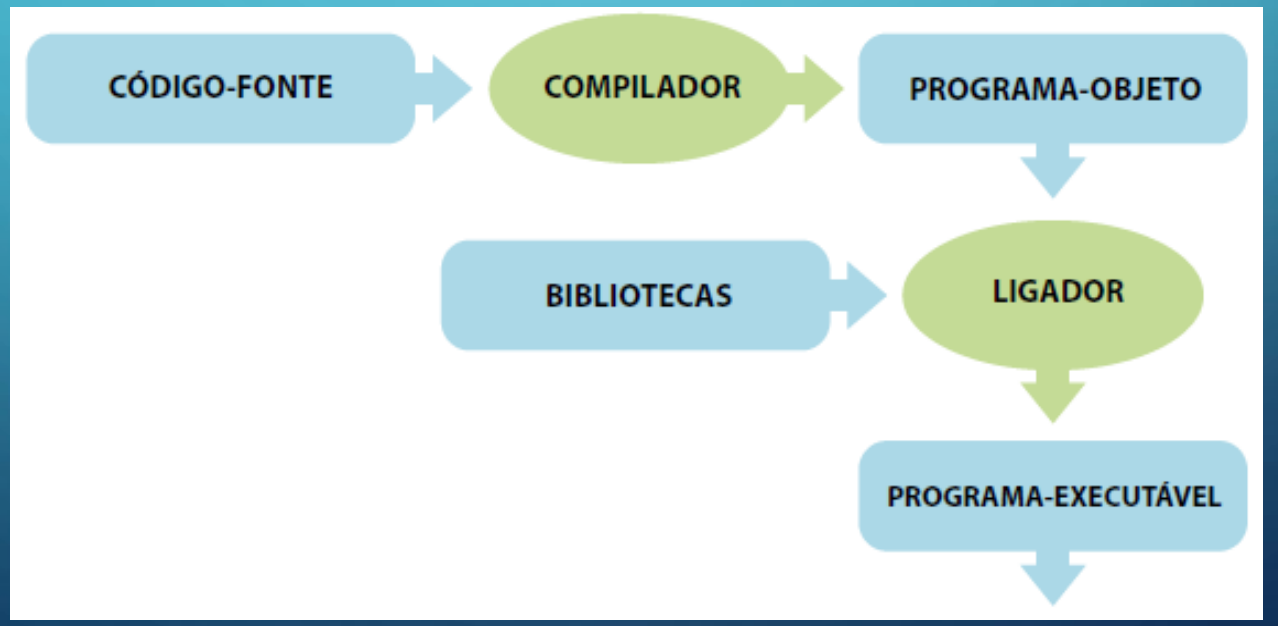
A LINGUAGEM C E OS **CONCEITOS INICIAIS** DE PROGRAMAÇÃO

Prof. Dr. Robyson Aggio
https://www.aggioirati.com.br

A LINGUAGEM **C** FOI CONCEBIDA E IMPLEMENTADA, INICIALMENTE, PARA O SISTEMA OPERACIONAL UNIX, NA DÉCADA DE 70, POR DENNIS RITCHIE, NOS LABORATÓRIOS BELL (KERNIGHAN; RITCHIE, 1988).

ALÉM DISSO, NÃO ESTÁ VINCULADA A UM HARDWARE ESPECÍFICO OU A QUALQUER OUTRO SISTEMA, DE MODO QUE É FÁCIL ESCREVER PROGRAMAS QUE SERÃO EXECUTADOS SEM MUDANÇAS EM QUALQUER MÁQUINA QUE SUPORTA C (KERNIGHAN; RITCHIE, 1988).

O processo de compilação analisa o código fonte, do ponto de vista sintático, e o converte para um código-objeto, que é a versão em linguagem de máquina do programa. Se o programa possui chamada às funções de bibliotecas, o *linker* (ligador) reúne o programa-objeto com as bibliotecas referenciadas e gera o programa-executável (arquivo binário) (ROCHA, 2006).



A ESTRUTURA DE UM PROGRAMA EM C

Kernighan e Ritchie (1988) destacam que a única maneira de aprender uma nova linguagem de programação é escrevendo programas nela. Abaixo temos um “esqueleto” da linguagem C.

```
01    <inclusão de bibliotecas>
02    int main()
03    {
04        <conjunto de instruções>
05         return (0);
06    }
```

```
01    #include <stdio.h>
02    int main()
03    {
04          printf("Olá mundo!");
05          return (0);
06    }
```

Na primeira linha, temos a instrução #include <stdio.h>, a qual indica ao compilador que, durante o processo de compilação e linkedição, deve-se incluir o conteúdo do arquivo predefinido (é um arquivo que já existe – uma biblioteca com funcionalidades pré-programadas), chamado stdio.h. Este arquivo chama-se arquivo de cabeçalho e contém declarações de funções para entrada e saída de dados.

Portanto, não podemos esquecer de inserir esta linha em nossos programas.

```
01    #include <stdio.h>
02    int main()
03    {
04          printf("Olá mundo!");
05          return (0);
06    }
```

Em seguida, temos a instrução main(), que identifica a função principal de qualquer programa escrito em linguagem C, denominada **main**. Os programas em C são formados por chamadas de função. Obrigatoriamente, todo programa deve possuir uma função main, a qual é a primeira a ser chamada quando o programa é executado.

```
01    #include <stdio.h>
02    int main()
03    {
04          printf("Olá mundo!");
05          return (0);
06    }
```

Dentro das chaves, temos duas instruções: a primeira delas é o printf, uma função previamente definida no arquivo de cabeçalho stdio.h. Neste caso, o printf é responsável por imprimir, na tela, a sequência de caracteres “Olá mundo!”.

```
01    #include <stdio.h>
02    int main()
03    {
04          printf("Olá mundo!");
05          return (0);
06    }
```

A última linha, return(0), indica o valor de retorno da função. No caso, 0, por convenção, indica que o programa terminou sem erros.

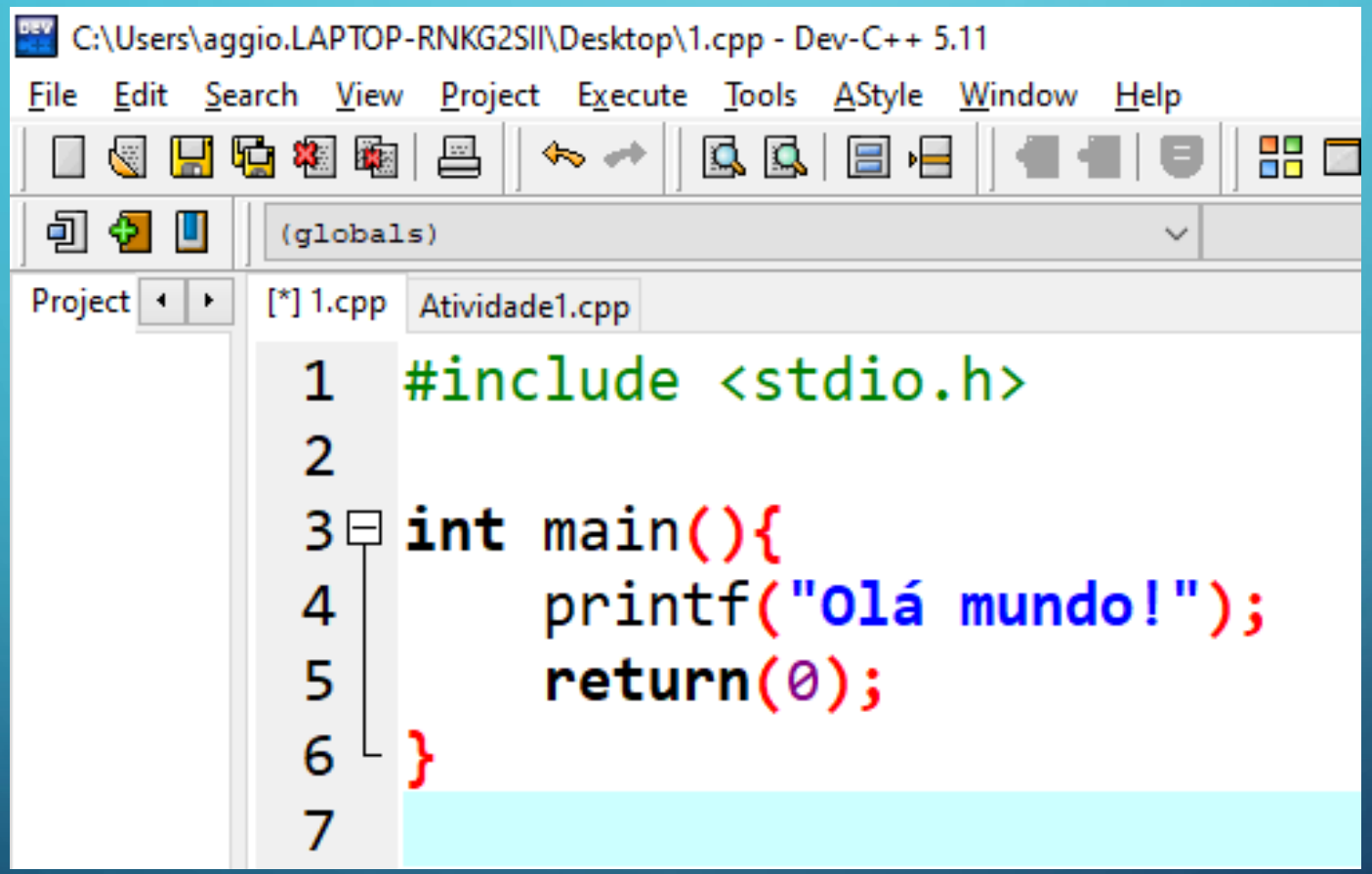
Observe que, ao final de cada instrução, há um ponto e vírgula, isto é, a grande maioria dos comandos em C terminam com “;”.

/* */ (barra asterisco e asterisco barra) é utilizado para escrever comentários em nosso código, isto é, o compilador desconsidera qualquer coisa que esteja entre estes dois pares de símbolos.

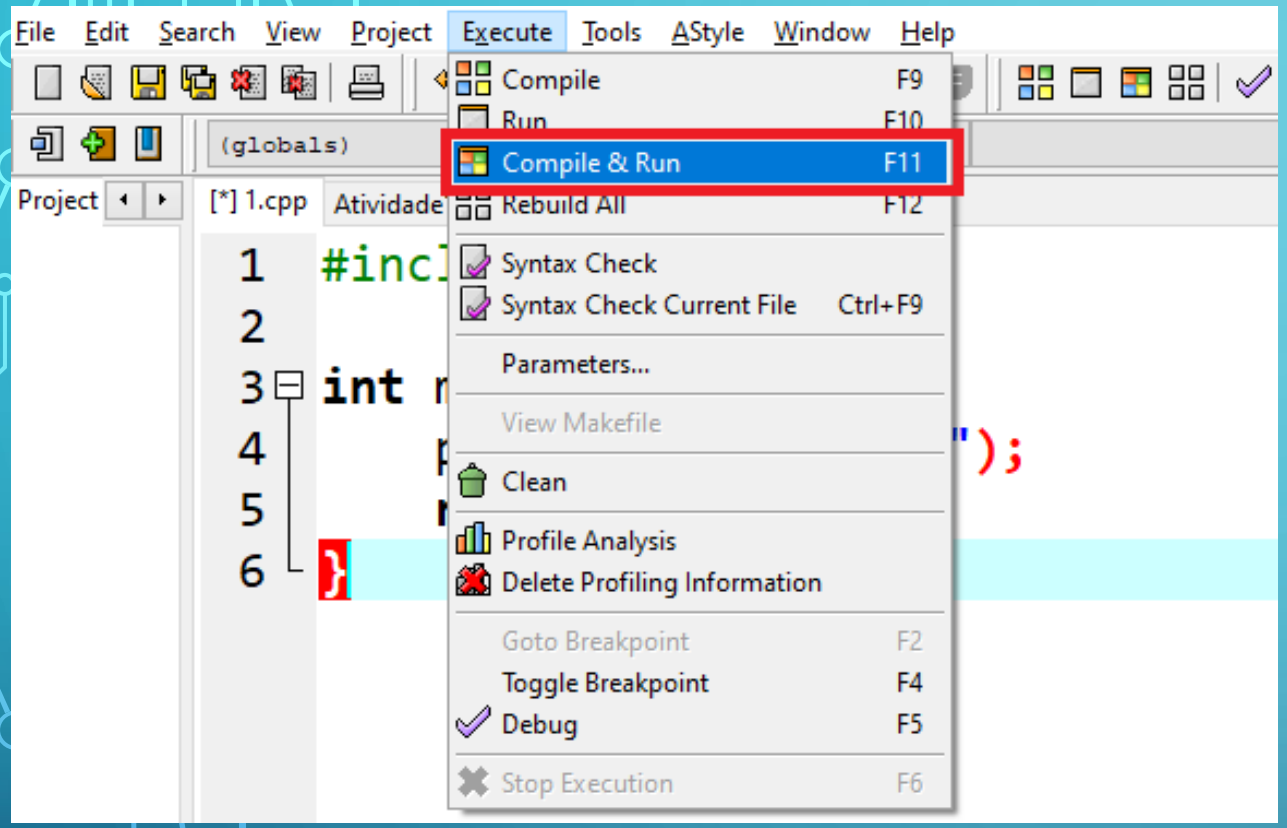
Obs: // (barra barra) também pode ser utilizado

```
1  /*Primeiro programa em C*/
2
3  #include <stdio.h> /*esta instrução insere o conteúdo do arquivo stdio.h*/
4  main() /*todo programa em C deve possuir essa linha*/
5  { /*delimita o inicio das instruçoes*/
6        printf("Hello, World"); /*esta instrução imprime uma mensagem na tela*/
7        return (0); /*este retorno indica que o programa está finalizando sem erro*/
8  } /*delimita o fim do conjunto de instruções da função main*/
9
```

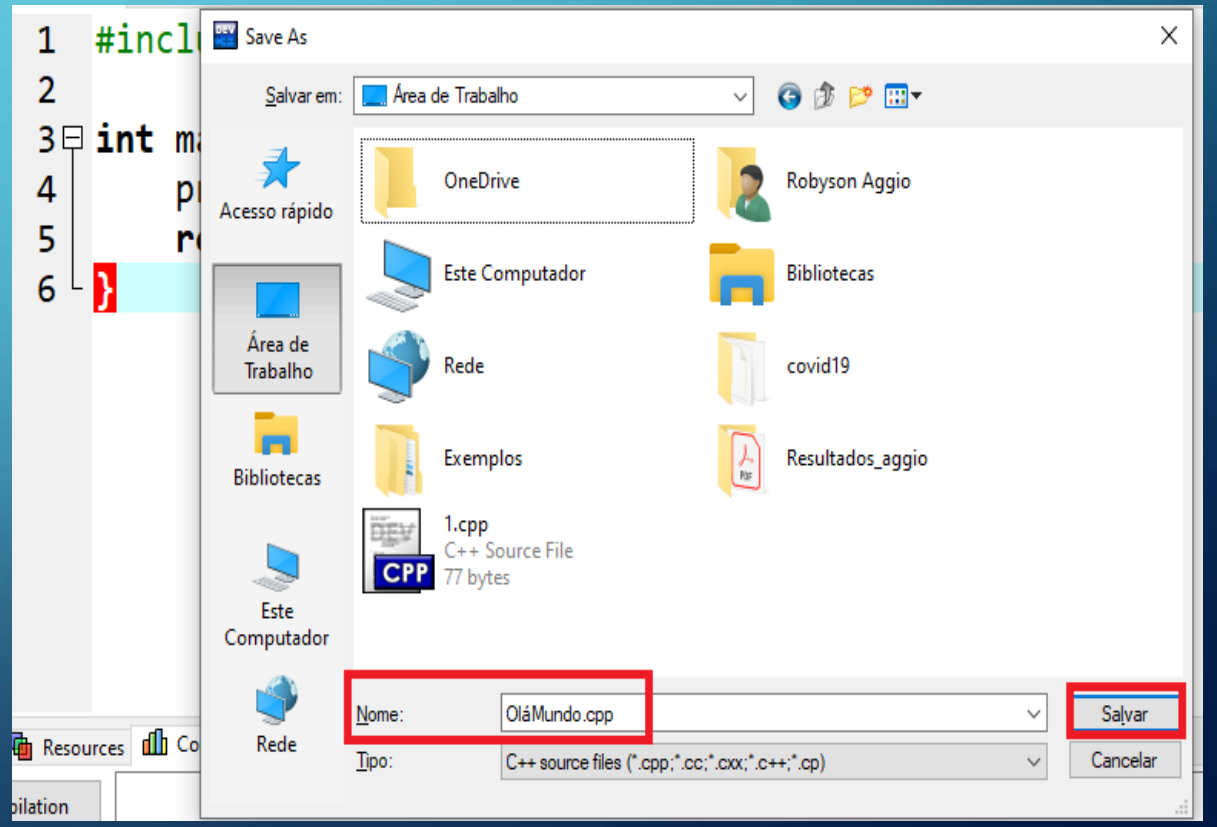
Escreveremos nossos programas utilizando o Dev-C++
Download: https://sourceforge.net/projects/orwelldevcpp/



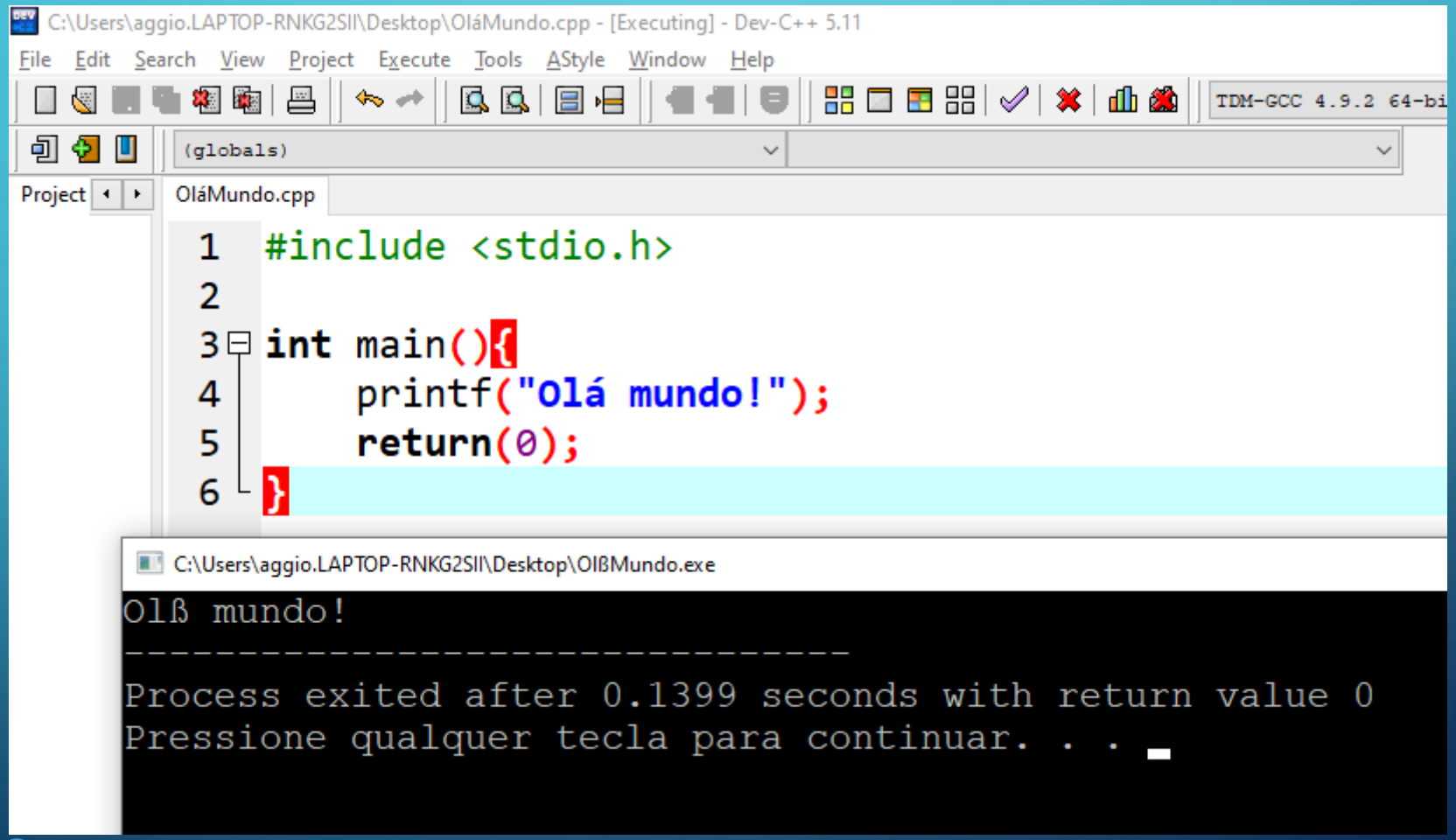
Para compilar e executar (F11)



Salve em alguma pasta e renomeie: OláMundo.cpp



Se tudo estiver correto, irá aparecer a tela (com fundo preto)
Obs: Verifique que a acentuação não foi formatada.



Para corrigir problemas com acentuação, devemos incluir:
Biblioteca → #include <locale.h>
Dentro do main() → setlocale(LC_ALL, “Portuguese”);

OláMundo.cpp

```
1  #include <stdio.h>
2  #include <locale.h>
3  int main(){
4      setlocale(LC_ALL, "Portuguese");
5      printf("Olá mundo!");
6      return(0);
7  }
```

C:\Users\aggio.LAPTOP-RNKG2SII\Desktop\OIßMundo.exe

```
Olá mundo!
--------------------------------
Process exited after 0.1438 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Arquivos de cabeçalho

Nome do Arquivo	Descrição
<stdio.h>	Tratamento de entrada/saída.
<locale.h>	Especifica constantes de acordo com a localização específica, como moeda, data, etc.
<math.h>	Funções matemáticas comuns em computação.
<stdlib.h>	Implementa funções para diversas operações, incluindo conversão, alocação de memória, controle de processo, funções de busca e ordenação.
<string.h>	Tratamento de strings.
<time.h>	Trata de tipos de data e hora.

OS IDENTIFICADORES, **OS TIPOS DE DADOS** e as Palavras Reservadas

A linguagem C é *case sensitive*, isto é, o compilador considera letras maiúsculas e minúsculas como caracteres distintos, diferenciando caixa alta de caixa baixa. Os comandos (palavras reservadas) em C só podem ser escritos em minúsculas, senão o compilador os interpretará como variáveis.

Identificadores válidos	Identificadores inválidos
A	2ª
a	b@
media	media idade
altura2	x*y
media_idade	#media
x36	idade!

Os tipos de dados:

Na linguagem C, as informações podem ser representadas por sete tipos básicos de dados: char, int, float, double, enum, void e *pointer* (ponteiro) (PAPPAS; MURRAY, 1991).

O tipo char é utilizado para representar caracteres simples e até *strings* (cadeia de caracteres). O tipo int são dados numéricos que não possuem componentes decimais ou fracionários. O tipo float, valores em ponto flutuante, são números que têm componente decimal ou fracionário. O tipo double são valores em ponto flutuante de precisão dupla, que apresentam alcance mais extenso. O tipo enum, dados enumerados, possibilitam os tipos definidos pelo usuário. O tipo void significa valores que ocupam 0 bits e não possuem valor algum, indicando uma espécie de "ausência de tipo". O *pointer*, apesar de não ser uma palavra reservada, representa um dado especial, que não contém uma informação propriamente dita, mas sim, uma localização de memória (endereço de memória) que, por sua vez, contém o dado verdadeiro.

As Palavras reservadas

Palavras reservadas			
asm	template	do	register
catch	this	double	return
clas	virtual	else	short
delete	_cs	enum	signed
_export	_ds	extern	sizeof
frient	_es	far	static
inline	_ss	float	struct
_loadds	auto	for	switch
new	break	goto	typedef
operator	case	huge	union
private	catch	if	unsigned
protected	cdecl	int	void
public	char	interrupt	volatile
_regparam	const	long	while
_saveregs	continue	near	
_seg	default	pascal	

AS VARIÁVEIS, **AS CONSTANTES,** as Expressões e os Operadores

Em nossos programas, precisamos armazenar algumas informações e, para isso, utilizamos as variáveis. Uma variável é um espaço na memória principal do computador que pode conter diferentes valores a cada instante de tempo (LOPES; GARCIA, 2002).

Na linguagem C, as variáveis são declaradas após a especificação de seus tipos, sendo os tipos mais utilizados: int, float e char. Note que, em C, não existe o tipo de dados boolean, pois considera verdadeiro qualquer valor diferente de 0. Além disso, não há um tipo especial para armazenar cadeia de caracteres (***strings***), sendo utilizado um vetor que contém vários elementos do tipo char para tal fim (ASCENCIO; CAMPOS, 2010).

A sintaxe para declaração de variáveis é dada por:
<tipo> <identificador>;

Declaração	
`int quantidade;`	Declara uma variável chamada quantidade, que pode armazenar um valor inteiro.
`float total;`	Declara uma variável chamada total, que pode armazenar um valor real.
`float valor, total;`	Declara duas variáveis denominadas valor e total, que podem armazenar valor real.
`char sexo;`	Declara uma variável denominada sexo, que pode armazenar um caractere.
`char endereco[30];`	Declara uma variável denominada endereço, que pode armazenar até 30 caracteres.

Uma constante armazena informações que não variam com o tempo, ou seja, o seu conteúdo é um valor fixo. Em C, podemos definir constantes por meio da seguinte sintaxe:

#define <identificador> <valor>

Observe que, na definição de constantes, **não** utilizamos o ; no final.

Operadores Aritméticos

Operação	Operador	Significado
Soma	+	Utilizado para efetuar a soma de duas ou mais variáveis. Dada uma variável A e outra B, temos que a soma delas é representada por A + B.
Subtração	-	Simboliza a subtração do valor de duas variáveis. Supondo uma variável A e B, a diferença entre elas é dada por: A – B.
Multiplicação	*	O produto entre duas variáveis A e B é representado por A * B.
Divisão	/	A divisão entre duas variáveis A e B é dada por: A/B. Em relação à divisão, é importante lembrar que não existe divisão por zero.
Resto de divisão	%	Usado quando se deseja encontrar o resto da divisão entre duas variáveis A e B. A representação é dada por A % B. Supondo A = 3 e B = 2, temos que A % B = 1, uma vez que 3 divididos por 2 resulta em 1, com resto igual a 1.

Em C, temos, também, os operadores aritméticos de atribuição, que são utilizados para representar, de maneira sintética, uma operação aritmética, seguida de uma operação de atribuição (ASCENCIO; CAMPOS, 2010).

Operador	Exemplo	Explicação
+=	x += y	Equivale a x = x + y.
-=	x -= y	Equivale a x = x - y.
*=	x *= y	Equivale a x = x * y.
/=	x /= y	Equivale a x = x / y.
%=	x %= y	Equivale a x = x % y.
++	x++	Equivale a x = x + 1.
	y = ++x	Equivale a x = x + 1 e, depois, y = x.
	y = x++	Equivale a y = x e, depois, x = x + 1.
--	x--	Equivale a x = x - 1.
	y = --x	Equivale a x = x - 1 e, depois, y = x.
	y = x--	Equivale a y = x e, depois, x = x - 1.

As expressões **relacionais** referem-se à comparação entre dois valores de um tipo básico. Tal comparação relacional, em linguagem C, pode resultar em verdadeiro ou falso. Os operadores relacionais são destacados a seguir, em que é possível visualizar o operador, o símbolo associado e a forma de uso.

Operador	Símbolo	Exemplo
Igual	==	A == 1
Diferente	!=	A != B
Maior	>	A > 5
Menor que	<	B < 12
Maior ou igual a	>=	A >= 6
Menor ou igual a	<=	B <=7

As expressões **lógicas** são aquelas cujo resultado consiste em um valor lógico verdadeiro ou falso. Neste tipo de expressão, podem ser usados os operadores relacionais, os operadores lógicos ou as expressões matemáticas.

Operador	Símbolo	Exemplo
Disjunção	||	A disjunção entre duas variáveis resulta em um valor verdadeiro quando, pelo menos, uma das variáveis é verdadeira.
Conjunção	&&	A conjunção entre duas variáveis resulta em um valor verdadeiro somente quando as duas variáveis são verdadeiras.
Negação	!	A negação inverte o valor de uma variável. Se a variável A é verdadeira, então, a negação de A torna o valor da variável falso.

AS FUNÇÕES **INTRÍNSECAS** e a atribuição

As funções intrínsecas são fórmulas matemáticas prontas que podemos utilizar em nossos programas. O quadro a seguir apresenta as principais funções da linguagem C.

Função	Exemplo	Objetivo
`ceil`	`ceil(x)`	Arredonda um número real para cima. Por exemplo, `ceil(2.3)` é 3.
`cos`	`cos(x)`	Calcula o cosseno de `x`. O valor de `x` deve estar em radianos.
`exp`	`exp(x)`	Obtém a constante de Euler elevada à potência `x`.
`abs`	`abs(x)`	Retorna o valor absoluto de `x`.
`floor`	`floor(x)`	Arredonda um número real para baixo. Por exemplo, `floor(2.3)` é 2.
`log`	`log(x)`	Retorna o logaritmo natural de `x`.
`log10`	`log10(x)`	Retorna o logaritmo de `x` na base 10.
`modf`	`z=modf(x,&y)`	Decompõe o número real, armazenado em `x`, em duas partes: `y` recebe a parte fracionária e
`pow`	`pow(x,y)`	Calcula a potência de `x` elevado a `y`.
`sin`	`sin(x)`	Calcula o seno de `x`.
`sqrt`	`sqrt(x)`	Calcula a raiz quadrada de `x`.
`tan`	`tan(x)`	Calcula a tangente de `x`.
`M_PI`	`M_PI`	Retorna o valor da constante trigonométrica π.

Atribuição

O comando de atribuição é usado para conceder valores ou operações a variáveis. O símbolo utilizado para a atribuição é o = (sinal de igualdade). A sintaxe para atribuição é dada por:

<identificador> = <expressão>

Exemplo	
`x = 3;`	O valor 3 é atribuído à variável `x`.
`x = x + 5;`	É atribuído à variável `x` o valor da própria variável `x` acrescido em 5.
`x = 2.5;`	O valor 2.5 é atribuído à variável `x`.
`sexo = 'M';`	O valor `'M'` é atribuído à variável denominada sexo.
`nome = "Maria";`	O valor `"Maria"` é atribuído à cadeia de caracteres chamada `nome`.

Note que, na linguagem C, os caracteres simples (char) são representados entre apóstrofos (‘), e as cadeias de caracteres (strings), entre aspas (“).

A ENTRADA DE DADOS em um programa

A entrada de dados permite receber os dados digitados pelo usuário por meio da entrada padrão do dispositivo computacional, geralmente, o teclado. Os dados recebidos são armazenados em variáveis. Na linguagem C, existem diversas funções para entrada de dados, algumas delas são scanf e gets (quadro a seguir) (ASCENCIO; CAMPOS, 2010).

Exemplo	
`gets(<variável>);`	Um ou mais caracteres digitados pelo usuário são armazenados na variável.
`scanf("<texto>", &<variável>);`	Um valor digitado pelo usuário é armazenado na **variável**.

A função scanf() é a mais utilizada

scanf(“<expressão de controle>”, <lista de variáveis>);

O argumento <expressão de controle> deve ser escrito entre aspas e contém os especificadores de formato, que indicam como os dados digitados devem ser armazenados (próximo slide).
No argumento <lista de variáveis>, as variáveis devem ser separadas por vírgulas e cada uma delas deve ser precedida pelo operador de endereço (&). As variáveis usadas para receber valores por meio da função scanf() deverão ser passadas pelos seus endereços. O operador de endereço indica o endereço da posição de memória para a variável (ROCHA, 2006).

Obs: Na leitura de cadeias de caracteres (*strings*), não se utiliza o operador de endereço (&), pois o identificador do vetor já é o endereço do primeiro elemento do vetor.

Código	Significado
`%c`	Leitura de um único **caractere**.
`%d`	Leitura de um número decimal inteiro.
`%i`	Leitura de um decimal inteiro.
`%u`	Leitura de um decimal sem sinal.
`%e`	Leitura de um número em ponto flutuante com sinal opcional.
`%f`	Leitura de um número em ponto flutuante com ponto opcional.
`%g`	Leitura de um número em ponto flutuante com expoente opcional.
`%o`	Leitura de um número em base octal.
`%s`	Leitura de uma *string*.
`%x`	Leitura de um número em base hexadecimal.
`%p`	Leitura de um ponteiro.

Tomemos como exemplo:
para ler uma variável que armazena a idade e outra que armazena o peso, o uso da função scanf() deve ser realizado do seguinte modo:

scanf("%d", &idade); → %d Leitura de um decimal inteiro
scanf("%f", &peso); → %f Leitura de um número em ponto flutuante

Lembrando que, para utilizar os comandos de entrada de dados, é necessário incluir a biblioteca **stdio.h**, utilizando o comando: #include **<stdio.h>**

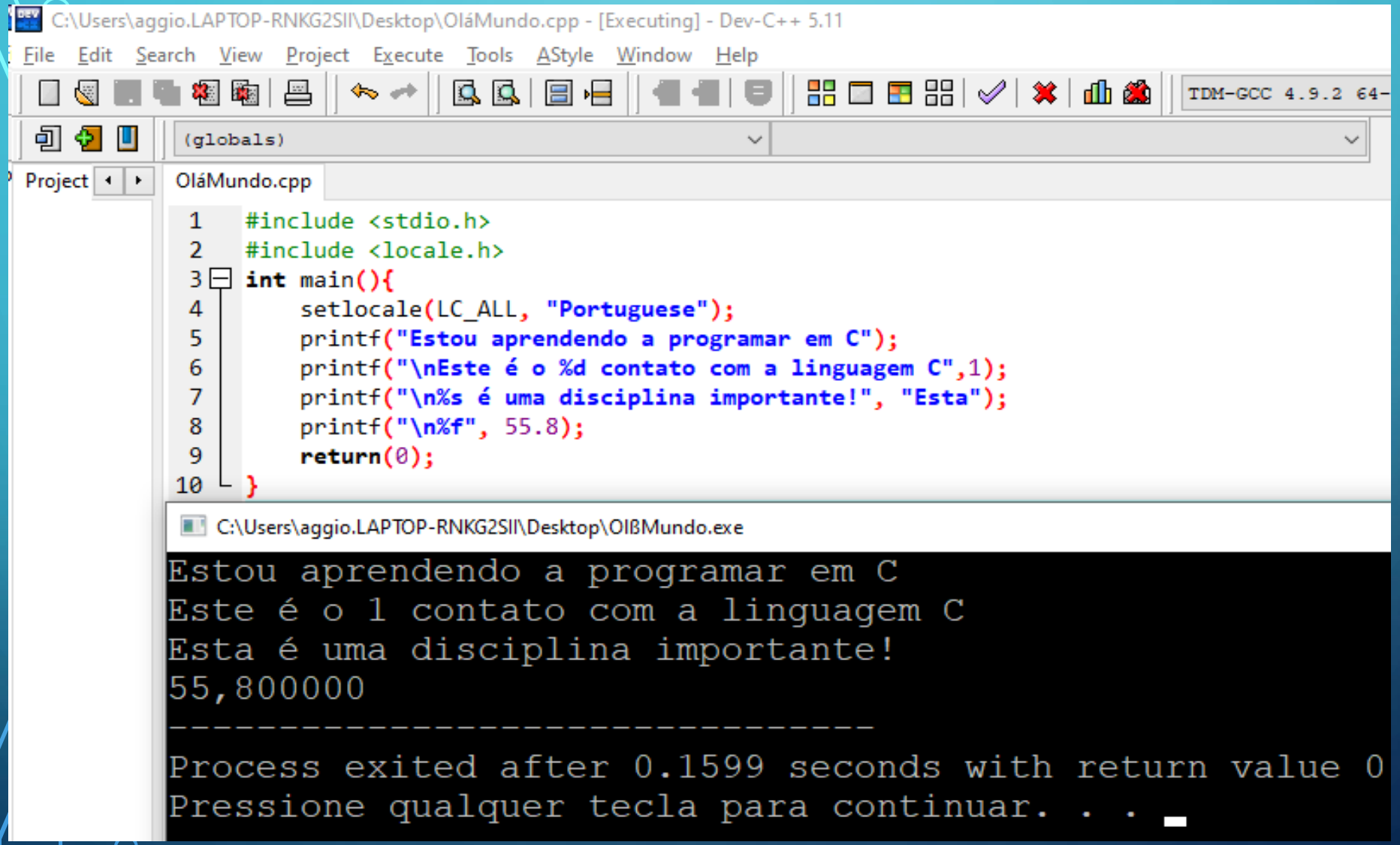
A SAÍDA DE DADOS em um programa

A saída de dados permite mostrar estes aos usuários. Na linguagem C, geralmente, utilizamos a função printf() para exibir resultados do processamento e mensagens.
A sintaxe desta função é:

printf(“<expressão de controle>”, <lista de argumentos>);

O argumento <expressão de controle> pode conter mensagens que serão exibidas na tela e os especificadores de formato que indicam o formato que os argumentos devem ser impressos, além de caracteres especiais indicadores de formatação de texto, como é o caso do \n, \t, etc. A <lista de argumentos> pode conter identificadores de variáveis, expressões aritméticas ou lógico-relacionais e valores constantes.

O programa a seguir apresenta alguns exemplos de uso da função printf().



Obs: "\n" antes dos printf(), para escrever na próxima linha

Construindo um programa

Vamos elaborar um programa que leia nome, idade e altura de uma pessoa e exiba nome, idade, altura e ano de nascimento dela. Para facilitar o entendimento do problema, o estruturaremos em três partes: entrada, processamento e saída.

Vamos elaborar um programa que leia nome, idade e altura de uma pessoa e exiba nome, idade, altura e ano de nascimento dela. Para facilitar o entendimento do problema, o estruturaremos em três partes: entrada, processamento e saída.

Entrada de dados: temos que obter os dados de nome, idade e altura. Cada um deles precisa ser armazenado em uma variável e, em sua leitura, utilizaremos a função scanf().

Processamento: temos que calcular o ano em que a pessoa nasceu.

Saída: devemos enviar para a tela, o nome, a idade, a altura e ano de nascimento. Para mostrar estas informações no monitor, utilizaremos a função printf().

[*] Exemplo1.cpp

```c
//Bibliotecas
#include <stdio.h> // Entrada e saída
#include <locale.h> //Idioma
//Programa Principal
int main(){
    setlocale(LC_ALL, "Portuguese");
    char nome[30];
    int idade, ano;
    float altura;
    //Entrada de dados
    printf("Digite o nome: "); //Não digite um nome composto.
    scanf("%s", nome); //Aqui é uma string, NÃO precisa do e comercial (&).
    printf("Digite a idade: ");
    scanf("%d", &idade); //Cuidado para não esquecer do e comercial (&).
    printf("Digite a altura: "); //O separador decimal é o ponto.
    scanf("%f", &altura); //Cuidado para não esquecer do e comercial (&).
    //Processamento
    ano = 2020 - idade;
    //Saída
    printf("------------------------------");
    printf("\nNome = %s",nome);
    printf("\nIdade = %d",idade);
    printf("\nAltura = %f", altura);
    printf("\nAno de Nascimento = %d", ano);
    return(0);
}
```

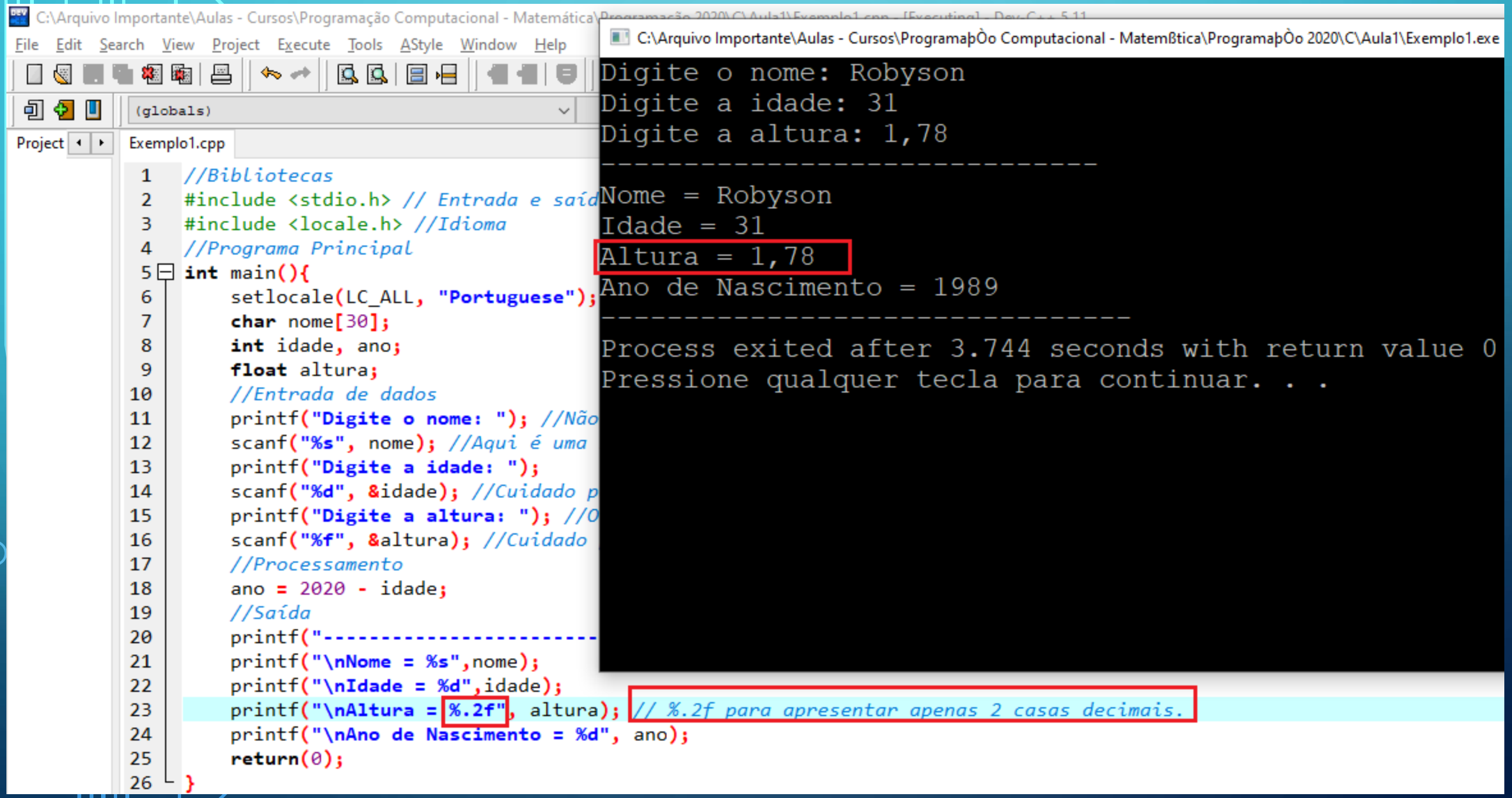
Obs: Como estamos incluindo o padrão do idioma para o português, o separador decimal é a vírgula! Caso contrário sempre será o ponto.

Exemplo1.cpp

```
1  //Bibliotecas
2  #include <stdio.h> // Entrada e saída
3  #include <locale.h> //Idioma
4  //Programa Principal
5  int main(){
6      setlocale(LC_ALL, "Portuguese");
7      char nome[30];
8      int idade, ano;
9      float altura;
10     //Entrada de dados
11     printf("Digite o nome: "); //Não digite u
12     scanf("%s", nome); //Aqui é uma string, N
13     printf("Digite a idade: ");
14     scanf("%d", &idade); //Cuidado para não e
15     printf("Digite a altura: "); //O separado
16     scanf("%f", &altura); //Cuidado para não
17     //Processamento
18     ano = 2020 - idade;
19     //Saída
20     printf("------------------------------");
21     printf("\nNome = %s",nome);
22     printf("\nIdade = %d",idade);
23     printf("\nAltura = %f", altura);
24     printf("\nAno de Nascimento = %d", ano);
25     return(0);
26 }
```

C:\Arquivo Importante\Aulas - Cursos\ProgramaþÒo Computacional - Matemßtica\ProgramaþÒo 2020\C\Aula1\Exemplo1.exe

```
Digite o nome: Robyson
Digite a idade: 31
Digite a altura: 1,78
------------------------------
Nome = Robyson
Idade = 31
Altura = 1,780000
Ano de Nascimento = 1989
--------------------------------
Process exited after 3.76 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```


C:\Arquivo Importante\Aulas - Cursos\Programação Computacional - Matemática\
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project
Exemplo1.cpp
1 //Bibliotecas
2 #include <stdio.h> // Entrada e saíd
3 #include <locale.h> //Idioma
4 //Programa Principal
5 int main(){
6 setlocale(LC_ALL, "Portuguese");
7 char nome[30];
8 int idade, ano;
9 float altura;
10 //Entrada de dados
11 printf("Digite o nome: "); //Não
12 scanf("%s", nome); //Aqui é uma
13 printf("Digite a idade: ");
14 scanf("%d", &idade); //Cuidado p
15 printf("Digite a altura: "); //O
16 scanf("%f", &altura); //Cuidado
17 //Processamento
18 ano = 2020 - idade;
19 //Saída
20 printf("----------
21 printf("\nNome = %s",nome);
22 printf("\nIdade = %d",idade);
23 printf("\nAltura = %.2f", altura); // %.2f para apresentar apenas 2 casas decimais.
24 printf("\nAno de Nascimento = %d", ano);
25 return(0);
26 }
C:\Arquivo Importante\Aulas - Cursos\ProgramaþÒo Computacional - Matemßtica\ProgramaþÒo 2020\C\Aula1\Exemplo1.exe
Digite o nome: Robyson
Digite a idade: 31
Digite a altura: 1,78
Nome = Robyson
Idade = 31
Altura = 1,78
Ano de Nascimento = 1989
Process exited after 3.744 seconds with return value 0
Pressione qualquer tecla para continuar. . .

- Na pasta em que você salvou o arquivo do código fonte, execute o arquivo com extensão “.exe” e verifique o que ocorrerá.
- Para contornar o problema acima: Inclua a biblioteca #include <stdlib.h> //Biblioteca para usar system("pause");
- Na linha acima do return(0), digite: system("pause");
- Compile e Execute (F11).
- Feche o devC++ e tente executar o arquivo .exe (verifique que agora o programa não fechará até ser pressionado uma tecla).
- Portanto: sempre inclua as três bibliotecas estudadas!
  #include <stdio.h> //Entrada e saída
  #include <stdlib.h> //System(“pause”)
  #include <locale.h> //Idioma Português

Exercícios

1) Escreva um programa que leia um número inteiro e apresente seu antecessor e seu sucessor.

2) Elabore um programa que receba quatro notas e calcule a média aritmética entre elas.

3) Faça um programa que receba o valor de um depósito e o valor da taxa de juros, calcule e apresente o valor do rendimento e o valor total (valor do depósito + valor do rendimento).

4) Escreva um programa que receba dois números, calcule e apresente o resultado do primeiro número elevado ao segundo. (Dica: inclua a biblioteca: #include <math.h>

5) Elabore um programa que calcule a área de um trapézio.

Gabarito

Exercício 1

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

int main(){
    setlocale(LC_ALL, "Portuguese");
    int numero;
    printf("Digite um número: ");
    scanf("%d", &numero);
    printf("Antecessor = %d", numero-1);
    printf("\nSucessor = %d \n", numero+1);
    system("pause");
    return(0);
}
```

Gabarito

Exercício 2

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

int main(){
    setlocale(LC_ALL, "Portuguese");
    float n1, n2, n3, n4, mediaAritmetica;
    printf("Digite a nota 1: ");
    scanf("%f", &n1);
    printf("Digite a nota 2: ");
    scanf("%f", &n2);
    printf("Digite a nota 3: ");
    scanf("%f", &n3);
    printf("Digite a nota 4: ");
    scanf("%f", &n4);
    mediaAritmetica = (n1+n2+n3+n4)/4.0;
    printf("Média final = %f \n", mediaAritmetica);
    system("pause");
    return(0);
}
```

Gabarito

Exercício 3

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

int main(){
    setlocale(LC_ALL, "Portuguese");
    float valorDeposito, taxaJuros, rendimentoJuros, rendimentoTotal;
    printf("Digite o valor do depósito: ");
    scanf("%f", &valorDeposito);
    printf("Digite a taxa de juros: ");
    scanf("%f", &taxaJuros);
    rendimentoJuros = valorDeposito*taxaJuros/100.0;
    printf("Rendimento = %f", rendimentoJuros);
    rendimentoTotal = valorDeposito + valorDeposito*taxaJuros/100.0;
    printf("\nValor total = %f\n", rendimentoTotal);
    system("pause");
    return(0);
}
```

Gabarito

Exercício 4

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <math.h>

int main(){
    setlocale(LC_ALL, "Portuguese");
    float num1, num2, resultado;
    printf("Digite o primeiro número: ");
    scanf("%f", &num1);
    printf("Digite o segundo número: ");
    scanf("%f", &num2);
    resultado = pow(num1,num2);
    printf("%.1f^%.1f = %.1f \n",num1,num2,resultado);
    system("pause");
    return(0);
}
```

Gabarito

Exercício 5

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

int main(){
    setlocale(LC_ALL, "Portuguese");
    float baseMaior, baseMenor, altura, area;
    printf("Digite o valor da base menor: ");
    scanf("%f", &baseMenor);
    printf("Digite o valor da base maior: ");
    scanf("%f", &baseMaior);
    printf("Digite o valor da altura: ");
    scanf("%f", &altura);
    area = (baseMenor + baseMaior)/2.0*altura;
    printf("A área do trapézio é = %f \n", area);
    system("pause");
    return(0);
}
```