Recapitulando a aula anterior

Como vimos na aula anterior, aprendemos um pouco sobre como variáveis inteiras(int) e reais(double) funcionam, bem como receber e escrever mensagens na tela do computador.

Implementação

O char, leitura e escrita.

O char. O **char**, assim como **double** e **int** é um tipo de dado em C. Mas diferentemente dos dois que armazenam números, o **char** armazena símbolo(ou desenho).

Tudo que digitamos são símbolos e depende do computador interpretar. Assim, se você digitar o símbolo **1**, pode interpretar como uma unidade ou também como simbolo um. Assim, depende de como você quer que leia. Logo, ao digitar **12**, pode-se interpretar como o número **doze** ou como dois símbolos, **1** e **2**, um seguido do outro.

Sempre que falarmos de algum símbolo, colocaremos então as aspas simples para diferenciar de um número, assim, temos que **'1'** representa o símbolo um enquanto **1** representa o número um.

Para declaração de um tipo **char**, é como mostrado pelo código 10-char.c

```c
/*   Codigo: 10-char.c
      Autor: Carlos Adir
  Descricao: Exemplifica a declaracao de um tipo char
*/
#include <stdio.h>
int main()
{
  char letra = 'a';
  char carac = '1';
  return 0;
}
```

Leitura. Para ler um caracter, utiliza-se o **scanf** mas no lugar de **d**(int) e **lf**(double), utiliza-se **c** tanto no **scanf** quanto **printf**. Assim, o código para ler um **char** é mostrado pelo arquivo 11-leitura_char.c.

```c
  char carac;
  scanf("%c", &carac);
```

Escrita. Para imprimir um caracter, assim como explicado na leitura, utiliza-se o **printf** com o **c** como mostrado no arquivo 12-impressao_char.c.

```c
  char carac = 'a';
  printf("%c\n", carac);
```

Um código para ler e escrever o caracter lido é mostrado por 13-exemplo_char.c.

```c
  char carac;
  printf("Digite um caracter: ");
  scanf("%c", &carac);
  printf("O caracter lido foi: %c\n", carac);
```

Getchar e buffer. Agora que sabemos como ler um caracter, vamos tentar ler dois caracteres! Teste o código 14-dois_char.c para ler **12** e depois **'1'**, pressione enter e **'2'**.

```
1  char c1, c2;
2  printf("Digite dois caracteres: ");
3  scanf("%c", &c1);
4  printf("Primeiro caracter lido: '%c'\n", c1);
5  printf("Digite o outro caracter: ");
6  scanf("%c", &c2);
7  printf("Segundo caracter lido: '%c'\n", c2);
8  printf("Os dois caracteres lidos foram: '%c' e '%c'\n", c1, c2);
```

Se tudo deu certo, no segundo caso não apareceu o **char '2'**. Mas o que aconteceu? Bem, toda vez que digitamos algo, esse algo vai para o **buffer** que é como uma fila de caracteres. Se você digitou **'1'** e **'2'**, então na fila fica **'1"2'**. Mas se você digitou **'1'**, enter(**'\n'**) e **'2'**, então o buffer fica **'1"\n"2'**

Quadro 1 Primeiro Buffer

'1'	'2'	

Quadro 2 Segundo Buffer

'1'	'\n'	'2'	

Assim, o primeiro **char** que está no buffer é o **'1'** em ambos casos e então fica armazenado na variável **c1**. No primeiro caso, como o próximo **char** é **'2'**, então o char **'2'** fica armazenado na variável **c2** enquanto no segundo caso em **c2** fica armazenado o **char '\n'**.
Contudo, precisamos toda vez que apertar o enter, retirar o **'\n'** do buffer e para isso podemos utilizar o comando **getchar** como indica o arquivo 15-usando_getchar.c:

```
1  char c;
2  printf("Digite um caracter: ");
3  scanf("%c", &c);
4  printf("Primeiro caracter lido: '%c'\n", c1);
5  getchar(); /* Retirando o '\n' do buffer */
6  printf("Digite outro caracter: ");
7  scanf("%c", &c);
8  printf("O outro caracter lido: %c", c);
```

String, leitura e escrita. String é um vetor de caracteres. Por questão de tempo, não será explicado vetores e sua utilização com exceção dessa parte de string. String é utilizado para capturar nomes e frases. Para declarar uma string, podemos usar como indica o arquivo 16-string.c

```
1  char palavra[100] = "Bicicleta";
```

Para representar string utilizamos as aspas duplas, como indica o exemplo acima. Para ler e escrever, utilizamos **s** no lugar de **d**, **lf** ou **c** e no scanf não precisamos do **&**. Assim, o exemplo 17-ler_escrever_string.c mostra como ler um nome e escrever.

```
1  char palavra[100];
2  printf("Digite seu nome: ");
3  scanf("%s", palavra);
4  printf("Ola %s, seja bem vindo!", palavra);
```

Mas, se você tentar colocar um nome composto, como *Carlos Adir*, será armazenado somente *Carlos* pois para ler string, sempre que encontrar espaço parará de ler. Para resolver esse problema, utilizamos **%[^ \n]** como indica o arquivo 18-ler_escrever_nome_composto.c.

```
1   char palavra[100];
2   printf("Digite seu nome: ");
3   scanf("%[^\n]", palavra);
4   printf("Ola %s, seja bem vindo!", palavra);
```