Punteros

1. ¿De qué tipo es cada una de las siguientes variables?:

a) `int *a, b;`

a puntero, b puntero

a puntero, b entero

a entero, b puntero

a entero, b entero

b) `int *a,*b;`

a puntero, b puntero

a puntero, b entero

a entero, b puntero

a entero, b entero

2. Comenta el siguiente programa

a)

```
int a, b;
int *pa;
a = 5;
pa = &a;
b = *pa;
```

b)

```
int i, j,*p;
 p=&i;
 *p=21;
 p=&j;
 *p=1;
```

3. **Si se declara:** `float x, *p;`

¿Cuál de las siguientes expresiones es correcta?

a. `p=&x;`

b. Ninguna de las restantes respuestas es correcta `x=p*;`

c. `&x=p;`

d. `&p=x;`

4. Declara a, b y c variables enteras, y p, q y r variables puntero a entero.

a. Declara m, n variables de tipo float y s y t variables puntero a float.

b. Asigna a p la dirección de a y a q la dirección de b.

c. Asigna a b el valor 40 usando el puntero q. Mostrar el valor de b, la dirección de b , el valor de q, la dirección de q y el valor contenido en la dirección almacenada en q.

d. Ingresar desde teclado el valor de a usando su dirección almacenada. Mostrar el valor de a, la dirección de a, el valor de p, la dirección de p y el valor contenido en la dirección almacenada en p.

e. Asigna a r la dirección de a.

f. Asigna a la posición de memoria contenida en el puntero r el valor 200. Mostrar el valor de a, la dirección de a , el valor de p, la dirección de p y el valor contenido en la dirección almacenada en p, y el valor de r, la dirección de r y el valor contenido en la dirección almacenada en r.

g. Emite un mensaje indicando si el valor del puntero p es igual al valor del puntero q

h. Ídem g indicando si el valor del puntero p es igual al valor del puntero r.

i. Emite un mensaje indicando si el contenido de la posición de memoria apuntada por p es igual al contenido de la posición de memoria apuntada por q.

5. Declara una variable de tipo puntero a entero y una variable entera. Asígnale un valor a la variable entera. Guarda la dirección. Luego lée el contenido de la variable puntero y muéstralo en hexadecimal.

6. Explica el error.

```
char c = 'A';
double *p = &c;
```

7. Un programa en C contiene las siguientes sentencias:

```
float a = 0.001, b = 0.003;
float c, *pa, *pb;
...
pa = &a;
*pa = 2 * a;
pb = &b;
c = 3 * (*pb - *pa);
```

Responda:

a) ¿Qué valor tiene a al finalizar el programa?

b) ¿Qué valor tiene b al finalizar el programa?

c) ¿Qué valor tiene c al finalizar el programa?

d) ¿Qué valor tiene (*pa) al finalizar el programa?

e) ¿Qué valor tiene (*pb) al finalizar el programa?

8. El siguiente código contiene un error, cuál es?:

```
main ()  {
int x = 5;
float y = 5;
int *xPtr = NULL;
xPtr = &y;
printf ("%d", *xPtr) ;
return 0;}
```

9. Cuál es la salida del siguiente programa? Una vez realizada su ejecución, comentar qué tarea realiza cada instrucción o línea del programa.

```
main(){
int a, b, *p, *q, *r;
char c, d, *m,*n;

a=5;
b=7;
q=&a;
p=q;
*p=b;
printf("%d\n\n\n", *q);
r=&b;
*r=*q;
printf("a=%d, b=%d, *p=%d, *q=%d,
*r=%d\n\n\n", a, b,*p,*q,*r);
c='A';
d='B';
m=&c;
n=&d;
*m=*n;

if (m==n) printf("m y n apuntan a la
misma dirección\n\n\n");
else printf("m y n apuntan a
direcciones distintas\n\n\n");
printf ("*m=%c\n\n\n", *m);

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

10. Realizar la traza del siguiente programa. Ejecutarlo y comentar qué tarea realiza cada instrucción o línea del programa.

```
int main(){

int *p;
inta=1, b=2;
int **s;
p=&a;

b=*p+1;
s=&p;
printf("p=%d, *p=%d, &p=%d\n", p,*p,
&p);
printf("s=%d, *s=%d, &s=%d\n", s,*s,
&s);

*s=&b;
printf("Se ejecutó *s=&b\n y
ahora…\n");
printf("s=%d, *s=%d, &s=%d\n", s,*s,
&s);

**s=79;
printf("a=%d, &a=%d\n", a, &a);
printf("b=%d, &b=%d\n", b, &b);

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

11. Comenta el siguiente programa

```
int main(){
   int a,b,c;
   int *p1,*p2;
   p1 = &a;
   *p1 = 1;
   p2 = &b;
   *p2 = 2;
   p1 = p2;
   *p1 = 0;
   p2 = &c;
   *p2 = 3;
   getche();
   return 0;
   }
```

12. Completa el siguiente programa:

```
int main(){

int num, n;
int *dir_n;
num=22;
n=7;
dir_n=&n;
printf("La variable num vale: %dln", ???);
printf("La dirección de memoria donde esta almacenada la variable num es: %p\n",
???);
printf("La dirección de memoria almacenada en dir_n es: %pln", ???);
printf("El valor de la variable apuntada por dir_n es: %d\n",???);
getche();
return 0;
}
```

13. Qué emite el siguiente programa?:

```
int main(){
int u = 3, v;
int *pu;
int *pv;
pu = &u;
v = *pu;
pv = &v;
printf("\nu=%d &u=%X pu=%X *pu = %d", u, &u, pu, *pu);
printf("\nv=%d &v=%X pv=%X *pv =%d", v, &v, pv, *pv);
getche();
```

```
    return 0;
}
```

14. Errores: realiza las declaraciones correspondientes considerando que las variables cuyo nombre comienza con p son punteros, encuentra si hay errores en cada línea y justifica:

- ✓ `pta = *a;`
- ✓ `ptb = &pta;`
- ✓ `pta = 8;`
- ✓ `ptb = ptb + 3;`
- ✓ `ptb = &NULL;`
- ✓ `b = 8;`

- ✓ `*ptb = b;`
- ✓ `ptb = a + 10;`
- ✓ `printf("%d", pta + ptb);`
- ✓ `int *pta,`
- ✓ `*ptb, a, b;`

15. De acuerdo al ejercicio anterior, construye un programa y haz:

- ✓ Que pta apunte a a
- ✓ Que ptb apunte al mismo sitio que pta
- ✓ Que pta cambie de contenido a 8
- ✓ Que ptb cambie de contenido aumentando 3 unidades
- ✓ Que ptb apunte a NULL
- ✓ Que b almacene 8
- ✓ Que ptb apunte a b
- ✓ Que ptb cambie de contenido a a + 10
- ✓ Que imprima la suma de los contenidos de pta y ptb

16. Investiga qué emite por pantalla el siguiente programa:

```
int main() {
int n = 8, * ptr;
printf("ptr es%p\n", ptr);
ptr = &n;
printf("n es %d\n", n);
printf("&n es %p\n", &n);
printf("*ptr es %d\n", *ptr);
printf("ptr es %p\n", ptr);
n--;
printf("n es despues de n-- %d\n", n);
printf("&n es %p\n", &n);
ptr++;
printf("ptr es despues de ptr++ %p\n", ptr);
printf("*ptr es %d\n", *ptr);
getche();
return 0;
}
```

17. Considera las siguientes instrucciones:

```
int *p; int i; int k;
i = 42;
k = i;
p = &i;
```

A. k = 75; B. *k = 75; C. p = 75; D. *p = 75;
E. Dos o más de las anteriores.

Luego de esas instrucciones, cuál de las siguientes cambia el valor de i a 75?

18. Explica el funcionamiento del siguiente programa.

```
int main(){
```

```
int a, *p;
    a=5;
    p=&a;
   *p+=7;
   printf("\nEl valor final de a es: %d\n", a);
   system("pause");
   return 0;
 }
```

19. Explica la salida del siguiente programa.

```
int main() {
int n;
int * ptr;
n=8;
printf("ptr es%p\n", ptr);
ptr=&n;
printf("n es %d\n", n);
printf("&n es %p\n", &n);
printf("*ptr es %d\n", *ptr);
printf("ptr es %p\n", ptr);
n--;
printf("n es despues de n-- %d\n", n);
printf("&n es %p\n", &n);
ptr++;
printf("ptr es despues de ptr++ %p\n", ptr);
printf("*ptr es %d\n", *ptr);
getch();
return 0;
}
```

20. Casting y punteros a void: Comenta el siguiente programa e identifica (si los hay) los errores.

```
int *p;
double
*q; void
*r; p=q;
p=(int*)q;
  p=r=q
```

21. En C, el carácter * tiene tres significados diferentes. ¿Cuáles son?

22. Transcribe el siguiente programa y extrae conclusiones:

```
int main()  {
   void *generico;
    int *pint, x=3;
    char *pchar = "Punteros  a caracteres";
    pint=&x;
    printf("El valor apuntado por puntint es %d (valor de x)\n",
*pint);
    generico=pchar;
    printf("\n\ngenerico (tipo void) apunta a objeto char:  %s",
generico);
    printf("\n\nImprimo ahora generico como char:\n\n");
    puts((char *)generico);
    getch();
return 0;
}
```

23. Sea a una variable entera: int a = 25; Se declaran 4 punteros:

```
int *ptrl, **ptr2, ***ptr3, ****ptr4;
```

¿Cómo podríamos imprimir el valor 25 de la variable a, con los cuatro punteros?. Construye el programa para verificar.

24. Determina la salida del siguiente programa:

```
int main ()

 {   int x = 5;
       int y = 10;
       int *xPtr = NULL;
       int *yPtr = NULL;
       xPtr = &x;
       yPtr = &x;
      *xPtr = *xPtr + 1;
      (*yPtr)--;
      printf ("x:   %d\n",x);
      printf ("y:   %d\n",y);
      return 0;
      }
```

25. Investiga qué emite por pantalla el siguiente programa:

```
int main() {
int n = 8, * ptr;
printf("ptr es%p\n", ptr);
ptr=&n;
printf("n es %d\n", n);
printf("&n es %p\n", &n);
printf("*ptr es %d\n", *ptr);
printf("ptr es %p\n", ptr);
n--;
printf("n es después de n--
%d\n", n);
printf("&n es %p\n", &n);
ptr++;
printf("ptr es después de ptr++
%p\n", ptr);
printf("*ptr es %d\n", *ptr);
getche();
return 0;
}
```

26. Analiza y comenta este ejercicio

```
int main()
{
int a,b,*pta,**ptb;
pta=&a;
ptb=&pta; //guardo la dire de pta en ptb (se puede acceder a través de los dos
punteros)
*pta=8;
      printf("ptb es: %d\n\n",ptb);
      printf("Contenidos de *ptb después de &pta es: %d\n\n",*ptb);
      printf("Contenidos de **ptb después de &pta es: %d\n\n",**ptb);

      **ptb=**ptb+3;//a toma el valor 11 a través del puntero a puntero ptb
      printf("Contenidos de **ptb es: %d\n\n",**ptb);
        printf("Contenidos de a es: %d\n\n",a);

      *ptb=NULL; // esta bien esto o seria **ptb=NULL // elimino la dirección
de pta de ptb
      printf("Contenidos de a después de *ptb=NULL, es: %d\n\n",a);
      printf("Contenidos de *ptb después de NULL es: %d\n\n",*ptb);

      b=8;
      pta=&b; //pta deja de apuntar a a  para apuntar a b, cuyo valor es 8
      printf("Contenidos de *ptb después de pta=&b, es: %d\n\n",*ptb);
      printf("Contenidos de a es: %d\n\n",a);
      printf("Contenidos de b después de pta=&b, es: %d\n\n",b);
      printf("Contenidos de *pta después de pta=&b, es: %d\n\n",*pta);//
muestra el valor de b
      printf("Contenidos de **ptb después de pta=&b, es: %d\n\n",**ptb);
       printf("ACA Contenidos de *ptb después de pta=&b, es: %d\n\n",*ptb);

      **ptb=a+10;

      printf("Contenidos de **ptb después de **ptb=a+10, es: %d\n\n",**ptb);
      printf("Contenidos de *pta después de **ptb=a+10, es: %d\n\n",*pta);
      printf("Contenidos de a después de **ptb=a+10, es: %d\n\n",a);
      printf("Contenidos de b después de **ptb=a+10, es: %d\n\n",b);
      printf("La suma de los contenidos de pta + ptb es: %d\n\n",(*pta) +
(**ptb));
      system("pause");
return 0;
}
```

27. Crea un programa para descomponer un entero corto sin signo en dos bytes utilizando exclusivamente operaciones con punteros (por supuesto, sin usar operadores a nivel bit).

```
int main(){
      unsigned short int dato=0xAA40;
      unsigned char *p1,*p2;

      p1=&dato; //uno de los punteros apunta al comienzo del dato de
      16 bits p2=p1+1; //el otro puntero apunta al siguiente dato de
      8 bits, es
decir, el byte más significativo de dato

      printf("Byte alto: %hx\n", *p2);
      printf("Byte bajo: %hx\n", *p1);}
```