Universidade Federal de Viçosa
Centro de Ciências Exatas e Tecnológicas
Departamento de Informática
INF101 – Introdução à Programação II
18/11/2015

Nome: _____ Matr.: _____ Turma: ___

2ª Prova (30%)

1. Assuma que os números inteiros do tipo `short int` do C++ em sua plataforma sejam armazenados em dois bytes e que o endereço inicial do arranjo esteja na posição 102500 da memória. A memória é endereçada por bytes.

   a) Declare um arranjo do tipo `short int` denominado `num` com 10 elementos e inicialize-os com os valores 0, 1, 2, ..., 9.

   b) Declare um ponteiro `nPtr` que aponta para uma variável do tipo `short int`.

   c) Dê dois comandos diferentes que atribuam o endereço inicial do arranjo `num` à variável `nPtr`.

   d) Assumindo que `nPtr` aponta para o início do arranjo `num`, qual é o endereço denotado por `nPtr+8`? Qual é o valor armazenado nesse endereço?

   e) Assumindo que `nPtr` aponta para `num[5]`, qual é o endereço denotado por `nPtr` após a execução de `nPtr -= 4`? Qual é o valor armazenado nesse endereço?

2. Considere o registro Aluno como definido abaixo:

```
struct Aluno {
   unsigned int matr;
   char nome[21];
   float notas[3];
   float nota_final;
};
```

   a) Defina em C++ um ponteiro para uma variável do tipo `Aluno`.

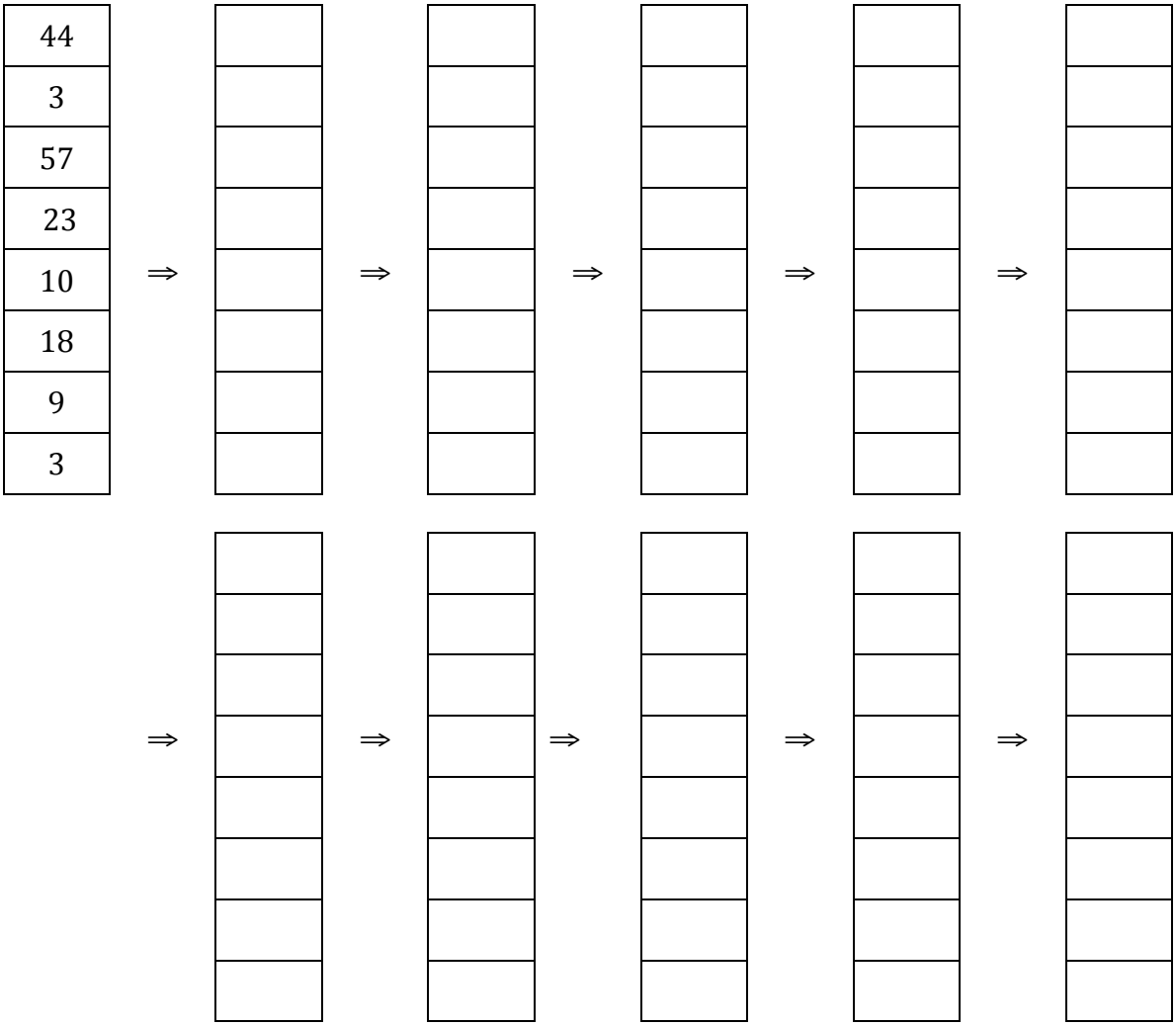
   b) Crie uma variável dinâmica do tipo `Aluno` apontada pelo ponteiro declarado acima.

   c) Inicialize todos os campos da variável dinâmica acima com os dados do aluno de matrícula `82807`, nome `Aprígio da Silveira`, notas `50,7  62,5 84,2`.

   d) Desaloque a variável dinâmica criada acima.

3. Mostre as configurações pelas quais o arranjo `A` com 8 números inteiros passa ao ser ordenado *crescentemente* pelo método *mergesort*. Use os diagramas fornecidos abaixo. Pode haver mais diagramas disponíveis que o necessário. O algoritmo segue abaixo:

```
void mergesort(int A[], int inicio, int fim)
{
   int meio;
   if (inicio < fim)
   {
      meio = (inicio + fim)/2;
      mergesort(A, inicio, meio);
      mergesort(A, meio+1, fim);
      intercale(A, inicio, meio, fim);
   }
}
```



4. Escreva uma função para pesquisar e imprimir todas as chaves `x` do tipo inteiro em um arranjo `A` de números inteiros. O tamanho de `A` é `n`.

5. Preencha as lacunas de modo que cada frase fique correta:

   a) Seja `int *xPtr;` então `*(xPtr + 5)` pode ser expresso também assim: _____ .

   b) Uma variável _____ é criada por meio do operador new e é destruída pelo operador _____ .

   c) O operador seta, −>, em C/C++, é usado com apontadores para _____ .

   d) Os operadores `&` e `*` são _____ um do outro.

   e) A operação de adição de dois ponteiros não é definida, enquanto a _____ de dois ponteiros é definida e o seu resultado é um offset.

   f) A recursividade é útil na prática por apresentar, em geral, maior _____.

   g) O método de busca (ou pesquisa) que divide ao meio o (sub)arranjo a cada comparação de chaves é denominado _____ .

   h) Um ponteiro contém o _____ de uma variável.

   i) Um arranjo em C/C++ não é nada mais nada menos que _____.

   j) As variáveis comuns de C/C++, por exemplo, `int x;` são alocadas _____.

6. Indique com V ou F nos espaços apropriados caso a respectiva frase seja verdadeira ou falsa respectivamente.
   a) ( ) Um ponteiro contém o endereço de memória de alguma outra variável.

   b) ( ) O operador `delete` em C++ destrói o ponteiro a que ele (o operador) se refere.

   c) ( ) O método de ordenação por seleção direta é eficiente podendo ser usado para arranjos grandes, com muitos elementos.

   d) ( ) Um membro `private` de uma classe pode ser usado por qualquer cliente da classe.

   e) ( ) Um ponteiro, quando é declarado apenas, contém o ponteiro nulo por *default.*

   f) ( ) Um parâmetro formal de função para ser passar um arranjo pode ser declarado como um ponteiro.

   g) ( ) Um nome de arranjo em C/C++ é um ponteiro constante.

   h) ( ) O jogo Torres de Hanoi implementado em computador e usando 64 discos nunca terminará.

   i) ( ) Toda função recursiva tem uma equivalente apenas iterativa.

   j) ( ) O algoritmo de pesquisa sequencial tem eficiência de tempo de execução proporcional a $\log_2 n$, onde n é o tamanho do arranjo que contém os dados.