Introdução à Programação #7

Tipo *int*

O tipo de dados *int* é, como já deve saber, referente a números inteiros. *Int* quer dizer *integer number*, ou seja, números inteiros. C, como é uma linguagem que trabalha muito com o *hardware* do computador, permite-nos definir o **tamanho** de cada variável deste (e de outros) tipos.

Primeiramente, como se declara uma variável *int* em C? É muito simples. No seguinte exemplo pode visualizar como declarar uma variável e, posteriormente, imprimi-la:

```
1.  #include <stdio.h>
2.
3.  int main() {
4.
5.      int a = 20;
6.
7.      printf("O número que guardei é %d.", a); //Imprime: O número que guardei é
     20.
8.      return 0;
9.  }
```

Antes de continuar, deve ter reparado que foi utilizado um *%d* dentro do primeiro parâmetro. Este caractere é substituído pelo valor da variável *a* quando é imprimido no ecrã.

Função printf e números inteiros:

Utiliza-se *%d* quando se quer imprimir o valor de uma variável dentro de uma frase. A variável deve ser colocada nos parâmetros seguintes, por ordem de ocorrência. Exemplo:

```
1.  #include <stdio.h>
2.
3.  int main() {
4.      int a = 20;
5.      int b = 100;
6.
7.      printf("O primeiro número é: %d.\n", a);
8.      //Imprime: O primeiro número é: 20.
9.      printf("O segundo número é: %d.\n", b);
10.     //Imprime: O segundo número é: 100.
11.     printf("O primeiro e segundo números são %d e %d.\n", a, b);
12.     //Imprime: O primeiro e segundo números são 20 e 100.
13.
14.     return 0;
15. }
```

Este tipo de variáveis ocupa, normalmente, entre 2 a 4 bytes na memória de um computador mas, e se quiser utilizar uma variável para um número pequeno? Não poderei gastar menos recursos? E se acontecer o contrário e precisar de um número maior?

Nestas situações, pode utilizar *long* e *short* para controlar os gastos de recursos. Estes modificadores permitem-nos criar variáveis que ocupem um maior ou menor número de bytes, respetivamente.

Um número inteiro de…

- 1 byte armazena de -128 a +127

- 2 bytes armazena de -32 768 a +32 767
- 4 bytes armazena de -2 147 483 648 a +2 147 483 647
- 8 bytes armazena de -9 223 372 036 854 775 808 a +9 223 372 036 854 775 807

A utilização destes modificadores é feita da seguinte forma:

```
1.  short int nomeDaVariavel = 20; //Ou "long"
```

O número de bytes atribuída utilizando um destes modificadores pode depender do computador onde o código está a ser executado. Para descobrirmos qual o tamanho de bytes que utiliza o seu sistema, basta utilizar a função *sizeof* da seguinte forma:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      printf("int : %d bytes\n", sizeof(int) );
6.      printf("short int: %d bytes\n", sizeof(short) );
7.      printf("long int: %d bytes\n", sizeof(long) );
8.      return 0;
9.  }
```

No computador que estou a utilizar, por exemplo, *short* refere-se a 2 bytes, *long* a 8 bytes e o tamanho padrão de*int* é 4 bytes.

Como sabe, os números inteiros podem assumir forma positiva e negativa. Por vezes, na programação, os números negativos podem atrapalhar (ou então ajudar), dependendo do caso.

Para termos controlo sobre a positividade ou negatividade de um número, podemos atribuir os modificadores*signed* e *unsigned*.

Para que uma variável possa conter tanto números positivos como negativos, devemos utilizar o modificador*signed*. Caso queira que o número seja apenas positivo, incluindo 0, utilize *unsigned*

Tipo float e double

Além dos números inteiros, existem outros tipos de dados que nos permitem armazenar números que, ao invés de serem inteiros, são decimais.

Existem dois tipos de dados que nos permitem armazenar valores do tipo decimal/real. Estes tipos são *float* e*double*. Devem ser utilizados da seguinte forma:

```
1.  float pi = 3.14;
2.  double pi = 3.14159265359;
```

Como pode ter reparado, em C (e na maioria das linguagens de programação), não se utiliza a vírgula, mas sim um ponto para separar a parte inteira da decimal.

A diferença entre *float* e *double* é que o segundo ocupa mais memória que o primeiro logo, consegue armazenar números de maior dimensão.

Normalmente, o tipo *float* ocupa 4 bytes de memória RAM enquanto o segundo tipo, *double*, ocupa 8 bytes de memória. Mais uma vez, relembro que estes valores podem alterar de máquina para máquina.

Para quem precise de fazer cálculos mais precisos, o tipo de dados *double* é o mais aconselhado pois é o que permite uma maior extensão do valor.

Função printf e números decimais:

Utiliza-se *%f* quando se quer imprimir o valor de uma variável dos tipos *float* ou *double* dentro de uma frase. A variável deve ser colocada nos parâmetros seguintes, por ordem de ocorrência. Exemplo:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      float piMinor = 3.14;
6.      double piMajor = 3.14159265359;
7.
8.      printf("Pi pode ser %f mas, de forma mais exata, é %f.", piMinor, piMajor)
    ;
9.      //Imprime: Pi pode ser 3.140000 mas, de forma mais exata, é 3.141593.
10.     return 0;
11. }
```

Pode visualizar que, quando se utiliza *%f*, é utilizado um número específico de casas decimais. Caso o número de casas decimais seja mais pequeno do que o da variável original, o número é arredondado.

Pode definir o número de casas decimais que quer que sejam apresentadas da seguinte forma: *%.{Número de Casas Decimais}f*. Veja o seguinte exemplo, baseado no anterior:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      float piMinor = 3.14;
6.      double piMajor = 3.14159265359;
7.
8.      printf("Pi pode ser %.2f mas, de forma mais exata, é %.11f.", piMinor, piM
    ajor);
9.      //Imprime: Pi pode ser 3.14 mas, de forma mais exata, é 3.14159265359.
10.     return 0;
11. }
```

Notação Científica

Certamente conhece notação científica, ou seja, números no formato **NUM1 X 10^{NUM2}** (NUM vezes dez elevado a NUM2). Podemos utilizar notação científica nas variáveis do tipo *float* e *double*. Veja o seguinte exemplo:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      float num = 24E-5; //24 x 10 elevado a -5
6.      printf("%f\n", num); //Imprime: 0.000240
```

```
7.
8.      num = 2.45E5; //2.45 x 10^5
9.      printf("%.0f", num);  //Imprime: 245000
10.
11.     return 0;
12. }
```

Tipo *char*

Há algumas semanas atrás, falámos do tipo *string*: um tipo de dados que nos permite armazenar sequências de caracteres, ou seja, de forma geral, frases.

Por agora, ainda não abordaremos este tipo, mas sim *char*. Este é um tipo de dados que nos permite armazenar um único caractere. É declarado da seguinte forma:

```
1.  char letra = 'P';
```

Como pode visualizar, a variável "letra" agora contém o caractere "P". Pode, ao invés de utilizar este tipo de notação, utilizar números hexadecimais, octais e decimais. Clique aqui para descarregar uma tabela ASCII em pdf com os códigos que pode utilizar em C.

Função printf e caracteres:

Utiliza-se *%c* quando se quer imprimir o valor de uma variável dos tipos *char* dentro de uma frase. A variável deve ser colocada nos parâmetros seguintes, por ordem de ocorrência. Exemplo:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      char letra = 'P';
6.
7.      printf("O nome Pplware começa por %c.", letra);
8.      return 0;
9.  }
```

Artigo original: http://pplware.sapo.pt/tutoriais/vamos-programar-introducao-a-programacao-7