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

Exercícios 09 :: Ponteiros

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 um programa em C que declara três variáveis (char, int e double) e imprime seus endereços.

2. Escreva uma função que recebe o endereço de duas variáveis inteiras. A função deve somar os valores contidos nessas variáveis por meio de seus endereços, colocando o resultado na primeira.
   `void add(int* a, int* b)`

3. Escreva uma função que recebe o comprimento e o endereço de um vetor, e imprime o seu conteúdo, sem utilizar o operador de índice [ ] para acessar os elementos do vetor.
   `void print_vector(int n, const int* v)`

4. Escreva uma função que recebe o comprimento e o endereço de um vetor, e imprime o seu conteúdo em ordem reversa, sem utilizar o operador de índice [ ] para acessar os elementos do vetor.
   `void print_vector_reverse(int n, const int* v)`

5. Escreva uma função que recebe o comprimento e o endereço de um vetor. A função deve retornar, via parâmetros min e max, o maior e o menor valor contido no vetor. Você não deve utilizar o operador de índice [ ] para acessar os elementos do vetor.
   `void get_min_max(int n, const int* v, int* min, int* max)`

6. Escreva uma função que recebe as dimensões e o endereço de uma matriz (vetor bidimensional). A função deve imprimir seu conteúdo, sem utilizar o operador de índice [ ] para acessar os elementos da matriz.
   `void print_vector2D(int rows, int cols, const int* v)`

   Exemplo de uso da função:

```
int v[3][4] = {
    {1, 2, 3, 4},
    {5, 6, 7, 8},
    {9,10,11,12}
};
print_vec2(3, 4, (int*)v);
```

   Saída:

```
01 02 03 04
05 06 07 08
09 10 11 12
```

7. Escreva uma função que recebe o endereço de uma string. A função deve concatenar, ao final da string original, uma barra vertical, seguida do conteúdo da string de forma invertida. Considere que o vetor possui comprimento suficiente para a adição dos novos caracteres. Você não deve utilizar o operador de índice [ ] para acessar os caracteres da string.

   void make_mirrored(char* str)

   Exemplo de uso da função:

```
char name[40] = "John Doe";
make_mirrored(name);
printf("%s\n", name);   // Saída "John Doe|eoD nhoJ"
```

8. Escreva uma função que recebe o endereço de duas strings e calcula a distância de Hamming entre ambas. A distância de Hamming corresponde à quantidade de caracteres diferentes em posições correspondentes nas duas strings. Considere que as strings terão o mesmo comprimento, mas certifique-se de que o código não acessa posições inválidas para strings de tamanhos distintos.

   void get_hamming_distance(const char* str1, const char* str2)

   Exemplo de uso da função:

```
int dist = get_hamming_distance("banana", "cabana"); // dist = 2
/*
      banana      << distância de Hamming é 2
      cabana
*/
```

9. Escreva uma função que recebe o endereço de uma string **str** e de um vetor de strings **words** (de até 49 caracteres). A função deve devolver o endereço da string de **words** que seja mais similar à **str**, isto é, com a menor distância de Hamming. OBS: faça chamadas à função do exercício anterior para encontrar as distâncias entre **str** e cada uma das palavras em **words**.

   char* find_most_similar(const char* str, int n, char list[n][50]);

   Exemplo de uso da função:

```
char words[][50] = {"cabana", "savana", "bacana", "halana"};

char* most_similar = find_most_similar("banana", 4, words);
printf("%s\n", most_similar);       // "bacana"
```

10. Escreva uma função que recebe o endereço de um Rect e imprime um tablado de acordo com a posição **x,y** e as dimensões **width e height**. Você pode considerar cada caractere do terminal como uma unidade para posição/tamanho.

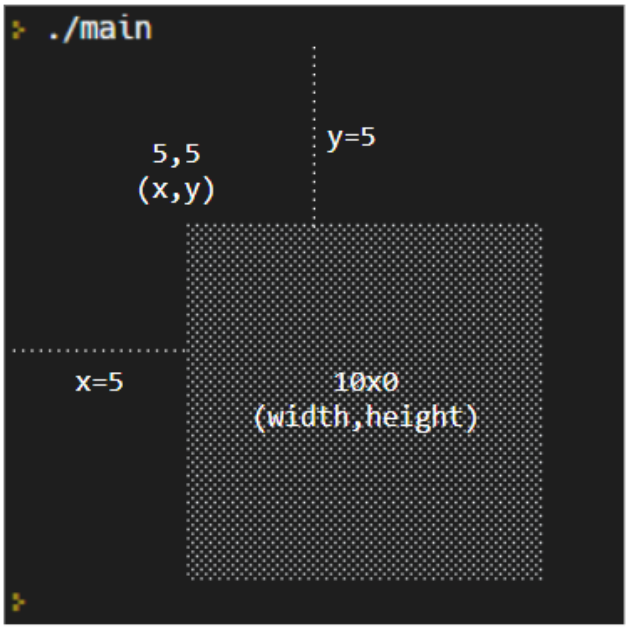
```
struct rect {
    int x, y, width, height;  // é possível declarar os campos em uma só linha
};
typedef struct rect Rect;

void print_board(Rect* board);
```

Exemplo de uso da função:

```
Rect rect = {5,5,10,10}; // posição 5,5 (x,y) e dimensões 10x10 (width,height)
print_board(&rect);
```

Saída:



OBS: neste exemplo, considerei dois espaços " " e dois caracteres "▒▒" como uma posição, pois assim obtive blocos mais quadrados, uma vez que os caracteres no terminal são mais altos do que largos. Opcionalmente, "limpe" a tela antes de desenhar o tablado com **`printf("\033[2J")`**.

11. Modifique a função do exercício anterior para que receba o endereço de um Point e o imprima dentro de um tablado de dimensões definido por um Rect. As coordenadas do Point devem ser relativas à do Rect, isto é, um Point na posição (0,0) deve ser desenhado no canto esquerdo superior do Rect. Caso a posição do Point extrapole os limites do Rect, não deve ser desenhado.

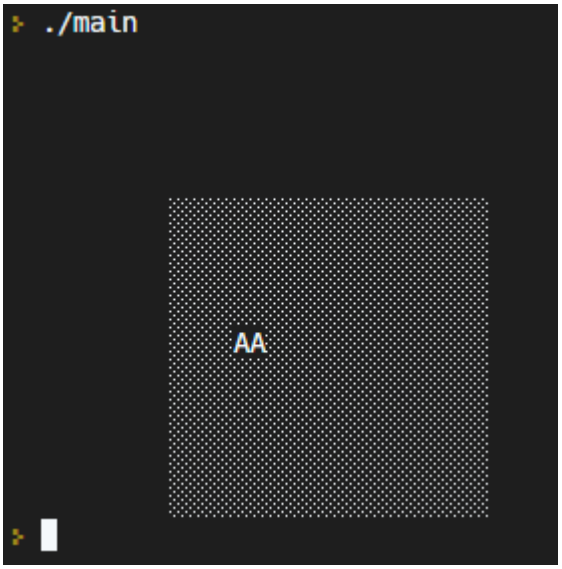
```
struct point {
    int x, y;                 // é possível declarar os campos em uma só linha
    char symbol;
};
typedef struct point Point;

void print_board2(Point* point, Rect* board);
```

Exemplos de usos da função:

```
Rect rect = {5,5,10,10}; // posição 5,5 (x,y) e dimensões 10x10 (width,height)
Point point = {0,0,'A'}; // posição 0,0 dentro do tablado
print_board2(&point, &rect);
```


```
Rect rect = {5,5,10,10}; // posição 5,5 (x,y) e dimensões 10x10 (width,height)
Point point = {2,4,'A'}; // posição 2,4 dentro do tablado
print_board2(&point, &rect);
```



12. Modifique a função do exercício anterior para que receba o endereço de vetor de Point. A função deve imprimir todos os pontos dentro do tablado, seguindo as mesmas orientações do exercício mencionado.

void print_board3(Point* points, Rect* board);

Exemplo de uso da função:

```
Rect rect = {5,5,10,10}; // posição 5,5 (x,y) e dimensões 10x10 (width,height)
Point points[] = { {0,0,'A'}, {2,7,'B'}, {9,9,'C'} };
print_board3(points, &rect);
```