ECLP
Lista 3 - 18/03/2020

Revisão de uso de memória em C.

1. Quais das seguintes instruções são corretas para declarar um ponteiro?
   a) int _ptr x; b) * int ptr; c) int * ptr; d) *x;

2. Dada a definição de variáveis:
```
int i, *pi, **ppi;
float f, *pf, **ppf;
```
   Quais das atribuições são permitidas?
   a) i = f; e) *pf =10; i) ppf = &pf;
   b) pf = &i; f) f = i; j) **ppi = 100;
   c) *pf = 5.9; h)*pi = 7.3; i) pi = i;

3. Qual é a forma correta de fazer referência à variável ***x1***, assumindo que o endereço de ***x1*** foi atribuído a ***ptr***?
   a) *ptr; b) x1; c) int * ptr; d) *x1;

4. Na expressão ***int *ptr;*** que é do tipo float?
   a) a variável *ptr* c) a variável apontada por *ptr*
   b) o endereço de *ptr* d) nenhuma das opções anteriores

5. Se o endereço de ***x*** foi atribuído a um apontador ***xptr***, quais das expressões a seguir são verdadeiras?
   a) x == & xptr b) x == *xptr c) xptr == *x d) xptr == &x

6. Supor a declaração: ***int mat[4], *p, x***;
   Quais expressões são válidas? Justifique.
   a) p = mat + 1; b) p = mat++; c) p = ++mat; d) x = (*mat)++;

7. Para ler o valor de ***x*** foi atribuído o endereço de ***x*** à variável ***xptr***, a instrução a seguir é correta? Justificar.
   scanf (“%d”, *xptr);

8. Que instrução deve ser agregada no programa a seguir para que ele execute corretamente?
```
int main(){
     int x, *xptr;
     *xptr = 5;
}
```

9. Supondo que o endereço da variável ***x*** foi atribuído a um ponteiro ***xptr***, escreva uma expresso que não use a variável ***x*** para dividir ***x*** por 3.

10. Dada a sequência de instruções em C:
```
int *p;
```

```
int i =5;
p = &i;
```

Quais das afirmativas a seguir são verdadeiras e quais falsas?

a) ***p*** armazena o endereço de ***i***.
b) ****p*** é igual a ***5***.
c) Quando é executada a instrução ****p = 10***; ***i*** passa a valer ***10***.
d) *p* é igual a 10.
e) Quando se modifica o valor de ****p***, o valor de ***i*** é modificado.
f) Quando se modifica o valor de ***i***, o valor de ****p*** é modificado.

11. 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);
}
```

12. Assumindo que **vet[]** é um vetor do tipo int, quais das seguintes expressões referenciam o valor do terceiro elemento do vetor?

a) *(vet + 2) b) *(vet + 4) c) vet + 4 d) vet + 2

13. 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?

14. Qual é o resultado do seguinte programa?

```
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);
       }
}
```

15. O que fazem os seguintes programas?

```
int main(){
       int vet[] = {4,9,13};
       int i;
       for(i=0; i<3; i++){
              printf("%d ",*(vet+i));
       }
```

```
	return 0;
}

int main(){
	int vet[] = {4,9,13};   int i;
	for(i=0; i<3; i++){
		printf("%d ",vet+i);
	}
}
```

16. Quais 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);
	/* (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;
}
```