UNIVERSIDADE DO ESTADO DA BAHIA
DEPARTAMENTO DE CIÊNCIAS EXATAS
E DA TERRA-I COLEGIADO DO CURSO
DE SISTEMAS DA INFORMAÇÃO

LISTA DE EXERCÍCIOS PARTE 1: PONTEIROS

SALVADOR
2024

PARTE 1: PONTEIROS

Atribuição de ponteiros:

Questão 1) Analise o programa em C abaixo, identificando possíveis erros e explicando sua saída (se for bem definida).

```c
#include <stdio.h>

int main() {
  float x = 5, y;
  int * p;

  p = & x;
  y = * p;

  printf("valor de p: %p\n", p);
  printf("valor de y: %f\n", y);

  return 0;
}
```

Questão 2) Analise o programa em C abaixo, identificando possíveis erros e explicando sua saída (se for bem definida).

Referência [2]

```c
#include <stdio.h>

int main() {
  int x = 5, y = 10;
  int * ptr1 = &x;
  int * ptr2 = &y;

  *ptr1 = *ptr2;
  ptr2 = ptr1;
  *ptr2 = 20;

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

  return 0;
}
```

Questão 3) Analise o programa em C abaixo, identificando possíveis erros e explicando sua saída (se for bem definida).

```c
#include <stdio.h>

int main() {
  int vet[5] = {
    10,
    20,
    30,
    40,
    50
  };
  int * p = vet;
  *(p + 2) = *(p + 4) + 5;
  p++;
  *(p + 1) = *p * 2;
  for (int i = 0; i < 5; i++) {
    printf("%d ", vet[i]);
  }
  return 0;
}
```

Questão 4) Analise o programa em C abaixo, identificando possíveis erros e explicando sua saída (se for bem definida).

```c
#include <stdio.h>

void troca(int ** pp, int * q) {
  int * temp = * pp;
  * pp = q;
  q = temp;
}

int main() {
  int a = 5, b = 10;
  int * pa = & a;
  int * pb = & b;

  troca( & pa, pb);

  printf("a = %d, b = %d\n", a, b);
  printf("*pa = %d, *pb = %d\n", * pa, * pb);

  return 0;
}
```

Ponteiros para funções:

Questão 5) Analise o seguinte programa em C, e informe: há outra forma de chamar as funções **strcmp()** e **numcmp()** como argumentos da função **check()**?

Referência [2].

questao5.c > ...

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "string.h"

void check(char *a, char *b, int (*cmp) (const char *, const char *));
int numcmp(const char *a, const char *b);

void main(void)
{
    char s1[80], s2[80];
    int (*p) ();
    int (*p2) ();

    p = strcmp;
    p2 = numcmp;

    gets(s1);
    gets(s2);

    if(isalpha(*s1))
        check(s1, s2, p);
    else
        check(s1, s2, p2);
}
```

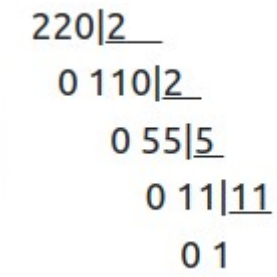
```c

void check(char *a, char *b, int (*cmp)(const char *, const char *))
{
    printf("testando igualdade\n");
    if(!(*cmp)(a,b))
        printf("igual");
    else
        printf("diferente");
}

int numcmp(const char *a, const char *b)
{
    if(atoi(a)==atoi(b))
        return 0;
    else
        return 1;
}
```

Questão 6) Faça uma função em C que:

- Recebe 3 parâmetros: um vetor de inteiros, um número inteiro n passado por valor e outro número x passado como ponteiro.
- Retorna um número inteiro.
- Decompõe o número *n* em fatores primos e armazena-os nas posições do vetor. O conteúdo de *x* deve receber o número de fatores primos encontrados. Caso o número de fatores encontrados seja maior que 10, a função deve retornar 1, do contrário deve retornar 0.
  Dica: Para decompor um número em números primos, deve-se dividi-lo pelo menor primo possível (restando zero), sucessivamente até que o quociente seja 1. Ex:



  Assim, a fatoração de 220 é: 2 x 2 x 5 x 11. Logo, o número de fatores primos encontrados nesse exemplo é 4.

Referência [3]

Questão 7) Faça uma função em C que receba duas strings como parâmetros e verifique se a segunda string ocorre dentro da primeira. Use aritmética de ponteiros para acessar os caracteres das strings.

Referência [4]

Questão 8) Faça uma função em C que calcule a área da superfície e o volume de uma esfera de raio R. Essa função deve obedecer ao protótipo:

void calcEsfera(float R, float *area, float *volume);

Referência [4]

Questão 9) Escreva uma função em C que receba um número real passado como parâmetro, retorne a parte inteira e a parte fracionária deste número. Essa função deve obedecer ao protótipo:

void frac(float num, int *parteInteira, int *parteFracionaria);

Referência [4]

Questão 10) Faça uma função em C que recebe como parâmetro um array com N valores, retornando o maior elemento do array e o número de vezes que esse elemento ocorreu no array. Por exemplo, para um array com os seguintes elementos: [5,2,15,3,7,15,8,6,15], a função deve retornar para o programa o valor 15 e o valor 3. (indicando que o número 15 ocorreu 3 vezes). A função deve ser do tipo void.

Referência [4]

Referências:

1. **NOTAS DE AULA - ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES**. Prof. Rodrigo de Oliveira. Ponteiros. [s.l: s.n.], 2004. Disponível em: https://www.ic.unicamp.br/~oliveira/doc/mc102_2s2004/aula9.pdf. Acesso em: 21 out. 2024.

2. SCHILDT, H.; MAYER, R. C. ***C completo e total***. São Paulo: Pearson Education do Brasil, 2008.

3. **INE5408-03208A (20121): Exercícios Adicionais em Linguagem “C” - Ponteiros e Alocação Dinâmica de Memória**. Moodle UFSC. Disponível em: https://legado.moodle.ufsc.br/mod/page/view.php?id=231983. Acesso em: 25 out. 2024.

4. **Exercícios: Ponteiros**. [s.l: s.n.]. Disponível em: https://www.facom.ufu.br/~backes/wordpress/ListaC09.pdf. Acesso em: 25 out. 2024.

5. **Exercícios: Alocação Dinâmica**. Disponível em: https://www.facom.ufu.br/~backes/wordpress/ListaC10.pdf. Acesso em: 25 out. 2024.