

INSTITUTO FEDERAL  
GOIÁS  
Campus Inhumas

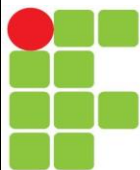
Ministério da Educação  
Secretaria de Educação Profissional e Tecnológica  
Instituto Federal de Educação, Ciência e Tecnologia de Goiás  
Campus de Inhumas  
Coordenação da Área de Informática

**ESTRUTURAS DE DADOS**

Curso:	BACHARELADO EM SISTEMAS DE INFORMAÇÃO		
Ano Letivo: 1º	Período: 3º	Ano: 2017	Professor (a): Rogério Sousa e Silva
Aluno:			

**PONTEIROS E ALOCAÇÃO DINÂMICA DE MEMÓRIA**

01. O que é um ponteiro? Para que serve um ponteiro?  
**Ponteiro é um tipo especial de variável que armazena endereços de memória.**
02. Explique as declarações abaixo:
- `int *x;` **Declaração de um ponteiro para um inteiro**
  - `int x;` **Declaração de uma variável inteira.**
03. Quais as maneiras de atribuir valor à variável a, após as declarações abaixo:
- `int a, *p;`  
**a = <valor>; Diretamente pela variável**  
**p = &a; \*p = <valor>; Através do ponteiro para a variável**
04. Na expressão **"float \*pont;"**, o que é do tipo float?
- A variável pont; **ERRADO – pont é um ponteiro para floats.**
  - O endereço de pont; **ERRADO – pont é um endereço!**
  - A variável apontada por pont; **CORRETO**
  - Nenhuma das anteriores. **ERRADO**
05. Considerando que o endereço da variável inteira x foi atribuído a um ponteiro p (1), quais das seguintes expressões estão corretas? Justifique: (1)p = &x;
- x == &p; **ERRADO, x possui valor inteiro e &p é o endereço da variável p.**
  - x == \*p; **CORRETO, o valor da variável apontada por p (\*p) recebeu o valor de x (1).**
  - p == \*x; **ERRADO, x não é um ponteiro**
  - p == &x; **CORRETO, p = &x(1), logo p == &x.**
06. Considere o seguinte trecho de código e responda:
- ```
int x;  
int *p;  
p = &x;
```
- qual dos comandos abaixo estão corretos (justifique sua resposta):
- `scanf("%i", &x);` **CORRETO, forma usual de entrada de dados inteiros pela função scanf.**
  - `scanf("%i", *x);` **ERRADO, x não é um ponteiro**
  - `scanf("%i", p);` **CORRETO, alternativa de entrada pelo ponteiro que endereça x.**
  - `scanf("%i", &p);` **ERRADO, apesar de sintaticamente aceitável, não é conveniente que o usuário entre com um endereço.**
07. Considerando que o endereço da variável x foi atribuída a um ponteiro p (1)p = &x;, escreva as instruções necessárias para dividir o valor de x por 5, sem utilizar diretamente a variável x.  
**<variável inteira> = \*p / 5; Através do ponteiro p que aponta para x.**
08. Considerando o código abaixo, quais valores serão impressos?
- ```
int main(){  
    int i=3, j=5;  
    int *p = &i, *q = &j;  
    printf("%i\n", p == &i);  
    printf("%i\n", *p-*q);  
    printf("%i\n", **&p);  
}
```
- 1, verdadeiro que p==&i.**  
**-2, 3 - 5**  
**3, o mesmo que \*p**  
**\*\*&p – Suponha que fossem declarados**  
**int x – um inteiro x;**  
**int \*p – um ponteiro p**



INSTITUTO FEDERAL  
GOIÁS  
Campus Inhumas

Ministério da Educação  
Secretaria de Educação Profissional e Tecnológica  
Instituto Federal de Educação, Ciência e Tecnologia de Goiás  
Campus de Inhumas  
Coordenação da Área de Informática

ESTRUTURAS DE DADOS

Curso:	BACHARELADO EM SISTEMAS DE INFORMAÇÃO		
Ano Letivo: 1º	Período: 3º	Ano: 2017	Professor (a): Rogério Sousa e Silva
Aluno:			

int \*\*pp – um ponteiro para um ponteiro pp

p = &x;

pp = &p;

assim, \*p seria o valor de x, e \*pp seria o valor do ponteiro p que é o endereço de x, \*&p é o endereço de p, \*(\*&)pp é o valor do endereço de p, ou seja o valor de x.

Veja:

```
printf("Resultados: p[%x] pp[%x] *p[%i] *pp[%x] &p[%x] *&p[%x] **&p[%i]\n\n", p, pp, *p, *pp, &p, *&p, *&p);
```

```
Resultados: p[60ff28] pp[60ff24] *p[10] *pp[60ff28] &p[60ff24] *&p[60ff28] **&p[10]
```

```
return 1;
```

```
}
```

09. Qual a saída deste programa?

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

p[0060FF2C] \*p+2[7] \*\*&p[5] 3\*\*p[15] \*\*&p+4[9], o valor impresso de p depende de cada execução.

10. Considerando as declarações do exercício 8 quais das seguintes atribuições estão incorretas? Justifique:

```
int i=3, j=5;
```

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

a. p=&i; CORRETA, ponteiro p recebe endereço de i

b. \*q=&j; ERRADO, valor de quem q aponta não pode receber um endereço.

c. p=&\*&i; CORRETO, mas não é usual, o mesmo que &i.

d. i=(\*&)j; ERRADO, operadores sem operando, devido aos parênteses. Sem os parênteses estaria correto e seria o mesmo que i=j;

e. i=\*&\*&j; CORRETO, o mesmo que i=j;

f. q=&p; CORRETO, porém o compilador adverte sobre incompatibilidade de tipos, neste caso o ponteiro q receberá o endereço do ponteiro p e não o endereço de quem p aponta.

g. i=(\*p)+++\*q; CORRETO, pós incremento do valor que p aponta somado com o valor de quem q aponta. Note que a prioridade é do operador de pós incremento sobre a soma, em ++, primeiro o ++ e em seguida +.

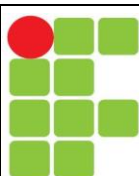
11. O seguinte programa está correto? Justifique sua resposta:

```
int main() {  
    int *pq;  
    pq = (int *) malloc(sizeof(int));  
    *pq = 3;  
    printf("Resultado: %d\n", ++*pq);  
    return 1;  
}
```

CORRETO, o ponteiro pq aponta para o endereço de memória obtido por alocação (comando malloc da biblioteca stdlib.h). Toda a manipulação é realizada sobre o ponteiro, não há a necessidade de uma variável estática.

12. Considere o seguinte trecho de programa:

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



INSTITUTO FEDERAL  
GOIÁS  
Campus Inhumas

Ministério da Educação  
Secretaria de Educação Profissional e Tecnológica  
Instituto Federal de Educação, Ciência e Tecnologia de Goiás  
Campus de Inhumas  
Coordenação da Área de Informática

**ESTRUTURAS DE DADOS**

<b>Curso:</b>	BACHARELADO EM SISTEMAS DE INFORMAÇÃO		
<b>Ano Letivo:</b> 1º	<b>Período:</b> 3º	<b>Ano:</b> 2017	<b>Professor (a):</b> Rogério Sousa e Silva
<b>Aluno:</b>			

q = &j;

Qual é o valor das seguintes expressões?

- p == &i; **1 (verdadeiro)**
- \*p - \*q; **-2**
- \*\*&p; **5**
- 3\* - \*p/( \*q)+7; **6 (observe as prioridades dos operadores)**

13. Quais serão as saídas do seguinte programa?

```
#include <stdio.h>

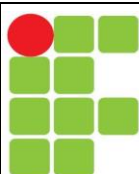
int main() {
    int    valor;
    int    *p1;
    float  temp;
    float  *p2;
    char   aux;
    char   *nome = "Algoritmos";
    char   *p3;
    int     idade;
    int     vetor[3];
    int     *p4;
    int     *p5;

    /* (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);
    /* (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);
}
```

**RESPOSTA IMPRESSA:**

- |          |        |
|----------|--------|
| (a) 20   | (g) o  |
| (b) 29.0 | (h) 31 |
| (c) A    | (i) 45 |
| (d) r    | (j) 27 |
| (e) A    | (l) 31 |
| (f) r    | (m) 45 |
|          | (n) 27 |



INSTITUTO FEDERAL  
GOIÁS  
Campus Inhumas

Ministério da Educação  
Secretaria de Educação Profissional e Tecnológica  
Instituto Federal de Educação, Ciência e Tecnologia de Goiás  
Campus de Inhumas  
Coordenação da Área de Informática

ESTRUTURAS DE DADOS

Curso:	BACHARELADO EM SISTEMAS DE INFORMAÇÃO		
Ano Letivo: 1º	Período: 3º	Ano: 2017	Professor (a): Rogério Sousa e Silva
Aluno:			

14. Qual é o resultado do seguinte programa?

```
#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);
    }
}
```

RESPOSTA:

índice	valor	valor	endereço	endereço
i = 0	vet[0] = 1.1	*(f + 0) = 1.1	&vet[0] = 60FF14	(f + 0) = 60FF14
i = 1	vet[1] = 2.2	*(f + 1) = 2.2	&vet[1] = 60FF18	(f + 1) = 60FF18
i = 2	vet[2] = 3.3	*(f + 2) = 3.3	&vet[2] = 60FF1C	(f + 2) = 60FF1C
i = 3	vet[3] = 4.4	*(f + 3) = 4.4	&vet[3] = 60FF20	(f + 3) = 60FF20
i = 4	vet[4] = 5.5	*(f + 4) = 5.5	&vet[4] = 60FF24	(f + 4) = 60FF24

OBSERVE as formas de manipulação dos elementos e dos endereços em um vetor.

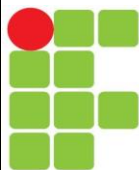
15. Assumindo que **pulo [10]** é um vetor do tipo int, quais das seguintes expressões referenciam o valor do terceiro elemento da matriz?
- \*(pulo + 2) **CORRETO**
  - \*(pulo + 4) **Valor do quinto elemento**
  - pulo + 4 **Endereço do quinto elemento**
  - pulo + 2 **Endereço do terceiro elemento**
16. Suponha a declaração: int mat[4], \*p, x; Quais expressões são válidas? Justifique:
- p = mat + 1; **VÁLIDA, p recebe o endereço do segundo elemento do vetor**
  - p = mat++; **INVÁLIDA, mat é o endereço do vetor, não pode ser incrementado pelo operador de pós-incremento.**
  - p = ++mat; **INVÁLIDA, mat é o endereço do vetor, não pode ser incrementado pelo operador de pré-incremento.**
  - x = ++ (\*mat); **VÁLIDA, pré-incrementa o valor do elemento 0 do vetor e o atribui para x.**
17. O que fazem os seguintes programas?

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

Mostra os elementos do vetor  
Observe que vet é o endereço do primeiro elemento e vet + i o endereço do i-ésimo elemento do vetor. O valor do elemento é apresentado devido ao \*.

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

Mostra os endereços dos elementos do vetor  
Observe que vet é o endereço do primeiro elemento e vet + i o endereço do i-ésimo elemento do vetor



INSTITUTO FEDERAL  
GOIÁS  
Campus Inhumas

Ministério da Educação  
Secretaria de Educação Profissional e Tecnológica  
Instituto Federal de Educação, Ciência e Tecnologia de Goiás  
Campus de Inhumas  
Coordenação da Área de Informática

**ESTRUTURAS DE DADOS**

Curso:	BACHARELADO EM SISTEMAS DE INFORMAÇÃO		
Ano Letivo: 1º	Período: 3º	Ano: 2017	Professor (a): Rogério Sousa e Silva
Aluno:			

18. O que fazem os seguintes programas quando executados?

<pre>#include &lt;stdio.h&gt; void main() {     int vet[] = {4,9,12};     int i,*ptr;     ptr = vet;     for(i = 0 ; i &lt; 3 ; i++) {         printf("%d ",*ptr++);     } }</pre> <p>Mostra os elementos do vetor OBSERVE a forma de saltar entre os elementos. O incremento no *ptr indica um salto para o próximo elemento do vetor</p> <p>(a)</p>	<pre>#include &lt;stdio.h&gt; void main(){     int vet[] = {4,9,12};     int i,*ptr;     ptr = vet;     for(i = 0 ; i &lt; 3 ; i++) {         printf("%d ",(*ptr)++);     } }</pre> <p>Mostra o primeiro elemento do vetor e os valores resultantes da soma de mais um a cada repetição. OBSERVE, neste caso, o incremento no (*ptr) indica um incremento no valor do elemento.</p> <p>(b)</p>
---	--

19. Seja vet um vetor de 4 elementos: vet[4]. Supor que depois da declaração, vet esteja armazenado no endereço de memória 0060FF1C (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 4 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:

- vet for declarado como char? 0060FF1D, 0060FF1E, 0060FF1F
- vet for declarado como int? 0060FF20, 0060FF24, 0060FF28
- vet for declarado como float? 0060FF20, 0060FF24, 0060FF28
- vet for declarado como double? 0060FF24, 0060FF2C, 0060FF34