Introdução à Programação #9

Interação com o utilizador

Numa aplicação, a interação com os utilizadores é algo extremamente fundamental. Se não houver interação com o utilizador, a aplicação é, na maioria das vezes, algo inútil.

Na sequência disto, irá aprender a receber dados do utilizador utilizando duas funções: *scanf* e *getchar*. Todas diferentes mas, ao mesmo tempo, todas úteis.

scanf

A função "scanf" permite-nos obter diversos tipos de dados do utilizador: números inteiros, decimais e também caracteres. Exemplo:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      int idade;
6.
7.      printf("Digite a sua idade: ");
8.      scanf("%d", &idade);
9.
10.     printf("A sua idade é %d", idade);
11.
12.     return 0;
13. }
```

Como pode ver na oitava linha, o primeiro argumento da função "scanf" contém o tipo de variável que vai ser inserido (tal como na função "printf").

Os argumentos seguintes serão o nome das variáveis, precedidos por um "&", a que os valores vão ser atribuídos. Pode ler a linha 8 da seguinte forma: "atribuir um número inteiro a idade".

Quando se executa o código acima, irá aparecer a mensagem "Digite a sua idade: " e o cursor irá posicionar-se logo após essa frase aguardando que um valor seja inserido. Depois de inserir o valor e pressionar ENTER, será imprimida uma mensagem com a idade que inseriu.

Relembrando, pode utilizar as seguintes expressões para definir o tipo de dados a ser introduzido:

- %d → Números inteiros (int)
- %f → Números decimais (float e double)
- %c → Caracteres (char)

Pedir mais do que um item

Podemos pedir mais do que um valor/caractere da seguinte forma:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      int num1, num2;
6.
```

```
7.      printf("Digite dois números: ");
8.      scanf("%d %d", &num1, &num2);
9.
10.     printf("Os números que digitou são %d e %d.", num1, num2);
11.
12.     return 0;
13. }
```

Ou seja, depois de aparecer a mensagem “Digite dois números:”, terá que inserir dois números e pode identificar a sua separação com um espaço, enter ou tab.

Como pode ver, é muito simples utilizar a função scanf. Apesar de apenas termos utilizado números inteiros, pode utilizar números decimais ou caracteres com esta função.

getchar

Existe outra forma, mais simples, de pedir caracteres ao utilizador: a função “getchar”. Compare os seguintes dois exemplos (primeiro com scanf e segundo com getchar):

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      char letra;
6.
7.      printf("Insira a primeira letra do seu nome: ");
8.      scanf("%c", &letra);
9.
10.     printf("A primeira letra do seu nome é %c.", letra);
11.     return 0;
12. }
```

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      char letra;
6.
7.      printf("Insira a primeira letra do seu nome: ");
8.      letra = getchar();
9.
10.     printf("A primeira letra do seu nome é %c.", letra);
11.     return 0;
12. }
```

Como pode verificar, a utilização da função “getchar” é bastante simples e permite-nos receber, mais facilmente, caracteres do utilizador.

Buffer e a sua limpeza

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      char letra1, letra2;
```

```
6.
7.      printf("Insira a primeira letra do seu nome: ");
8.      scanf("%c",&letra1);
9.
10.     printf("E agora a última: ");
11.     scanf("%c",&letra2);
12.
13.     printf("O seu nome começa com \"%c\" e termina com \"%c\".", letra1, letra
    2);
14.
15.     return 0;
16. }
```

Olhando para o código acima, irá pensar que coloco duas letras como “H” e “E” e que, no final é imprimido «O seu nome começa com “H” e termina com “E”.» mas não. Não é isso que acontece. Experimente correr o código.

Como deve ter verificado, logo após digitar a primeira letra, o programa terminando deixando o espaço para a segunda letra em branco.

O que aconteceu aqui? Quando nós clicamos no ENTER para submeter a primeira letra, além dessa letra, o ENTER (“\n”) é submetido para a memória e quando aparece um novo pedido para inserir um caractere, o “\n” é automaticamente assumido como esse novo caractere.

Para que isto não aconteça, basta **limpar o buffer** antes de utilizar novamente a função. No Windows utilizamos a função “fflush(stdin)” e no Linux “__fpurge(stdin)” de forma a limpar o buffer do stdin (teclado).

Então, ficaria assim:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      char letra1, letra2;
6.
7.      printf("Insira a primeira letra do seu nome: ");
8.      scanf("%c",&letra1);
9.
10.     fflush(stdin);
11.     __fpurge(stdin);
12.
13.     printf("E agora a última: ");
14.     scanf("%c",&letra2);
15.
16.     printf("O seu nome começa com \"%c\" e termina com \"%c\".", letra1, letra
    2);
17.
18.     return 0;
19. }
```

http://pplware.sapo.pt/tutoriais/vamos-programar-introducao-a-programacao-9