UFU Universidade Federal de Uberlândia

FEELT

FEELT31201 - Programação Procedimental - 2022/1

Laboratório 05

Prazo: 08/01 - 23:59h

Básico 1

1.25 pontos

- Nome do arquivo para o código-fonte: `mdc.c`
- Tarefa a cumprir:

  Implemente o algoritmo de cálculo do *maior divisor comum* (mdc):

  $$mdc(a,b) = \begin{cases} a & \text{se } b = 0 \\ mdc(b, a \bmod b) & \text{caso contrário} \end{cases}$$

  O máximo divisor comum entre dois ou mais números é o maior número que é fator de tais números. Por exemplo, os divisores de 12 (1, 2, 3, 4, 6, 12) e de 18 (1, 2, 3, 6, 9, 18) tem em comum os divisores 1, 2, 3 e 6; logo, o $mdc(12, 18) = 6$.

  Peça ao usuário para inserir dois números. Calcule o mdc entre eles.
- Exemplo: O usuário entra com os valores 42 e 24. O mdc entre eles é 6.
- Método de entrada do usuário:

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

- Casos de teste:

Entrada do usuário	A saída esperada contém
12 18	6
42 24	6
1069 1223	1
62 155	31
1302 5082	42

- *Dicas:* O algoritmo apresentado é recursivo.

Básico 2

1.25 pontos

- Nome do arquivo para o código-fonte: `fib.c`
- Tarefa a cumprir:

  Implemente o algoritmo para calcular a *sequência de Fibonacci*:

$$fib(n) = \begin{cases} 0 & \text{se } n = 0 \\ 1 & \text{se } n = 1 \text{ ou } n = 2 \\ fib(n-1) + fib(n-2) & \text{caso contrário} \end{cases}$$

  A sequência começa com 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181...

  Peça ao usuário um índice da sequência e apresente o número de Fibonacci equivalente na sequência.

- Exemplo: se o usuário informar 15, o programa apresenta 610.
- Método de entrada do usuário:

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

- Casos de teste:

Entrada do usuário	A saída esperada contém
15	`610`
0	`0`
2	`1`
19	`4181`
21	`10946`

- *Dicas:* O algoritmo apresentado é recursivo.

Básico 3

1.25 pontos

- Nome do arquivo para o código-fonte: `altura.c`
- Tarefa a cumprir:

  Implemente uma `struct` que represente uma medida de comprimento, com valores inteiros para `metros` e `centimetros`. Peça para o usuário informar uma altura em metros, receba as informações e separe metros e centímetros na nova estrutura. Apresente a altura na forma `[metros]m[centimetros]`.
- Exemplo: se o usuário informar `1.71`, seu programa apresenta `Altura 1m71`.
- Método de entrada do usuário:

```
// mensagem para o usuário
scanf("%d.%d", &altura.metros, &altura.centimetros);
```

- Casos de teste:

Entrada do usuário	A saída esperada contém
1.71	Altura 1m71
0.93	Altura 0m93
2.1	Altura 2m10
1.8752	Altura 1m87

- *Dicas:* Note que a seção *método de entrada do usuário* usa a especificação do `scanf` para dividir a informação de entrada a partir de uma entrada com ponto flutuante. Para armazenar a informação dos centímetros, use uma estratégia para garantir dois dígitos de maneira coerente; por exemplo, a seguinte função recursiva:

```
int cm(int x) {
    if(x < 10) return x*10;
    if(x > 99) return cm(x/10);
    return x;
}
```

  com chamada para garantir a coerência:

```
altura.centimetros = cm(altura.centimetros);
```

Básico 4

1.25 pontos

- Nome do arquivo para o código-fonte: `datas.c`
- Tarefa a cumprir:

  Implemente uma `struct` que represente datas (inteiros para dia, mês e ano). Peça para o usuário uma data no formato `D/M/YYYY` (Dia/Mês/Ano sem zeros à esquerda). Apresente para o usuário a mesma data nos formatos:

  - `DD.MM.yyyy` (Dia.Mês.Ano com zeros à esquerda)
  - `MM-DD-YYYY` (Mês-Dia-Ano com zeros à esquerda)
  - `DD/MM/YY` (Dia/Mês/Ano com zeros à esquerda e ano com dois últimos dígitos)
  - `DDMMMYYYY` (DiaMêsAno com zeros à esquerda e mês de três letras)

- Exemplo: o usuário informa `3/10/2022` e o programa retorna `03.10.2022`, `10-03-2022`, `03/10/22` e `03OUT2022`.
- Método de entrada do usuário:

```
 // mensagem para o usuário
scanf("%d/%d/%d", &data.dia, &data.mes, &data.ano);
```

- Casos de teste:

Entrada do usuário	A saída esperada contém
3/10/2022	03.10.2022 10-03-2022 03/10/22 03OUT2022
1/4/1984	01.04.1984 04-01-1984 01/04/84 01ABR1984
13/9/2013	13.09.2013 09-13-2013 13/09/13 13SET2013
1/1/2001	01.01.2001 01-01-2001 01/01/01 01JAN2001
31/12/1999	31.12.1999 12-31-1999 31/12/99 31DEZ1999

- *Dicas:* Note que a seção *método de entrada do usuário* usa a especificação do `scanf` para dividir a informação de entrada a partir de uma entrada com separação por barras. O especificador do `scanf` para mostrar zeros à esquerda é `%02d`. Para meses com três letras, você pode usar o comando `switch`, ou a seguinte estratégia:

```
const char strmes[13][4] = {
    "", "JAN", "FEV", "MAR", "ABR", "MAI", "JUN",
    "JUL", "AGO", "SET", "OUT", "NOV", "DEZ"
};
```

  Note que `strmes[0]` está vazio apenas para facilitar o acesso correto ao mês apropriado (por exemplo, `strmes[1]` é a string `JAN`).

Médio 1

1.5 pontos

- Nome do arquivo para o código-fonte: **`aproxpi.c`**
- Tarefa a cumprir:

  Leibniz em 1682 desenvolveu a seguinte série para calcular aproximações para o número irracional π:

  $$S_n = \sum_{i=0}^{n} 4\,\frac{(-1)^i}{2i+1}$$

  A idéia é que teríamos $S_n \to \pi$ quando $n \to \infty$. O algoritmo para o cálculo dessa série é:

  $$S(n) = \begin{cases} 4 & \text{se } n = 0 \\ S(n-1) + 4\,\frac{(-1)^n}{2n+1} & \text{caso contrário} \end{cases}$$

  Peça ao usuário o limite superior n da série $(0 \ldots n)$. O programa deve mostrar o resultado da série que é a aproximação de π referente a esse limite superior.
- Exemplo: o usuário informa $n = 1000$ e o programa retorna a aproximação $\pi \approx 3.142592$.
- Método de entrada do usuário:

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

- Casos de teste:

Entrada do usuário	A saída esperada contém
1000	`3.142592`
1001	`3.140595`
100	`3.151493`
29	`3.108269`
233333	`3.141588`

- *Dicas:* Implemente o algoritmo recursivo apresentado.

Médio 2

1.5 pontos

- Nome do arquivo para o código-fonte: `racionais.c`
- Tarefa a cumprir:

  Implemente uma nova `struct` para **números racionais** com a capacidade de **adicionar**, **subtrair**, **multiplicar** e **dividir**. Em matemática, um número racional é um número que pode ser expresso como o quociente ou fração de dois números inteiros, um **numerador** (p) e um **denominador** diferente de zero (q). Peça ao usuário dois números racionais e mostre esses números após a simplificação e os resultados de sua adição, subtração, multiplicação e divisão. As regras que você deve implementar:

  - Os números racionais devem ser simplificados sempre que forem modificados. Isso significa que você deve implementar o algoritmo *maior divisor comum* (mdc):

    $$mdc(a,b) = \begin{cases} a & \text{if} \quad b = 0 \\ mdc(b, a \bmod b) & \text{caso contrário} \end{cases}$$

    e uma função simplificada para transformá-lo como $Q = \dfrac{p}{q} = \dfrac{p/\text{mdc}(p,q)}{q/\text{mdc}(p,q)}$;
  - Quando o numerador for 0 (*zero*), o denominador deverá ser 1; também, se o denominador for *negativo*, você deve alterar o sinal do numerador e do denominador (se $q < 0$ então p agora é $-p$ e q agora é $-q$) ;
  - Adição e subtração são realizadas como: $Q_1 \pm Q_2 = \dfrac{p_1}{q_1} + \dfrac{p_2}{q_2} = \dfrac{p_1 \cdot q_2 \pm p_2 \ cdot q_1}{q_1 \cdot q_2}$;
  - A multiplicação é realizada como: $Q_1 \cdot Q_2 = \dfrac{p_1}{q_1} \cdot \dfrac{p_2}{q_2} = \dfrac{p_1 \cdot p_2}{q_1 \cdot q_2}$;
  - A divisão é realizada como: $Q_1/Q_2 = \dfrac{p_1}{q_1} / \dfrac{p_2}{q_2} = \dfrac{p_1 \cdot q_2}{q_1 \cdot p_2}$;
  - Não se esqueça de simplificar os resultados após cada operação.

- Exemplo: o usuário informa dois números racionais $\frac{3}{-6}$ ($mdc = 3$) e $\frac{525}{75}$ ($mdc = 75$). Em seguida, o programa os simplifica para $-\frac{1}{2}$ e $\frac{7}{1}$, respectivamente, e mostra esses números ao usuário junto com os resultados de:
  - Adição: $-\frac{1}{2} + \frac{7}{1} = \frac{-1 \cdot 1 + 7 \cdot 2}{2 \cdot 1} = \frac{13}{2}$;
  - Subtração: $-\frac{1}{2} - \frac{7}{1} = \frac{-1 \cdot 1 - 7 \cdot 2}{2 \cdot 1} = -\frac{15}{2}$;
  - Multiplicação: $-\frac{1}{2} \cdot \frac{7}{1} = -\frac{1 \cdot 7}{2 \cdot 1} = -\frac{7}{2}$;
  - Divisão: $-\frac{1}{2} / \frac{7}{1} = -\frac{1 \cdot 1}{2 \cdot 7} = -\frac{1}{14}$;
- Método de entrada do usuário:

```
// mensagem para o usuário
// qualquer estratégia de captura de racional1
// mensagem para o usuário
// qualquer estratégia de captura de racional2
```

- Casos de teste:

Entrada do usuário	**A saída esperada contém**
3 -6 525 75	`-1/2  7/1  13/2  -15/2  - 7/2  -1/14`
2 8 -3 12	`1/4  -1/4  0/1  1/2  -1 /16  -1/1`
13 11 21 77	`13/11  3/11  16/11  10/11  39/121  13/3`
0 42 42 -1	`0/1  -42/1  -42/1  42/1  0 /1  0/1`
7 1 5 1	`7/1  5/1  12/1  2/1  35/1  7/5`

- *Dicas:* Leia atentamente as instruções e veja o exemplo para entender melhor.

Difícil

2.0 points

- Nome do arquivo para o código-fonte: `cadastro.c`
- Tarefa a cumprir:

  Crie uma `struct` para cadastrar uma pessoa, com os seguintes dados: **nome** (string), **aniversário** (struct data), **altura** (struct comprimento) e **peso** (float, em kg). Peça ao usuário o número de pessoas que irá cadastrar. Receba as informações e apresente a listagem das pessoas cadastradas.
- Exemplo: O usuário informa que quer cadastrar duas pessoas. Depois entra com `Maria Silva`; `12/1/1984`; `1.65`; `54.5` e `Pedro Silva`; `5/12/2005`; `1.4`; `48`. O programa então lista `Maria Silva; 12JAN1984; 1m65; 54.5kg` e `Pedro Silva; 05DEZ2005; 1m40; 48.0kg`.
- Método de entrada do usuário:

```
// mensagem para o usuário
scanf("%d", &n);
// laço para captura de informações, controle i
   // qualquer estratégia de captura de estrutura[i]
```

- Casos de teste:

Entrada do usuário	A saída esperada contém
3 Maria Silva 12/1/1984 1.65 54.5 Pedro Silva 5/12/2005 1.4 48 Jonas Silva 23/4/1980 1.7892 70.2	Maria Silva; 12JAN1984; 1m65; 54.5kg Pedro Silva; 05DEZ2005; 1m40; 48.0kg Jonas Silva; 23ABR1980; 1m78; 70.2kg

- **Dicas:** Use funções e estruturas que você desenvolveu para os outros programas. A estrutura de cadastro utilizará membros com outras estruturas (pense em usar `typedef`). Você também precisará de um vetor de estruturas para armazenar os cadastros necessários.