



Disciplina: Linguagens de Programação		Nota: 1,3	Coordenador:
Professor: Abrantes Araújo Silva Filho			
Aluno: Lucas Augusto Ferraz			
Turma: CC3M		Semestre:	Valor: 10 pontos
Data: 27/11/2023		Avaliação: 2ª Avaliação	

INSTRUÇÕES PARA A REALIZAÇÃO DESTA AVALIAÇÃO:

- Esta é a segunda avaliação da disciplina Linguagens de Programação, e refere-se ao conteúdo sobre tipos de dados, expressões e sentenças de atribuição, estruturas de controle, subprogramas, tipos abstratos de dados, e paradigmas das linguagens de programação.
- Esta avaliação contém xx questões discursivas, **todas obrigatórias**, que devem ser respondidas no papel almaço fornecido pelo professor. **ATENÇÃO:** se o papel almaço for preenchido incorretamente, de cabeça para baixo ou de trás para frente, sua nota será penalizada, imediatamente e sem possibilidade de discussão, em 3 (três) pontos.
- No cabeçalho do papel almaço, escreva seu nome completo (não abrevie nenhum nome) e seu número de matrícula. Se você utilizar mais de uma folha de papel almaço, repita o cabeçalho na nova página também. **ATENÇÃO:** se o cabeçalho estiver errado, sua nota será penalizada, imediatamente e sem possibilidade de discussão, em 1 (um) ponto.
- **RESPONDA AS QUESTÕES EM ORDEM!** Se você responder as questões fora de ordem sua prova será penalizada, imediatamente e sem possibilidade de discussão, em 2 (dois) pontos.
- Provas sem identificação no papel almaço não serão corrigidas e a nota do aluno será lançada como 0 (zero), sem possibilidade de discussão.
- As questões podem ser respondidas, na prova, com lápis ou caneta.
- A nota desta prova será calculada de 0 (zero) a 10 (dez). O peso dessa prova na nota do bimestre será de 40%.
- Ao final da prova, **DEVOLVA A PROVA E O PAPEL ALMAÇO** para o professor.
- **Desligue o celular** antes de começar. A avaliação é **individual e sem consulta**.
- Siga todas as normas de **Integridade Acadêmica** da disciplina. Alunos que forem flagrados com qualquer espécie de "cola" ou trocando informações com outros alunos terão suas avaliações recolhidas, as notas zeradas, e a situação será encaminhada para a coordenação para a aplicação das penalidades previstas pela universidade.
- Com 90 minutos de avaliação você terá pouco mais de 2 min por questão, em média.
- Boa avaliação!

1. Analise em detalhes o seguinte código em C, que implementa um tipo de dado não abstrato, chamado de ImagemPB, para armazenar uma imagem em preto e branco (dimensões da imagem e valores dos pixels, de 0-255): [10 pts]

```
1  #include <cs50.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* Struct e tipo de dado para armazenar uma imagem em PB */
6  struct st_imagemPB {
7      int nx;      // n.º de pixels em uma linha (largura da imagem)
8      int ny;      // n.º de pixels em uma coluna (altura da imagem)
9      int np;      // numero total de pixels
10     int p[];     // array para os pixels, em prioridade de coluna
11 };
12 typedef struct st_imagemPB ImagemPB;
13
14 /* Protótipo das funções */
15 ImagemPB *criar_ImagemPB(int x, int y);
16 int get_pixel(ImagemPB *i, int x, int y);
17 void set_pixel(ImagemPB *i, int x, int y, int valor);
18
19 /* Função main */
20 int main(void) {
21     ImagemPB *i = criar_ImagemPB(get_int("Lar: "), get_int("Alt: "));
22     for (int x = 0; x < i->nx; x++) {
23         for (int y = 0; y < i->ny; y++) {
24             set_pixel(i, x, y, get_int("Pixel (%i, %i): ", y, x));
25         }
26     }
27
28     for (int y = 0; y < i->ny; y++) {
29         for (int x = 0; x < i->nx; x++) {
30             printf("%3i\t", get_pixel(i, x, y));
31         }
32         printf("\n");
33     }
34
35     free(i); i = NULL; return 0;
36 }
37
38 /* Função para criar uma imagem pb */
39 ImagemPB *criar_ImagemPB(int nx, int ny) {
40     ImagemPB *t = malloc(sizeof(ImagemPB) + sizeof(int[nx * ny]));
41     t->nx = nx;
42     t->ny = ny;
43     t->np = nx * ny;
44     return t;
45 }
46
47 /* Função para retornar o valor de um pixel */
48 int get_pixel(ImagemPB *i, int x, int y) {
49     // TODO
50 }
51
52 /* Função para atribuir o valor de um pixel */
53 void set_pixel(ImagemPB *i, int x, int y, int valor) {
54     // TODO
55 }
```

Note que no tipo de dado `ImagemPB`, temos o seguinte:

- `nx`: é o número total de pixels em uma linha da imagem, ou seja, `nx` representa a **largura** da imagem (em pixels);
- `ny`: é o número total de pixel em uma coluna da imagem, ou seja, `ny` representa a **altura** da imagem (em pixels);
- `np`: é o número total de pixels na imagem; e
- `p[]`: é um *array* unidimensional que armazena o valor de cada pixel da imagem, sendo que os pixels são armazenados em ordem de **prioridade de coluna**.

Um pixel, em uma imagem preto e branca, pode variar de 0 (o preto mais intenso) até 255 (o branco mais intenso). Considere a seguinte a imagem 2×3 , abaixo:



A imagem acima poderia ser armazenada em um *struct* `ImagemPB` com os seguintes membros:

```
nx = 2;
ny = 3;
np = 6; → nx * ny
p[] = {0, 50, 100, 50, 100, 255};
```

O código C acima começa perguntando a altura e a largura da imagem, e depois solicita cada pixel individualmente e imprime o array em duas dimensões formado pelos pixels da imagem. Para a imagem acima o comportamento esperado do programa é o seguinte:

```
$ ./imagem
Alt: 3
Lar: 2
Valor do pixel (0, 0): 0
Valor do pixel (1, 0): 50
Valor do pixel (2, 0): 100
Valor do pixel (0, 1): 50
Valor do pixel (1, 1): 100
Valor do pixel (2, 1): 255
  0  50
  50 100
 100 255
```

O problema é que o código da função `get_pixel` (na linha 48) e o código da função `set_pixel` (na linha 53) não estão prontos! Faça o seguinte:

- Crie a função `get_pixel` considerando que a matriz bidimensional formada pelos pixels da imagem foi armazenada em um array unidimensional em ordem de prioridade de coluna; e
- Crie a função `set_pixel` também considerando o armazenamento em um *array* unidimensional em ordem de prioridade de coluna.

Obs.: não é necessário incluir nenhum código de validação das entradas dos usuários nem verificações de erros diversos. Faça apenas o código principal das funções.

2. Responda às seguintes questões:

[10 pts]

- O que é um array associativo? Crie um exemplo, em Python, para armazenar as notas da prova final de três alunos.
- O que é um ponteiro? Crie um exemplo, em linguagem C, onde você tem um ponteiro chamado "nome" na área stack de memória, apontando para a string "UVV" na área heap de memória. Também informe como evitar dois problemas comuns com o uso de ponteiros, o vazamento de memória e o ocorrência de ponteiros pendurados.

3. Considere o seguinte programa em uma linguagem muito parecida com a C, mas na qual a ordem de avaliação de expressões é não-determinística: [10 pts]

```
#incluir <entrada_saida.h>

inteiro a = 15;

inteiro teste() {
    a = 27;
    retornar 13;
}

inteiro principal(vazio) {
    a = a + teste();
    imprimir(a);
    retornar 0;
}
```

Há alguma coisa errada com o programa acima? Se não houver nada de errado, explique como o programa funciona e o que será impresso no final da execução da função principal. Se houver alguma coisa de errado, explique o que está errado e o que será impresso no final da execução da função principal.

4. Considere os seguintes programas, A e B, abaixo:

[10 pts]

<pre>// Programa A int x = 10; int y = 5; x = y++; print(x)</pre>	<pre>// Programa B int x = 10; int y = 5; x = ++y; print(x)</pre>
---	---

Qual o valor de x impresso pelo programa A e pelo programa B? Por que são iguais ou diferentes?

5. Em relação aos subprogramas, explique o que são os seguintes termos:

[15 pts]

- Declaração e definição de um subprograma
- Assinatura, cabeçalho e protocolo de um subprograma
- Parâmetro e argumento de um subprograma

6. O n-ésimo número de Fibonacci, $fib(n)$, é dado pela seguinte função matemática recursiva:

[15 pts]

$$fib(n) = \begin{cases} 0, & \text{se } n = 0 \\ 1, & \text{se } n = 1 \\ fib(n-1) + fib(n-2), & \text{se } n > 1 \end{cases} \quad (1)$$

O programa Lisp, abaixo, implementa o cálculo do n-ésimo número de Fibonacci utilizando um algoritmo **recursivo**, exatamente como a função matemática acima:

```
(defun fib(n)
  (if (or (= n 0) (= n 1))
      n
      (+ (fib (- n 1)) (fib (- n 2)))))
```

- (a) Reescreva o código para calcular o n-ésimo número de Fibonacci utilizando um algoritmo iterativo, em Python ou C.
- (b) Algoritmos iterativos fazem uso de **estruturas de repetição**. Quais são as três estruturas de repetição mais comuns nas linguagens de programação? Explique as vantagens e desvantagens de cada uma, indicando em que situações são mais indicadas.
7. Em relação aos fechamentos (*closures*): [15 pts]
- (a) Explique o que é e como funciona um fechamento; e
- (b) Crie um programa, em JavaScript, Python ou Lisp, que demonstre corretamente a criação e o uso de um fechamento. Uma idéia é criar uma função como uma "cria_somador" que retornará um fechamento com uma variável vinculada a um argumento passado para a função cria_somador.
8. Em relação aos diferentes paradigmas das linguagens de programação, responda: [15 pts]
- (a) Qual a característica principal de cada paradigma?
- (b) Crie um quadro comparativo com as vantagens e desvantagens de cada paradigma.

Tabela de correção, não escreva nada aqui!

Questão	Pontos	Nota
1	10	0
2	10	6
3	10	1,0
4	10	0
5	15	6
6	15	0
7	15	0
8	15	0
TOTAL	100	13