Lista de Exercícios (Ponteiros)

Como submeter sua lista

Resolva os exercícios abaixo e envie suas respostas via SIGAA em um arquivo de texto simples (.TXT), obedecendo o seguinte padrão:

```
Nome: <seu nome>
Matrícula: <sua matrícula>

Questão 1:

  <resposta da questão 1>
---------------------------
Questão 2:

    <resposta da questão 2>
---------------------------
Questão 3:

    <resposta da questão 3>
```

As questões

1. Seja o seguinte trecho de programa:

   ```c
   int i=3,j=5;
   int *p, *q;
   p = &i;
   q = &j;
   ```

   Determine o valor das seguintes expressões:

   ```c
   p == &i;
   *p - *q;
   **&p;
   3 - *p/(*q) + 7;
   ```

2. Mostre o que será impresso por programa supondo que `i` ocupa o endereço 4094 na memória e que uma variável do tipo `int` ocupa 2 bytes na memória virtual.

   ```c
   int i=5, *p;
   p = &i;
   ```

```
printf("%x %d %d %d %d", p,*p+2,**&p,3**p,**&p+4);
```

3. Se `i` e `j` são variáveis do tipo `int` e p e q ponteiros para int, quais das seguintes expressões de atribuição são ilegais?

```
p = i;
q = &j;
p = &*&i;
i = (*&)j;
i = *&j;
i = *&*&j;
q = *p;
i = (*p)++ + *q;
```

4. Determine o que será mostrado pelo seguinte programa (compile-o, execute-o e verifique se foram obtidas as respostas esperadas).

```
int main() {
  int    valor;
  int   *p1;
  float  temp;
  float *p2;
  char   aux;
  char  *nome = "Ponteiros";
  char  *p3;
  int    idade;
  int    vetor[3];
  int   *p4;
  int   *p5;

  /* (a) */
  valor = 10;
  p1 = &valor;
  *p1 = 20;
  printf("%d \n", valor);

  /* (b) */
  temp = 26.5;
  p2 = &temp;
  *p2 = 29.0;
  printf("%.1f \n", temp);

  /* (c) */
  p3 = &nome[0];
  aux = *p3;
  printf("%c \n", aux);

  /* (d) */
```

```
  p3 = &nome[4];
  aux = *p3;
  printf("%c \n", aux);

  /* (e) */
  p3 = nome;
  printf("%c \n", *p3);

  /* (f) */
  p3 = p3 + 4;
  printf("%c \n", *p3);

  /* (g) */
  p3--;
  printf("%c \n", *p3);

  /* (h) */
  vetor[0] = 31;
  vetor[1] = 45;
  vetor[2] = 27;
  p4 = vetor;
  idade = *p4;
  printf("%d \n", idade);

  /* (i) */
  p5 = p4 + 1;
  idade = *p5;
  printf("%d \n", idade);

  /* (j) */
  p4 = p5 + 1;
  idade = *p4;
  printf("%d \n", idade);

  /* (l) */
  p4 = p4 - 2;
  idade = *p4;
  printf("%d \n", idade);

  /* (m) */
  p5 = &vetor[2] - 1;
  printf("%d \n", *p5);

  /* (n) */
  p5++;
  printf("%d \n", *p5);
  return(0);
}
```

5. Determine o que será mostrado pelo seguinte programa (compile-o, execute-o e verifique se

foram obtidas as respostas esperadas).

```
int main(void){
  float vet[5] = {1.1,2.2,3.3,4.4,5.5};
  float *f;
  int i;
  f = vet;
  printf("contador/valor/valor/endereco/endereco");
  for(i = 0 ; i <= 4 ; i++){
    printf("\ni = %d",i);
    printf("vet[%d] = %.1f",i, vet[i]);
    printf("*(f + %d) = %.1f",i, *(f+i));
    printf("&vet[%d] = %X",i, &vet[i]);
    printf("(f + %d) = %X",i, f+i);
  }
}
```

6. Assumindo que `pulo[]` é um vetor do tipo int, quais das seguintes expressões referenciam o valor do terceiro elemento do vetor?

```
*(pulo + 2);
*(pulo + 3);
pulo + 3;
pulo + 2;
```

7. Considerando a declaração `int mat[4], *p, x;`, quais das seguintes expressões são válidas? Justifique.

```
p = mat + 1;
p = mat++;
p = ++mat;
x = (*mat)++;
```

8. O que fazem os seguintes programas em C?

```
int main(){
  int vet[] = {4,9,13};
  int i;
  for(i=0;i<3;i++){
    printf("%d ",*(vet+i));
  }
}
```

```
int main(){
  int vet[] = {4,9,13};
```

```
  int i;
  for(i=0;i<3;i++){
    printf("%X ",vet+i);
  }
}
```

9. Seja `x` um vetor de 4 elementos, declarado da forma `TIPO x[4];`. Suponha que depois da declaração, `x` esteja armazenado no endereço de memória `4092` (ou seja, o endereço de `x[0]`). Suponha também que na máquina seja usada uma variável do tipo `char` ocupa 1 byte, do tipo `int` ocupa 2 bytes, do tipo `float` ocupa 4 bytes e do tipo `double` ocupa 8 bytes. Quais serão os valores de `x+1`, `x+2` e `x+3` se:
   - x for declarado como char?
   - x for declarado como int?
   - x for declarado como float?
   - x for declarado como double?
10. Implemente um programa de computador para testar as suposições da questão anterior e compare as respostas oferecidas pelo programa com as respostas que você idealizou.
11. Suponha que as seguintes declarações tenham sido realizadas:

```
float aloha[10], coisas[10][5], *pf, value = 2.2;
int i=3;
```

    Identifique quais dos seguintes comandos é válido ou inválido:

```
aloha[2] = value;
scanf("%f", &aloha);
aloha = value;
printf("%f", aloha);
coisas[4][4] = aloha[3];
coisas[5] = aloha;
pf = value;
pf = aloha;
```

12. O que é um ponteiro para uma função? Pesquise na Internet referências sobre o assunto e escreva um pequeno programa exemplificando o uso deste recurso.
13. Implemente em linguagem C uma função em um programa de computador que leia n valores do tipo float e os apresente em ordem crescente. Utilize alocação dinâmica de memória para realizar a tarefa.
14. Reimplemente o programa da questão anterior utilizando a função `qsort()` do C. Comente o seu código, explicando o que faz cada uma das linhas.
15. Utilize a ideia do ponteiro para função pela função qsort() para implementar sua própria função de ordenação, mas que seja capaz de ordenar apenas inteiros do tipo `int`. Para isso, sua função deverá receber, entre outros argumentos, um ponteiro para a função de comparação que

determinará como os elementos do array serão ordenados.

16. Procure na internet mecanismos que possibilitem medir tempos de execução de rotinas computacionais. Geralmente, estas medidas são realizadas com o auxílio de funções em C que lêem a hora no sistema (sistemas Unix e Windows geralmente usam funções diferentes). Utilizando os conhecimentos que você obteve com sua pesquisa, meça os tempos de execução das implementações que você criou para os dois problemas de ordenação anteriores, considerando apenas arrays de elementos tipo `int` e compare os resultados obtidos. O que se conclui nesse caso?

17. Escreva uma função em linguagem C que escreva em um vetor a soma dos elementos correspondentes de outros dois vetores (os tamanhos dos vetores devem ser fornecidos pelo usuário). Por exemplo, se o primeiro vetor contiver os elementos 1, 3, 0 e -2, e o segundo vetor contiver os elementos 3, 5, -3 e 1, o vetor de soma terá valores resultantes iguais a 4, 8, -3 e -1. A função deve receber 4 argumentos: os nomes dos três vetores e o número de elementos presentes em cada vetor. Exemplo:

    ```
    soma_vetores(vet1, vet2, resultado, 4);
    ```

18. Crie uma função capaz de realizar multiplicação matricial da forma `C = AxB`. A função deve receber 6 argumentos: os ponteiros para as matrizes A, B e C, o número de linhas e colunas de A e o número de colunas de B (assuma que o número de coluna de A é igual ao número de linhas de B). O resultado da multiplicação deve ficar armazenado em C. Crie um programa para testar sua implementação, capaz de utilizar a função de multiplicação e imprimir as três matrizes. A função criada para multiplicação não deve realizar nenhum tipo de saída de dados no terminal. Exemplo: para multiplicar duas matrizes (A e B) de dimensões 2x3 e 3x4, respectivamente (o resultado deve ficar armazenado em C).

    ```
    multiplica_matrizes(A, B, C, 2, 3, 4);
    ```