Algoritmos e Estruturas de Dados I (DCC/003)

Aula Prática 03

- **Instruções:**
  - Os exercícios deverão ser feitos em aula de laboratório durante o tempo da aula;
  - O professor irá esclarecer dúvidas em aula;
  - Crie uma pasta com seu nome e vá gravando seus programas implementados.

Exercício 1

Escreva um programa que leia um número positivo N e imprima N linhas do triângulo de Floyd:

1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21

No exemplo acima o usuário digitou N = 6. Seu programa deve verificar, antes de executar, se o usuário digitou um número válido. Caso ele não tenha digitado um número válido, exiba novamente a opção para digitar um número até que um número válido seja lido. [salve o seu código com o nome: **ap03-ex1.c**]

Exercício 2

Escreva um programa em C para cálculo do máximo divisor comum (MDC) entre dois números.

O programa é simples, deve seguir o seguinte algoritmo:

```
1- Receba 2 valores inteiros A e B (use scanf);
2- Enquanto B for diferente de zero:
   {
        3- resto = resto da divisão de A por B;
        4- A = B;
        5- B = resto;
   }
6- resultadoMDC = valor que fica em A ao sair do loop (estrutura de repetição);
```

Ao final imprima o valor do resultado do MDC.

Execute o código para testar seu programa de computador e verificar se o resultado está correto. [salve o seu código com o nome: **ap03ex2.c**]

Exercício 3

Vamos agora fixar o conceito de função, que é o seguinte: um código que é escrito para realizar alguma ação específica e retornar o resultado.

Vamos aproveitar o código do exercício anterior e vamos agora aprender esse novo conceito.

Faça o seguinte então:

1) Declare uma função chamada MDC. Isso deverá ser feito após os *includes* de seu código:

```
int MDC (int A, int B){

}
```

2) Agora dentro do código desta função MDC que você criou, escreva o código para calcular o MDC (idêntico ao que você fez no número 1) – parte do **while**;
3) Vai ser necessário agora declarar dentro da função as variáveis que você vai usar e que não foram recebidas como parâmetro (que serão **resto** e **resultadoMDC**);
4) Depois de sair do **while** você vai colocar em o resultado do MDC numa variável (que chamamos aí de **resultadoMDC**) e dar o seguinte comando que vai retornar o valor do resultado:
return (resultadoMDC);

Você acabou de criar uma função que recebe dois valores inteiros, faz o cálculo do MDC e depois retorna o resultado.

Então agora vamos usá-la na continuação do exercício.

Exercício 3 (continuação)

Seguindo o que você fez no exercício anterior, crie um código principal (**main**), que vai:

- declarar as variáveis que serão necessárias (dois números inteiros num1 e num2) e também uma variável para guardar o resultado do MDC (resultado);

- Leia os 2 valores de num1 e num2 do teclado (scanf);

- Chame a função que você criou e guarde o valor do resultado dela na variável resultado:

```
Resultado = MDC (num1, num2);
```

- Imprima o resultado;

Simples, não? Você acabou de calcular o MDC usando a função que criou.

Execute o código para testar seu programa de computador e verificar se o resultado está correto. [salve o seu código com o nome: **ap03-ex3.c**]

Exercício 4

Por fim, agora vamos criar um novo código **ap03-ex4.c**, que vai fazer algo a mais no código anterior, que é o seguinte:
- Ficar executando até que o usuário resolva sair do programa. Como faremos isso? Pense aí um pouco a respeito!

Bem, pode ser feito algo assim:
- Acrescente um loop (estrutura de repetição) que fique executando até que o usuário não queira mais continuar, ou seja:

```
do{
        - seu código anterior que calcula MDC usando a função (copie ele aqui);
 }
 while ((c == 'S') || (c == 's'));
```

Para que possamos ver se o usuário quer ou não continuar, façamos o seguinte: vamos pedir a ele para digitar a tecla “s” se quiser continuar.

```
printf("Continua? (S/N) ");
c = getche();
printf("\n");
```

Isso vai ser colocado no fim do seu comando do-while para receber o valor do usuário. Apenas para esclarecer, a função **getche()** (no Windows) ou **getchar()** (no Linux ou Mac) lê um caracter resultante de um evento de teclado. Pode-se usar antes um comando para sincronização do teclado (entrada de dados): fflush(stdin) ou fpurge(stdin) ou scanf(“ %c”,).