Quarta Lista de Exercícios – IPC

Prof. Bruno A. N. Travençolo – FACOM-UFU

1) Faça um algoritmo utilizando o comando **while** que mostra uma contagem regressiva na tela, iniciando em 10 e terminando em 0. Mostrar uma mensagem "FIM!" após a contagem.

   Exemplo de saída:

```
<< Contagem regressiva >>
10.. 9.. 8.. 7.. 6.. 5.. 4.. 3.. 2.. 1.. 0.. FIM!
```

2) Refaça o exercício anterior utilizando o comando **do .. while;**
3) Refaça o exercício anterior utilizando o comando **for**
4) Faça um algoritmo utilizando o comando **while** que mostra uma contagem regressiva na tela, iniciando em um valor que o usuário determina e terminando em 0. Mostrar uma mensagem "FIM!" após a contagem.

   Exemplo de saída:

```
<< Contagem regressiva >>
Qual é o número inicial? 12
12.. 11.. 10.. 9.. 8.. 7.. 6.. 5.. 4.. 3.. 2.. 1.. 0.. FIM!
```

5) Refaça o exercício anterior utilizando o comando **do .. while;**
6) Refaça o exercício anterior utilizando o comando **for**

7) Faça um algoritmo utilizando o comando **while** que mostra uma contagem progressiva na tela, iniciando em um valor que o usuário determina e terminando em 0. Mostrar uma mensagem "FIM!" após a contagem.

   Exemplo de saída:

```
<< Contagem progressiva até 0 >>
Qual é o número inicial? -5
-5.. -4.. -3.. -2.. -1.. 0.. FIM!
```

8) Refaça o exercício anterior utilizando o comando **do .. while;** Discuta por que o do-while não é a melhor estrutura para resolver este exercício.
9) Refaça o exercício anterior utilizando o comando **for**

10) Dado um número inteiro positivo *n*, calcular a soma dos *n* primeiros números naturais (faça a soma utilizando **while** .

    Exemplo de saída:

```
<< Soma de n valores naturais >>
Quantos números deseja somar? 5
A soma dos 5 primeiros números naturais é : 15
```

11) Refaça o exercício anterior utilizando o comando **do .. while**;
12) Refaça o exercício anterior utilizando o comando **for**

13) Dados n e dois números inteiros positivos i e j diferentes de 0, imprimir em ordem crescente os n primeiros naturais que são múltiplos de i ou de j e ou de ambos.
Exemplo: Para n = 6 , i = 2 e j = 3 a saída deverá ser : 0,2,3,4,6,8.

Exemplo de saída:

```
<< Múltiplos >>
Entre com o valor de n: 6
Entre com o valor de i: 2
Entre com o valor de j: 3
Os múltiplos de i ou j são : 0,2,3,4,6,8
```

14) Dado um inteiro positivo *p*, verificar se *p* é primo.
Exemplo de saída:

```
<< Números primos >>
Entre com o valor: 15
O número 15 não é primo
```

Exemplo de saída:

```
<< Números primos >>
Entre com o valor: 7
O número 7 é primo
```

15) Faça um conversor binário – decimal. O usuário deve digitar o número de bits do número binário e um bit deve ser lido por vez

Exemplo de saída:

```
<< Conversor binário-decimal >>
Entre com o número de bits: 3
Digite o bit # 1: 1
Digite o bit # 2: 0
Digite o bit # 3: 0
O número binário 100 em decimal é 4
```

dica para juntar os bits para mostrar na tela: 101 = 1(10^2) + 0*(10^1) + 1*(10^0)

16) Faça um algoritmo que mostre qual a quantidade máxima de endereços de memória que um computador com 1, 2, 4, 8, 16, 32, 64 e 128 bits pode representar. O programa deve conter um loop e uma variável. Use a função pow()

pow()
Syntax:
#include <math.h>
double pow(double base, double exp);

Description:
The pow() function returns base raised to the exp power. There's a domain error if base is zero and exp is less than or equal to zero. There's also a domain error if base is negative and exp is not an integer. There's a range error if there's an overflow.

Observe que esta função retorna um tipo double (use %f no printf).

Exemplo de saída:

```
Com 2 bits é possível endereçar 4 posições de memória
Com 4 bits é possível endereçar 16 posições de memória
Com 8 bits é possível endereçar 256 posições de memória
Com 16 bits é possível endereçar 65536 posições de memória
Com 32 bits é possível endereçar 4294967296 posições de memória
Com 64 bits é possível endereçar 1.84467440737096E19 posições de memória
Com 128 bits é possível endereçar 3.40282366920938E38 posições de memória
```

17) Faça um programa que seja semelhante ao jogo de forca, mas com uma única letra. A letra que o usuário deve adivinhar deve ser definida no código do programa. O usuário tem 5 chances de acertar a letra. O programa finaliza sua execução quando o usuário acerta a letra ou quando acabam suas chances

Exemplo de saída:

```
<<Forca de uma letra só>>
Qual a letra? o
Errado! Você tem mais  4 chances

Qual a letra? d
Errado! Você tem mais  3 chances

Qual a letra? w
Errado! Você tem mais  2 chances

Qual a letra? q
ACERTOU!
```

Exemplo de saída 2:

```
<<Forca de uma letra só>>
Qual a letra? d
Errado! Você tem mais  4 chances

Qual a letra? g
Errado! Você tem mais  3 chances
```

```
Qual a letra? k
Errado! Você tem mais  2 chances

Qual a letra? r
Última chance!!!

Qual a letra? o
Acabaram suas chances. A letra correta é 'q'
```

18) Faça um algoritmo que converta uma velocidade expressa em km/h para m/s e vice versa. Você deve criar um menu com as duas opções de conversão e com uma opção para finalizar o programa. O usuário poderá fazer quantas conversões desejar, sendo que o programa só será finalizado quando a opção de finalizar for escolhida (no caso, caso ele escolha a opção 'q')

Exemplo de saída:

```
<< Conversor >>
 Digite uma opção
   1 - para converter de km/h para m/s
   2 - para converter de m/s para hm/h
   q - para sair
1
Digite a velocidade (km/h): 100
A velocidade em m/s é 27.7777777777778
 << Conversor >>
 Digite uma opção
   1 - para converter de km/h para m/s
   2 - para converter de m/s para hm/h
   q - para sair
2
Digite a velocidade (m/s): 30
A velocidade em km/h é 108
 << Conversor >>
 Digite uma opção
   1 - para converter de km/h para m/s
   2 - para converter de m/s para hm/h
   q - para sair
q
```

19) José possui no banco R$50.000,00 e Carlos R$ 25.000,00. Carlos poupa R$ 3.000,00 por mês, enquanto que o José poupa R$800,00. Quantos anos levarão para que o Carlos fique com mais dinheiro que o José? Faça um algoritmo para mostrar esse cálculo.

20) Escreva um programa que leia um numero inteiro positivo **n** e em seguida imprima n linhas do chamado Triangulo de Floyd:
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21

21) Faça um programa que mostre o valor de 1!, 2!, 3!, até 10!. Utilize somente variáveis inteiras

Exemplo de saída:

```
<< Fatorial >>
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
```

22) Altere o programa anterior para calcular o10! até 15!. É esperado que ocorra um erro. Explique o motivo do erro e qual a solução.

Exemplo de saída:

```
<< Fatorial >>
11 ! = 39 916 800
12 ! = 479 001 600
13 ! = 6 227 020 800
14 ! = 87 178 291 200
15 ! = 1 307 674 368 000
```

23) Calcule o número neperiano usando uma série (ver abaixo). Peça para o usuário entrar com o número de termos que serão somados (note que, quanto maior o número, a resposta estará mais próxima do valor *e*)

O número e pode ser representado e calculado por meio da utilização da série de Taylor para e^x quando x=1, como a soma da seguinte série infinita:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots$$

Aqui $n!$ representa o fatorial de n.

(fonte: Wikipédia)

Exemplo de saída:

```
<< Número neperiano >>
Entre com o número de termos: 100
e ~  2.71828182845905
```