

# Portugol → C

## O Compilador C

C é uma linguagem de programação desenvolvida nos anos 70 por Dennis Ritchie. Trata-se de uma linguagem de uso geral estruturada e imperativa.

Basicamente, o processo de compilação consiste em:

Escrita do arquivo fonte → Geração do arquivo-objeto → Geração do arquivo executável		
Os comandos são escritos em arquivos de texto comuns e salvos, por padrão, com a extensão .c, por exemplo: calculadora.c	Nessa etapa o compilador gera um objeto intermediário, que será automaticamente nominado com o nome do arquivo fonte, porém, com extensão .o: Exemplo: calculadora.o	Trata-se do arquivo final, que será executado pelo computador realizando as tarefas que lhe foram programadas. Exemplo: calculadora.exe

Os arquivos compilados que tornam-se executáveis só serão devidamente executados na plataforma na qual tenham sido compilados, por exemplo, se o arquivo fonte C foi compilado em ambiente Windows, o mesmo só será executado neste ambiente, se for compilado em ambiente Linux, só será possível sua execução no ambiente Linux, assim para Mac OS X, Unix e assim por diante.

Para a programação do arquivo fonte vamos utilizar a IDE (Interface de desenvolvimento) Code::Blocks. Entenda que o Code::Blocks não é um compilador, é um ambiente de programação que facilita a tarefa de programar, tornando mais fácil detecção de erros sintáticos e semânticos no código, assim como utilizar um esquema de cores que facilita o entendimento e ferramentas de depuração e Debug, muito úteis na busca por erros no código ou no estudo da otimização de códigos gerados.

O link para download do Code::Blocks que já possui acoplado um compilador (GCC) para instalação automática está disponível no Moodle, no endereço:

<http://sourceforge.net/projects/codeblocks/files/Binaries/12.11/Windows/codeblocks-12.11mingw-setup.exe>

Vamos agora a estrutura básico de um arquivo fonte. Faremos analogia diretamente com o Portugol, ambiente de ensino de algoritmos de programação mais didático que utilizamos até o momento:

Portugol	C
Início	main(){ ←Chave de abertura do bloco main (início)
...	...
...	...
fim	} ←Chave de fechamento do bloco main (fim)

A definição acima é a mais elementar. Afim de que possamos trabalhar com alguns comandos de leitura e escrita do C, faremos um incremento no código, que fica da seguinte forma

Portugol	C
Início	#include <stdio.h> ← Biblioteca “Standart I/O”. main(){ ←Chave de abertura do bloco main (início)
...	...
...	...
fim	} ←Chave de fechamento do bloco main (fim)

É comum encontrarmos algumas variações no código básico do bloco main, como por exemplo:

```
#include <stdio.h>
int main(void){
    ...
    ...
    return 0
}
```

O exemplo acima, tal quais os demais citados, funcionam da mesma forma.

Declaração de variáveis:

Portugol	C		
Início inteiro numero real valor caracter sexo texto modelo ... fim	→ → → →	<pre>#include &lt;stdio.h&gt; main(){     int numero;     float valor;     char sexo;     char modelo[255];     ... }</pre>	Obs.: Na linguagem C, cada final de declaração deverá conter o ;(ponto e vírgula), que indica o final da instrução. Na verdade, não só nas declarações de variáveis, mas em qualquer final de instrução será utilizado o ponto e vírgula. Uma variável do tipo texto, definida como char modelo[255], tem tamanho máximo de 255 caracteres e é chamada string.

Declarando mais variáveis do mesmo tipo:

Portugol	C		
Início inteiro numero, temp, aux real valor, bruto, liquido ... fim	→ →	<pre>#include &lt;stdio.h&gt; main(){     int numero, temp, aux;     float valor, bruto, liquido;     ... }</pre>	Obs.: A exemplo do que usamos no Portugol, C também permite a declaração de variáveis do mesmo tipo utilizando a , (vírgula) como separador.

Declarar vetores e matrizes:

Portugol	C		
Início inteiro vetor[10] inteiro matriz[5][5] ... fim	→ →	<pre>#include &lt;stdio.h&gt; main(){     int vetor[10];     int matriz[5][5];     ... }</pre>	

Comando de atribuição:

Portugol	C		
Início inteiro x, y x ← 2 y ← x + 3 fim	<pre>main(){     int x, y;     x = 2;     y = x + 3; }</pre>		

No C o comando de atribuição é o sinal = (igual) ao invés da seta ← .

Obs.: No Portugal, ao declararmos uma variável do tipo inteiro ou real, o valor atribuído automaticamente a tais variáveis é 0 (zero). No C esse processo não ocorre de forma automática, portanto, precisamos atribuir o 0 (zero) à variável, especialmente nos caso em que haja necessidade de incremento do uma variável, por exemplo: `cont = cont + 1`. Se não inicializarmos a variável `cont` com 0 (zero), o programa será executado com erro. Para declarar variáveis atribuindo valores, procede-se como a seguir:

```
#include <stdio.h>
main(){
    int numero, cont=0, aux=2;
    float valor, bruto, liquido=0.0;
    char sexo = ' f ';
    ...
}
```

Como vimos acima, o sinal de = (igual) é o comando de atribuição no C. No Portugal, o sinal de = significava igualdade. Vejamos nas tabelas abaixo como ficam os sinais aritméticos, relacionas e lógicos na linguagem C:

Relacionais		
Portugol	C	
<	<	menor
>	>	maior
<=	<=	Menor ou igual
>=	>=	Maior ou igual
=	==	igual
!=	!=	diferente

Aritméticos		
Portugol	C	
+	+	soma
-	-	subtração
*	*	multiplicação
/	/	divisão
%	%	Módulo (resto)

Lógicos	
Portugol	C
e	&&
ou	
não	!

Escrever:

Portugol	C
Inicio escrever “Boa tarde” fim	<pre>#include &lt;stdio.h&gt; main(){     printf (“Boa Tarde”); }</pre>

## Escrever texto e variáveis:

### Portugol

#### Início

```
inteiro idade
real altura
caracter sexo
idade ← 23
altura ← 1.78
sexo ← "f"
escrever "Sua idade é ", idade, " sua altura é", altura, "e seu sexo é", sexo
```

fim

### C

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

```
main(){
```

```
    int idade;
```

```
    float altura;
```

```
    char sexo;
```

```
    idade = 23;
```

```
    altura = 1.78;
```

```
    sexo = 'f';
```

```
    printf ("Sua idade é %i sua altura é %f e seu sexo é %c", idade, altura, sexo);
```

```
}
```

Na linguagem C, um caractere é definido utilizando-se aspas simples '. Aspas duplas " são utilizadas para definir strings (texto)

Vejamos a estrutura do comando printf:

```
printf ("Sua idade é %i sua altura é %f e seu sexo é %c", idade, altura, sexo);
```

Cada espaço indicado pelos sinais %i, %f e %c serão substituídos pelas suas respectivas variáveis, idade, altura e sexo. É a ordem que determina a substituição das variáveis, e não o tipo. Como idade é a primeira, o primeiro espaço, determinado pelo sinal %i exibirá o valor da variável, e assim por diante.

Tipos de variáveis:

%i ou %d	Int
%f	Float
%c	char
%s	String (texto)

Nova linha

Portugol	C
\n	\n

Exemplo:

```
printf ("Sua idade é %i\n sua altura é %f\n e seu sexo é %c\n", idade, altura, sexo);
```

## Ler

A leitura de variáveis tem variações conforme o tipo que se deseje ler.

Portugol	C
<pre>Inicio inteiro idade real altura caracter sexo     escrever "Digite a idade"     ler idade     escrever "Digite a altura"     ler altura     escrever "Digite o sexo"     ler sexo     escrever "Sua idade é ", idade, " sua altura é", altura, "e seu sexo é", sexo fim</pre>	<pre>#include &lt;stdio.h&gt; main(){     int idade;     float altura;     char sexo;     printf ("Digite a idade");     scanf (" %d ", &amp;idade);     printf ("Digite a altura");     scanf (" %f ", &amp;altura);     printf ("Digite o sexo");     scanf (" %c ", &amp;sexo);     printf ("Sua idade é %d sua altura é %f e seu sexo é %c", idade, altura, sexo); }</pre>

Observe que a regra é a mesma para a escrita, %i ou %d para variáveis do tipo inteiro, %f para float e %c para caracteres. A regra NÃO se aplica para leitura de strings, para tal, veremos o procedimento mais adiante.

Todavia, se compilarmos e executarmos o trecho de código, um erro ocorre ao ler a variável sexo. O problema é que, na linguagem C, a tecla ENTER é lida como um caractere, o que causa o problema, pois ao lermos a variável altura, pressionarmos a tecla ENTER, e a mesma é lida como um caractere que está sendo esperado e é atribuído à variável sexo. Para contornarmos tal problema, toda vez que lermos uma variável do tipo char (caractere), vamos utilizar o comando getchar(). Vejamos o exemplo:

```
#include <stdio.h>
main(){
    int idade;
    float altura;
    char sexo;
    printf ("Digite a idade")
    scanf ("%d",&idade);
    printf ("Digite a altura")
    scanf ("%f",&altura);
    printf ("Digite o sexo")
    getchar();
    scanf ("%c",&sexo);
    printf ("Sua idade é %i sua altura é %f e seu sexo é %c", idade, altura, sexo);
}
```

O getchar() tem por finalidade “limpar” o ENTER do buffer do teclado antes de ler a variável sexo.

Leitura de variáveis String: NÃO EXISTE scanf(" %s ", &variável), para ler uma variável do tipo string, utilizamos o comando gets(nome da variável), antecedido por getchar() pelos mesmos motivos da leitura de caracteres. Todavia, para utilizar o comando gets, precisamos declarar a biblioteca **string.h**, da mesma forma e no mesmo local que declaramos a stdio.h. Vejamos um exemplo:

```

#include <stdio.h>
#include <string.h>
main(){
    int idade;
    char nome[100];
    printf ("Digite a idade");
    scanf ("%d",&idade);
    printf ("Digite o nome");
    getchar();
    gets(nome);
    printf ("Seu nome é %s sua idade é %d", nome, idade);
}

```

Declarando constantes:

Portugol	C
<p>Inicio</p> <p>    constante inteiro IDADE ← 23</p> <p>    escrever “a idade é”, idade</p> <p>fim</p>	<pre> #include &lt;stdio.h&gt; #define IDADE 23 main(){     printf (“a idade é %d”, IDADE); } </pre>

Condição SE

Portugol	C
<p>Se (condição) então</p> <p>    Bloco se verdadeiro</p> <p>fimse</p>	<pre> If (condição) {     Bloco se verdadeiro } </pre>
<p>Se (condição) então</p> <p>    Bloco se verdadeiro</p> <p>senão</p> <p>    Bloco se falso</p> <p>fimse</p>	<pre> If (condição) {     Bloco se verdadeiro } else{     Bloco se falso } </pre>
<p>Inicio</p> <p>    inteiro idade</p> <p>    escrever “Digite a idade”</p> <p>    ler idade</p> <p>    se idade &gt;= 18 então</p> <p>        escrever “Pode ter CNH”</p> <p>    senão</p> <p>        escrever “Não pode ter CNH”</p> <p>    fimse</p> <p>fim</p>	<pre> #include &lt;stdio.h&gt; #include &lt;string.h&gt; main(){     int idade;     printf (“Digite a idade”);     scanf( “ %d ”, &amp;idade);     if (idade &gt;= 18){         printf (“Pode ter CNH”);     }     else {         printf (“Não pode ter CNH”);     } } </pre> <div> <p>Bloco Verdadeiro</p> <p>Bloco Falso</p> </div>

OBS.: A condição deve estar entre parênteses if (idade >= 18).

Exemplo com uso de operadores lógicos:

Portugol	C
Se sexo = “f” ou sexo = “m” então ...	If ((sexo == ' f ')    (sexo == ' m ')) { ...
Se idade > 18 e idade < 36 então ...	If ((idade > 18) && (idade < 36)) { ...

If encadeado: C permite utilizarmos o comando if – eles de forma encadeada, como na sentaxe a seguir:

```

if (condição){
    Execução se condição verdadeira
}
else if (condição){
    Execução se condição verdadeira
}
else if (condição){
    Execução se condição verdadeira
}
else{
    Execução se todas as condições
    anteriores forem falsas
}

```

Escolhe – caso – defeito → switch – case – default

Portugol	C
<p>Inicio</p> <p>inteiro opcao</p> <p>escrever “Digite a opção”</p> <p>ler opcao</p> <p>escolhe opção</p> <p>caso 1:</p> <p>escrever “Gerente”</p> <p>caso 2:</p> <p>escrever “Supervisor”</p> <p>caso 3:</p> <p>escrever “Atendente”</p> <p>defeito:</p> <p>escrever “Opção inválida”</p> <p>fimescolhe</p> <p>fim</p>	<pre> #include &lt;stdio.h&gt; #include &lt;string.h&gt; main(){     int opcao;     printf (“Digite a opção”);     scanf ("%d", &amp; opcao);     switch (opcao){         case 1: {             printf (“Gerente”);         }         break;         case 2: {             printf (“Supervisor”);         }         break;         case 3: {             printf (“Atendente”);         }         break;         default:{             printf (“Opção inválida”);         }     } } </pre>

### Switch com múltiplos testes:

Portugol	C
<pre> Inicio     inteiro posicao     escrever "Digite a posição"     ler posicao     escolhe opção         caso 1, 2, 3:             escrever "Classificado"          caso 4, 5, 6:             escrever "Repescagem"          defeito:             escrever "Desclassificado"      fimescolhe fim         </pre>	<pre> #include &lt;stdio.h&gt; #include &lt;string.h&gt; main(){     int posicao;     printf ("Digite a posição");     scanf ("%d", &amp; posicao);     switch (posicao){         case 1:         case 2:         case 3 :{             printf ("Classificado");         } break;         case 4:         case 5:         case 6: {             printf ("Repescagem");         } break;         default:{             printf ("Desclassificado");         }     } }         </pre>

### Enquanto – fimenquanto → while

Portugol	C
<pre> Inicio     inteiro cont     cont ← 0     enquanto (cont &lt; 10)         escrever "Contador ", cont         cont ← cont + 1     fimenquanto     escrever "Finalizando" fim         </pre>	<pre> #include &lt;stdio.h&gt; #include &lt;string.h&gt; main(){     int cont;     cont = 0;     while (cont &lt; 10) {         printf ("Contador %i", cont);         cont = cont + 1; (ou ainda cont++)     }     printf ("Finalizando"); }         </pre>



Faz – enquanto → do – while

Portugol	C
<pre>Inicio   inteiro cont   cont ← 0   faz     escrever “Contador ”, cont     cont ← cont + 1   enquanto (cont &lt; 10)     escrever “Finalizando” fim</pre>	<pre>#include &lt;stdio.h&gt; #include &lt;string.h&gt; main(){   int cont;   cont = 0;   do {     printf (“Contador %i”, cont);     cont = cont + 1; (ou ainda cont++)   } while (cont &lt; 10);   printf (“Finalizando”); }</pre>

Para → for

Portugol	C
<pre>Inicio   inteiro cont   para cont de 0 até 10 passo 1     escrever “Contador ”, cont   próximo   escrever “Finalizando” fim</pre>	<pre>#include &lt;stdio.h&gt; #include &lt;string.h&gt; main(){   int cont;   for (cont = 0; cont &lt; 10; cont++){     printf (“Contador %i”, cont);   }   printf (“Finalizando”); }</pre>

Sintaxe do comando for:

```
for (  cont =0;  cont < 10;  cont++){
      inicio    condição    passo
```

Incremento:

```
for (  cont =10;  cont > 0;  cont--){
      inicio    condição    passo
```

Incremento/ decremento maior que 1:

```
for (  cont =0;  cont < 10;  cont = cont + 2){
      inicio    condição    passo
```

```
for (  cont =20;  cont > 0;  cont = cont - 3){
      inicio    condição    passo
```

Comentários em código C:

Utilize // no início de uma linha para cometá-la, ou /\* comentário \*/ para comentar várias

Exemplo:

```
#include <stdio.h>
#include <string.h>
main(){
    /* esse é um comentário do código
       código que possui mais de uma
       linha */
    int idade;
    float altura;
    char sexo;
    //int peso; ← comentário de uma linha
    idade = 23;
    altura = 1.78;
    sexo = 'f';
    printf ("Sua idade é %d sua altura é %f e seu sexo é %c", idade, altura, sexo);
}
```

Gerar números aleatórios (randômicos) no C:

Para gerar números randivos em C, temos duas funções que trabalham de forma semelhante:

- rand  
- random.

Estas duas funções geram números aleatórios de 0 a valor\_maximo. Se você quiser gerar valores randômicos em uma determinada faixa numérica, basta fazer assim:

numero= rand() % valor\_maximo;  
ou  
numero= random() % valor\_maximo;

Para usá-las, porém, é necessário inicializar o gerador de números aleatórios com a função srand.

Outra observação muito importante, é que as funções rand, random e srand pertencem a biblioteca stdlib.h, que deverá ser declarada tal qual fizemos com stdio.h e string.h.

Vejam um exemplo:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
main() {
    int i;
    srand(100); /* inicializar o gerador de números aleatórios */
    for (i=0; i<5; i++) {
        printf("%d ", rand() % 50); /* para gerar números aleatórios de 0 a 50 */
    }
}
```

O exemplo acima gera números randômicos, todavia, sempre a mesma sequencia. Para sanar tal

problema, vejamos o exemplo abaixo:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
main() {
    int i;
    srand (time(NULL)); /* inicializar o gerador de números aleatórios */
    for (i=0; i<5; i++) {
        printf("%d ", rand() % 50); /* para gerar números aleatórios de 0 a 50 */
    }
}
```

O comando `srand (time(NULL))`, que é o inicializador de números aleatórios, utilizará o relógio do sistema(`time`) como semente para a geração dos números, portanto, a cada momento gera números diferentes.