Aula Prática 3
Funções e Ponteiros

Instruções para Submissão

Na aula prática de hoje, você terá que elaborar programas para resolver problemas diversos, conforme descrito abaixo. Cada uma das soluções deverá ser implementada em seu próprio arquivo com extensão `.c`. Por exemplo, a solução para o problema 1 deverá ser implementada em um arquivo chamado `problema1.c`, a solução para o problema 2 deverá ser implementada no arquivo `problema2.c` e assim por diante. Para os problemas que solicitarem a criação de um módulo, você deverá fornecer dois arquivos: um arquivo contendo a declaração das funções (arquivo com extensão `.h`), e um arquivo contendo a definição das funções (com extensão `.c`). Finalmente, submeta cada um dos arquivos pelo Moodle.

Problema 1 [problema1.h e problema1.c]

Você foi contratado para realizar o cálculo da folha de pagamento de uma empresa. Para isso, escreva uma função chamada `pagamento` que recebe como **parâmetros** o valor da hora trabalhada (tipo `double`) e a quantidade de horas trabalhadas (tipo `unsigned short`) e **retorna** o salário após os descontos do Imposto de Renda (IR) (tipo `double`), conforme as regras abaixo.

Salário bruto	**Porcentagem de desconto do IR**
Salários até até R$ 900,00	Isento de desconto do IR
Maior que R$ 900,00 até R$ 1500,00	Desconto de 5%
Maior que R$ 1500,00 até R$ 2500,00	Desconto de 10%
Maior que R$ 2500,00	Desconto de 20%

Observação 1: O cabeçalho da função deve ser exatamente como especificado acima (tipo de retorno, nome da função, tipos e quantidade de parâmetros).

Observação 2: Você terá que implementar um módulo com a função descrita acima. Logo, você deverá submeter dois arquivos: um arquivo contendo a declaração das funções (arquivo `problema1.h`), e um arquivo contendo a definição das funções (arquivo `problema1.c`)

Observação 3: Você não precisa realizar a entrada e saída de dados (não precisa usar as funções `scanf()` e `printf()`).

O programa que irá testar o seu módulo no Moodle é apresentado abaixo:

```c
#include "problema1.h"
#include <stdio.h>

int main() {
   unsigned short qtde;
   double valor_hora, salario = 0;

   printf("Digite: valor_hora, quantidade \n");
   scanf("%lf, %hu", &valor_hora, &qtde);

   salario = pagamento(valor_hora, qtde);
   printf("Salario: %.2lf\n", salario);
   return 0;
}
```

Exemplo de execução do programa:

Digite: valor_hora, quantidade
90, 10
Salario: **900.00**

Problema 2 [problema2.h e problema2.c]

Um posto está vendendo combustíveis de acordo com os descontos a seguir:

- Álcool (‘a’)
  - abastecendo até 20 litros, o preço por litro tem desconto de 3%.
  - abastecendo acima de 20 litros, o preço por litro tem desconto de 5%.
- Gasolina (‘g’)
  - abastecendo até 20 litros, o preço por litro tem desconto de 4%.
  - abastecendo acima de 20 litros, o preço por litro tem desconto de 6%.

Escreva uma função chamada `calcula_valor` que recebe como **parâmetros** o preço do litro de combustível (tipo `double`), a quantidade de litros abastecidos (tipo `unsigned int`) e o tipo de combustível utilizado ('a' ou 'g')(tipo `char`) e **retorna** o valor a ser pago de acordo com as regras acima (tipo `double`).

Observação 1: O cabeçalho da função deve ser exatamente como especificado acima (tipo de retorno, nome da função, tipos e quantidade de parâmetros).

Observação 2: Você terá que implementar um módulo com a função descrita acima. Logo, você deverá submeter dois arquivos: um arquivo contendo a declaração das funções (arquivo `problema2.h`), e um arquivo contendo a definição das funções (arquivo `problema2.c`)

Observação 3: Você não precisa realizar a entrada e saída de dados (não precisa usar as funções `scanf()` e `printf()`).

O programa que irá testar o seu módulo no Moodle é apresentado abaixo:

```c
#include "problema2.h"
#include <stdio.h>

int main() {
   char tipo;
   unsigned int qtde;
   double preco, valor = 0;

   printf("Digite: preco, quantidade, tipo \n");
   scanf("%lf, %u, %c", &preco, &qtde, &tipo);

   valor = calcula_valor(preco, qtde, tipo);
   printf("Valor: %.2lf \n", valor);

   return 0;
}
```

Exemplo de execução do programa:

Digite: preco, quantidade, tipo
2.5, 20, a
Valor: **48.50**

Problema 3 [problema3.h e problema3.c]

Faça uma função chamada `operacao` que receba como **parâmetros** dois números (tipos `float`) e um símbolo (tipo `char`). Este símbolo representará uma operação que deverá ser efetuada com os dois números. Se o símbolo for '+', a função deve retornar a adição; se for '-', a subtração; se for '*', a multiplicação; e se for '/', a divisão. O tipo de **retorno** da função deve ser `float`.

Observação 1: O cabeçalho da função deve ser exatamente como especificado acima (tipo de retorno,

nome da função, tipos e quantidade de parâmetros).

Observação 2: Você terá que implementar um módulo com a função descrita acima. Logo, você deverá submeter dois arquivos: um arquivo contendo a declaração das funções (arquivo `problema3.h`), e um arquivo contendo a definição das funções (arquivo `problema3.c`)

Observação 3: Você não precisa realizar a entrada e saída de dados (não precisa usar as funções `scanf()` e `printf()`).

O programa que irá testar o seu módulo no Moodle é apresentado abaixo:

```c
#include "problema3.h"
#include <stdio.h>

int main() {
   char op;
   float num1, num2, res = 0;

   printf("Digite: num1, num2, op \n");
   scanf("%f, %f, %c", &num1, &num2, &op);

   res = operacao(num1, num2, op);
   printf("Resultado: %.2f \n", res);

   return 0;
}
```

Exemplo de execução do programa:

Digite: num1, num2, op
2.5, 2.5, +
Resultado: **5.00**

Problema 4 [problema4.h e problema4.c]

Escreva uma função chamada `valor_energia` que recebe como **parâmetros** a quantidade de kWh consumida no mês (tipo `unsigned int`) e o tipo de instalação ('R' para residencial, 'C' para comercial e 'I' para industrial) (tipo `char`). A função deve **retornar** o preço total a pagar pelo fornecimento de energia elétrica (tipo `float`), conforme as regras definidas na tabela a seguir:
Observação 1: O cabeçalho da função deve ser exatamente como especificado acima (tipo de retorno, nome da função, tipos e quantidade de parâmetros).

Observação 2: Você terá que implementar um módulo com a função descrita acima. Logo, você deverá submeter dois arquivos: um arquivo contendo a declaração das funções (arquivo `problema4.h`), e um arquivo contendo a definição das funções (arquivo `problema4.c`)

Observação 3: Você não precisa realizar a entrada e saída de dados (não precisa usar as funções `scanf()` e `printf()`).

Preço por tipo e faixa de consumo		
Tipo	**Faixa(kWh)**	**Preço do kWh**
Residencial	Até 500	R$ 0,40
	Acima de 500	R$ 0,65
Comercial	Até 1000	R$ 0,55
	Acima de 1000	R$ 0,60
Industrial	Até 5000	R$ 0,30
	Acima de 5000	R$ 0,35

O programa que irá testar o seu módulo no Moodle é apresentado abaixo:

```c
#include "problema4.h"
#include <stdio.h>

int main() {
   char tipo;
   unsigned int consumo;
   float total = 0;

   printf("Digite: consumo, tipo \n");
   scanf("%u, %c", &consumo, &tipo);

   total = valor_energia(consumo, tipo);
   printf("Total: %.2f \n", total);

   return 0;
}
```

Exemplo de execução do programa:

Digite: consumo, tipo
500, R
Total: **200.00**

Problema 5 [problema5.h e problema5.c]

Escreva uma função chamada `estacionamento` que recebe como **parâmetros** a hora e minuto de entrada, e hora e minuto de saída de um estacionamento (todos os quatro do tipo `unsigned short`) e **retorna** o valor total devido (tipo `float`) de acordo com as seguintes regras:

Quantidade de Horas	Tarifa
Até duas horas	R$ 1.00 para cada hora
Entre três e quatro horas	R$ 1.40 para cada hora
Acima de quatro horas	R$ 2.00 para cada hora

O número de horas a pagar é sempre inteiro e arredondado por excesso. Por exemplo, quem estacionar por 1 minuto pagará por 1 hora. Quem estacionar durante 61 minutos pagará por 2 horas. Perceba que os momentos de entrada e saída do estacionamento são apresentados na forma de pares de inteiros (`unsigned short`), representando horas e minutos. Por exemplo, o par (12, 50) representará "dez para a uma da tarde". Admite-se que a entrada e a saída se dão com intervalo não superior a 24 horas. Se o horário de entrada for igual ao horário de saída, então o tempo de permanência é de 24 horas.

Observação 1: O cabeçalho da função deve ser exatamente como especificado acima (tipo de retorno, nome da função, tipos e quantidade de parâmetros).

Observação 2: Você terá que implementar um módulo com a função descrita acima. Logo, você deverá submeter dois arquivos: um arquivo contendo a declaração das funções (arquivo `problema5.h`), e um arquivo contendo a definição das funções (arquivo `problema5.c`)

Observação 3: Você não precisa realizar a entrada e saída de dados (não precisa usar as funções `scanf()` e `printf()`).

O programa que irá testar o seu módulo no Moodle é apresentado abaixo:

```c
#include "problema5.h"
#include <stdio.h>

int main() {
   unsigned short h_e, m_e, h_s, m_s;
   float total = 0;

   printf("Digite: h_e, m_e, h_s, m_s \n");
   scanf("%hu, %hu, %hu, %hu", &h_e, &m_e, &h_s, &m_s);

   total = estacionamento(h_e, m_e, h_s, m_s);
   printf("Total: %.2f \n", total);

   return 0;
}
```

Exemplo de execução do programa:

Digite: h_e, m_e, h_s, m_s
12, 20, 17, 21
Total: **12.00**

Problema 6 [problema6.h e problema6.c]

Escreva um procedimento chamado `troca` que recebe como **parâmetros** dois endereços de memória de variáveis do tipo `int` (`end_valor1` e `end_valor2`) (que tipo de variável é capaz de armazenar endereços de memória?). Esse procedimento deve trocar o conteúdo armazenado nesses endereços. Ou seja, o inteiro armazenado no primeiro endereço (endereço armazenado em `end_valor1`) deve ser armazenado no segundo endereço (endereço armazenado em `end_valor2`), e vice-versa. Por se

tratar de um procedimento, o tipo de **retorno** tem que ser `void`.

Observação 1: O cabeçalho da função deve ser exatamente como especificado acima (tipo de retorno, nome da função, tipos e quantidade de parâmetros).

Observação 2: Você terá que implementar um módulo com a função descrita acima. Logo, você deverá submeter dois arquivos: um arquivo contendo a declaração das funções (arquivo `problema6.h`), e um arquivo contendo a definição das funções (arquivo `problema6.c`)

Observação 3: Você não precisa realizar a entrada e saída de dados (não precisa usar as funções `scanf()` e `printf()`).

O programa que irá testar o seu módulo no Moodle é apresentado abaixo:

```c
#include "problema6.h"
#include <stdio.h>

int main() {
   int x, y;

   printf("Digite: x, y \n");
   scanf("%d, %d", &x, &y);

   troca(&x, &y);
   printf("%d, %d \n", x, y);

   return 0;
}
```

Exemplo de execução do programa:

Digite: x, y
1, 2
2, 1

Problema 7 [problema7.h e problema7.c]

Escreva uma função chamada `equacao` que recebe como **parâmetros** três valores do tipo `int` e dois endereços de memória de variáveis do tipo `double` (`end_x1` e `end_x2`). Ou seja, a função possui 5 parâmetros. Assuma que os três primeiros parâmetros (a, b e c) são os coeficientes de uma equação de segundo grau:

$$ax^2 + bx + c = 0$$

Primeiramente, a função deve calcular o valor de Δ:

$$\Delta = b^2 - 4ac$$

- Se $\Delta < 0$, não existe raiz real. Logo, a função deve simplesmente **retornar** o inteiro 0.

- Se $\Delta = 0$, existe uma raiz real. A função deve calcular a raiz e armazená-la no primeiro endereço recebido (endereço armazenado em `end_x1`). Finalmente, a função deve **retornar** o inteiro 1.
- Se $\Delta > 0$, existem duas raízes reais. A função deve calcular as duas raízes e armazená-las nos dois endereços recebidos (endereços armazenados em `end_x1` e `end_x2`). Finalmente, a função deve **retornar** o inteiro 2.

Lembrando que:

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}$$

Observação 1: O cabeçalho da função deve ser exatamente como especificado acima (tipo de retorno, nome da função, tipos e quantidade de parâmetros).

Observação 2: Você terá que implementar um módulo com a função descrita acima. Logo, você deverá submeter dois arquivos: um arquivo contendo a declaração das funções (arquivo `problema7.h`), e um arquivo contendo a definição das funções (arquivo `problema7.c`)

Observação 3: Você não precisa realizar a entrada e saída de dados (não precisa usar as funções `scanf()` e `printf()`).

O programa que irá testar o seu módulo no Moodle é apresentado abaixo:

```c
#include "problema7.h"
#include <stdio.h>

int main() {
   int a, b, c, r = 2;
   double x1 = 0, x2 = 0;

   printf("Digite: a, b, c \n");
   scanf("%d, %d, %d", &a, &b, &c);

   r = equacao(a, b, c, &x1, &x2);

   if (r == 0) {
      printf("Nao existe raiz real\n");
   } else if (r == 1) {
      printf("Raiz unica\n");
      printf("Raiz: %.2lf\n", x1);
   } else {
      printf("Raiz 1: %.2lf\n", x1);
      printf("Raiz 2: %.2lf\n", x2);
   }

   return 0;
}
```

Exemplo de execução do programa:

Digite: a, b, c
3, 12, 9
Raiz 1: **-1.00**
Raiz 2: **-3.00**

Problema 8 [problema8.h e problema8.c]

Escreva um procedimento chamado `ordena` que recebe como **parâmetros** três endereços de memória de variáveis do tipo `int` (`end_x1`, `end_x2` e `end_x3`). O procedimento deve então reorganizar os valores armazenados nesses endereços de modo que:

- O menor inteiro seja armazenado no primeiro endereço (endereço armazenado em `end_x1`).
- O inteiro de valor intermediário seja armazenado no segundo endereço (endereço armazenado em `end_x2`).
- O maior inteiro seja armazenado no terceiro endereço (endereço armazenado em `end_x3`).

No desenvolvimento do seu módulo (`problema8.h`), você deverá obrigatoriamente utilizar a função `troca` do módulo desenvolvido no Problema 6 (`problema6.h`). Ao submeter o seu módulo no Moodle, **não precisa submeter** os arquivos do módulo desenvolvido no Problema 6 (`problema6.h` e `problema6.c`).

Observação 1: O cabeçalho da função deve ser exatamente como especificado acima (tipo de retorno, nome da função, tipos e quantidade de parâmetros).

Observação 2: Você terá que implementar um módulo com a função descrita acima. Logo, você deverá submeter dois arquivos: um arquivo contendo a declaração das funções (arquivo `problema8.h`), e um arquivo contendo a definição das funções (arquivo `problema8.c`)

Observação 3: Você não precisa realizar a entrada e saída de dados (não precisa usar as funções `scanf()` e `printf()`).

O programa que irá testar o seu módulo no Moodle é apresentado abaixo:

```c
#include "problema8.h"
#include <stdio.h>

int main() {
    int x, y, z;

    printf("Digite: x, y, z\n");
    scanf("%d, %d, %d", &x, &y, &z);

    ordena(&x, &y, &z);

    printf("%d, %d, %d\n", x, y, z);

    return 0;
}
```

Exemplo de execução do programa:

Digite: x, y, z
3, 1, 2
1, 2, 3

Problema 9 [problema9.h e problema9.c]

Escreva um procedimento chamado `media` que recebe como **parâmetros** duas notas de um aluno (tipos `float`), um ponteiro para um `float` (`end_media`) e um ponteiro para um `char` (`end_status`). O procedimento deve então calcular a média do aluno e armazenar o resultado da média na variável apontada por `end_media`. Finalmente, o procedimento deve armazenar a situação do aluno (caracteres 'A', 'E' ou 'R') na variável apontada por `end_status` conforme as regras a seguir:

- Se média ≥ 7.0: armazenar o caractere 'A' (situação "Aprovado").
- Se média ≥ 4.0 e < 7.0: armazenar o caractere 'E' (situação "Especial").
- Se média < 4.0: armazenar o caractere 'R'(situação "Reprovado").

Observação 1: O cabeçalho da função deve ser exatamente como especificado acima (tipo de retorno, nome da função, tipos e quantidade de parâmetros).

Observação 2: Você terá que implementar um módulo com a função descrita acima. Logo, você deverá submeter dois arquivos: um arquivo contendo a declaração das funções (arquivo `problema9.h`), e um arquivo contendo a definição das funções (arquivo `problema9.c`)

Observação 3: Você não precisa realizar a entrada e saída de dados (não precisa usar as funções `scanf()` e `printf()`).

O programa que irá testar o seu módulo no Moodle é apresentado abaixo:

```c
#include "problema9.h"
#include <stdio.h>

int main() {
   char status = ' ';
   float n1, n2, m = 0;

   printf("Digite: n1, n2\n");
   scanf("%f, %f", &n1, &n2);

   media(n1, n2, &m, &status);
   printf("Media: %.2f\n", m);
   printf("Status: %c\n", status);

   return 0;
}
```

Exemplo de execução do programa:

Digite: n1, n2
7, 7
Media: **7.0**
Status: **A**

Problema 10 [problema10.h e problema10.c]

Escreva um procedimento chamado `valida_data` que recebe como **parâmetros** o dia, o mês e o ano de uma data (tipos `int`), um ponteiro para um `int` (`end_valida`) e um outro ponteiro para um `int` (`end_bissexto`). O procedimento deve então verificar se a data fornecida é válida e atualizar as variáveis apontadas pelos ponteiros `end_valida` e `end_bissexto` conforme as regras abaixo:

- Se a data for válida, a variável apontada por `end_valida` deve armazenar 1. Caso contrário, deve armazenar 0.
- Se o ano informado for bissexto, a variável apontada por `end_bissexto` deve armazenar 1. Caso contrário, deve armazenar 0. A data não precisa necessariamente ser válida para indicar se o ano é bissexto.

Para verificar se uma data é válida, verifique se o mês está entre 1 e 12, e se o dia existe naquele mês. Note que fevereiro tem 29 dias em anos bissextos, e 28 dias em anos não bissextos. Além disso, os meses de abril, junho, setembro e novembro possuem 30 dias. Um ano é bissexto se ele for múltiplo de 4 e não for múltiplo de 100, ou se ele for múltiplo de 400.

Observação 1: O cabeçalho da função deve ser exatamente como especificado acima (tipo de retorno, nome da função, tipos e quantidade de parâmetros).

Observação 2: Você terá que implementar um módulo com a função descrita acima. Logo, você deverá submeter dois arquivos: um arquivo contendo a declaração das funções (arquivo `problema10.h`), e um arquivo contendo a definição das funções (arquivo `problema10.c`)

Observação 3: Você não precisa realizar a entrada e saída de dados (não precisa usar as funções `scanf()` e `printf()`).

O programa que irá testar o seu módulo no Moodle é apresentado abaixo:

```c
#include "problema10.h"
#include <stdio.h>

int main() {
   int dia, mes, ano;
   int valida = 0, bissexto = 0;

   printf("Digite: dia/mes/ano\n");
   scanf("%d/%d/%d", &dia, &mes, &ano);

   valida_data(dia, mes, ano, &valida, &bissexto);
```

```
   if (valida)
      printf("Valida: Sim\n");
   else
      printf("Valida: Nao\n");
   if (bissexto)
      printf("Bissexto: Sim\n");
   else
      printf("Bissexto: Nao\n");

   return 0;
}
```

Exemplo de execução do programa:

Digite: dia/mes/ano
31/4/2020
Valida: **Nao**
Bissexto: **Sim**