restritas que linguagens naturais (português, inglês, vietnamita..), são bem **mais legíveis** que **linguagem de máquina** (binário).
- Exemplos: **C**, C++, C#, Java, Python, PHP,

Linguagem de Programação:

- Um programa escrito numa linguagem de programação qualquer é um **arquivo de texto**, contendo as instruções que o computador deve executar.

Linguagem de Programação:

- Um programa escrito numa linguagem de programação qualquer é um **arquivo de texto**, contendo as instruções que o computador deve executar.
- Este arquivo é chamado de **código fonte**.

Linguagem de Programação:

- Para ser executado, o programa precisa ser **traduzido** para a **linguagem de máquina**.
 - A linguagem de computadores é o sistema binário: 0 e 1.

Linguagem de Programação:

- Para ser executado, o programa precisa ser **traduzido** para a **linguagem de máquina**.
 - A linguagem de computadores é o sistema binário: 0 e 1.

Linguagem C

```
#include <stdio.h>

int main() {
    printf("Ola, mundo!\n");
    return(0);
}
```

Linguagem Binária

```
0110011101011010000101
0101010101010101011001
0111010101100101010101
0101111010100001101010
0100101001111101010111
1100000010100110
```

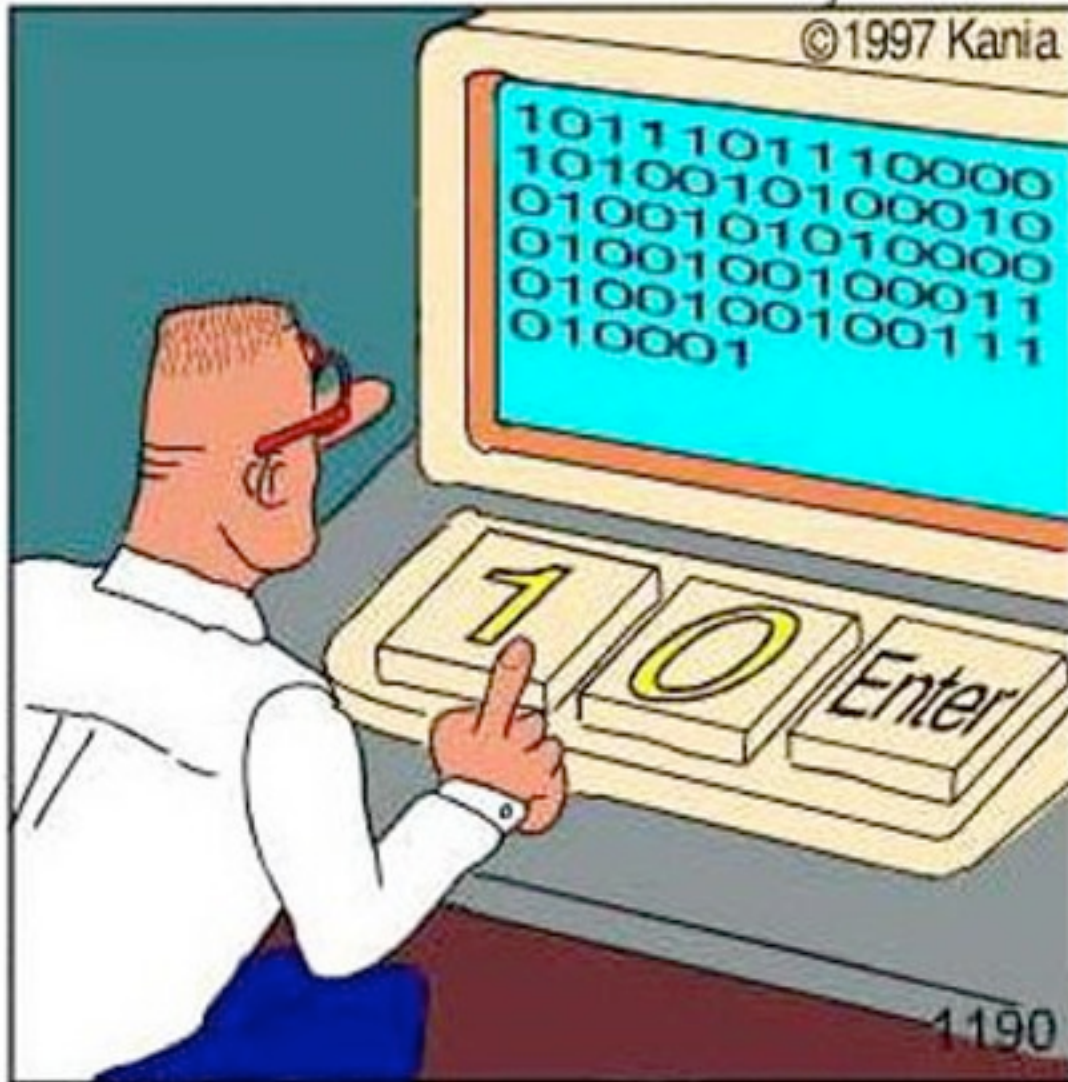
Linguagem de Programação:

- Para se traduzir

– A linguagem binária

```
#include
```

```
int main(  
    print  
    return  
}
```



Real programmers code in binary.

isa ser
quina.

ma

n Binária

```
010000101  
101011001  
101010101  
001101010  
101010111  
110
```

Linguagem de Programação:

- O processo de **tradução** é feito por **outro programa**, que pode se chamar:

Linguagem de Programação:

- O processo de **tradução** é feito por **outro programa**, que pode se chamar:
 - **Compilador**: lê o código-fonte inteiro e cria um arquivo executável (.exe) contendo as mesmas instruções traduzidas para linguagem de máquina.
 - A linguagem C é compilada!

Linguagem de Programação:

- O processo de **tradução** é feito por **outro programa**, que pode se chamar:
 - **Compilador**: lê o código-fonte inteiro e cria um arquivo executável (.exe) contendo as mesmas instruções traduzidas para linguagem de máquina.
 - A linguagem C é compilada!
 - **Interpretador**: lê o código fonte linha a linha, e vai executando as instruções à medida que são lidas (sem criar um arquivo executável).
 - O PHP, por exemplo, é uma linguagem interpretada.

Processos

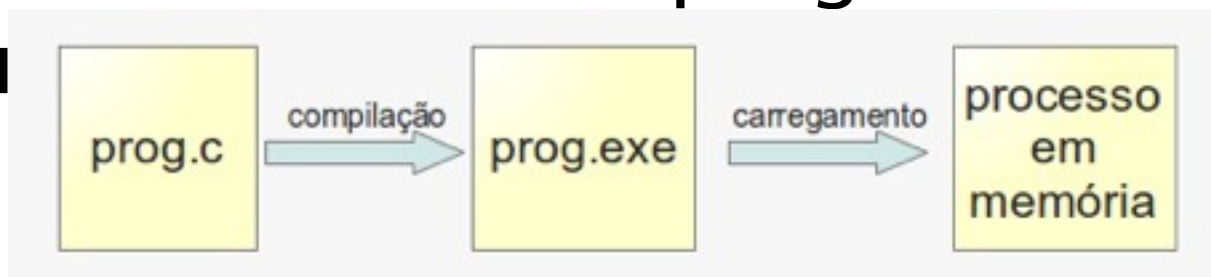
- Os arquivos executáveis (.exe) são **inertes até que o sistema operacional os carregue em memória**, e passe o controle para eles.

Processos

- Os arquivos executáveis (.exe) são **inertes até que o sistema operacional os carregue em memória**, e passe o controle para eles.
 - É o que ocorre quando ‘abrimos’ um programa, por exemplo, ao clicar em um ícone na área de trabalho.

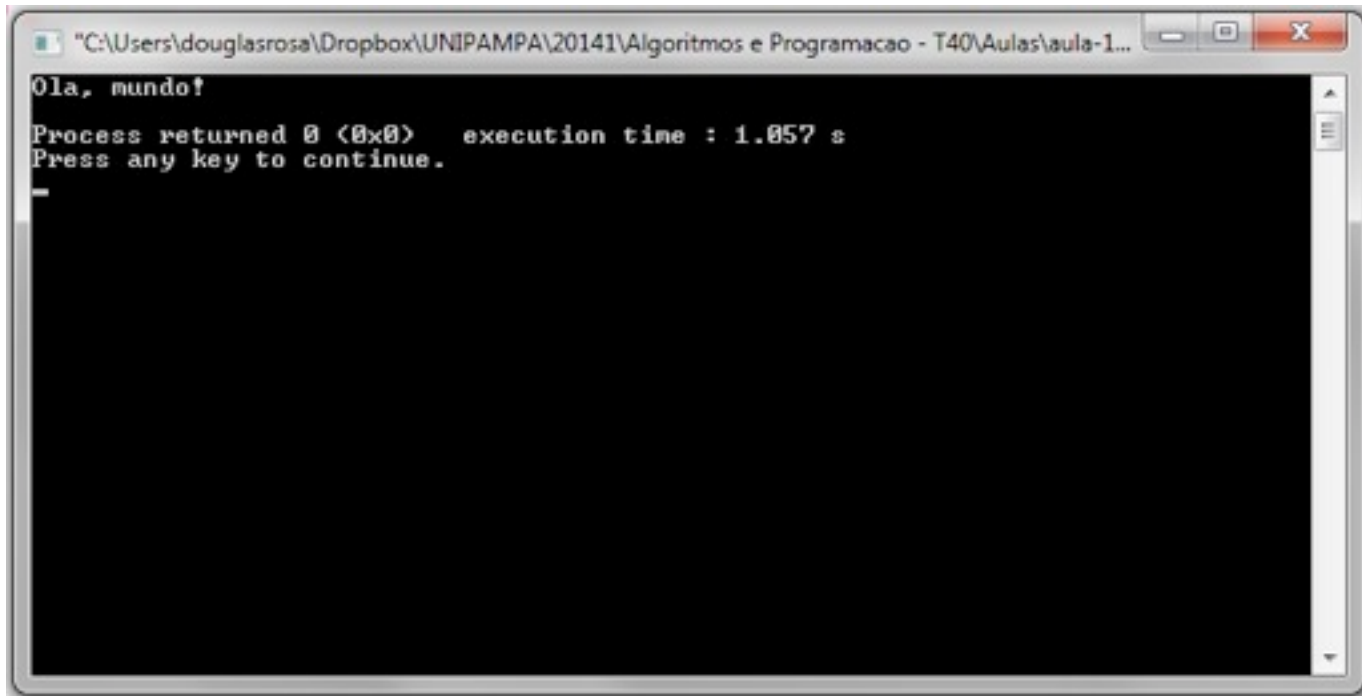
Processos

- Os arquivos executáveis (.exe) são **inertes até que o sistema operacional os carregue em memória**, e passe o controle para eles.
 - É o que ocorre quando ‘abrimos’ um programa, por exemplo, ao clicar em um ícone na área de trabalho.
- Uma instância de um programa em execu



Processos

- Programa “ola-mundo” sendo executado:



A screenshot of a Windows command prompt window. The title bar at the top reads: "C:\Users\douglasrosa\Dropbox\UNIPAMPA\20141\Algoritmos e Programacao - T40\Aulas\aula-1...". The window has standard Windows window controls (minimize, maximize, close) on the right. The command prompt area is black with white text. The text displayed is: "Ola, mundo!", "Process returned 0 (0x0) execution time : 1.057 s", and "Press any key to continue.". A single underscore character "_" is visible on the line following the prompt message.

```
Ola, mundo!  
Process returned 0 (0x0) execution time : 1.057 s  
Press any key to continue.  
_
```

Ambientes de desenvolvimento

- Um **ambiente de desenvolvimento integrado** (IDE) é um programa que contém diversas ferramentas para o desenvolvimento de programas.
- Componentes de um IDE:
 - Editor de texto;
 - Compilador ou interpretador;
 - Depurador (software para observar o processo em execução e identificar erros);
 - Documentação da linguagem de programação e bibliotecas
- Nesta disciplina utilizaremos recursos do sistema operacional Linux
- Opcionalmente, use o IDE C/C++: **Code::Blocks**
 - Link para download: <http://sourceforge.net/projects/codeblocks/files/Binaries/13.12/Windows/codeblocks-13.12mingw->

Estrutura de um programa C

- Programas em C são compostos por **declarações globais**, **declarações de funções auxiliares** e pela **função main**.

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#define PI 3.1415
```

```
int fatorial(int n){
    ...
}

float circunferencia(int r){
    ...
}
```

```
int main(){
    ...
    //aqui começa e
    termina //a execução do
    programa
}
```

Estrutura de um programa C

- Programas em C são compostos por **declarações globais**, **declarações de funções auxiliares** e pela **função main**.
- A função **main** tem presença obrigatória: é onde a execução de um programa começa e acaba.
- A **função main** deve aparecer após as declarações globais e as declarações de funções auxiliares.

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#define PI 3.1415
```

```
int fatorial(int n){
    ...
}
float circunferencia(int r){
    ...
}
```

```
int main(){
    ...
    //aqui começa e
    termina //a execução do
    programa
}
```

Estrutura de um programa C

- Programas em C são compostos por **declarações globais**, **declarações de funções auxiliares** e pela **função main**.
- A função **main** tem presença obrigatória: é onde a execução de um programa começa e acaba.
- A **função main** deve aparecer após as declarações globais e as declarações de funções auxiliares.

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#define PI 3.1415
```

```
int fatorial(int n){
    ...
}
float circunferencia(int r){
    ...
}
```

```
int main(){
    ...
    //aqui começa e
    termina //a execução do
    programa
}
```

Conteúdo

- Linguagens de Programação
- **Declarações de variáveis**
- Atribuição e expressões aritméticas
- Entrada e saída de dados

Declaração de variáveis

- A declaração de uma variável aloca uma parte da memória do computador, e dá um nome a ela para uso dentro do programa.

Declaração de variáveis

- A declaração de uma variável aloca uma parte da memória do computador, e dá um nome a ela para uso dentro do programa.

Possui o seguinte formato:

```
tipo var1, var2, ..., varn;
```

Declaração de variáveis

- A declaração de uma variável aloca uma parte da memória do computador, e dá um nome a ela para uso dentro do programa.

Possui o seguinte formato:

```
tipo var1, var2, ..., varn;
```

Onde **tipo** é um dos **tipos básicos (primitivos)**, e **var1** a **varn** são **nome válidos (identificadores)**.

```
int x, y, z;  
double largura, altura;
```

Identificadores

- Em C, podemos dar nomes para posições de memória (variáveis), funções e parâmetros.
 - Estes nomes são chamados de **identificadores** (assim como fizemos em pseudocódigo).

Identificadores

- Em C, podemos dar nomes para posições de memória (variáveis), funções e parâmetros.
 - Estes nomes são chamados de **identificadores** (assim como fizemos em pseudocódigo).
- Possuem regras para formação:
 - Começam por uma letra (maiúscula ou minúscula) ou um ‘_’ (underline).
 - Não podem começar por um dígito (número).
 - Após, podem conter letras, números ou ‘_’.
 - As letras não podem ser acentuadas.
 - Não pode haver espaços em branco.
 - Letras maiúsculas e minúsculas são consideradas diferentes (case sensitive – sensível a caixa).

Identificadores

- Exemplos de identificadores

Válidos	Inválidos
contador	0cont
media	média
soma_de_numer	soma de
Media	!media
_um_nome_	_Um nome_

Note que, **media** e **Media** são identificadores diferentes por conta da diferenciação entre maiúsculas e minúsculas.

Tipos de dados (tipos primitivos)

- C possui cinco tipos de dados básicos (primitivos):
 - **int** (números inteiros)
 - **char** (caracteres do teclado)
 - **float** (números fracionários em ponto flutuante)
 - **double** (números fracionários com precisão dupla)
 - **void** (nada)

Tipo `int`

Variáveis do tipo `int` armazenam números inteiros (positivos e negativos).

Números podem ser escritos por extenso ou em notação científica:

Como escrever	Valor
230	230
-12	-12
1.2e3	$1,2 \times 10^3 =$

Tipo char

O tipo char armazena valores numéricos (inteiros positivos) representando o código ASCII de um caractere do teclado.

Sintaxe especial – caractere entre **aspas simples**: ‘a’, ‘Z’, ‘\$’.

Alguns caracteres não são imprimíveis diretamente, portanto há códigos de escape definidos para eles:

Como escrever	Valor
‘A’	‘a’ maiúsculo = 65
‘a’	‘a’ minúsculo = 97
‘\n’	nova linha = 10
‘\t’	tabulação = 9
‘\\’	contrabarra = 92

Importante: use aspas simples! As aspas duplas representam sequências de caracteres (strings), que serão discutidas

Tipo `float` e `double`

Representam números em ponto flutuante (fracionários).

Diferença:

- **float** – precisão simples (4 bytes)
- **double** – precisão dupla (8 bytes)

Números podem ser escritos por extenso (com ponto no lugar da vírgula), ou em notação científica:

Como	Valor
230	230
-12.5	-12,5
1.2345e2	1,2345 x 10 ² = 123,45

Tipo `void`

O tipo `void` significa nada.

Dois usos básicos em C:

- Indicar que uma função não retorna nenhum valor.
- Permitir a definição de ponteiros para posições de memória sem indicar o tipo apontado

Estes usos serão detalhados

... e os tipos lógicos?

Em pseudocódigo utilizamos tipos lógicos para definir valores biestáveis (que podem assumir somente 2 valores – verdadeiro e falso).

... e os tipos lógicos?

Em pseudocódigo utilizamos tipos lógicos para definir valores biestáveis (que podem assumir somente 2 valores – verdadeiro e falso).

Muitas linguagens de programação possuem um tipo específico para valores lógicos (verdadeiro e falso), também chamados de **booleanos**.

... e os tipos lógicos?

Em pseudocódigo utilizamos tipos lógicos para definir valores biestáveis (que podem assumir somente 2 valores – verdadeiro e falso).

Muitas linguagens de programação possuem um tipo específico para valores lógicos (verdadeiro e falso), também chamados de **booleanos**.

C representa valores lógicos utilizando uma codificação numérica, e podemos utilizar variáveis inteiras (char e int) para armazená-los.

Como escrever	Valor
0	Falso
Qualquer número diferente de 0: 1, -1,	Verdadeiro

Valores iniciais de variáveis

Podemos, na hora da declaração, definir um valor inicial para a variável.

Valores iniciais de variáveis

Podemos, na hora da declaração, definir um valor inicial para a variável.

Exemplo:

```
int num_pessoas = 5, num_lugares = 25;  
char tecla_digitada = 'a';  
float temperatura = 27.5;
```

Valores iniciais de variáveis

Podemos, na hora da declaração, definir um valor inicial para a variável.

Exemplo:

```
int num_pessoas = 5, num_lugares = 25;  
char tecla_digitada = 'a';  
float temperatura = 27.5;
```

Importante: variáveis não inicializadas podem conter **qualquer valor** (lixo de memória).

Sempre inicialize as variáveis antes de utilizá-las, seja na declaração, ou no código.

Conteúdo

- Linguagens de Programação
- Declarações de variáveis
- **Atribuição e expressões aritméticas**
- Entrada e saída de dados

Atribuição

Dentro da função main (ou de outras funções), uma atribuição é um comando que **atualiza** o valor de uma variável.

Atribuição

Dentro da função main (ou de outras funções), uma atribuição é um comando que **atualiza** o valor de uma variável.

Sintaxe:

```
var = expressao;
```

Atribuição

Dentro da função main (ou de outras funções), uma atribuição é um comando que **atualiza** o valor de uma variável.

Sintaxe:

```
var = expressao;
```

Semântica: calcula a expressão ao lado direito até chegar a um valor, e então armazena na variável do lado esquerdo.

```
n = n + 2;  
Tecla_digitada = 'd';  
Media = (a + b) / 2;
```

Atribuição

Dentro da função main (ou de outras funções), uma atribuição é um comando que **atualiza** o valor de uma variável.

Sintaxe:

```
var = expressao;
```

Semântica: calcula a expressão ao lado direito até chegar a um valor, e então armazena na variável do lado esquerdo.

```
n = n + 2;  
Tecla_digitada = 'd';  
Media = (a + b) / 2;
```

Importante: o operador = em C é sempre atribuição,

Operadores Aritméticos

A expressão que vai no lado direito de uma atribuição pode ser bastante complexa, e permite diversos tipos de cálculos.

Abaixo segue uma tabela das operações aritméticas básicas em C:

Símbol	Operação	Operação	
+	Soma	Soma	
*	Multiplicação		
-	Inversão $(-X)$ ou subtração $(A - B)$		
/	Divisão (inteira ou fracionária)		
%	Resto da divisão inteira		
++	Incremento $(++x$ ou $x++)$	Multiplicação	
--	Decremento $(--x$ ou $x--)$		
+	Soma		
*	Multiplicação		
-	Inversão $(-X)$ ou subtração $(A - B)$		
/	Divisão (inteira ou fracionária)		
%	Resto da divisão inteira		
++	Incremento $(++x$ ou $x++)$	Divisão (inteira ou fracionária)	
--	Decremento $(--x$ ou $x--)$		
+	Soma		
*	Multiplicação		
-	Inversão $(-X)$ ou subtração $(A - B)$		
/	Divisão (inteira ou fracionária)	Resto da divisão inteira	
%	Resto da divisão inteira		
++	Incremento $(++x$ ou $x++)$		
--	Decremento $(--x$ ou $x--)$		
+	Soma	Divisão (inteira ou fracionária)	
*	Multiplicação		
-	Inversão $(-X)$ ou subtração $(A - B)$		
/	Divisão (inteira ou fracionária)		
%	Resto da divisão inteira		
++	Incremento $(++x$ ou $x++)$	Resto da divisão inteira	
--	Decremento $(--x$ ou $x--)$		
+	Soma		
*	Multiplicação		
-	Inversão $(-X)$ ou subtração $(A - B)$		
/	Divisão (inteira ou fracionária)	Divisão (inteira ou fracionária)	
%	Resto da divisão inteira		
++	Incremento $(++x$ ou $x++)$		
--	Decremento $(--x$ ou $x--)$		
+	Soma	Resto da divisão inteira	
*	Multiplicação		
-	Inversão $(-X)$ ou subtração $(A - B)$		
/	Divisão (inteira ou fracionária)		
%	Resto da divisão inteira		
++	Incremento $(++x$ ou $x++)$	Resto da divisão inteira	
--	Decremento $(--x$ ou $x--)$		
+	Soma		
*	Multiplicação		
-	Inversão $(-X)$ ou subtração $(A - B)$		
/	Divisão (inteira ou fracionária)	Resto da divisão inteira	
%	Resto da divisão inteira		
++	Incremento $(++x$ ou $x++)$		
--	Decremento $(--x$ ou $x--)$		

Precedência de operadores

- Assim como na matemática (e em pseudocódigo), existe uma ordem de prioridade entre os operadores aritméticos.

$$\text{Precedência de Operadores} = \begin{cases} +, -, \text{unário} \\ *, / \\ +, - \end{cases}$$

A expressão

```
x = - 3 + 6 * 2 / 3;
```

É equivalente a

```
x = (- 3) + ((6 * 2) / 3);
```

Conteúdo

- Linguagens de Programação
- Declarações de variáveis
- Atribuição e expressões aritméticas
- **Entrada e saída de dados**

Entrada e saída de dados

Existem várias formas de um **programa** se **comunicar** com o **meio externo**:

Entrada e saída de dados

Existem várias formas de um **programa** se **comunicar** com o **meio externo**:

- Ler o que foi digitado no teclado
- Perceber a movimentação e cliques nos botões do mouse
- Desenhar gráficos na tela
- Escrever texto na tela
- Interação através de janelas, botões, formulários
- Ler e escrever em arquivos no disco
- Ler e escrever em portas de comunicação via rede

Entrada e saída de dados

Existem várias formas de um **programa** se **comunicar** com o **meio externo**:

- Ler o que foi digitado no teclado
- Perceber a movimentação e cliques nos botões do mouse
- Desenhar gráficos na tela
- Escrever texto na tela
- Interação através de janelas, botões, formulários
- Ler e escrever em arquivos no disco
- Ler e escrever em portas de comunicação via rede

Nesta disciplina, vamos nos limitar à forma de comunicação mais simples possível, representada pelo uso do terminal (monitor + teclado)

Biblioteca de entrada e saída

Para realizarmos a entrada e saída de dados via terminal, precisamos da biblioteca de software `stdio.h` (standard input/output).

Biblioteca de entrada e saída

Para realizarmos a entrada e saída de dados via terminal, precisamos da biblioteca de software `stdio.h` (standard input/output).

Adicionamos as funções da biblioteca ao nosso programa usando a diretiva de compilação `#include`, no início do programa.

```
#include <stdio.h>
```

Biblioteca de entrada e saída

Para realizarmos a entrada e saída de dados via terminal, precisamos da biblioteca de software `stdio.h` (standard input/output).

Adicionamos as funções da biblioteca ao nosso programa usando a diretiva de compilação `#include`, no início do programa.

```
#include <stdio.h>
```

Outras bibliotecas de funções podem ser usadas utilizando `#include` (veremos posteriormente).

Biblioteca de entrada e saída

Para realizarmos a entrada e saída de dados via terminal, precisamos da biblioteca de software `stdio.h` (standard input/output).

Adicionamos as funções da biblioteca ao nosso programa usando a diretiva de compilação `#include`, no início do programa.

```
#include <stdio.h>
```

Outras bibliotecas de funções podem ser usadas utilizando `#include` (veremos posteriormente).

A biblioteca `stdio.h` fornece as funções `printf` (impressão formatada) e `scanf` (leitura formatada).

Escrita com `printf`

A função `printf` escreve texto no terminal.

Escrita com `printf`

A função `printf` escreve texto no terminal.

Texto = string: sequência de caracteres entre aspas duplas, sendo permitido espaço em branco, símbolos e códigos de especiais.

Escrita com `printf`

A função `printf` escreve texto no terminal.

Texto = string: sequência de caracteres entre aspas duplas, sendo permitido espaço em branco, símbolos e códigos de especiais.

```
#include <stdio.h>

int main() {
    printf("Olá, mundo!\n");
    printf("Adeus, mundo!\n");
}
```

Escrita com `printf`

Além de texto, podemos imprimir o conteúdo de variáveis e o resultado de cálculos com `printf`.

Escrita com `printf`

Além de texto, podemos imprimir o conteúdo de variáveis e o resultado de cálculos com `printf`.

Inserimos no texto um **código especial** que diz **onde** e **como** escrever o valor, e, após, colocamos a(s) expressão(ões) a serem escritas (separadas por vírgula).

Escrita com `printf`

Além de texto, podemos imprimir o conteúdo de variáveis e o resultado de cálculos com `printf`.

Inserimos no texto um **código especial** que diz **onde** e **como** escrever o valor, e, após, colocamos a(s) expressão(ões) a serem escritas (separadas por vírgula).

```
#include <stdio.h>
int main(){
    int x = 10;
    float y = 20.5;
    printf("O valor de x é %d e o valor de y é %f\n", x, y);
    printf("O resultado do cálculo é %d\n", 3 * x + y);
}
```

Escrita com `printf`

Tabela de códigos de formatação mais comuns:

Código	Descrição
<code>%c</code>	Caracteres simples
<code>%d</code>	Inteiros decimais com sinal
<code>%u</code>	Inteiros decimais sem sinal
<code>%e</code>	Notação científica
<code>%f</code>	Ponto flutuante decimal
<code>%g</code>	Usa <code>%e</code> ou <code>%f</code> (o que for mais
<code>%s</code>	Sequência de caracteres
<code>%%</code>	Símbolo de porcentagem (%)

Leitura com `scanf`

A função `scanf` lê o que o usuário digita, e armazena em variáveis.

Leitura com `scanf`

A função `scanf` lê o que o usuário digita, e armazena em variáveis.

Usamos uma **string** de formato para indicar para como a função deve ler, e, após, devemos indicar um **endereço de variável** (nome da variável precedido de `&`).

Leitura com `scanf`

A função `scanf` lê o que o usuário digita, e armazena em variáveis.

Usamos uma **string** de formato para indicar para como a função deve ler, e, após, devemos indicar um **endereço de variável** (nome da variável precedido de `&`).

```
#include <stdio.h>
int main(){
    int idade;
    printf("Escreva sua idade: ");
    scanf("%d", &idade);
    printf("Voce diz ter %d anos!\n", idade);
}
```

Vamos programar!

Olá mundo! (em C)

- O programa “Olá mundo” é um programa mínimo, que simplesmente imprime uma mensagem na tela.
- Muito famoso por seu uso em livros de algoritmos e programação.

```
#include <stdio.h>

int main() {
    printf("Olá,mundo!\n");
}
```

Tarefa para casa

- Instale o ambiente Code::Blocks
 - Tutorial de instalação: <http://aulasdec.wordpress.com/configurando-o-codeblocks-no-windows/>
- Obtenha o arquivo ola-mundo.c (no moodle)
- Carregue o arquivo no IDE (menu file → Open, escolha o arquivo ola-mundo.c)
- Compile e execute o programa (tecla F9)
- Se tudo deu certo, uma nova janela de terminal surgirá contendo a mensagem

Exercícios

- 1. Faça um programa em C que, ao completar o tanque de combustível de um automóvel, calcule o valor gasto para abastecê-lo, o consumo efetuado em e a autonomia que o carro ainda teria antes do abastecimento (em km/l). Considere que o veículo sempre seja abastecido até encher o tanque. Os dados fornecidos para o algoritmo são apenas a capacidade do tanque, a quantidade de litros abastecida, a quilometragem percorrida desde o último abastecimento e o preço do litro do combustível.
2. Faça um programa em C que recebe três números: H, M e S, que representam o número de horas, minutos e segundos, respectivamente. Seu programa deve ter como saída o tempo total em segundos.
3. Construa um programa que calcule e informe a média aritmética entre quatro notas bimestrais quaisquer fornecidas pelo usuário.
4. Faça um programa em C que recebe dois valores quaisquer e os soma, logo em seguida recebe mais três valores quaisquer e calcula média entre estes três valores. A saída deve ser o produto entre a soma e a média.