FEELT31201 - Programação Procedimental - 2022/1

Laboratório 01

Prazo: 04/11 - 23:59h

Básico 1

1.25 pontos

- Nome do arquivo com código-fonte: **`aritmInt.c`**
- Tarefa a cumprir:

  Peça ao usuário por dois números inteiros. Mostre o resultado da soma (+), subtração (-), multiplicação (*), divisão (/) e módulo (%) entre eles.
- Exemplo: se o usuário informar 42 e 34, teremos como resposta: `42+34 = 76`, `42-34 = 8`, `42*34 = 1428`, `42/34 = 1` e `42%34 = 8`.
- Método sugerido para entrada de dados pelo usuário:

```
// mensagem para o usuário
scanf("%d %d", &primeiroInt, &segundoInt);
```

- Casos de teste:

Entrada		Saída esperada contém
42	34	`42+34 = 76, 42-34 = 8, 42*34 = 1428, 42/34 = 1, 42%34 = 8`
34	42	`34+42 = 76, 34-42 = -8, 34*42 = 1428, 34/42 = 0, 34%42 = 34`
-231	25	`-231+25 = -206, -231-25 = -256, -231*25 = -5775, -231/25 = -9, -231%25 = -6`
-1	-25	`-1+-25 = -26, -1-25 = 24, -1*-25 = 25, -1/-25 = 0, -1%-25 = -1`

- *Dica(s):* Você pode usar o especificador `%%` para imprimir o caractere `%` dentro da *string* de um `printf`.

Básico 2

1.25 pontos

- Nome do arquivo com código-fonte: **`somaGauss.c`**
- Tarefa a cumprir:

  Peça um número inteiro n para o usuário que seja maior do que 1. Calcule a soma de Gauss desse número, ou seja, a soma da sequência de números de 1 a n.
- Exemplo: se o usuário informar 5, a resposta $(1 + 2 + 3 + 4 + 5)$ será `15`.
- Método sugerido para entrada de dados pelo usuário:

```
// mensagem para o usuário
scanf("%d", &numero);
```

- Casos de teste:

Entrada	**Saída esperada contém**
5	`15`
15	`120`
18	`171`
42	`903`

- *Dica(s):* Existe uma fórmula para a soma de Gauss que você pode pesquisar sobre e usá-la, ou você pode resolver por "força bruta" usando apenas um laço. Se desejar, implemente as duas estratégias e verifique se o resultado é o mesmo.

Básico 3

1.25 pontos

- Nome do arquivo com código-fonte: `fatorial.c`
- Tarefa a cumprir:

  Peça um número inteiro para o usuário que seja maior do que 1. Calcule o fatorial desse número. Um número n tem seu fatorial $n!$ definido como:

$$n! = (n) = \prod_{i=1}^{n} i$$

- Exemplo: se o usuário informar 5, a resposta $(1 \cdot 2 \cdot 3 \cdot 4 \cdot 5)$ será `5! = 120`.
- Método sugerido para entrada de dados pelo usuário:

```
// mensagem para o usuário
scanf("%d", &numero);
```

- Casos de teste:

Entrada	Saída esperada contém
5	`5! = 120`
12	`12! = 479001600`
13	`13! = 6227020800`
20	`20! = 2432902008176640000`

- *Dica(s):* Por que podemos calcular com `int` o fatorial de 12, mas não o de 13? Na biblioteca `stdint.h` temos o tipo de dado `int64_t`; qual seria o maior número que poderíamos calcular seu fatorial usando esse tipo de dado? (como ref. 21! = 51090942171709440000)

Básico 4

1.25 pontos

- Nome do arquivo com código-fonte: `imc.c`
- Tarefa a cumprir:

  Pergunte ao usuário qual seu peso em [kg] e qual sua altura em [m]. Calcule o IMC baseado na fórmula:

  $$IMC = \frac{peso}{altura^2}$$

  Informe também o resultado como descrito na tabela a seguir:

faixa, menor IMC (incluso)	**faixa, maior IMC (excluso)**	**Resultado**
–	16	Perigo de vida
16	17	Muito abaixo do peso
17	18,5	Abaixo do peso
18,5	25	Peso normal
25	30	Acima do peso
30	35	Obesidade grau I
35	40	Obesidade grau II
40	–	Obesidade grau III

- Exemplo: Para um peso de 60,0kg e uma altura de 1,60m, temos um `IMC = 23,44` que equivale a `Peso normal`.
- Método sugerido para entrada de dados pelo usuário:

```
// mensagem para o usuário
scanf("%f", &peso);
// mensagem para o usuário
scanf("%f", &altura);
```

- Casos de teste:

Entrada	**Saída esperada contém**
60 1.6	`23.44 (Peso normal)`
103 1.6	`40.23 (Obesidade grau III)`
63 1.97	`16.23 (Muito abaixo do peso)`
90 1.8	`27.78 (Acima do peso)`
103 1.82	`31.10 (Obesidade grau I)`

- *Dica(s):* Não precisa usar a biblioteca `math.h`, uma vez que $x^2 = x \cdot x$. Para guiar condicionais, use faixas de valores com o operador booleano "E" para definir o resultado; ex. `x >= 1 && x < 4` significa um número na faixa entre 1 (incluso) e 4 (excluso).

Médio 1

1.5 pontos

- Nome do arquivo com código-fonte: **`somaDivisores.c`**
- Tarefa a cumprir:

  Dado um inteiro positivo informado pelo usuário, calcule a **soma dos divisores positivos** $\sigma_1(\cdot)$ desse número:

  $$\sigma_1(n) = \sum_{d|n} d$$

  onde $d \mid n$ é uma abreviação de "d divide n".
- Exemplo: se o usuário entrar com 12, todos os divisores para ele são 1, 2, 3, 4, 6 e 12. Portanto, $\sigma_1(12) = 28$, como a soma de todos os divisores.
- Método sugerido para entrada de dados pelo usuário:

```
// mensagem para o usuário
scanf("%d", &n);
```

- Casos de teste:

Entrada	Saída esperada contém
12	28
64	127
121	133
331	332

- **Dica(s):** `https://en.wikipedia.org/wiki/Divisor_function`; pense em um laço de 1 a n e, para testar se um número x divide o outro y, basta verificar se é verdade que `y % x == 0`.

Médio 2

1.5 pontos

- Nome do arquivo com código-fonte: `fibonacci.c`
- Tarefa a cumprir:

  A sequência de Fibonacci tem aplicações na análise de mercados financeiros, na ciência da computação e na teoria dos jogos. Também aparece em configurações biológicas, como, por exemplo, na disposição dos galhos das árvores ou das folhas em uma haste, no arranjo do cone da alcachofra, do abacaxi, ou no desenrolar da samambaia. Os números de Fibonacci são, portanto, os números que compõem a seguinte sequência (sequência A000045 na OEIS): 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, ...

  Peça para o usuário entrar com a posição do máximo termo desejado na sequência e imprima a sequência até a posição informada (para efeitos do algoritmo a seguir, considere '1' a posição do termo '0'), terminado com "...". Mas a tarefa aqui não é fazer usando qualquer estratégia, o desafio aqui é converter o algoritmo a seguir em um laço do tipo `for` com uma única instrução `printf("%d, ", i;)` em seu corpo, ou seja, colocar o algoritmo para funcionar **dentro** dos argumentos do comando `for`.

```
Algoritmo fib(n):
    j ← 1
    i ← 0
    para k de 1 até n faça:
        t ← i+j
        i ← j
        j ← t
    reposta em i
```

- Exemplo: colocando 7, temos `0, 1, 1, 2, 3, 5, 8, ...`
- Método sugerido para entrada de dados pelo usuário:

```
// mensagem para o usuário
scanf("%d", &n);
```

- Casos de teste:

Entrada	Saída esperada contém
1	`0, ...`
10	`0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...`
20	`0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, ...`

- **Dica(s):** `https://pt.wikipedia.org/wiki/Sequ%C3%AAncia_de_Fibonacci`; o comando `for` é bastante versátil como, por exemplo, em
  `for(int i=0, j=10; i*j != 25; i++, j-=1) printf("%d, %d | ", i, j);`
  que imprime `0, 10 | 1, 9 | 2, 8 | 3, 7 | 4, 6 |`.

Difícil

2.0 points

- Nome do arquivo com código-fonte: `cosTaylor.c`
- Tarefa a cumprir:

  Peça ao usuário os números `int` n e `double` x. O último deve ser considerado como um fator para π, ou seja, a entrada deve ser considerada como $x\,\pi$. Calcule a série de Taylor truncada para $\cos(x\,\pi)$:

  $$\cos(x\,\pi) = \sum_{i=0}^{n} (-1)^i \cdot \frac{(x\,\pi)^{2i}}{(2i)!}$$

- Exemplo: para $n = 2$, a série truncada é:

  $$\cos(x\,\pi) = \sum_{i=0}^{2} (-1)^i \cdot \frac{(x\,\pi)^{2i}}{(2i)!} = 1 - \frac{(x\,\pi)^2}{2!} + \frac{(x\,\pi)^4}{4!}$$

  e ao calcular $x = 0,25$ (processado como $\pi/4$), a resposta é $\approx 0,707429$.

- Método sugerido para entrada de dados pelo usuário:

```
// mensagem para o usuário
scanf("%d", &int_var);
// mensagem para o usuário
scanf("%f", &float_var);
```

- Casos de teste:

Entrada	Saída esperada contém
1 0.0	1.00000
2 0.25	≈ 0.70743
3 0.25	≈ 0.70710
7 0.667	≈ -0.50091

- **Dica(s):** `https://en.wikipedia.org/wiki/Taylor_series`; use o código que você fez no Básico 3 (`fatorial.c`) para ajudar aqui, preferencialmente encapsulado em uma função *fatorial* definida para isso. Tente apresentar os resultados com 5 casas decimais. Você consegue imaginar como faria para que essa série se tornasse "infinita" dentro do computador, ou seja, independente de n?