COMANDOS DE SAÍDA DE DADOS

Comandos de saída

Contextualização

A partir de agora vamos conhecer os comandos de saída, utilizados na programação para permitir a exibição de informações ao usuário. Além disso, construiremos nosso primeiro programa em C. Você se lembra do teste estilo BuzzFeed indicado no início deste conteúdo?

Programa *Hello World!*

Você sabe qual a relação entre um programa e a linguagem C? Confira as suas respectivas definições:

Programa

É uma sequência de instruções dadas para resolver um problema.

Linguagem C

É a forma de dar essas orientações ao computador.

O nosso primeiro programa em C será o mais conhecido no mundo da programação: o *Hello World*.

```
#include <stdio.h>
int main(){
printf("Hello World");
return 0;
}
```

Função Printf()

Dentro da função **main()**, inserimos as instruções que serão executadas. Usam-se as chaves **{ }** para delimitar o que está incluso no corpo dessa função.

A primeira linha **#include <stdio.h>** é uma diretiva de pré-compilação e não uma instrução, por isso não é seguida por ponto e vírgula. A diretiva serve para incluir funções que estejam na biblioteca por meio das tags <>. Entende-se, então, que a biblioteca **stdio.h** tem funções que serão usadas em **main()**.

Em **main()**, nota-se uma única função, representada por **printf()**, que faz parte da biblioteca **stdio.h**. Por esse motivo, é preciso incluir a biblioteca no início do arquivo. *Printf*, traduzido do inglês como escrever formatado (*print* + *format*), tem como principal objetivo realizar a escrita na tela. Mas você pode estar se perguntando:

O que essa função exibe para o usuário?

Ela exibe o parâmetro recebido dentro dos parênteses! No exemplo anterior, printf() recebeu *Hello World* como parâmetro. Perceba que a string (cadeia de caracteres) está entre aspas, uma vez que servem para delimitá-la.

Para testar os conhecimentos adquiridos até aqui, tente fazer sozinho um programa que escreva o seu nome completo na tela.

Observe estas instruções:

```
#include <stdio.h>

int main(){

printf("Primeira linha");

printf("Segunda linha");

return 0;

}
```

Observe que a função printf() não faz a quebra de linha automática ao final da string. Em função disso, devemos inserir o caractere especial ‘\n’, ajustando o programa anterior para:

```
#include <stdio.h>

int main(){

printf("Primeira linha\n");

printf("Segunda linha");

return 0;

}
```

A função **printf()** também permite a utilização de variáveis para compor o que será escrito na tela. Para indicar a posição de entrada de conteúdo de variáveis

dos tipos **int** e **char** utilizam-se, respectivamente, os símbolos **%d** e **%c**. Vejamos, a seguir, a utilização dessas variáveis.

Observe o exemplo:

```
#include <stdio.h>

int main(){

int a = 10;

char ch = 'Z';

printf("Atualmente, temos a = %d e ch = %c.\n", a, ch);

return 0;

}
```

Também podemos utilizar mais de uma variável do mesmo tipo, desde que sejam passadas, corretamente, quais delas preencherão a frase. Será seguida, então, a ordem invocada em printf(), com os conteúdos das variáveis acompanhando a sequência de uso dos símbolos %d ou %c e a correspondente passagem de parâmetros. Vejamos como aplicar essas variáveis. Observe o exemplo:

```
#include <stdio.h>

int main(){

int a, b, c;

a = 10;

b = a + 1;

c = b + 2;

printf("Atualmente, temos a = %d, b = %d e c = %d.\n", a, b, c);

return 0;

}
```

Você também pode escrever uma expressão matemática como parâmetro da função printf() por meio destas linhas:

```
#include <stdio.h>

int main(){

int a;

a = 10;

printf("A variavel a vale %d. Seu sucessor e o %d.\n", a, a + 1);
```

```
return 0;
}
```

No próximo exemplo, utilizamos variáveis do tipo char. Confira:

```
#include <stdio.h>
int main(){
char ch1, ch2, ch3;
ch1 = 'H';
ch2 = 'o';
ch3 = 'W';
printf("%cell%c %corld.\n", ch1, ch2, ch3);
return 0;
}
```

Para ampliar seus conhecimentos, listamos os principais formatos de escrita e leitura das variáveis, usados com a função **printf()**:

Tipo	Formato	Observações
char	%c	Um único **caractere**
int	%d ou %i	Um inteiro (Base **d**ecimal)
int	%o	Um inteiro (Base **o**ctal)
int	%x ou %X	Um inteiro (Base he**x**adecimal)
short int	%hd	Um s**h**ort inteiro (Base **d**ecimal)
long int	%ld	Um **l**ong inteiro (Base **d**ecimal)
unsigned short int	%hu	Short inteiro positivo

Tipo	Formato	Observações
unsigned int	%u	Inteiro positivo
unsigned long int	%lu	Long inteiro positivo
float	%f ou %e ou %E	
double	%f ou %e ou %E	

Tabela: Formatos de escrita e leitura das variáveis usados com a função printf().
Humberto Henriques de Arruda.

O próximo exemplo mostra o uso de **printf** com variável do tipo **float**.

Observe:

```
#include <stdio.h>

int main(){

float a;

a = 12.5;

printf("a = %f\n", a);

return 0;
```

Repare que a variável do tipo **float** é armazenada com seis casas decimais. Para reduzir esse número, utiliza-se %.**1f**, %.**2f**, entre outros. O número entre "**.**" e "**f**" indica as casas decimais exibidas. É importante lembrar que o conteúdo da variável permanece inalterado, visto que a mudança afeta apenas a forma como será feita a escrita na tela. Vamos fazer um teste!

No exemplo anterior, caso alterássemos a última linha para:

```
printf("a = %.1f\n", a);
```

O resultado seria:

a = 12.5

Vamos praticar

Você receberá agora uma série de práticas para realizar em seu ambiente de programação. Tente executá-las. Vamos lá!

Prática 1

```
#include <stdio.h>

int main(){

int a, b, c;

a = 1;

b = a + 3;

c = a;

printf("b = %d e c = %d.\n", b, c);

return 0;

}
```

Ao usar o símbolo %d, o conteúdo das variáveis b e c será colocado na frase e será exibido o seguinte resultado:

b = 4 e c = 1

Prática 2

```
#include<stdio.h>

int main()

{

printf("Valor total: %.1f\n", 9.1415169265);

return 0;

}
```

Ao usar o símbolo %.1f, o conteúdo da variável será exibido com apenas uma casa decimal:

Valor total: 9.1

Prática 3

Vamos determinar qual é a função que as strings “%d”, “%f” e “%s” estão usualmente associadas na linguagem C.

A solução é que os símbolos **%d**, **%f** e **%s** são utilizados para compor a frase que a função **printf()** vai exibir na tela.