

Fundação Hermínio Ometto

Bacharelado em Sistemas de Informação

SIF009 - Linguagem de Programação I

Prof. Dr. Sérgio Luis Antonello

Aula 02

11/08/2020

Sumário da aula

Primeiro momento (revisão)

- ✓ Linguagem de programação estruturada
- ✓ Compilação de programas

Segundo momento (conteúdo)

- ✓ Dados
- ✓ Operadores e precedência
- ✓ Comandos de I/O
- ✓ Depuração de código

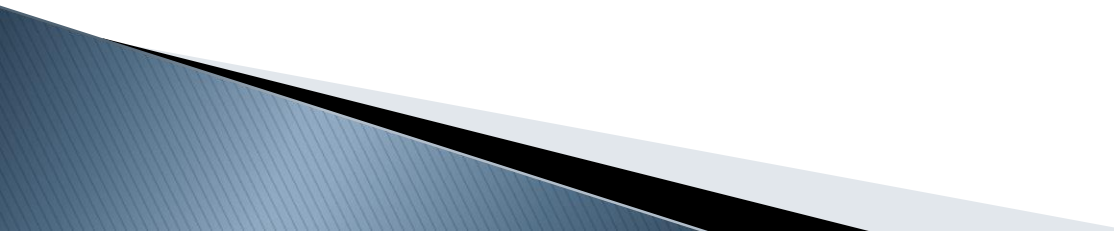
Terceiro momento (síntese)

- ✓ Retome pontos importantes da aula

1. Primeiro momento: revisão

A programação estruturada possibilita códigos modulares, organizados de forma hierárquica, associadas às estruturas básicas de controle, que são: sequência, decisão e repetição.

1. Primeiro momento: revisão

- ▶ Qual o papel do compilador?
 - ▶ Qual o papel do linkeditor?
 - ▶ Porque a linguagem C é considerada case sensitive?
 - ▶ O que são palavras reservadas?
- 

1. Primeiro momento: revisão

Um compilador gera o código objeto a partir de um código fonte. O processo de compilação realiza as seguintes tipos de análises:

- ▶ Léxica: separa cada símbolo que tenha algum significado para a linguagem.
- ▶ Sintática: série de regras que descrevem quais são as construções válidas da linguagem.
- ▶ Semântica: verifica incoerências quanto ao significado das construções utilizadas pelo programador (variáveis declaradas, funções, etc).

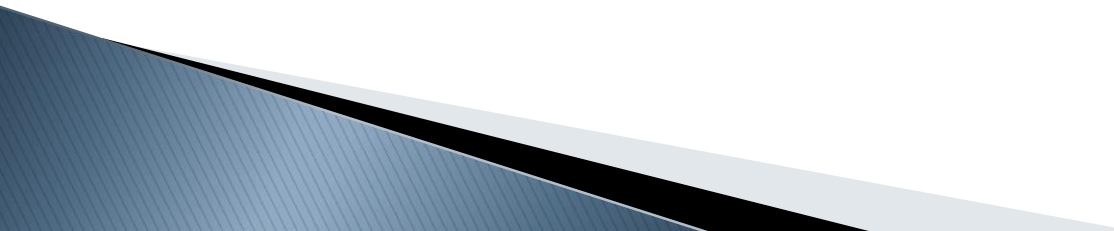
O linker (linkeditor) gera o código executável a partir de um ou vários códigos objetos.



1. Primeiro momento: revisão

Correção de exercícios.

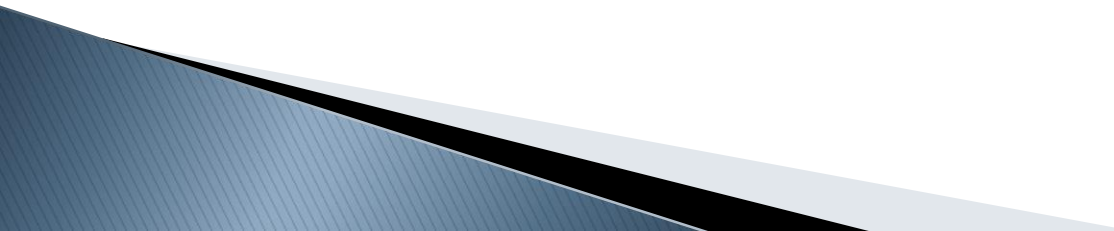
2. Segundo momento

- ▶ Dados
 - ▶ Operadores e precedência
 - ▶ Comandos de I/O
 - ▶ Depuração de código
 - ▶ Exercícios de aplicação
- 

3. Tipo de dados

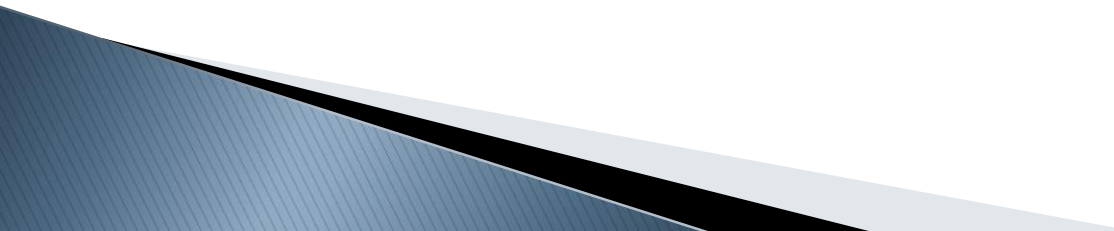


3. Tipo de dados

- ▶ Variáveis e Constantes são os elementos básicos que um programa manipula.
 - ▶ Uma variável é um espaço reservado na memória para armazenar determinado **tipo de dado**.
 - ▶ Tipos de Dados podem ser, por exemplo: inteiros, reais, caracteres, etc.
- 

3. Tipo de dados

Os tipos de dados básicos em linguagem C são:

- ▶ **char**: para armazenar um caractere.
 - ▶ **int**: número inteiro.
 - ▶ **float**: número em ponto flutuante de precisão simples (números reais).
 - ▶ **double**: número em ponto flutuante de precisão dupla.
- 

3. Tipo de dados

É adequado que um determinado produto pode ser armazenado e transportado por qualquer tipo de embalagem?

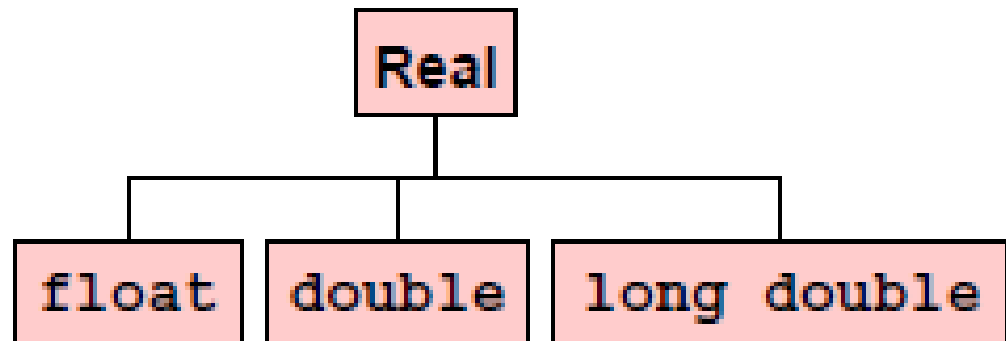
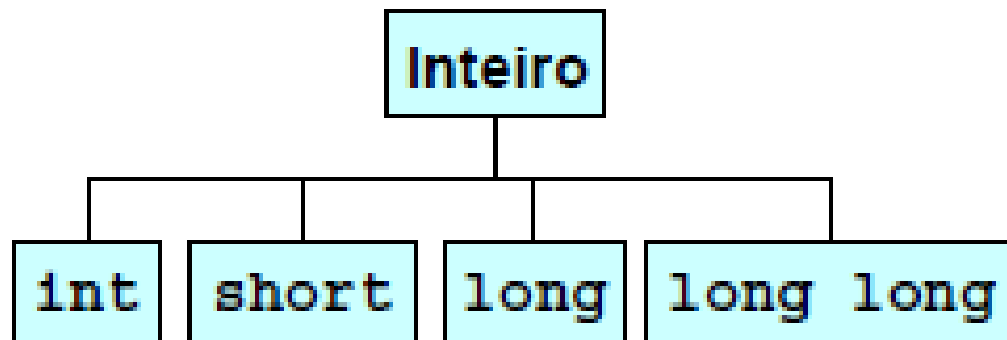


3. Tipo de dados

Tipo	Tamanho em Bytes	Faixa Mínima
char	1	-127 a 127
unsigned char	1	0 a 255
signed char	1	-127 a 127
int	4	-2.147.483.648 a 2.147.483.647
unsigned int	4	0 a 4.294.967.295
signed int	4	-2.147.483.648 a 2.147.483.647
short int	2	-32.768 a 32.767
unsigned short int	2	0 a 65.535
signed short int	2	-32.768 a 32.767
long int	4	-2.147.483.648 a 2.147.483.647
signed long int	4	-2.147.483.648 a 2.147.483.647
unsigned long int	4	0 a 4.294.967.295
float	4	Seis dígitos de precisão
double	8	Dez dígitos de precisão
long double	10	Dez dígitos de precisão

4. Modificadores de tipo

Exemplo para os tipos inteiro e real.



5. Variáveis e Constantes

Uma variável é um espaço reservado na memória para armazenar determinado tipo de dado.

- `int A;`
- `float B;`

Escopo de variáveis

- Local
 - Global
- 

5. Variáveis e Constantes

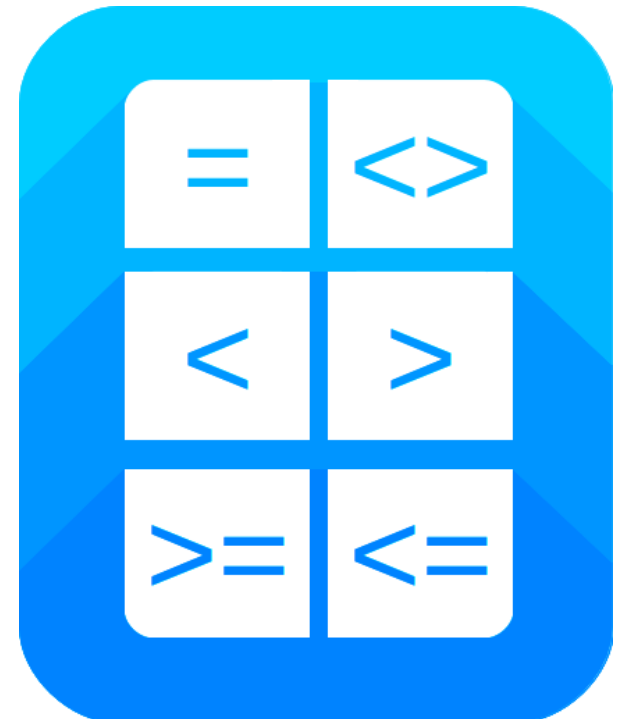
Constantes são usadas em expressões para representar vários tipos de valores.

- Constante inteira : 100, -3
- Constante em ponto Flutuante: 123.45
- Constante caractere: 'a'
- Constante cadeia de caracteres: 'olá mundo'

```
const int num = 10;
```



6. Operadores



6. Operadores aritméticos

Operador	Ação
-	Subtração, também menos unário
+	Adição
*	Multiplicação
/	Divisão
%	Módulo da divisão (resto)
--	Decremento
++	Incremento

6. Operadores aritméticos

A linguagem C possui operadores unário, binário e ternário:

- ▶ **Unário**: agem sobre uma variável (operando). Exemplo o operador “-”
- ▶ **Binário**: usam dois operandos (variáveis). Exemplo: os operadores adição e subtração.
- ▶ **Ternário**: usam três operandos para substituir certas sentenças de forma if-then-else. Exemplo: o operador “?”

6. Operadores aritméticos

- ▶ O operador “/” (divisão) quando aplicado a variáveis inteiras ou caractere, nos fornece o resultado da divisão inteira, ou seja, o resto é truncado.

```
int x = 5, y = 2;  
printf("%d", x/y);
```

Mostra na tela o número 2



- ▶ O operador / (divisão) quando aplicado a variáveis em ponto flutuante nos fornece o resultado da divisão “real”.

```
float x = 5, y = 2;  
printf("%f", x/y);
```

Mostra na tela o número 2.500000




6. Operadores aritméticos

- ▶ O operador % (módulo) quando aplicado a variáveis inteiras ou caractere, nos fornece o resto de uma divisão inteira.

```
int x = 5, y = 2;  
printf("%d", x%y);
```

Imprime na tela o resto da divisão, portanto, o número 1



- ▶ Atenção: O operador % (módulo) não pode ser usado nos tipos em ponto flutuante.

6. Operadores aritméticos

- Observe atentamente o funcionamento dos operadores de incremento e decremento quando usados em uma expressão com outros operadores.

```
x = 10;
```

```
y = ++x;
```

Executa o incremento antes de usar o valor do operando para atualizar y. Resultado: X = 11 e Y = 11

```
x = 10;
```

```
y = x++;
```

Usa o valor do operando para atualizar y antes de incrementar x. Resultado: X = 11 e Y = 10

6. Operadores aritméticos

- Exemplo de uso dos operadores.

```
1 ☐ int main(){  
2  
3   int A, B, X;  
4   A = B = 5;  
5   X = A++;  
6   A = -A;  
7   B++;  
8   X = ++B;  
9 }
```

Resultado na memória RAM da execução de cada linha de comandos.

A	B	X
5	5	
6	5	5
-6	5	5
-6	6	5
-6	7	7

6. Operadores aritméticos

Exemplo de uso dos operadores

$y = x > 9 ? 100 : 200;$

- ▶ y recebe o valor 100 se a expressão $x > 9$ for verdadeira.
- ▶ y recebe o valor 200 se a expressão for falsa.

7. Operadores relacionais

Operador	Ação
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a
==	Igual a
!=	Diferente de

7. Operadores relacionais

Uma expressão com operadores relacionais retornam verdadeiro (1) ou falso (0).

Em linguagem C, verdadeiro é qualquer valor diferente de zero.

```
int a = 10, b = 1, c = 12;  
a > b + c;
```

O resultado dessa expressão é **falso**.

7. Operadores lógicos

Operador	Ação
&&	And (E)
 	Or (Ou)
!	Not (Não)

8. Operadores bit a bit

Operador	Ação
&	And
	Or
^	Or exclusivo (Xor)
~	Complemento de um
>>	Deslocamento à esquerda
<<	Deslocamento à direita

8. Operadores bit a bit

- Operação bit a bit refere-se a testar, atribuir ou deslocar os bits efetivos em um byte ou uma palavra.
- Operações bit **não** podem ser usadas em float, double, long double, void ou outros tipos mais complexos.
- Operadores bit a bit encontram aplicações mais frequentes em “drivers” de dispositivos – como em programas de modem, rotinas de arquivos em disco e impressoras.

8. Operadores bit a bit

Um exemplo de uso

- Os operadores de deslocamento “>>” e “<<” movem todos os bits de uma variável para a direita ou para a esquerda, respectivamente.
 - Deslocamento à direita:
 - variável >> número de posições de bits
 - Deslocamento à esquerda:
 - variável << número de posições de bits
- Conforme os bits são deslocados para uma extremidade, zeros são colocados na outra.

8. Operadores bit a bit

Um exemplo de uso.

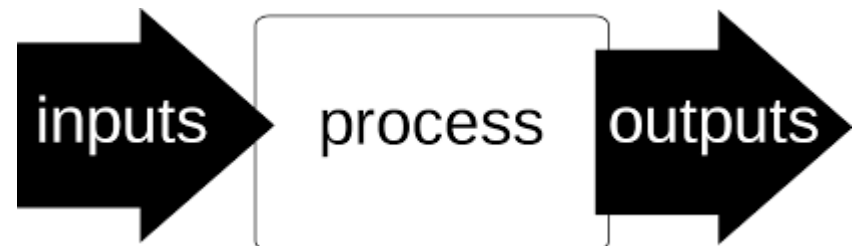
```
1 int main(){
2
3     int X;
4     X = 7;
5     printf ("\nX=%d", X);
6
7     X = X<<1;
8     printf ("\nX<<1=%d", X);
9
10    X = X<<3;
11    printf ("\nX<<3=%d", X);
12
13    X = X<<2;
14    printf ("\nX<<2=%d", X);
15
16    X = X>>1;
17    printf ("\nX<<1=%d", X);
18
19    X = X>>2;
20    printf ("\nX<<2=%d", X);
21 }
```

X a cada execução	Valor de x
00000111	7
00001110	14
01110000	112
11000000	192
01100000	96
00011000	24

9. Precedência de Operadores

Operador	Tipo de operação
postfix ++ e postfix --	
prefix ++ e prefix --	
* / %	Mult. Divi. Resto div.
+ -	Adição e Subtração
== <> < > <= >=	Relacional
!	Negação
&&	Lógico E
	Lógico OU

10. Comandos de I/O



10. Comandos de I/O

A princípio serão vistos dois comandos para entrada e saída de dados.

- **scanf()**: para “input” (entrada) de dados, embora existam outros.
- **printf()**: para “output” (saída) de dados, embora existam outros.

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

```
int main()
{
    int num;
    printf("Digite um número: ");
    scanf("%d", &num);
    printf("%d", num);
    return 0;
}
```

10. Comandos de I/O

Tanto na entrada de dados usando `scanf()` quanto na saída de dados usando a função `printf()`, deve-se especificar qual o formato do dado.

Exemplos:

```
scanf("%f", &salario);
```

```
scanf("%d", &codigo);
```

```
scanf("%s", nome);
```

```
printf("%f", salario);
```

```
printf("%d", codigo);
```

```
printf("%s", nome);
```

Código	Formato
<code>%c</code>	Caractere simples (<code>char</code>)
<code>%d</code>	Decimal (<code>int</code>)
<code>%e</code>	Notação científica
<code>%f</code>	Ponto flutuante (<code>float</code>)
<code>%g</code>	<code>%e</code> ou <code>%f</code> (o mais curto)
<code>%o</code>	Octal
<code>%s</code>	Cadeia de caracteres
<code>%u</code>	Decimal sem sinal
<code>%x</code>	Hexadecimal
<code>%ld</code>	Decimal longo
<code>%lf</code>	Ponto flutuante longo (<code>double</code>)
<code>%p</code>	Ponteiro

11. Depuração de código



11. Depuração de código

“O hífen mais caro da história”

Em 22 de julho de 1962, o Mariner I estava em sua plataforma, pronto para fazer história. Depois do investimento de anos de construção, cálculos e financiamento, a NASA tinha grandes esperanças de que seu foguete realizaria com sucesso um voo de reconhecimento até Vênus, o que daria impulso para a corrida espacial americana. Em todos os sentidos, a NASA estava prestes a estabelecer um precedente em viagens espaciais.

Mas assim que o foguete foi lançado, ficou claro que não havia o que comemorar: menos de cinco minutos depois do início de voo, o Mariner I explodiu, dando um prejuízo de US\$80 milhões (hoje seriam US\$630 milhões). O que causou este desastre? Um simples hífen que foi omitido num código matemático escrito à mão.

11. Depuração de código

“Mars Climate Orbiter”

Em 1999, a Mars Climate Orbiter, que custou US\$125 milhões (seriam US\$172 milhões hoje), voou para fora da rota e se desintegrou pois os engenheiros espaciais se esqueceram de converter as medidas do sistema imperial para unidades métricas.

Enquanto a equipe do Jet Propulsion Laboratory estava usando o sistema métrico (milímetros e metros) em seus cálculos, a Lockheed Martin Astronautics, que construiu a nave, estava usando medições em polegadas, pés e libras.

11. Depuração de código

A Mariner I e a Mars Climate Orbiter vão servir sempre como um lembrete para todos, desde os humildes editores de blogs até os engenheiros da NASA, da importância universal da revisão.

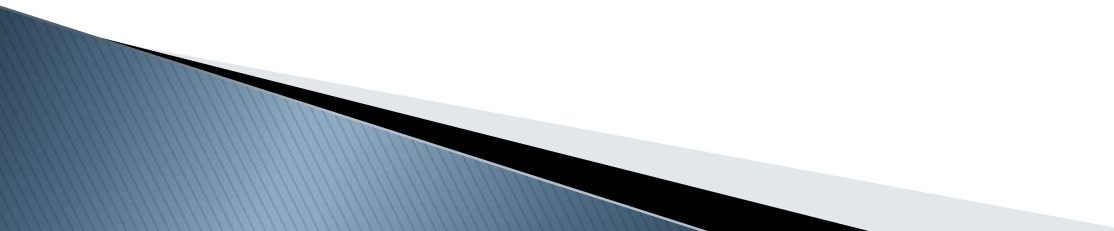
Nesse meio tempo, vamos nos consolar com o fato de que nossos erros de digitação podem não acabar custando US\$80 milhões.

Fonte: <http://gizmodo.uol.com.br/hifen-nasa>.



11. Depuração de código

Devemos depurar um código para solucionar possíveis problemas:

- **Erros de Sintaxe** - comandos escritos de maneira incorreta, falta de “;” ao término de um comando, abertura e fechamento de bloco de códigos, dentre outros.
 - **Erros de lógica** - precedência dos operadores, cálculos incorretos, dentre outros.
- 

12. Codificação e Depuração de código em Linguagem C

Para programar em linguagem C precisamos, basicamente, de um **editor de texto** qualquer e um **compilador C**:

- O editor de textos você pode usar o de sua preferência: Bloco de Notas, Notepad++, Sublime Text, dentre outros.
- O compilador que será usado nesta disciplina é o **GCC**, porém existem outros compiladores no mercado.
- Outra opção é usar uma IDE, que trata-se de um ambiente para desenvolvimento de softwares. Nesta disciplina será usado o **Dev C++**.

13. IDE DevC++

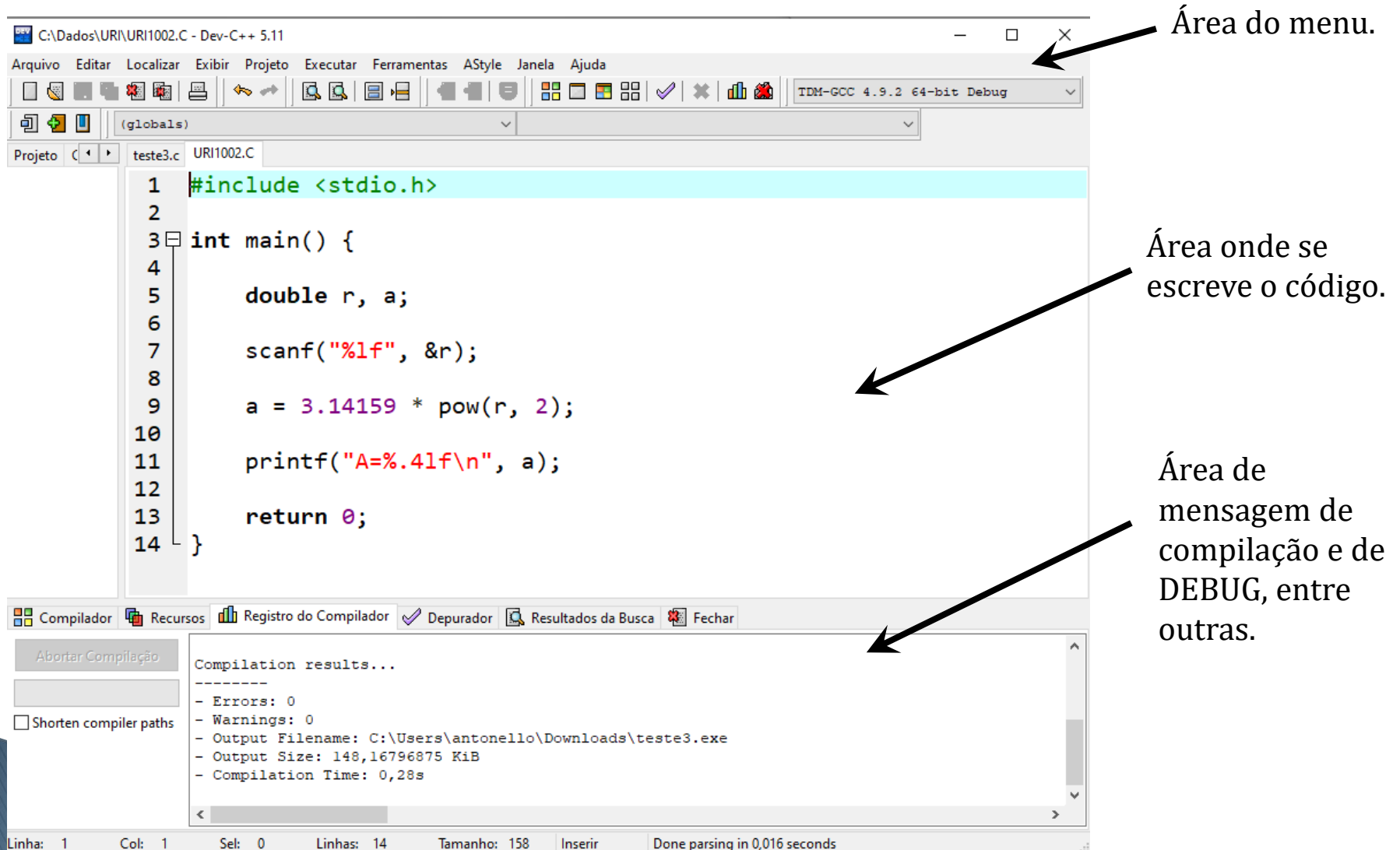
Um Integrated Development Environment (IDE) ou Ambiente de Desenvolvimento Integrado, trata-se de software que reúne ferramentas específicas para o apoio aos desenvolvedores de softwares, buscando facilitar e agilizar o processo de desenvolvimento.

Dev-C++ 5.0 beta 9.2 (4.9.9.2) (9.0 MB) with Mingw/GCC 3.4.2

<https://www.bloodshed.net/dev/devcpp.html>



13. IDE DevC++



14. Exercícios

**Vamos
Programar!**



14. Exercícios

1) Usando o Dev C++, escreva e compile o código C abaixo, observando:

- ▶ O código apresenta algum tipo de erro na compilação?
- ▶ Quais os tipos de erros (Sintaxe e/ou Lógica)?
- ▶ O que deve ser feito para o código executar normalmente? (Liste todos os erros e as soluções – indique as linhas)

```
1  #include <stdio.h>
2
3  int main(){
4      int num;
5      float x, total;
6
7      num = 2
8      x = 135.4;
9
10     total = num * x;
11
12     printf("Multiplicacao = %.2f\n", total)
13
14     return 0;
15 }
```

14. Exercícios

2) Repetindo o exercício anterior, usando o Dev C++, escreva e compile o código C abaixo, observando:

- ▶ O código apresenta algum tipo de erro na compilação?
- ▶ Quais os tipos de erros (Sintaxe e/ou Lógica)?
- ▶ O que deve ser feito para o código executar normalmente? (Liste todos os erros e as soluções – indique as linhas)

```
1  #include <stdio.h>
2
3  int main(){
4      int numero, x;
5
6      scanf("%d", numero);
7
8      scanf("%d", &x);
9
10     printf("Soma = %d\n", numero + x)
11
12     return 0;
13 }
```

14. Exercícios

3) Na mesma linha dos anteriores, trabalhe no código abaixo que deve calcular a média das notas de um aluno, que fez duas provas durante o semestre.

- ▶ O código apresenta algum tipo de erro na compilação?
- ▶ Quais os tipos de erros (Sintaxe e/ou Lógica)?
- ▶ O que deve ser feito para o código executar normalmente? (Liste todos os erros e as soluções – indique as linhas)

```
#include <stdio.h>

int main(){
    float nota1, nota2, media;

    printf("Digite a nota 1: ");
    scanf("%f", &nota1);
    printf("Digite a nota 2: ");
    scanf("%f", &nota2);

    media = (nota1 + nota2) / 5;

    printf("Resultado da media: %.1f", media);

    return 0;
}
```

14. Exercícios

4) URI 1014 Consumo

<https://www.urionlinejudge.com.br/judge/pt/problems/view/1014>

5) URI 1008 Salário

<https://www.urionlinejudge.com.br/judge/pt/problems/view/1008>



15. Terceiro momento: síntese

- Os dados são usados no processamento definido para a solução de um problema.
- Na linguagem C, deve-se observar o tipo de dado específico para cada conteúdo de entrada, processamento ou saída.
- O processamento usa expressões formadas por variáveis e constantes associadas aos operadores.
- Os operadores mais comuns são os aritméticos, relacionais e lógicos. Observe que existe uma precedência de execução entre eles.
- A depuração de código (debug) é importante para avaliar o funcionamento do código e encontrar erros, principalmente os erros de lógica.