




Disciplina:

Programação Computacional

Prof. Fernando Rodrigues
e-mail: fernandorodrigues@sobral.ufc.br

Aula 15: Programação em C

❖ Conceitos Avançados:

- Configurando a localidade;
 - Geração de valores aleatórios;
 - Passar argumentos na linha de comando;
 - Diretivas de compilação;
 - O Padrão C99;
 - Funções de Manipulação de Strings;
 - Exercícios.
- 

Configurando a localidade

- ▶ Usamos a função *setlocale()* da biblioteca *locale.h*:
 - `char* setlocale(int categoria, const char* local);`
- ▶ onde **categoria** é uma das macros definidas para localidade e **local** é o identificador de localidade do sistema especificado.
- ▶ Os valores possíveis de macro para **categoria** são:

Categoria	Descrição
LC_ALL	Afeta todo o local da linguagem.
LC_COLLATE	Afeta as funções strcoll() e strxfrm() que trabalham com strings.
LC_CTYPE	Afeta as funções que manipulam caracteres.
LC_MONETARY	Afeta a informação de formatação monetária.
LC_NUMERIC	Afeta a formatação numérica da localidade C.
LC_TIME	Afeta a função de formatação de data e hora strftime() .

Configurando a localidade

- ▶ Com relação ao **local**, trata-se de um valor específico do sistema. Porém, dois valores sempre existirão:
 - 1º) “C” : para a localidade mínima da linguagem.
 - 2º) “” : localidade-padrão do sistema.
 - Além destes, “ptb” é usado para Português do Brasil.
- ▶ Para configurar a localidade de todo o ambiente da linguagem para Português do Brasil, fazemos:
 - `setlocale(LC_ALL, “ptb”);` ou
 - `setlocale(LC_ALL, “portuguese”);`
- ▶ Lembrando que a biblioteca *locale.h* precisa ser importada: `#include <locale.h>`

Geração de valores (pseudo-)aleatórios

- ▶ Para geração de valores aleatórios (entre 0 e `RAND_MAX`, onde `RAND_MAX` é uma constante definida na biblioteca *stdlib.h*) usamos a função:
 - `int rand(void)`
- ▶ Por exemplo, para a variável inteira 'z' receber um valor aleatório, devemos fazer:
 - `int z = rand();`
- ▶ Para inicializarmos a semente (seed) de geração de números aleatórios, utilizamos a função:
 - `void srand(unsigned int)`
- ▶ Antes da chamada a função *rand()*.

Geração de valores (pseudo-)aleatórios

- ▶ Para que seja gerada uma “boa semente” randômica, a dica é utilizar a função *time(NULL)* (da biblioteca *time.h*) como parâmetro da função *srand()*, da seguinte forma:

- `srand(time(NULL));`

▶ Ex:

```
#include <time.h>
int main(){
    int i, aleatorio[10];
    srand(time(NULL));
    for(i=0 ; i < 10 ; i++)
        aleatorio[i] = rand();
}
```

```
aleatorio[0] = 13056
aleatorio[1] = 11423
aleatorio[2] = 12394
aleatorio[3] = 7366
aleatorio[4] = 21356
aleatorio[5] = 21367
aleatorio[6] = 25043
aleatorio[7] = 23545
aleatorio[8] = 19399
aleatorio[9] = 14287
```

- ▶ Para escolher a faixa de valores, vamos usar operações matemáticas, principalmente o operador de módulo, também conhecido como “resto da divisão”: %

Geração de valores (pseudo-)aleatórios

- ▶ Para fazer com que um número 'x' receba um valor entre 0 e 9, fazemos:
 - $x = \text{rand()} \% 10;$
- ▶ Para fazer com que um número 'x' receba um valor entre 1 e 10, fazemos:
 - $x = 1 + (\text{rand()} \% 10);$
- ▶ Para fazer com que um número 'x' receba um valor entre 0.00 e 1.00, primeiro geramos números inteiros, entre 0 e 100:
 - $x = \text{rand()} \% 101;$
- ▶ Para ter os valores decimais, dividimos por 100:
 - $x = x/100;$

Passando argumentos na linha de comando

- ▶ A função main pode ser definida de tal maneira que o programa receba parâmetros que foram dados na linha de comando do sistema operacional. Para receber esses parâmetros, a função main adquire a seguinte forma:

```
int main(int argc, char *argv[ ]) {  
    //sequência de comandos }
```

- ▶ Agora a função main recebe dois parâmetros de entrada:
 - int argc: trata-se de um valor inteiro que indica o número de parâmetros com os quais a função main foi chamada na linha de comando.

Passando argumentos na linha de comando

- `char *argv[]`: trata-se de um ponteiro para uma matriz de strings. Cada uma das strings contidas nessa matriz é um dos parâmetros com os quais a função `main` foi chamada na linha de comando.
- Observe que o valor de `argc` é sempre maior ou igual a 1. Esse parâmetro vale 1 se o programa foi chamado sem nenhum parâmetro (o nome do programa é contado como argumento da função), e é somado +1 em `argc` para cada parâmetro passado para o programa.
- Ao todo, existem `argc` strings guardadas em `argv`.

Passando argumentos na linha de comando

Exemplo: parâmetros da linha de comando

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(int argc, char* argv[]){
04      if(argc == 1){
05          printf("Nenhum parametro passado para o programa %s\n",argv[0]);
06      }else{
07          int i;
08          printf("Parametros passados para o programa %s:\n",argv[0]);
09          for(i=1; i<argc; i++)
10              printf("Parametro %d: %s\n",i,argv[i]);
11      }
12      system("pause");
13      return 0;
14  }
```