UNIVERSIDADE FEDERAL DO RIO GRANDE
ESTRUTURA DE DADOS – Danúbia Espíndola
LISTA – PONTEIROS E ALOCAÇÃO DINÂMICA
danubiafurg@gmail.com

1. Seja o seguinte trecho de programa:

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

Qual é o valor das seguintes expressões ?

a) `p == &i;` TRUE b) `*p - *q` -2 c) `**&p` 3 d) `3* - *p/(*q)+7` 2 5,2

$\left[3 \cdot \left(\frac{-3}{5}\right)\right] + 7$

$2 = 5{,}2$

2. Qual será a saída deste programa supondo que **`i`** ocupa o endereço **`4094`** na memória?

```
main() {
  int i=5, *p;
  p = &i;
  printf("%x %d  %d  %d  %d \n", p,*p+2,**&p,3**p,**&p+4);
}
```

2 4094 7 5 15 9

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

a) `p = &i;` ✓ b) `*q = &j;` ✗ c) `p = &*&i;` ✓ d) `i = (*&)j;` ✗
e) `i = *&j;` ✓ f) `i = *&*&j;` ✓ g) `q = *p;` ✗ h) `i = (*p)++ + *q` ✓

4. Qual serão as saídas do seguinte programa?

```
#include <stdio.h>
#include <conio.h>

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

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

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

(a) 20

(b) 29.0

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

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

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

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

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

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

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

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

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

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

  /* (n) */
  p5++;
  printf("(n) %d \n", *p5);


  return(0);
}
```

C = "A"

D = "r"

C = "A"

F = "r"

g = "0"

h = 31

i = 45

j = 27

l = 31

m = 45

n = 27

5. Qual é o resultado do seguinte programa?

```
#include <conio.h>
#include <stdio.h>
void main(){
      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);
      }
}
```

-> Imprime o índice que está sendo iterado

-> Imprime o valor que há dentro índice iterado

-> Imprime o mesmo valor, porém o acessa como um ponteiro não como um vetor

-> Imprime o endereço de memória do índice iterado

-> Imprime o mesmo endereço de memória, porém o acessa como um ponteiro não como um vetor

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

a) `*(pulo + 2)` b) `*(pulo + 4)` c) `pulo + 4` d) `pulo + 2`

7. Supor a declaração: `int mat[4], *p, x;` Quais expressões são válidas? Justifique:

a) `p = mat + 1;` ✓ É permitido fazer um ponteiro apontar para qualquer um dos índices de um dado vetor.

b) `p = mat++;` ✗ Não é permitido realizar qualquer incremento em um ponteiro enquanto utiliza o mesmo para definir uma outra variável.

c) `p = ++mat;` ✗

d) `x = (*mat)++;` ✓ É permitido realizar qualquer incremento em um inteiro enquanto o utiliza para definir outra variável.

8. O que fazem os seguintes programas?

```
#include <conio.h>
#include <stdio.h>
void main(){
  int vet[] = {4,9,13};
  int i;
  for(i=0;i<3;i++){
    printf("%d ",*(vet+i));
  }
}
```

Imprime o valor dos elementos de vet

```
#include <conio.h>
#include <stdio.h>
void main(){
  int vet[] = {4,9,13};
  int i;
  for(i=0;i<3;i++){
    printf("%X ",vet+i);
  }
}
```

Imprime os endereços de memória dos elementos de vet

```
#include <conio.h>
#include <stdio.h>
void main(){
  int vet[] = {4,9,13};
  int i;
  for(i=0;i<3;i++){
    printf("%d ",vet+i);
  }
}
```

Aponta erro de compilação, pois estamos tentando imprimir um endereço de memória na formatação de números inteiros.

9. O que faz o seguinte programa quando executado?

```
#include <conio.h>
#include <stdio.h>
void main() {
      int vet[] = {4,9,12};
      int i,*ptr;
      ptr = vet;
      for(i = 0 ; i < 3 ; i++) {
            printf("%d ",*ptr++);
      }
}
```

(a)

Imprime o valor que há dentro do índice de vet que está sendo iterado e depois incrementa o endereço de memória para qual ptr está apontando, para assim iterar o próximo índice de vet.

```
#include <conio.h>
#include <stdio.h>
void main(){
      int vet[] = {4,9,12};
      int i,*ptr;
      ptr = vet;
      for(i = 0 ; i < 3 ; i++) {
            printf("%d ",(*ptr)++);
      }
}
```

(b)

O ponteiro não é incrementado em nenhum momento, então o código imprime o valor de vet[0] e depois incrementa este valor. Este processo é repetido 3 vezes, onde ao final do programa, o código terá feito a seguinte expressão: 4 5 6.

10. Seja **vet** um vetor de 4 elementos: **TIPO vet[4]**. Supor que depois da declaração, **vet** esteja armazenado no endereço de memória `4092` (ou seja, o endereço de `vet[0]`). Supor também que na máquina 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.

Qual o valor de `vet+1`, `vet+2` e `vet+3` se:

a) **vet** for declarado como `char`? Vet+1 = 4093 | Vet+2 = 4094 | Vet+3 = 4095
b) **vet** for declarado como `int`? Vet+1 = 4094 | Vet+2 = 4096 | Vet+3 = 4098
c) **vet** for declarado como `float`? Vet+1 = 4096 | Vet+2 = 4100 | Vet+3 = 4104
d) **vet** for declarado como `double`? Vet+1 = 4100 | Vet+2 = 4108 | Vet+3 = 4116

11. Faça um programa que leia um valor *n* e crie dinamicamente um vetor de *n* elementos e passe esse vetor para uma função que vai ler os elementos desse vetor. Depois, no programa principal, o vetor preenchido deve ser impresso. Além disso, antes de finalizar o programa, deve-se liberar a área de memória alocada.

12. Faça uma função que receba um valor *n* e crie dinamicamente um vetor de *n* elementos e retorne um ponteiro. Crie uma função que receba um ponteiro para um vetor e um valor *n* e imprima os *n* elementos desse vetor. Construa também uma função que receba um ponteiro para um vetor e libere esta área de memória. Ao final, crie uma função principal que leia um valor *n* e chame a função criada acima. Depois, a função principal deve ler os *n* elementos desse vetor. Então, a função principal deve chamar a função de impressão dos *n* elementos do vetor criado e, finalmente, liberar a memória alocada através da função criada para liberação.

13. Construa um programa (main) que aloque em tempo de execução (dinamicamente) uma matriz de ordem *m* x *n* (linha por coluna), usando 1+*m* chamadas a função malloc. Agora, aproveite este programa para construir uma função que recebendo os parametros m e n aloque uma matriz de ordem *m* x *n* e retorne um ponteiro para esta matriz alocada. Crie ainda uma função para liberar a área de memória alocada pela matriz. Finalmente, crie um novo programa (main) que teste/use as duas funções criadas acima.