INSTITUTO FEDERAL
GOIÁS

Atividade sobre Ponteiros

1. Explique a diferença entre:
   a. `p++;`
   b. `(*p)++;`
   c. `*(p++);`

2. O que pode acontecer se for atribuído algum valor a um ponteiro que não tenha sido inicializado.

```
float *p;
*p = 2000;
```

3. Seja o seguinte trecho de programa:

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

   Qual é o valor das expressões?

   a. `P == &i`
   b. `*p - *q`
   c. `**&p`
   d. `#* - *0 / (*q) + 7`

4. Assumindo que o endereço de vox foi atribuído a um ponteiro variável invox, quais das seguintes expressões são verdadeiras?
   a. `vox == &invox`
   b. `vox == *invox`
   c. `invox == *vox`
   d. `invox == &vox`
   a. `&vox == &invox`

5. Qual é a maneira correta de referenciar o valor de ch, assumindo que o endereço de ch foi atribuído ao ponteiro indica?
   a. `*indica`
   b. `int *indica`
   c. `&indica`
   d. `&ch`
   e. `*ch`

6. Qual o resultado do código abaixo? Explique cada linha.

```
int x = 100, *p, **pp;
p = &x;
pp = &p;
printf("Valor de pp : %d\n", **pp);
```

7. Faça um programa usando ponteiros, para ordenar 3 números e mostrá-los ordenados na tela.

8. Escreva um programa que a partir da leitura de duas Strings qualquer, informe o número de caracteres da concatenação das duas Strings.

9. Assumindo que o endereço da variável var foi atribuído a um ponteiro variável ptrvar, escreva um programa que não usa var e dívida var por 10.

10. Faça um programa que declare 3 variáveis, dos tipos inteiro, real e caracter, atribua valores para estas variáveis e mostre seus valores assim como seu tamanho em bytes e seu endereço na memória (inteiro e hexa).

11. Crie um programa que declare duas variáveis (inteiro e real) atribua valores para elas e crie dois ponteiros, cada um apontando para uma destas variáveis. Mostre na tela: O valor e o endereço de memória das variáveis, o valor do ponteiro, o endereço do ponteiro e o valor apontado pelo ponteiro.

12. Implemente um programa em que obtenha do teclado o tamanho (inteiro) de uma frase (string), após isso aloque dinamicamente (malloc) um vetor de caracteres com tamanho obtido e então leia novamente do teclado a frase digitada pelo usuário armazenando-a no novo vetor alocado.

13. Implemente um programa em C que obtenha do teclado uma frase e armazene-a em uma variável ponteiro. Seu programa irá realocar o tamanho deste ponteiro (realloc) a cada caracter digitado. O programa só deve terminar de capturar cacteres quando a tecla enter for pressionada. Após isso imprime o conteúdo desse ponteiro, o seu total de caracteres e as letras que se encontram nas posições pares, incluindo a posição 0.

14. Em C, não se pode fazer uma função que retorne dois valores, exceto com o uso de estruturas.

15. Uma forma de solucionar esta restrição é com o uso de ponteiros, pois a função pode receber qual - quer numero de variáveis por referência. Faca programa, que possuindo duas variáveis inteiras a=2 e b=3, chame a função void troca(int *a1, int *b1), que deve fazer a inversão dos valores de a e b, ou seja, b passa a valer o que a valia e a passa a valer o que b valia. Apos a chamada da função troca(), imprimir os valores de a e b. A função troca também deve ser implementada, e deve ter tipo de retorno void.

16. Declare vetores de inteiro, char, float, double, long int com 5 posições, da seguinte forma:

```
int v[5] = { 2, 5, 1, 4, 0 };
char c[5] = { 'a','b', 'm', '4', '-' };
float v[5] = { 2.66, 0.125, 1.0, 4.99, 2.009 };
```

Usando a função printf, com o argumento “%p”, e com vetores apontando para cada um dos tipos de dados, descubra quantos bytes é alocado pelo seu compilador a cada tipo de dados. Não se esqueça que para um vetor apontar para um vetor de float, ele deve ser do tipo float. Como se sabe que em um vetor as posições são contínuas, se for

impresso o endereço de duas posições, pela diferença entre os dois endereços pode-se descobrir quantos bytes são alocados.

17. O que imprime o programa a seguir? Tente entendê-lo e responder. A seguir, execute-o e comprove o resultado.

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
	int t, i, M[3][4];

	for (t = 0; t < 3; ++t) {
		for (i = 0; i < 4; ++i) {
			M[t][i] = (t * 4) + i + 1;
		}
	}

	for (t = 0; t < 3; ++t) {
		for (i = 0; i < 4; ++i) {
			printf("%3d ", M[t][i]);
		}
		printf("\n");
	}

	system("pause");
	return 0;
}
```

18. Qual o valor de y no final do programa? Tente primeiro descobrir e depois verifique no computador o resultado. A seguir, escreva um /* comentário */ em cada comando de atribuição explicando o que ele faz e o valor da variável à esquerda do '=' após sua execução.

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
	int y, *p, x;
	y = 0;
	p = &y;
	x = *p;
	x = 4;
	(*p)++;
	x;
	(*p) += x;
	printf("y = %d\n", y);

	system("pause");
	return 0;
}
```

19. Reescreva o programa abaixo usando ponteiros

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
	float matrx[50][50];
	int i, j;

	for (i = 0; i < 50; i++) {
		for (j = 0; j < 50; j++) {
			matrx[i][j] = 0.0;
		}
	}

	system("pause");
	return 0;
}
```

20. Diga quais expressões abaixo são válidas ou não.

```c
int vetor[10];
int *ponteiro;
```

Considere as declarações.

a. `vetor = vetor + 2;`
b. `vetor++;`
c. `vetor = ponteiro;`
d. `ponteiro = vetor;`
e. `ponteiro = vetor+2;`

21. Explique o que faz o programa abaixo

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
	float vet[5] = { 1.1, 2.2, 3.3, 4.4, 5.5 }, *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\n", i, f + i);
	}

	system("pause");
	return 0;
}
```

22. Assumindo que M1[] é um vetor do tipo int, quais das seguintes expressões referenciam o valor do terceiro elemento de M1?

```
a. *(M1 + 2)
b. *(M1 + 4)
c. M1 + 4
d. M1 + 2
```

23. Considere a declaração:

```c
int mat[4], *p, x;
```

Quais expressões são válidas? Justifique:

```
a. p = mat + 1;
b. p = mat++;
c. p = ++mat;
d. x = (*mat)++;
```

24. Explique o que o programa abaixo faz

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {

	int vet[] = { 4, 9, 12 }, i, *ptr;
	ptr = vet;

	for (i = 0; i < 3; i++)
		printf("%d ", *ptr++);

	system("pause");
	return 0;
}
```

25. 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?
    b. vet for declarado como int?
    c. vet for declarado como float?
    d. vet for declarado como double?

26. Considere um micro cujo barramento de endereços possui 16 bits. Considere um ponteiro p apontando para a primeira posição de memória da figura abaixo.

Responda:

a. Qual o valor de p?
b. Qual o valor de *p?
c. Qual o valor de &p?
d. Qual o valor de *(p + 1)?
e. Qual o valor de (p + 5)?
f. Qual o valor de (p + A)?
g. Qual o valor de *(p + 2)?

27. Verifique o programa abaixo. Encontre o seu erro e corrija-o para que escreva o número 10 na tela.

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {

	int x, *p, **q;
	p = &x;
	q = &p;
	x = 10;

	printf("\n%d\n", &q);

	system("pause");
	return 0;
}
```

28. Escreva um programa que declare uma matriz 100x100 de inteiros. Você deve inicializar a matriz com zeros usando ponteiros para endereçar seus elementos. Preencha depois a matriz com os números de 1 a 10000, também usando ponteiros.

29. Implemente um algoritmo em C para cadastrar a nota de vários alunos de uma disciplina utilizando alocação dinâmica. Crie uma função para receber o código de matrícula de um aluno e retornar se o aluno está aprovado ou não na disciplina.

30. Criar uma função para receber o numerador e o denominador de uma fração e retornar o valor decimal correspondente. Por exemplo, Entrada → 1,2 (referente à fração ½), Saída 0,5.

31. Criar uma função para receber o valor de uma determinada temperatura em graus Fahrenheit e exibir o valor correspondente em Celsius. A fórmula de conversão é C=5/9 (F-32))

32. Implemente uma função em C que realize a troca dos elementos de um vetor de tamanho 2. Use passagem de parâmetros por referência. Crie a função main() invocando a função implementada.

33. Um ponteiro pode ser usado para dizer a uma função onde ela deve depositar o resultado de seus cálculos. Escreva uma função hm que converta minutos em horas-e-minutos. A função recebe um inteiro mnts e os endereços de duas variáveis inteiras, digamos h e m, e atribui valores a essas variáveis de modo que m seja menor que 60 e que 60*h + m seja igual a mnts. Escreva também uma função main que use a função hm.

34. Escreva uma função mm que receba um vetor inteiro v[0..n-1] e os endereços de duas variáveis inteiras, digamos min e max, e deposite nessas variáveis o valor de um elemento mínimo e o valor de um elemento máximo do vetor. Escreva também uma função main que use a função mm.

35. Crie um algoritmo em C, utilizando a técnica de modularização, para calcular a média final de vários alunos de uma disciplina.