UNILASALLE RIO DE JANEIRO

PROGRAMAÇÃO II - RELAÇÃO DE EXERCÍCIOS

PONTEIROS

Prof: Maria Inês Vasconcellos Furtado

1. Quais das seguintes instruções são corretas para declarar um ponteiro?

```
a)int _ptr x;        b) int *ptr;        c) *int ptr;        d) *x;
```

2. Como referenciar **ch**, assumindo que o endereço de **ch** foi atribuído ao ponteiro **indica**?

```
a)*indica;      b)int *indica;      c)*indic;      d)ch      e)*ch;
```

3. Na expressão **float *pont**; o que é do tipo float?

a) a variável pont.
b) o endereço de pont.
c) a variável apontada por pont.
d) nenhuma das anteriores.

4. Assumindo que o endereço de **num** foi atribuído a um ponteiro **pnum**, quais das seguintes expressões são verdadeiras?

```
a) num == &pnum      b) num == *pnum      c) pnum == *num      d) pnum == &num
```

5. Assumindo que queremos ler o valor de **x**, e o endereço de **x** foi atribuído a **px**, a instrução **scanf ("%d", *px);** é correta? Por que?

6. Qual é a instrução que deve ser adicionada ao programa seguinte para que ele trabalhe corretamente?

```
main ( )
{    int j, *pj;
     *pj = 3;  }
```

7. Assumindo que o endereço da variável x foi atribuído a um ponteiro px, escreva uma expressão que não usa x e divida x por 5.

8. Qual o valor das seguintes expressões:

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

```
a) p == &i      b) *p - *q      c) **&p
```

9. Se **i** e **j** são variáveis inteiras e **p** e **q** ponteiros para inteiros, quais das expressões de atribuição são ilegais?

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

10. Qual afirmativa é falsa, considerando a seguinte sequência de instruções em um programa C:

```
int *pti;
int i = 10;
pti = &i;
```

a) pti armazena o endereço de i
b) *pti é igual a 10
c) ao se executar *pti = 20; i passará a ter o valor 20
d) ao se alterar o valor de i, *pti será modificado
e) pti é igual a 10

11. Considerando as variáveis e ponteiros definidos abaixo; quais são as atribuições permitidas?

```
int x, *ptx, **pp;
float a, *pta, **pf;
```

```
a) x = 100;      d) *pf = &a;      g) *ptx = 20;      j) pf = &pta;
b) *pta = &a;    e) pp = &pta;     h) ptx = &x;
c) ptx = &a;     f)**pf = 7.9;     i) pp = &x;
```

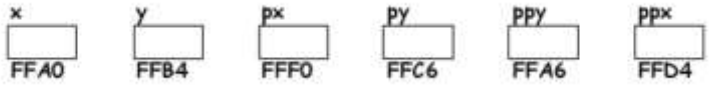
12. Considerando as variáveis e ponteiros definidos abaixo; quais são as atribuições permitidas?

```
int i, *pi, **ppi;
float f, *pf, **ppf;
```

a) `i = f;`
b) `pf = &i;`
c) `*pf = 5.9;`
d) `*ppi = &pi;`
e) `*pf = 10;`
f) `f = i;`
g) `pi = &f;`
h) `*pi = 7.3;`
i) `ppf = &pf;`
j) `**ppi = 100;`

13. Dadas as declarações abaixo; qual é o valor dos itens:

```
int x = 10, *px = &x, **ppx = &x;
float y = 5.9, *py = &y, **ppy = &py;
```



a) x =
b) *py =
c) px =
d) &y =
e) *px =
f) y =
g) *ppx =
h) py =
i) &x =
j) py++ =
k) *px-- =
l) **ppy =
m) &ppy =
n) *&px =
o)**ppx++=
p) px++ =
q) &ppx =

14. Faça o teste de mesa nos trechos de programas.

A

```
int vet[5]={7,4,6,1,5};
char str [ ]="boa sorte";
int num=1,i=0, *ptr;
ptr=&num;
*ptr = vet[3];
printf("%d", num);
ptr = vet;
vet[0] = --num;
printf("%d\n %c", *ptr, str[num]);
num=0;
while(i<3)    {
   num+=vet[i];
   i++;     }
printf("%d\n %s", num, str);
```

B

```
float vetor[ ]={2.5, 3.0, 5.5, 10.0};
float var=0.0, *p;
int x, y;
y=sizeof (vetor);
printf("%d", y++);
p = vetor;
x = (int) *p;
printf("%d", x);
p++;
*p += y;
printf("%d", vetor[1]);
x=0;
while(var < 13.0){
   var+= vetor[x];
   x++;     }
printf("%d %d", var, x);
```

C

```
char c, lin[ ]="CAIXA";;
int n=5, ts=55, *ptr;
ptr = &n;
c = (n < 10)?'3':'9';
*ptr += sizeof(c);
printf("%d \n %c", n, c);
ptr = &ts;
*ptr = -1;
printf("%d", ++ts);
for(n=0, ts=1; lin[n] !='\0';n++)
   ts += n;
printf("%d\n%c", ts, lin[--n]);
```

D

```
int t1=1,i=0,vet[3]={3,2,4};
int *p;
char str[ ]="DOMINGO";
do {
   t1 *= vet[i++];
} while (i<3);
p=&t1;
printf("%d",*p);
t1 = sizeof(str) + i;
printf("%d\n%c",t1, str[i);
*p = 0;
printf("%d",vet[t1]);
```

E

```
char texto[]="PENSAR!";
int a=10, b=2, c=0, *pin;
float vf=0.0;
pin = &a;
a = ++b;
printf("%d", *pin);
c = (b>2)?15:25;
pin = &b;
printf("%s\n"%d",texto, c);
*pin = 1;
printf("%c", texto[b]);
a-=1;
vf = c/a;
printf("%d", vf);
```

F

```
int i, n, x, vet[]={3, 5, 9, 0, 4};
int *ptr;
ptr = vet;
printf("%d", *ptr);
*ptr = 2;
for(i=n=x=0; i<4; i++)
{  n+=vet[i];
   x = (i%2)?x+1:x+2;
}
printf("%d\n%d\n", n,x);
printf("%d\n", i*sizeof(vet));
ptr++;
printf("%d", *ptr);
```

G

```
   int a, b, vet[4] = {6, 9, 2, 5}, *ptr;
   a=0;  b=1;
   char txt[10]="2PROVA";
   ptr = &a;
   b = *ptr;
   printf("%d", vet[b]);
   ptr = vet;
   a = ++b;
   printf("%d", a);
   ptr++;
   printf("%d", *ptr);
   printf("%d",sizeof(txt)-strlen(txt));
   for(a=1, b=0; txt[b]!='\0'; b++)
      if(isalpha(txt[b]))
         a*=2;
   printf("%d", a);
```

H

```
#define dif(y, z)   (y>z)?y-z+2:z-y

int main( )
  { int a=6, i=1, *ptr;
   char frase[ ]="a+b23*c";
   char outra[12]="ESTUDE#PROG";
   ptr = &i;
   for(i=0; frase[i]!='\0'; i++)
      if (isdigit(frase[i]))
        a+=i;
   printf("%d\n%d\n", a++,*ptr);
   *ptr = strlen(frase)-2;
   printf("%c", outra[--i]);
   printf("%d", dif(a, i));
   outra[6] = frase [1];
   printf("%s", outra);
```

I

```
int main( )
 {
  int i , V[3];
  char z = 'c', *p;
  int Funcao (char , int , char *);
  p= &z;
  for (i=0; i<3 ; i++)
    *(V+i) = i * i ;
  i=Funcao('Y', *(V+2), &z);
  printf("1-%d\n2-%c\n3-%c", ++i,z,*p );
  }
int Funcao (char a, int b, char *w)
 {
  printf("\n4-%c", a ) ;
  printf("\n5-%d", b-- );
  (*w)++;
  return ( b * b );
  return 0;
  }
```

J

```
int X(float *V, int t, float c)
  { int i;
   for (i=0; i<t; i++)
     if (*(V+i) !=c)
       *(V+i) *=10;
     else   return i;
   return -1;  }

int main ( )
 {int X(float *, int, float), i, tam, kx;
  float AR[2], num=2;
  tam = 2;
  for (i=tam-1; i>=0; i--)
    *(AR+i) = i+1;
  if ((kx = X(AR, tam, num))<0)
  printf("Erro\n");
  else  printf("%d\n", kx);
  return 0;
  }
```

K

```
void mx(float, float, float *);

int main()
{float x=2.0, y=3.0, max;
 mx(x, y, &max);
 mx(x, max, &y);
 printf("%.1f %.2f %.0f",x, y, max);
  return 0;
 }

void mx(float A, float B, float *max)
{
  if (A > B)
    *max=A;
  if (*max<=A)
    B=*max+A;
  else
    *max=B;
  *max/=B;
  }
```

L

```
int main()
  { int a=1, b=6;
   void funcunica (int *, int *);
   funcunica (&a, &b);
   if(( a))
      printf("%.1f\n",(float) a+b);
   else
     printf("Erro\n");
   if((!b))
     printf("%.1f\n",(float) a/b);
   else printf("Erro\n");
   puts ("FIM");
  return 0;
  }
void funcunica (int *num1, int *num2)
{   int a = 1;
    int b = 6;
   *num1 = 10;
   *num2 = 4;
   }
```

15. Escrever um programa contendo uma função **int divisao (int dividendo, int divisor, int *resto)**, que retorna a divisão inteira de dividendo por divisor e armazena no parâmetro resto, passado por referência, o resto da divisão.

```
int r, d;
d = divisao(5, 2, &r);
printf("Resultado:%d - Resto:%d", d, r); /* Resultado:2 - Resto:1 */
```

16. Escrever um programa que declara um vetor *inteiro* com 30 elementos na função *main.* Em seguida, em uma função chamada leitura, ler um valor maior que 0 e menor que 1000 para cada posição do vetor. Finalmente, em uma função chamada calculo, deve ser impresso o maior e o menor valor armazenado no vetor.

    Não pode ser declarada nenhuma variável global.

17. Faça um programa que declara e carrega um vetor de 20 elementos inteiros na função *main*. Em seguida exibir seu conteúdo na exibe, seguindo o seguinte: se a soma dos valores for um número par, mostrar o vetor a partir do primeiro elemento até o último, caso contrário, mostrar o vetor a partir do último elemento até o primeiro. O programa não pode utilizar nenhuma variável global.

18. Escreva um programa completo que possui as funções:

    a) **int ultima (char *string, char c)** que retorna a última posição na string em que aparece o caracter c.

    ```
    char str[ ]=”teste”;
    q=ultima(str, ‘t’); /* q recebe 3 */
    ```

    b) **int primeira(char *string, char c)** que retorna a primeira posição na string em que aparece o caracter c..

    ```
    char str[ ]=”teste”;
    q=primeira(str, ‘e’); /* q recebe 1 */
    ```

    OBS.: Se o caracter não aparecer, retornar –1.

19. Escrever uma função **int substitui(char *string, char c1, char c2)**, que troca, na string recebida como parâmetro, toda a ocorrência do caracter c1 pelo caracter c2. A função deve retornar o número de substituições que foram feitas.

    ```
    char txt[ ] = "recupera";
    int num;
    num = substitui(txt, 'e', 'X');
    printf("%d - %s", num, txt); /* 2 - rXcupXra */
    ```

20. Escrever uma função **int totalpos(char *string, char let)**, que retorna a soma das posições (índices) da string onde aparece o caracter let. Se o caracter não aparece na string, retornar -1.

    ```
    num = totalpos(“internet”, ‘e’); /*retorna 9 (3+6) */
    num = totalpos(“internet”, ‘i’); /*retorna 0 (0) */
    num = totalpos(“internet”, ‘a’); /*retorna -1 */
    ```

21. Escrever uma função **int contadepois(char *string, char let)** que retorna quantos caracteres a string possui após a primeira posição onde aparece o caracter let. Se a string não possuir o caracter let a funcão deve retornar -1.

    ```
    var = contadepois(“avaliando”, ‘a’); /* var recebe 8 */
    var = contadepois(“avaliando”, ‘o’); /* var recebe 0 */
    var = contadepois(“avaliando”, ‘x’); /* var recebe -1 */
    ```