# LINGUAGEM C – Notas de Aula II

**Comandos de repetição**

WHILE:

```
while (condição(ões))
{
  comando(s);
}
```

DO:

```
do
{
  comando(s);
} while (condição(ões));
```

Exercício: colocar o exercício anterior em *loop* até acertar a data

Solução:

```
main()
{
  int d,m,a,ok;
  do
  {
    printf("\nDigite dia: ");

    ...

    if (ok==0) { printf("\nData Invalida. Digite novamente..."); }
  } while (ok==0);
  printf("Data OK!");
  getch();
}
```

FOR:

```
for (expressão_inicial; condição; incremento)
{
  comando(s);
}
```

Exercícios básicos para treino:

Gere 10 números (de 0 a 10) aleatórios

```
#include <stdlib.h>

main()
{
  int x,i;
  randomize();
  clrscr();
  for (i=0;i<=20;i++)
  {
    x=random(11);
    printf("\n%d",x);
```

```
  }
  getch();
}
```

Cálculo do fatorial de 7:

```
main()
{
  int i,res=1;
  clrscr();
  printf("Fatorial de 7\n");
  for (i=2;i<=7;i++)
    res = res * i;
  printf("Resultado: %d",res);
  getch();
}
```

O Fatorial de 8 (no Turbo C) dá um número negativo! O motivo é o estouro do tamanho em bits destinado a um int. A solução é declarar “res” como long int e no printf usar “%ld” para a exibição.

Exercícios mais complexos:

1-Soma dos 10 primeiros termos de uma PA começando com 2, fator 5.

(solução em sala)

2-Escreva um programa que faça o sorteio de cinco dezenas para a Loto
(de 00 a 99, sem repetição)

```
#include <stdlib.h>
main()
{
  int d1,d2,d3,d4,d5;
  clrscr();
  randomize();
  printf("sorteio da loto\n");
  d1 = random(100);
  do d2 = random(100); while (d2==d1);
  do d3 = random(100); while ( (d3==d1) || (d3==d2) );
  do d4 = random(100); while ( (d4==d3) || (d4==d2) || (d4==d1) );
  do d5 = random(100); while ( (d5==d4) || (d5==d3) || (d5==d2) || (d5==d1) );
  printf("dezenas sorteadas: %d-%d-%d-%d-%d",d1,d2,d3,d4,d5);
  getch();
}
```

**Estruturas de Dados Básicas**

Vetor

Declaração:

tipo nome[n]; /* declara um vetor de n elementos (índice de 0 a n-1) */

Uso:

nome[índice]

Exemplo: Dados 10 números inteiros, imprimi-los na ordem inversa a da leitura.

```
#include <stdio.h>
main()
{
    int i, a[10];
    clrscr();
    printf("digite 10 números inteiros:\n");
    for (i=0; i<10; i++)
      scanf("%d",&a[i]);
    printf("em ordem inversa:\n");
    for (i=9; i>=0; i--)
      printf("%d\n",a[i]);
}
```

"strings" em C

Via de regra, strings são cadeias de caracteres, portanto podem ser entendidas como vetores de caracteres...

Exemplo:

```
char nome[40];
```

A linha acima declara um vetor de caracteres chamado "nome". Podemos usar esse vetor como um string:

```
char nome[40]="Joao da Silva";
```

ATENCAO! O trecho abaixo não funciona!

```
char nome[40];
nome = "joao da silva"; /* não funciona! */
```

Para atribuir após declarar, podemos usar:

```
char nome[40];
strcpy(nome,"joao da silva");
/* aí funciona (talvez seja preciso #include <string.h> */
```

Para ler do teclado:

```
scanf("%s",nome);
/* não utilizar '&' na leitura de strings, pois o nome do vetor já define o endereço dele */
```

O problema com o scanf acima é que ele lê apenas o primeiro nome (não armazena espaços em branco). Uma das soluções possíveis é usar outra função de leitura: o gets, que lê uma string do teclado:

```
gets(nome);
```

Para escrever strings, também existe o puts:

```
puts(nome);
```

Os "gets" e "puts" existem para outros tipos, como getd(i), por exemplo, para ler inteiros

Matriz:

É um vetor de vetor. Exemplo: int m[10][3];

Struct (registro):

```
struct nome_opcional
{
  tipo campo;
} variavel_opcional;
```

se usar nome (r, por exemplo), para declarar:

```
      struct r r1;
```

Exercícios mais complexos:

1-Pedir 10 números inteiros ao usuário e escrevê-los na tela ordenados.

```
main()
{
  int i,num[10],x,ordena;
  clrscr();
  printf("digite 10 números:\n");
  for (i=0;i<10;i++) scanf("%d",&num[i]);

  ordena=1; /* flag de ordenar verdadeira */
  while (ordena==1)
  {
    ordena=0; /* flag de ordenar falsa */
    for (i=0;i<9;i++) /* cuidado para não colocar i<10... */
    {
      if (num[i]>num[i+1])
      {
        x=num[i];
        num[i]=num[i+1];
        num[i+1]=x;
        ordena=1; /* continuar ordenando */
      }
    }
  }

  printf("ordenados:\n");
  for (i=0;i<10;i++) printf("%d-",num[i]);
  getch();
}
```

Sortear 100 números de 0 a 99. Ver maior, menor, média e no. de maior incidência:

```
#include <stdlib.h>

main()
{
  int i,n,maior=0,menor=99,inc[100],mi=0;
  long int soma=0;
  clrscr();
  for (i=0;i<100;i++) inc[i]=0;
  randomize();
  for (i=0;i<1000;i++)
  {
    n=random(100);
    if (n>maior) maior=n;
    if (n<menor) menor=n;
    soma=soma+n;
    inc[n]++;
  }
  printf("maior numero sorteado: %d\n",maior);
  printf("menor numero sorteado: %d\n",menor);
  printf("media dos numeros sorteados: %f\n",soma/1000.0);
  for (i=0;i<100;i++) if (inc[i]>mi) mi=inc[i];
  printf("numeros de maior incidencia: %d vezes sortados\n",mi);
  for (i=0;i<100;i++) if (inc[i]==mi) printf("%d-",i);
  getch();
}
```

**Funções, Recursividade**

```
tipo nome(parâmetros)
{
  ...
  return(resultado);
}
```

Lembrete: o tipo do resultado deve “bater” com o tipo da função.

Exemplos:

Função de Quadrado:

```
int quadrado(int base)
{
  return (base*base);
}

main()
{
  clrscr();
  printf("O quadrado de 12 é = %d \n",quadrado(12));
  getch();
}
```

Refazendo o cálculo do fatorial de 8, usando recursividade:

```
long int fat(int p)
{
  if (p==0)
    return(1);
  else
    return (p*fat(p-1));
}

main()
{
  clrscr();
  printf("Fatorial de 8 = %ld \n",fat(8));
  getch();
}
```