

# INTRODUÇÃO À PROGRAMAÇÃO

Prof.<sup>a</sup> Priscilla Abreu

[priscilla.braz@rj.senac.br](mailto:priscilla.braz@rj.senac.br)



# Introdução à Programação



## Roteiro de Aula

- Objetivo da aula
  - Conceitos de Programação e linguagem C
- Exercícios

# Introdução à Programação



## Objetivo da aula

Introduzir conceitos básicos da linguagem C.

# REVISANDO...

# Introdução à Programação



## Revisando...

- Algoritmos
- Linguagem de Programação
- Representações de algoritmos

# LINGUAGEM DE PROGRAMAÇÃO

# Introdução à Programação



## LINGUAGEM DE PROGRAMAÇÃO

1. Qual o tipo de linguagem que o computador entende?

Linguagem de máquina.

2. Como podemos fazer a tradução de uma linguagem de programação para linguagem de máquina?

Usando um compilador.

3. Como um programa é executado no computador?

Primeiro ele é carregado na memória e depois cada instrução é executada de acordo com a lógica do programa.

# Introdução à Programação



## COMPILAÇÃO X INTERPRETAÇÃO

### – Compilação

- Finalidade de traduzir ou converter um programa (código-fonte) escrito em uma determinada linguagem para um programa (código-objeto) que pode ser executado pelo computador.
- Mais rápidos: geram **executável** específico para um SO.



# Introdução à Programação



## COMPILAÇÃO X INTERPRETAÇÃO

### – Interpretação

- O interpretador age como uma simulação de software de uma máquina, lidando com instruções em linguagem de alto nível em vez de instruções de máquina.
- Processo mais lento: análise e geração de código em tempo de execução.

# Introdução à Programação



## LINGUAGEM DE PROGRAMAÇÃO

Paradigmas de programação

- Estilos de programação:
  - Programação Estruturada
  - Programação Orientada a Objetos
  - Programação Orientada a Aspectos
  - Programação Funcional
  - Programação Lógica
  - ...

# Introdução à Programação



## LINGUAGEM DE PROGRAMAÇÃO

Programação Estruturada – método de escrita de programas.

Qual são os objetivos da programação estruturada?

- Facilitar a escrita dos programas;
- Facilitar a leitura dos programas;
- Permitir a verificação prévia dos programas;
- Facilitar a manutenção e modificação dos programas.

# LINGUAGEM C

# Introdução à Programação



## LINGUAGEM C

- Portabilidade entre máquinas e sistemas operacionais;
- Baseada em construções simples, com uso de recursos da máquina de forma eficiente;
- Linguagem estrutural (funções e procedimentos);
- Uso amplamente difundido;
- Possui diversas funcionalidades através de bibliotecas de rotinas padronizadas.

# Introdução à Programação



## LINGUAGEM C

- A linguagem de programação C foi originalmente projetada para ser implementada no sistema operacional UNIX;
- C foi desenvolvido a partir de duas linguagens anteriores: BCPL e B.
- A linguagem foi desenvolvida por Dennis Ritchie em 1970.

# Introdução à Programação



## LINGUAGEM C x C++

- C++ é uma versão estendida de C que é projetada para suportar Programação Orientada a Objetos.
- C++ contém e suporta toda a linguagem C e mais um conjunto de extensões orientadas a objetos.
- Os compiladores C++ podem ser utilizados para compilar C.

## LINGUAGEM C – CARACTERÍSTICAS

- C é uma linguagem compilada: lê todo o código fonte e gera o código objeto (linguagem de máquina) uma única vez.
- Sempre que o código fonte for alterado ele deve ser novamente compilado.
- C é sensível ao contexto, ou seja, maiúsculas são diferentes de minúsculas.



# Introdução à Programação



## LINGUAGEM C – ESTRUTURA BÁSICA

- Um programa em C é composto por dados e funções, que são basicamente pequenas partes de um código organizadas em módulos agrupados para realizar determinadas tarefas.
- O C possui uma função especial, que é a principal função do programa. É onde ocorre o início de qualquer programa desenvolvido em C.

# Introdução à Programação



## LINGUAGEM C – ESTRUTURA BÁSICA

```
/* Primeiro Programa em C */  
#include <stdio.h>  
  
int main()  
{  
    printf("Meu primeiro programa em C\n");  
    printf("INTRODUÇÃO À PROGRAMAÇÃO!");  
}
```

Comentário

Comandos de  
pré-processamento  
biblioteca

Função  
Principal

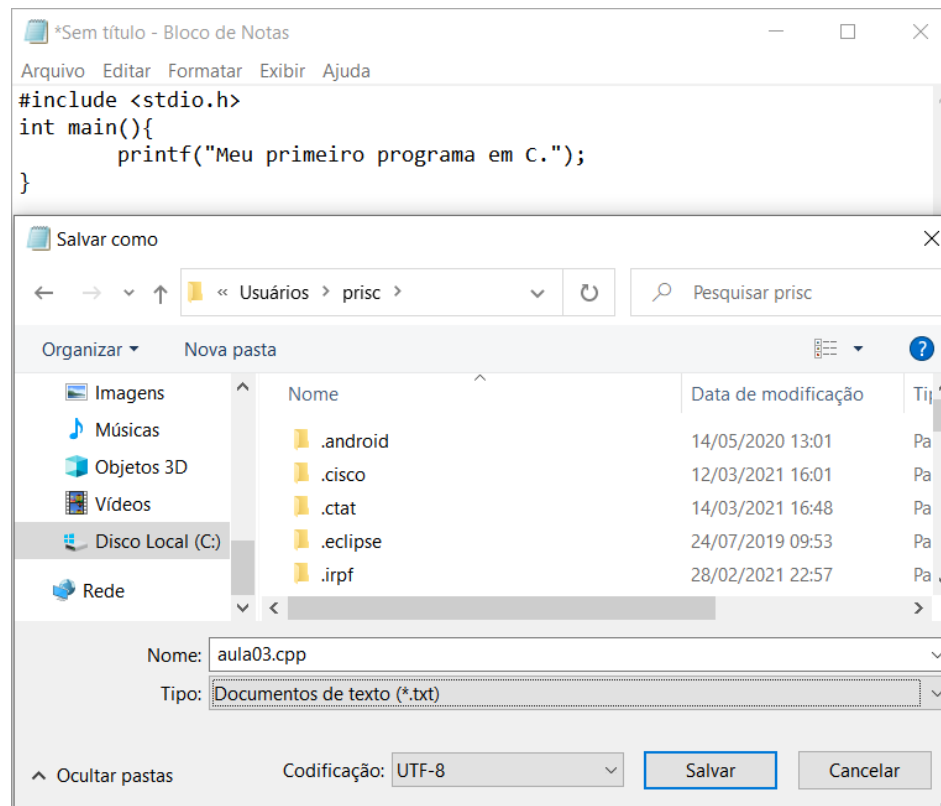
Escrevendo na  
tela

# Introdução à Programação



## EXECUTANDO UM PROGRAMA EM C

- Escrever o código;



# Introdução à Programação



## EXECUTANDO UM PROGRAMA EM C

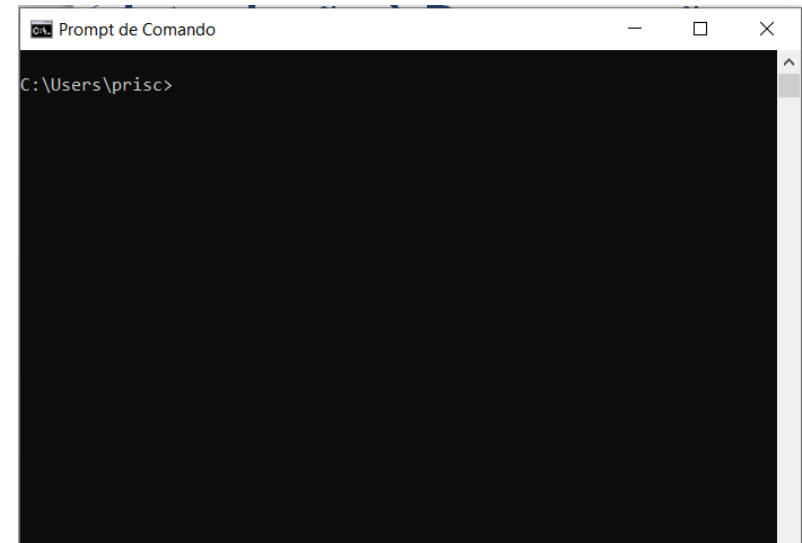
- Escrever o código;
- Compilar e executar o programa:
  - Pelo terminal (prompt de linha de comando);  
ou
  - Utilizando algum ambiente de desenvolvimento.

# Introdução à Programação



## EXECUTANDO UM PROGRAMA EM C

- Compilar e executar o programa:
  - Pelo terminal (prompt de linha de comando);
    - É preciso instalar um compilador para a linguagem C: gcc
    - Após instalação e configuração, utilizamos o terminal para iniciar o Processo de compilação e execução

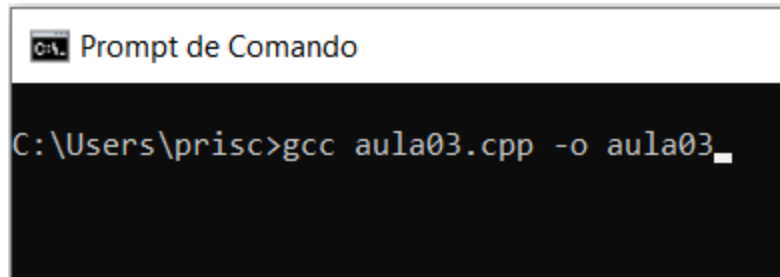


# Introdução à Programação



## EXECUTANDO UM PROGRAMA EM C

- Compilar e executar o programa:
  - Pelo terminal (prompt de linha de comando);
    - Comando para compilar o arquivo:
      - `gcc nome_arquivo.cpp -o nome_executável`

A screenshot of a Windows Command Prompt window. The title bar reads "Prompt de Comando". The command prompt shows the directory "C:\Users\prisc>" followed by the command "gcc aula03.cpp -o aula03\_".

```
Prompt de Comando

C:\Users\prisc>gcc aula03.cpp -o aula03_
```

# Introdução à Programação



## EXECUTANDO UM PROGRAMA EM C

- Compilar e executar o programa:
  - Pelo terminal (prompt de linha de comando);
    - Executando o arquivo gerado (programa):  
nome\_executável

Resultado da  
execução

C:\> Prompt de Comando

```
C:\Users\prisc>gcc aula03.cpp -o aula03
```

```
C:\Users\prisc>aula03_
```

C:\> Prompt de Comando

```
C:\Users\prisc>gcc aula03.cpp -o aula03
```

```
C:\Users\prisc>aula03
```

```
Meu primeiro programa em C.
```

```
C:\Users\prisc>
```

# Introdução à Programação



## EXECUTANDO UM PROGRAMA EM C

- Compilar e executar o programa:
  - Pelo terminal (prompt de linha de comando);
    - Em caso de erro...

Ponto onde se encontra o erro

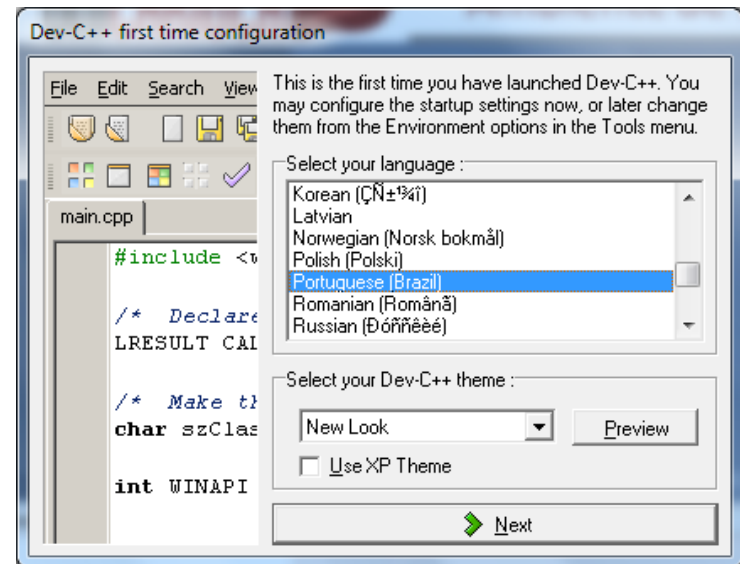
```
C:\Users\prisc> gcc aula03.cpp -o aula03
aula03.cpp: In function 'int main()':
aula03.cpp:3:37: error: 'print' was not declared in this scope
  print("Meu primeiro programa em C.");
  ^
```



# Introdução à Programação



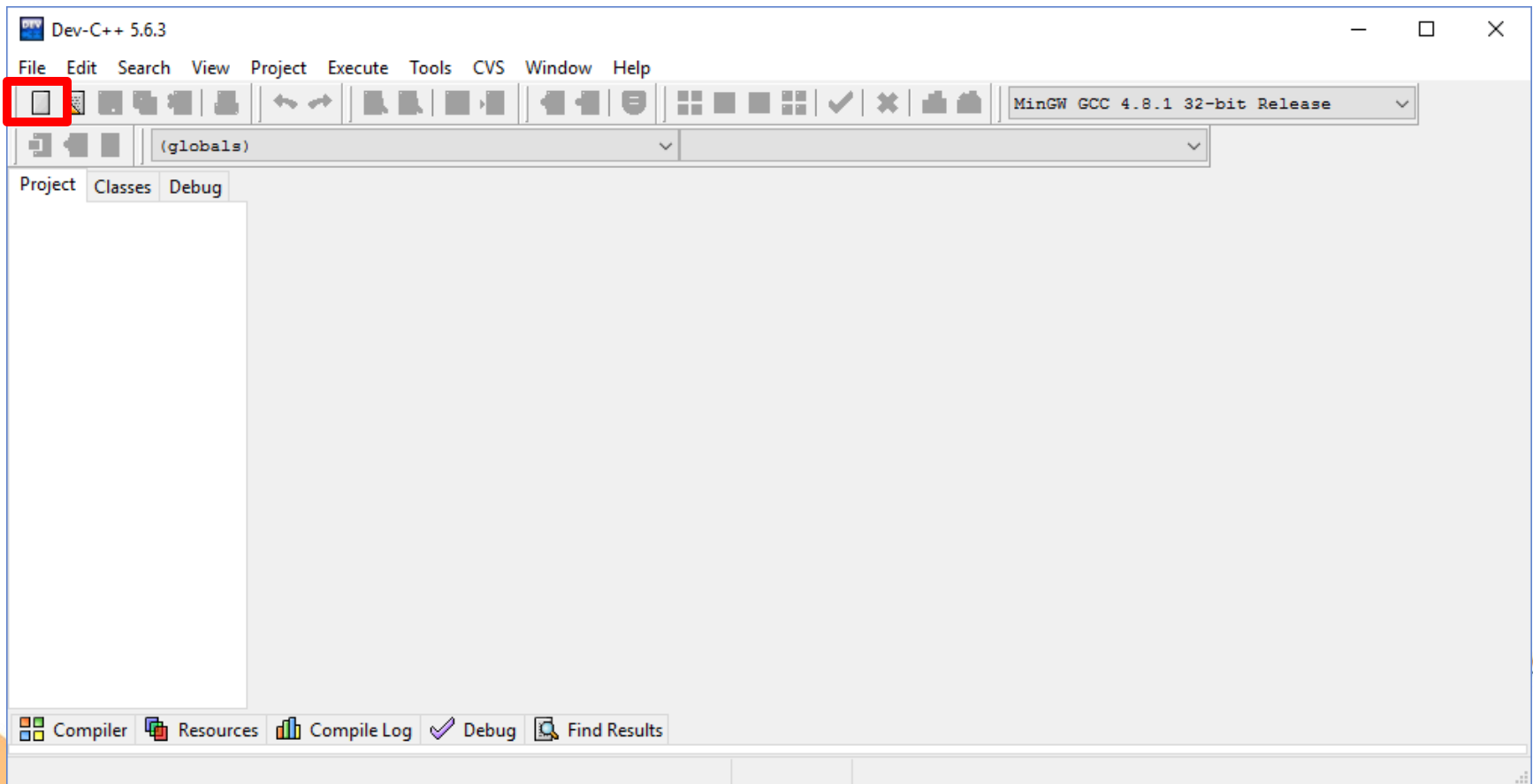
## LINGUAGEM C – AMBIENTE



# Introdução à Programação



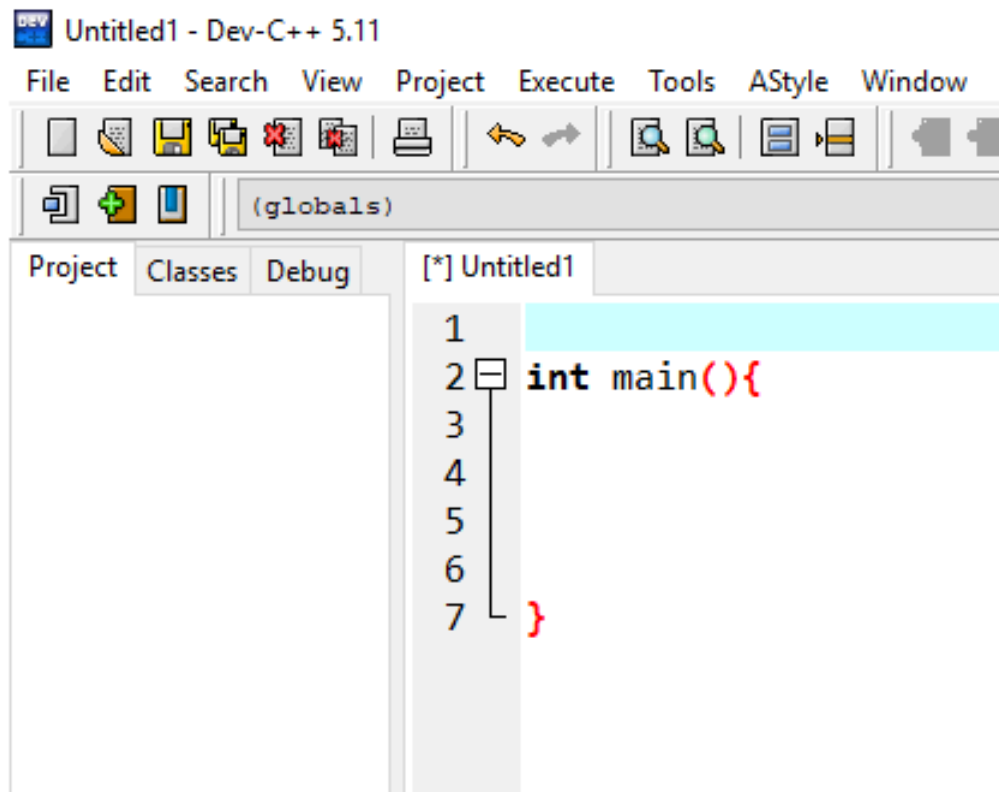
## LINGUAGEM C – AMBIENTE



# Introdução à Programação



## LINGUAGEM C – ESTRUTURA BÁSICA

A screenshot of the Dev-C++ 5.11 IDE. The title bar reads "Untitled1 - Dev-C++ 5.11". The menu bar includes File, Edit, Search, View, Project, Execute, Tools, AStyle, and Window. Below the menu is a toolbar with icons for file operations and execution. A status bar at the bottom shows "(globals)". The main editor window, titled "[\*] Untitled1", displays a C program structure with line numbers 1 through 7. Line 1 is a blank line with a light blue background. Line 2 contains the code "int main(){" in black text. Line 3 is a blank line. Line 4 is a blank line. Line 5 is a blank line. Line 6 is a blank line. Line 7 contains a closing curly brace "}" in red text. A vertical line connects the opening brace on line 2 to the closing brace on line 7, indicating the scope of the main function. The "Project" tab is selected in the left sidebar, and the "Classes" and "Debug" tabs are also visible.

```
1  
2 int main(){  
3  
4  
5  
6  
7 }
```

# Introdução à Programação



## LINGUAGEM C – PALAVRAS RESERVADAS

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

- C entende tais palavras apenas em letras minúsculas.

# Introdução à Programação



## LINGUAGEM C – BIBLIOTECA

- Conjunto de funções para realizar tarefas específicas.
- Biblioteca padrão C - ANSI - funções básicas.
- As primeiras linhas do programa indicam as bibliotecas utilizadas.

`#include "minha_biblioteca.h" ou`  
`#include <minha_biblioteca.h>`

# CONCEITOS BÁSICOS ALGORITMOS E PROGRAMAÇÃO

# Introdução à Programação



## INSTRUÇÕES

Considerando que um algoritmo é uma sequência de passos para a resolução de problemas, todo algoritmo ou programa apresentará instruções;

Elas representam os comandos a serem executados pelo computador;

Aprenderemos diversos comandos utilizando a linguagem de programação C;

Em C, ao final das instruções colocamos o ‘;’.

# Introdução à Programação



## Dados

De um modo geral, um programa manipula dados o tempo todo e ao utilizar uma linguagem de programação para programar é preciso considerar isso.

Algoritmos e linguagens de programação diferenciam os tipos dos dados manipulados em um programa.



# Introdução à Programação



## Tipos de dados

Os computadores trabalham com sistema numérico binário, em que os dados são transformados em 0 e 1 para serem armazenados na memória.

Todos os caracteres possuem um correspondente numérico que é transformado em binário.

De acordo com o tipo de dado a ser armazenado, o espaço de memória reservado será maior ou menor.

# Introdução à Programação



## Tipos de dados

Nesse contexto, as informações manipuladas são divididas em quatro tipos primitivos:

- Inteiro
- Real
- Caracter
- Cadeia
- Lógico

# Introdução à Programação



## Tipos de dados – INTEIROS

Os números inteiros podem ser positivos ou negativos e não possuem parte fracionária.

Exemplos:

-23

98

0

-2

Em C, o tipo inteiro é representado pelo termo *int*.

# Introdução à Programação



## Tipos de dados – REAIS

Os números reais podem ser positivos ou negativos e possuem parte fracionária (separada por “.”).

Exemplos:

23.45

346.89

-34.88

Em C, o tipo real é representado pelos termos *float* e *double*.

# Introdução à Programação



## TIPOS DE DADOS

### Numéricos:

Inteiros: int

- longos: long
- curtos: short
- não-negativos: unsigned (int, long, short, char)

Ponto Flutuante:

- simples: float (6 dígitos de precisão)
- duplo: double (12 dígitos de precisão)

# Introdução à Programação



## Tipos de dados – CHARACTER

São dados formados por um único caractere. Tais caracteres podem ser letras, números e os caracteres especiais (&, #, @, ?, +, ...).

Exemplos:

'a'

'1'

'+'

'\*'

Em C, o tipo caracter é representado pelo termo *char*

# Introdução à Programação



## Tipos de dados – CADEIA

São dados formados por uma cadeia de caracteres. Tais caracteres podem ser letras, números e os caracteres especiais (&, #, @, ?, +, ...).

Exemplos:

“aluno”

“1234”

“1 + 2”

Em C, o tipo cadeia é representado como um tipo composto e será visto posteriormente.

# Introdução à Programação



## Tipos de dados – LÓGICO

São também denominados dados booleanos e podem assumir os valores verdadeiro ou falso.

Em C, o tipo lógico não tem um termo específico, mas é representado pelos valores 0 (falso) e 1 (verdadeiro).



# Introdução à Programação



## DENIFINDO VARIÁVEIS...

Na construção de qualquer algoritmo, a informação é essencial, pois é o dado de entrada que será processado pelo computador para gerar um resultado posteriormente.

Esses dados precisam ser armazenados no computador para serem utilizados no processamento.

Esse armazenamento é feito na memória.

# Introdução à Programação



## DENIFINDO VARIÁVEIS...

Uma variável representa uma posição de memória. São espaços de memória para armazenar valores que podem ser alterados durante a execução de um programa;

- Representam dados do programa;
- Possuem nome e valor;
- Cada variável armazena um tipo de dado específico;
- São representadas por identificadores;

# Introdução à Programação



## DENIFINDO VARIÁVEIS...

As variáveis podem ter seu conteúdo modificado durante a execução do programa/algoritmo. No entanto, uma variável de um tipo primitivo só armazena um valor a cada instante.

### Sintaxe:

tipo <identificador>;

– Exemplo:

```
int idade;
```

### Memória

idade	-	206
	-	202

# Introdução à Programação



## CONSTANTES

Em algumas situações pode ser necessário que um determinado parâmetro não tenha seu valor alterado durante a execução do programa.

Para isso, existem as constantes.

Constante é um espaço de memória com dado que durante a execução de um algoritmo permanece com os seus valores inalterados;

# Introdução à Programação



## CONSTANTES

Para declarar uma constante basta adicionar a palavra reservada `*const*` seguida do tipo de dado, pelo nome da constante e atribuir um valor a ela.

Sintaxe:

```
const tipo NOME_DA_CONSTANTE = <valor>
```

Exemplo:

```
const int maximo = 10;
```

# Introdução à Programação



## FORMAÇÃO DE IDENTIFICADORES

Identificadores representam nomes das variáveis, dos algoritmos, das constantes, ...

Regras básicas:

- Caracteres permitidos: números, letras maiúsculas/minúsculas e o caractere “\_”;
- Primeiro caractere: letra ou sublinhado;
- Não são permitidos espaços em branco e caracteres especiais;
- É recomendável que os nomes sejam os mais significativos possíveis.

# Introdução à Programação



## FORMAÇÃO DE IDENTIFICADORES

Exemplos:

- Salario
- Tel1
- Tel2
- nome

# Introdução à Programação



## DECLARAÇÃO – VARIÁVEIS E CONSTANTES

Utilizada para indicar a existência de uma variável/constante e o seu tipo em um algoritmo. Associa-se o identificador escolhido com a respectiva posição de memória que ele passa a representar.

Uma vez declarada (variável ou constante), qualquer referência ao seu identificador implica na referência ao conteúdo armazenado no local da memória que ele representa.





# Introdução à Programação



## DECLARAÇÃO – VARIÁVEIS E CONSTANTES

Exemplos:

```
const float PI = 3.14
```

```
int idade;
```

```
char sexo;
```

```
float nota1;
```

```
float nota2;
```

# Introdução à Programação



## COMANDO DE ATRIBUIÇÃO

Um comando é a descrição de uma ação a ser executada em um dado momento.

O comando de atribuição permite fornecer um valor a uma certa variável, onde este valor deve ter o mesmo tipo de dado da variável em questão.

### **Forma geral:**

identificador = expressão OU valor;

# Introdução à Programação



## COMANDO DE ATRIBUIÇÃO

### Exemplos

- `A = 2;`
- `SEXO = 'F';`
- `MEDIA = SOMA / 4;`

# Introdução à Programação



## COMANDO DE ATRIBUIÇÃO

Exemplo [Phyton Tutor](#)

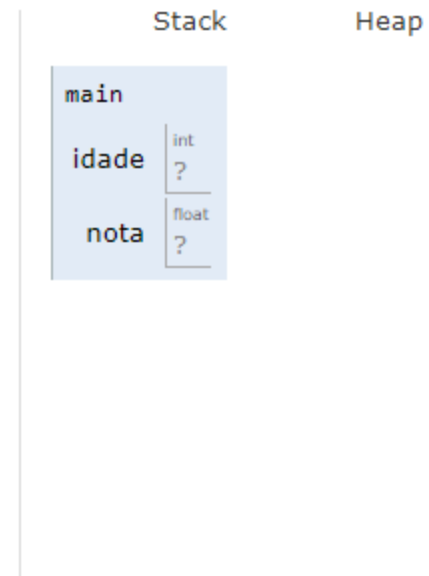
C (gcc 4.8, C11)  
**EXPERIMENTAL!** [known bugs/limitations](#)

```
1 int main() {  
2     int idade;  
3     float nota;  
→ 4     idade= 4;  
5     nota=3.4;  
6     idade = 8;  
7     return 0;  
8 }
```

[Edit this code](#)

→ line that has just executed

→ next line to execute



# DÚVIDAS???