Introdução à Programação #22

Henrique Dias
henrique.dias@pplware.com

28 de janeiro de 2015

O artigo de hoje irá incidir sobre algumas sugestões a referir dadas por leitores no artigo da semana passada, mas também iremos começar a abordar outro tópico.

1 Limpeza do *buffer*

No artigo número nove da presente saga introduzimos as seguintes funções:

```
1 _fpurge();
2 fflush();
```

Ambas as funções permitem-nos limpar o *buffer* de entrada, porém não são muito seguras. Em programação profissional de C (e outras linguagens), estas funções não são utilizadas por motivos de segurança.

Claro que existem "soluções" alternativas. O leitor José Carlos Ferreira, no artigo da semana anterior, sugeriu-nos o seguinte:

```
1 #define CLEAR_INPUT while (!getchar ())
```

O que nos permite utilizar o trecho de código:

```
1 while (!getchar ())
```

Recorrendo apenas ao nome "CLEAR_INPUT".

2 Alternativa a *gets* e a *scanf*

Devido à falta de uma solução efetiva ao problema existente com a função *scanf* e à existência de alguns contra-tempos com a função *gets*) foi sugerido, pelos leitores Zé e lmx a utilização da função *fgets*.

Sintaxe Esta função tem algumas diferenças relativamente às funções mencionadas no título. Ora veja a sintaxe:

```
1 fgets(char *str, int n, FILE *stream);
```

Onde:

- **str** - O apontador para um array de caracteres onde a *string* será armazenada.

- **n** - O número máximo de caracteres a serem lidos (incluindo o delimitador final). Geralmente é igual ao tamanho do array.
- **stream** - O apontador para o ficheiro ou objeto donde serão lidos os caracteres.

Exemplo Vejamos então um exemplo baseado no boletim de informação criado no artigo anterior.

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

int main()
{
    char nome[21],
         apelido[21],
         morada[51],
         codigoPostal[11];

    printf("Por favor insira os seus dados conforme pedido:\n\n");
    printf("Primeiro nome: ");
    fgets(nome, 21, stdin);

    printf("Último nome: ");
    fgets(apelido, 21, stdin);

    printf("Morada: ");
    fgets(morada, 51, stdin);

    printf("Código Postal: ");
    fgets(codigoPostal, 11, stdin);

    printf("\nO seu Cartão de Identificação:\n");
    printf("Nome: %s, %s\n", apelido, nome);
    printf("Morada: %s\n", morada);
    printf("Código Postal: %s\n", codigoPostal);
    return 0;
}
```

2.1 Remoção das Mudanças de Linha

Mas, se corrermos o código acima, iremos receber algo parecido com o seguinte:

```
O seu Cartão de Identificação:
Nome: Apelido
 , Nome

Morada: Morada

Código Postal: CP
```

Como pode verificar, isto mostra que os arrays ficaram, também com os delimitadores da mudança de linha ("\n"). Para removermos estes delimitadores podemos recorrer à função *strtok*.

Sintaxe Ora veja a sintaxe desta função.

```
strtok(char *str, const char *delim);
```

Onde:

- **str** - O apontador para um array de caracteres onde a *string* está armazenada.
- **delim** - O delimitador em questão ou uma constante que o contenha.

Exemplo Então, para que o exemplo acima imprima corretamente o Bilhete de Informação, devem ser adicionadas as seguintes linhas antes de ser imprimida qualquer informação:

```
1 strtok(nome, "\n");
2 strtok(apelido, "\n");
3 strtok(morada, "\n");
4 strtok(codigoPostal, "\n");
```