Algoritmos e Programação 2

Exercícios para praticar

1. Escreva uma função que receba uma string e substitua cada segmento de dois ou mais espaços por um só caractere ' '.

2. Escreva uma função que receba uma string de 0s e 1s, interprete essa string como um número em notação binária e devolva o valor desse número.

3. Escreva uma função que receba *strings* `x` e `s` e devolve o índice da posição a partir da qual `x` ocorre em `s`.

4. Escreva uma função que modifica uma *string* substituindo todas as ocorrências de `foo` por `xxx`. Por exemplo, para "food fool", a saída deverá "ser xxxd xxxl". Faça a função da forma mais enxuta possível, sem perder a legibilidade do código.

5. Qual será o valor de `s1` após a execução do código abaixo?

```
strcpy(s1, "computer");
strcpy(s2, "science");
if (strcmp(s1, s2) < 0)
  strcat(s1, s2);
else
  strcat(s2, s1);
s1[strlen(s1)-6] = '\0';
```

6. Se uma *string* for declarada como: `char str[20];` o número máximo de caracteres que poderão ser lidos e armazenados nela é:

   (a) 18

   (b) 19

   (c) 20

   (d) 21

7. Declare uma estrutura para um número complexo (parte real e parte imaginária do tipo `double`. Crie funções para ler, escrever e somar números complexos.

8. Considere uma estrutura para armazenar data (dia, mês e ano). Escreva uma função:

   (a) que recebe duas datas (verifique se as datas são válidas) e devolve o número de dias que decorreram entre as duas datas;

   (b) recebe uma data e retorna o dia do ano (1 a 366);

(c) recebe duas datas `d1` e `d2` e compara-as retornando -1 se `d1` $<$ `d2`, 0 se `d1` $=$ `d2` e 1 se `d1` $>$ `d2`.

9. Monte um programa para cadastrar imóveis a serem alugados ou vendidos, contendo os seguintes dados: tipo (loja, apartamento, casa, kit), endereço, bairro, valor, situação (aluguel ou venda). Ao final, solicitar ao usuário a situação a ser pesquisada e mostrar todos os dados dos imóveis enquadrados na solicitação;

10. Escreva duas versões de um programa que lê uma mensagem e verfica se a mesma é um palíndromo. A primeira versão deve utilizar inteiros para controlar as posições do vetor. A segunda versão deve utilizar ponteiros.

11. Considere algoritmo abaixo:

```
int main()
{
  int i, j, *p_1, *p_2, **p_p_1, **p_p_2;
  i = 4;
  j = 5;
  p_1 = &i;
  p_2 = &j;
  p_p_1 = &p_2;
  p_p_2 = &p_1;
  return 0;
}
```

Considere também a seguinte alocação de memória (variável - endereço alocado para a mesma):

- `i` - 1000
- `j` - 1004
- `p_1` - 1057
- `p_2` - 1042
- `p_p_1` - 1547
- `p_p_2` - 1147

Indique o valor de:

(a) `i`
(b) `j`
(c) `p_1`
(d) `p_2`
(e) `p_p_1`
(f) `p_p_2`
(g) `&i`
(h) `&j`
(i) `&p_1`
(j) `&p_2`
(k) `&p_p_1`
(l) `&p_p_2`
(m) `*p_1`
(n) `*p_2`
(o) `*p_p_1`
(p) `*p_p_2`
(q) `**p_p_1`
(r) `**p_p_2`
(s) `&*p_1`
(t) `*&p_2`

12. Implemente uma função que receba como parâmetro um vetor de números inteiros (`vet`) de tamanho n e retorne quantos números pares, ímpares e primos estão armazenados no vetor. A função deverá obedecer o protótipo:

```
int consulta_vet (int n, int *vet, int *pares, int *impares);
```

13. Seja `f` a seguinte função:

```
int f(char *s, char *t){
  char *p1, *p2;
  for (p1 = s; *p1; p1++) {
    for (p2 = t; *p2; p2++) {
      if (*p1 == *p2) break;
    if (*p2 == '\0') break;
  }
  return p1-s;
}
```

(a) qual é o valor de `f("abcd", "babc")`?

(b) qual é o valor de `f("abcd", "bcd")`?

(c) em geral, qual o valor que `f` retorna ao passar duas *strings* `s` e `t`

14. Considere a seguinte declaração de variáveis.

```
int a[] = {5, 15, 34, 54, 14, 2, 52, 72};
int *p = &a[1], *q = &a[5];
```

(a) Qual é o valor de `*(p+3)`?

(b) Qual é o valor de `*(q-3)`?

(c) Qual é o valor de `q-p`?

(d) A condição `p < q` é verdadeira ou falsa?

(e) A condição `*p < *q` é verdadeira ou falsa?

15. O que será impresso pelo programa abaixo?

```
#include <stdio.h>

typedef struct T{
  int x;
  int y;
};

void f1(T * a);
void f2(int * b);
int main(){
  T a, b, *c, *d;
  c = &a;
  a.x = 2;
  a.y = 4;
  b.x = 2;
```

```
  b.y = 2;
  d = c;
  f1(d);
  b = *d;
  printf("x: %d --- y: %d\n",b.x,b.y);
}

void f1(T *a){
  f2(&(a->x));
  f2(&(a->y));
}

void f2(int *b){
  *b = 2
  *(*b);
}
```

16. Seja a seguinte sequência de instruções em um programa C:

```
int *pti;
int i = 10;
pti = &i;
```

Qual afirmativa é falsa?

(a) pti armazena o endereço de i

(b) *pti é igual a 10

(c) ao se executar *pti = 20; i passará a ter o valor 20

(d) ao se alterar o valor de i, *pti será modificado

(e) pti é igual a 10

17. Se i e j são variáveis inteiras e pi e pj são ponteiros para inteiro, qual atribuição é ilegal?

(a) pi = &i;

(b) *pj = &j;

(c) pj = &*&j;

(d) i = *&*&j;

(e) i = (*pi)+++*pj;

18. Seja a seguinte sequência de instruções:

```
int *pti;
int veti[]={10,7,2,6,3};
pti = veti;
```

Qual afirmativa é falsa?

(a) *pti é igual a 10

(b) *(pti+2) é igual a 2

(c) pti[4] é igual a 3

(d) pti[1] é igual a 10

(e) *(veti+3) é igual a 6

19. Na seqeência de instruções abaixo:

```
float f;
float *pf;
pf = &f;
scanf("%f", pf);
```

(a) Efetuamos a leitura de f

(b) Não efetuamos a leitura de f

(c) Temos um erro de sintaxe

(d) Deveríamos estar usando &pf no scanf

(e) Nenhuma das opções anteriores

20. Escreva um algoritmo recursivo para o calculo de $a * b$, onde a e b são inteiros não-negativos.

21. Execute `fusc(7,0)` (desenhe a árvore de recursão).

```
int fusc (int n, int profund) {
  int i;
  for (i = 0; i < profund; i++)
    printf ("  ");
  printf ("fusc (%d,%d)\n", n, profund);
  if (n = 1)
    return 1;
  if (n % 2 == 0)
    return fusc (n/2, profund+1);
  return fusc ((n-1)/2, profund+1) + fusc ((n+1)/2, profund+1);
}
```