PROGRAMAÇÃO 2 – PROVA 1

Prof. Thiago Cavalcante

- Não use celular/computador e não converse com ninguém, a prova é individual.
- Sinta-se à vontade para tirar dúvidas (**razoáveis**) ou pedir esclarecimentos sobre as questões.
- Use **letra legível**! não posso dar nota para algo que não consigo ler.
- Lembre-se de **assinar seu nome nas suas folhas**. Se usar **mais de uma** folha, **enumere cada página**.
- **Seja organizado:** especifique número e letra da questão que você está respondendo e deixe um espaço entre as respostas, para não ficar tudo amontoado. Você pode pegar mais folhas, se precisar.

NOME: _____

1. (1,0 pt) Verdadeiro ou falso.

   (a) Para acessar um elemento em um array, especificamos o nome do array e o valor do elemento em particular.

   (b) Uma definição de array reserva espaço na memória para o array.

   (c) Para indicar que 100 locais devem ser reservados para o array de inteiros **p**, escreva **p[100]**;

   (d) Um programa que inicializa os elementos de um array de 15 elementos com valor 0 precisa conter um laço **for**.

   (e) Estruturas devem conter variáveis com apenas um tipo de dados.

   (f) O nome de uma estrutura **struct nome { ... };** é opcional.

   (g) Os campos de diferentes estruturas devem ter nomes também diferentes.

   (h) A palavra-chave **typedef** é usada para definir novos tipos de dados.

   (i) As estruturas são sempre passadas por referência para funções.

   (j) O operador * retorna o local na memória em que seu operando está armazenado.

2. (2,4 pt) Escreva instruções que executem as tarefas abaixo.

   (a) Imprimir na tela o valor do 7º elemento do array de caracteres **f**.

   (b) Copiar o array **a** na primeira parte do array **b**. Considere a declaração **double a[11], b[34];**

   (c) Ler do teclado os 12 valores do array de ponto flutuante **temp_por_mes**.

   (d) Inicializar a string **u** com o valor **"ufal"**.

   (e) Determinar o comprimento de uma string **s**.

   (f) Definir uma estrutura chamada **ferramenta** que contenha a variável **int numero_ferramenta** e a string **nome_ferramenta** de até 25 caracteres (o '\0' não está incluso na contagem dos caracteres).

   (g) Definir **Ferramenta** como um sinônimo do tipo **struct ferramenta**.

   (h) Usar **Ferramenta** para declarar a variável **a** e o array **b[10]**, ambos do tipo **struct ferramenta**, e a variável **ptr** do tipo ponteiro para **struct ferramenta**.

   (i) Ler do teclado um número de ferramenta e um nome de ferramenta para os campos da variável **a**.

   (j) Atribuir os valores dos campos da variável **a** ao elemento de índice 3 do array **b**.

   (k) Atribuir o endereço do array **b** ao ponteiro **ptr**.

   (l) Imprimir os valores dos campos do elemento de índice 3 do array **b** usando a variável **ptr** e o operador de ponteiro da estrutura (->).

3. (0,8 pt) Dê o **protótipo de função** para cada um dos seguintes itens:

   (a) Função **hipotenusa**, que utiliza dois argumentos de ponto flutuante, **lado1** e **lado2**, e retorna um resultado de ponto flutuante.

   (b) Função **minimo**, que utiliza três inteiros, **x**, **y**, **z**, e retorna um inteiro.

   (c) Função **instrucoes**, que não recebe argumento algum e não retorna um valor.

   (d) Função **intParaFloat**, que utiliza um argumento inteiro, **number**, e retorna um resultado de ponto flutuante.

4. (2,2 pt) Explique o(s) problema(s) nos fragmentos de código abaixo

(a) **[arrays]**

```
int a[3];
printf("$d %d %d\n", a[1], a[2], a[3]);
```

(b) **[arrays]**

```
double f[3] = {1.1, 2.2, 3.3, 4.4};
```

(c) **[arrays]**

```
double d[2][10];
d[1, 9] = 2.345;
```

(d) **[strings]**

```
printf("%s", 'a');
```

(e) **[strings]**

```
char s[12];
strcpy(s, "Seja bem-vindo, visitante!");
```

(f) **[tipos def. pelo prog.]**

```
struct pessoa {
  char nome[15];
  char sobrenome[15];
  int idade;
}
pessoa d;
```

(g) **[funções]**

```
int g(void) {
  printf("Dentro da função g\n");

  int h(void) {
    printf("Dentro da função h\n");
  }
}
```

(h) **[funções]**

```
int soma(int x, int y) {
  int resultado;
  resultado = x + y;
}
```

(i) **[ponteiros]**

```
int *x;
int y;

x = y;
```

(j) **[funções]**

```
void produto(void) {
  int a, b, c, resultado;
  printf("Digite três inteiros: ")
  scanf("%d %d %d", &a, &b, &c);
  resultado = a * b * c;
  printf("Resultado é %d", resultado);
  return resultado;
}
```

(k) **[funções, recursão]**

```
int soma(int n) {
  if (n == 0)
    return 0;
  else
    return n + soma(n);
}
```

5. (3,6 pt) O que os programas abaixo fazem? E o que eles imprimem na tela?

(a) **[arrays, recursão]**

```
#include <stdio.h>

int funcao(int *b, int p) {
  if (p == 1)
    return b[0];
  else
    return b[p - 1] + funcao(b, p - 1);
}

int main() {
  int x;
  int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

  x = funcao(a, 10);
  printf("O resultado é %d\n", x);
  return 0;
}
```

(b) **[strings]**

```
#include <string.h>

int main() {
  char s1[50] = "ufal";
  char s2[50] = "penedo";
  char s3[50];

  strcpy(s3, s1);
  strcat(s3, "_em_");
  strcat(s3, s2);

  printf("%d", strlen(s1) + strlen(s2));
  printf("%s", s3);
  printf("%d", strlen(s3));
}
```

(c) **[strings, ponteiros]**

```
#include <stdio.h>

int funcao(char *s1, char *s2) {
  while (*s1 != '\0' && *s2 != '\0') {
    if (*s1 != *s2) {
      return 0;
    }
    s1++;
    s2++;
  }
  return 1;
}

int main() {
  char str1[80];
  char str2[80];
  int r;
  printf("Digite duas strings: ");
  scanf("%s%s", str1 , str2);

  r = funcao(str1, str2);
  printf("O resultado é %d\n", r);
  return 0;
}
```