M en C Gabriel Castillo Hernández

Ejercicios sobre apuntadores.

Para las siguientes preguntas considere que los números enteros utilizan 4 celdas de memoria y que el siguiente programa imprimió un 45621

```
int main(){
   int a[] = {1,6,3,5};
   int *p;
   p = a;
   printf("%d",&a);
   return 0;
}
```

Para cada pregunta suponga que se parte del programa arriba mostrado (Es decir los cambios introducidos por cada pregunta no son acumulados para responder la siguiente preguta)

1. ¿Qué imprimirá si ejecutamos la línea: `printf("%d",a+3);` antes del return?
   _____
2. ¿Qué imprimirá si ejecutamos la línea: `printf("%d",*a+3);` antes del return?
   _____
3. ¿Qué imprimirá si ejecutamos la línea: `printf("%d",*(a+3));` antes del return?
   _____
4. ¿Qué imprimirá si ejecutamos la línea: `printf("%d",p+3);` antes del return?
   _____
5. ¿Qué imprimirá si ejecutamos la línea: `printf("%d",*p+3));` antes del return?
   _____

Sean las siguientes declaraciones, contenido de memoria y tabla de símbolos;

```
int a;

int *p, *q;

int v[3];

int b;
```

símbolo	tipo	dirección
a	int	102
p	int *	101
q	int *	110
v	int []	100
b	int	106

100	345
101	112
102	10
103	88
104	25
105	34
106	23
107	92
108	21
109	31
110	43
111	103

considere que los enteros y las direcciones ocupan una celda para almacenarse en memoria es decir sizeof(int) es igual a 1

Con base en el esquema anterior, responda las siguientes preguntas. Cada pregunta siempre parte del estado de memoria y variables que se ha establecido originalmente (las modificaciones no se acumulan con cada pregunta).

6. Para el siguiente segmento de código se imprime el valor: _____

```
a =5;

b= 10;

p =&a;

printf("%d",*(p+1));
```

7. Para el siguiente segmento de código se imprime el valor:_____

```
a =5;
b= 10;
p =&a;
printf("%d",*p+1);
```

8. Para el siguiente segmento de código se imprime el valor: _____

```
a =5;
b= 10;
p =&a;
*(p+3)=50;
printf("%d",v[3]);
```

9. En este segmento de código se imprime el valor: _____

```
a =5;
b= 10;
v[3] = 0
p =&a;
printf("%d",b);
```

10. En este segmento de código se imprime el valor: _____

```
printf("%d",*v+1);
```

11. En este segmento de código se imprime el valor: _____

```
printf("%d",v[1]);
```

12. En este segmento de código se imprime el valor: _____

```
printf("%d",v+1);
```

PRECEDENCIA DE LOS OPERADORES: En ausencia de paréntesis, los operadores del primer renglón se ejecutan siempre antes que los operadores del segundo renglón.

Operador	Descripción	Asociatividad
++ -- () [] . ->	Post- incremento y decremento Llamada a función Elemento de vector Selección de elemento por referencia Selección de elemento con puntero	Izquierda a derecha
++ -- + - ! ~ (type) * & sizeof	Pre- incremento y decremento Suma y resta unitaria NOT lógico y NOT binario Conversión de tipo Indirección Dirección de Tamaño de	Derecha a izquierda

Para cada uno de los siguientes programas, escríbalo en CodeBlocks, ejecútelo, capture su salida, haga un esquema de cómo se vería la memoria y qué cambios irían sufriendo las variables y justifique su respuesta.

13. Programa:

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

int main()
{
    double x[5] = {1.3,2.4,3.3,6.7,0.0};
    double *p;
    int k;

    for (k=0;k<5;k++)
        printf("%5.2f\n",x[k]);
    printf("---------------\n");

    return 0;
}
```

14. Programa:

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

int main()
{
    double x[5] = {1.3,2.4,3.3,6.7,0.0};
    double *p;
    int k;

    p = &x[0];
    for (k=0;k<5;k++)
        printf("%5.2f\n",*p+k);
    printf("---------------\n");

    return 0;
}
```

15. Programa:

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

int main()
{
    double x[5] = {1.3,2.4,3.3,6.7,0.0};
    double *p;
    int k;

    p = &x[0];
    for (k=0;k<5;k++)
        printf("%5.2f\n",p[k]);
    printf("---------------\n");

    return 0;
}
```

16. Programa:

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

int main()
{
    double x[5] = {1.3,2.4,3.3,6.7,0.0};
    double *p;
    int k;

    p = &x[0];
    printf("Tamanio de un double: %d bytes\n\n",sizeof(double));
    for (k=0;k<5;k++) {
        printf("%d\n",p);
        p = p+1;
    }

    return 0;
}
```

17. Programa:

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

int main()
{
    double x[5] = {1.3,2.4,3.3,6.7,0.0};
    int k;


    printf("Tamanio de un double: %d bytes\n\n",sizeof(double));
    for (k=0;k<5;k++) {
        printf("%d\n",&(x[k]));
    }

    return 0;
}
```

Programa:

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

int main()
{
    double x[5] = {1.3,2.4,3.3,6.7,0.0};
    int k;


    for (k=0;k<5;k++) {
        printf("%5.2f\n",*(x+k));
    }

    return 0;
}
```

18. En el siguiente programa el usuario espera que se tenga la siguiente salida:

```
direccion de k: 6356736
direccion de a: 6356712
23423.912300
10
20
30
40
50

Process returned 0 (0x0)   execution time : 0.013 s
Press any key to continue.
```

Sin embargo, eso no sucede, indique qué salida se obtiene. Para explicarlo, haga un diagrama de memoria donde se muestren las variables y explique a que se debe esto.

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

int main()
{
    double k=23423.9123;
    int a[] = {0,0,0,0};


    int j;

    a[0] = 10;
    a[1] = 20;
    a[2] = 30;
    a[3] = 40;
    a[4] = 50;
    a[5] = 60;

    printf("direccion de k: %d\n",&k);
    printf("direccion de a: %d\n",&a);
    printf("%f\n",k);
    for (j=0;j<5;j++)
        printf("%d\n",a[j]);

    return 0;
}
```

19. A partir de estos ejercicios escriba un ensayo donde exponga el concepto de apuntadores, la aritmética de apuntadores y la relación que existe entre apuntadores y arreglos.