UTFPR - Universidade Tecnológica Federal do Paraná
DACOM - Departamento de Computação
BCC2001 :: Algoritmos

Exercícios 08 :: Structs

Instruções Gerais

- Faça todos os exercícios em um único arquivo .c. Utilize a função main() para fazer chamadas de testes às funções solicitadas pelos exercícios.
- Utilize a extensão .c, o compilador gcc e o editor de sua preferência: VS Code, Dev C++, etc.
  - Alternativamente, utilize https://replit.com/languages/c.
- Você pode utilizar as seguintes funções disponíveis na biblioteca (lib) padrão da linguagem C:
  - `rand()` **<stdlib.h>**
  - `printf(), sprintf()` **<stdio.h>**
  - `strlen(), strcpy(), strcmp(), strcat()` **<string.h>**

1. Escreva uma função que recebe um ponto x,y (**Point**) e o imprime com 2 casas de precisão.

```c
struct point {
  float x;
  float y;
};
typedef struct point Point;
```

float print_point(Point p)

2. Escreva uma função que recebe dois pontos x,y (**Point**) e devolve a distância entre os mesmos. Utilize **sqrt()** da lib <math.h> para obter a raiz quadrada: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

float distance2D(Point p1, Point p2)

Exemplo de uso função:

```c
Point p1 = {-2.0f, 7.5f};
Point p2 = {5.0f, 12.4f};
float dist = distance(p1, p2);
```

OBS: para compilar com a lib <math.h>, talvez seja necessário acrescentar o argumento -lm à chamada do gcc:

$ gcc programa.c -lm

3. Escreva uma função que recebe um vetor de pontos (**Point**), bem como seu tamanho, e informa (imprime) a distância a cada dois pontos consecutivos do vetor.

float distance_vector(int n, Point v[])

4. Escreva uma função que recebe um vetor de pontos (**Point**) e os inicia com valores **float** aleatórios entre [-50, 50].

   `void random_points(int n, Point v[])`

   Para obter um float randômico, consideremos como exemplo o intervalo [-2, 2]:

   a. Dividimos o resultado de rand() por RAND_MAX, obtendo um float [0, 1]:

      `rand()/(float)RAND_MAX  →  [0, 1]`

   b. Multiplicamos o resultado anterior pela quantidade desejada de números, subtraída de 1. No exemplo, de -2 à 2, temos 5 números [0, 4]. Então multiplicamos por 5-1=4, pois os 5 números vão de 0 à 4:

      `rand()/(float)RAND_MAX * 4  →  [0, 4]`

   c. Por fim, devemos subtrair o resultado pelo valor inicial do intervalor [-2, 2]:

      `float rand_number = rand()/(float)RAND_MAX * 4 - 2; // [-2, 2]`

5. Escreva uma função que recebe um vetor de Point e encontra os dois pontos mais próximos, isto é, aqueles que possuem a menor distância. A função deve imprimir os dois pontos mais próximos - (X_1,Y_1) e (X_2,Y_2) -, além da distância. Obviamente, você não deve considerar a distância de um ponto a ele mesmo, que será sempre zero.

   `void nearest_points(int n, Point v[])`

6. Escreva uma função que recebe um **Contact** (ver struct abaixo) e imprime seu conteúdo. A função também recebe um vetor de strings contendo os nomes dos tipos **type_names** e seu tamanho **n**. O campo **type** do contato deve ser utilizado como índice no vetor **type_names**.

   `void print_contact(Contact c, int n, char type_names[n][50])`

```c
struct contact {
   int id;
   char name[51];
   char email[51];
   int type;
};
typedef struct contact Contact;
```

Exemplo de uso da função:

```c
char types[5][50] = {"Família","Amigos","Trabalho","Escola","Outros"};
Contact c = {1, "John Doe", "john.doe@email.com", 2};
print_contact(c, 5, types);
// saída: {1, John Doe, john.doe@email.com, Trabalho}
```

7. Escreva uma função que recebe um vetor de Contact e imprime os contatos agrupados pelo seu tipo.

   void print_contact_list(int n1, Contact list[n1], int n2, char types[n2][50])

   Exemplo de uso da função:

```
char types[][50] = {"Família","Amigo","Trabalho","Escola","Outros"};
Contact list[] = {
      {1, "Marcus Fenix", "fenix@gow.com", 2},
      {41,"Blue Mary", "mary@ff3snk.net", 0},
      {17,"Barry Burton", "bburton@re.cap", 0},
      {8, "Charlie Nash", "nash@ssz.com", 2},
      {2, "Ada Wong", "wong@re2.net", 4},
      {5, "Chris Redfield", "crfield@re.cap", 0}
};
print_contact_list(6, list, 5, types);

/* SAÍDA DA FUNÇÃO: */
/*
Família
      Blue Mary
      Barry Burton
      Chris Redfield

Trabalho
      Marcus Fenix
      Charlie Nash

Outros
      Ada Wong
*/
```

8. Escreva uma função que recebe um vetor de empregados (Employee) e seu tamanho. A função deve imprimir os nomes dos três empregados de maiores salários, ou seja, as pessoas que recebem os três maiores salários.

   void print_best3(int n, Employee v[n]);

```
struct employee {
  char name[50];
  float salary;
  int type;  // 0 - Developer, 1 - Designer, 2 - Manager, 3 - Support
};
typedef struct employee Employee;
```

9. Escreva uma função que recebe uma lista de empregados (**Employee**) e seu tamanho. A função deve calcular e imprimir: o total dos salários, a média dos salários e as médias dos salários por tipo de empregado (“Developer”, “Designer”, “Manager” ou “Support”).

   `void income_report(int n, Employee v[n]);`

10. Escreva uma função que recebe um nome, o verifica e corrige, se necessário. Ao final, o nome deverá conter apenas letras e espaços, com as iniciais de cada nome em maiúscula e o restante em minúsculas. Deve haver somente um espaço entre cada nome e não devem haver espaços no início e final.

    `void fix_name(char name[])`

    Exemplo de uso da função:

    ```
    char name[] = "  JoHN   DOE# 23!    "
    fix_name(name);
    // Após a chamada, name: "John Doe"
    ```

11. Escreva uma função que preenche e devolve um Contact (**struct** do exercício 6), definindo os campos da seguinte forma:
    a. id: definido pelo parâmetro ***id***;
    b. name: definido pelo parâmetro ***name***. Deve ser verificado e corrigido, se necessário. Para tanto, chame a função do exercício anterior.
    c. email: deve ser gerado com base no nome e respeitando as seguintes características
        i. Todas as letras devem estar minúsculas;
        ii. Espaço deve ser substituído por “.” (ponto);
        iii. Acrescentar o sufixo “@mail.br”.
    d. type: definido pelo parâmetro ***type***.

    `Contact create_contact(int id, char name[], int type)`

    Exemplo de uso da função:

    ```
    Contact c = create_contact(4, "  JoHn! Do5e3   SILVA", 1);
    // Após a chamada, c = {4, "John Doe Silva", "john.doe.silva@mail.br",1}
    ```

    OBS: para montar o campo e-mail, você pode usar strcat() ou sprintf(). Pesquise sobre essas funções na web, para compreender melhor como utilizá-las.

12. Escreva uma função que recebe um vetor de Contact e o preenche. O campo ***id*** deve ser preenchido incrementalmente, iniciando em 1. O campo ***type*** deve ser randomizado entre um dos 5 tipos possíveis, exceto para contatos de pessoas da família “Doe”, que devem ser definidos com o tipo “Amigo”. Utilize a função do exercício anterior.

    `void fill_contact_list(int n, Contact list[n])`

    Exemplo de uso da função:

    ```
    Contact list[15];
    fill_contact_list(15, list);
    // Após a chamada, a lista estará preenchida.
    ```

13. Escreva uma função que recebe um vetor de Contact e um outro vetor, com uma lista de nomes a buscar. A função deve retornar a quantidade de Contatos que possuem, ao menos, um dos nomes procurados.

    `int find_by_name(int n1, Contact list[n1], int n2, char names[n2][50])`

    Exemplo de uso da função:

```
char names[5][10] = {"joanna", "john", "mike", "leonor", "caroline"};
// a chamada abaixo contará os nomes em “list” que contêm ao menos
// uma das palavras em “names”.
int count = find_by_name(30, list, 5, names);
```