UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA
Departamento de Ciencias da Computación e Tecnoloxías da Información
Programación II

Ejercicios de punteros

1. Suponga que tenemos las declaraciones:

```
typedef int tIndice;
typedef tIndice* tPIndice;

tIndice i;
tPIndice apuntI;
```

   a) ¿Qué contiene entonces `apuntI`?

   b) Si en seguida ejecutamos la codificación...

```
apuntI = malloc(sizeof(tIndice));
*apuntI = 2;
i = 4;
```

   ..¿qué contendrá entonces `apuntI`? ¿qué contendrá `*apuntI`?

2. Sea el siguiente fragmento de codificación:

```
1  typedef int* tPEntero;
2
3  int c, cc;
4  tPEntero apuntC, apuntCC;
5
6  apuntC = NULL;
7  apuntCC = malloc(sizeof(int));
```

   Contestar a las siguientes preguntas:

   a) Después de la ejecución de la línea 7, ¿cuáles de las siguientes variables contienen basura?

      I) `apuntC`
      II) `apuntCC`
      III) `c`
      IV) `cc`
      V) `*apuntC`
      VI) `*apuntCC`

   b) Representar gráficamente la situación de todas las variables–enteras, punteros y dinámicas.

   c) Si añadimos la sentencia `*apuntCC = 3;` ¿cuál será entonces la situación? Dibuje un diagrama de apuntadores.

3. Suponga que:

```
typedef float tFloat;
typedef tFloat* tPFloat;

tPFloat A1, A2;
```

¿cuáles de los siguientes enunciados son sintácticamente correctos en ese caso?

a) `A1 = 1.1;`

b) `A1 = 1.1*;`

c) `A1 = malloc(sizeof(float));`

d) `A1 = NULL;`

e) `*A1 = 1.1;`

f) `*A1 = malloc(sizeof(float));`

g) `A2 = A1;`

h) `A2 = *1.1;`

i) `A2 = *A1;`

4. ¿Qué exhibe el siguiente fragmento de código?

```
typedef char* tPCaracter;

tPCaracter A1, A2;

A1 = malloc(sizeof(char));
A2 = malloc(sizeof(char));
*A1 = 'A';
*A2 = 'B';
A1 = A2;
printf("*A1 = %d",*A1);
printf("*A2 = %d",*A2);
```

5. Dadas las siguientes definiciones y declaraciones:

```
typedef int* tPEntero;
typedef char* tPCaracter;

tPEntero P1, P2;
tPCaracter Q1, Q2;
```

¿qué hacen los siguientes fragmentos de código?

a)

```
P1 = malloc(sizeof(int));
P2 = malloc(sizeof(int));
Q1 = malloc(sizeof(char));
scanf(" %c",Q1);
*P2 = P1;
printf("*Q1 igual a %c, *Q2 igual a %c", *Q1,*Q2);
```

b)

```
P1 = malloc(sizeof(int));
P1 = P2;
P1 = 3.5 * (*P1);
```

c)

```
P1 = malloc(sizeof(int));
Q2 = malloc(sizeof(char));
*P1 = 48;
*Q2 = (char)(*P1);
P1 = Q2;
```

d)

```
P1 = malloc(sizeof(int));
Q2 = malloc(sizeof(char));
*P1 = 6;
*Q2 = (char)(*P1+59);
printf("*P1 igual a %d, *Q2 igual a %c", *P1, *Q2);
```

e)

```
Q1 = malloc(sizeof(char));
Q2 = malloc(sizeof(char));
*Q1 ='Z';
*Q2= (*Q1) - 1;
printf("*Q1 igual a %c, *Q2 igual a %c", *Q1, *Q2);
```

6. Asumir las siguientes declaraciones:

```
typedef int* tPEntero;
typedef float* tPFloat;

int X;
tPEntero P1, P2;
tPFloat Q1, Q2;
```

¿Qué es incorrecto (si lo hay) en cada una de las sentencias?

a) `printf("%i",P1);`

b) `P1 = Q1;`

c) `if (*P1 == NULL)  Q1 = Q2;`

d) `scanf("%d",*P1)`

e) `X = malloc(sizeof(int))`

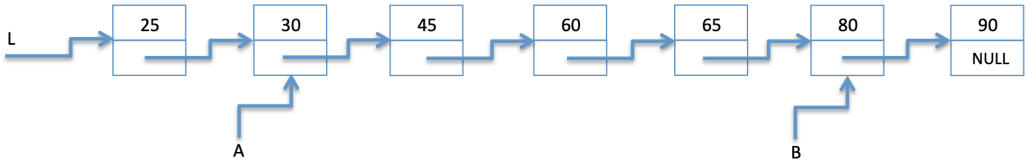
f)

```
*P1 = 17;
P1 = malloc(sizeof(int));
```

g)

```
P1 = malloc(sizeof(int));
*P1 = 17;
P1 = malloc(sizeof(int));
```

7. Sea la siguiente figura que representa una estructura enlazada que almacena una colección de elementos:



y sean:

```
typedef int tDato;
typedef struct tNodo * tEnlace;
struct tNodo
    { tDato dato;
      tEnlace sig;
    };
typedef tEnlace tColeccion;

tColeccion L;
tEnlace A,B;

// (*A).dato es equivalente a A->dato
// La reserva de memoria se haria como L = malloc(sizeof(struct tNodo))
```

a) Dar los valores de las siguientes expresiones:

I) `A->dato`

II) `B->sig->dato`

III) `L->sig->sig->dato`

b) Decir si se verifican las siguientes igualdades:

I) `L->sig == A`

II) `A->sig->dato == 60`

III) `B->sig == NULL`

c) Indicar si la sintaxis de las siguientes sentencias son correctas o no, y explicar cuál es el problema, si lo hay.

I) `L->sig = A->sig`

II) `L->sig = B`

III) `L->dato = B`

IV) `B = A->sig->dato`

V) `L = B->sig->sig`

VI) `B = B->sig->sig->sig` (NULL no tiene siguiente)

d) Escribir una sentencia para cada una de las siguientes acciones:

I) Hacer que `L` apunte al nodo que contiene `45`

II) Hacer que `B` apunte al último nodo de la lista

III) Hacer que `L` apunte a una colección vacía

e) Mostrar lo que escribe el siguiente segmento de código:

```
tEnlace Ptr;
...
L=malloc(sizeof(struct tNodo));
Ptr=malloc(sizeof(struct tNodo));
L->dato = 2;
Ptr->dato = 5;
L = Ptr;
Ptr->dato = 7;
printf(" %d %d",Ptr->dato, L->dato);
```

f) Mostrar lo que escribe el siguiente segmento de código:

```
tEnlace Ptr;
...
L=malloc(sizeof(struct tNodo));
L->dato = 10;
Ptr=malloc(sizeof(struct tNodo));
Ptr->dato = 18;
Ptr->sig = NULL;
L->sig = Ptr;
Ptr=malloc(sizeof(struct tNodo));
Ptr->dato = 20;
Ptr->sig = L;
L = Ptr;
while (Ptr != NULL)
{   printf(" %d \n",Ptr->dato);
    Ptr = Ptr->sig;
}
```

8. Dadas las declaraciones siguientes:

```
struct NodoNumero
       {   int datos;
           struct NodoNumero * sig;
       };

typedef struct NodoNumero tNodoNumero;
tNodoNumero* p1, *p2;
int* p3;
```

y suponiendo que se han ejecutado previamente las instrucciones

```
p1 = malloc(sizeof(tNodoNumero));
p2 = malloc(sizeof( tNodoNumero));
p3 = malloc(sizeof(int));
```

¿Qué resultado se visualizará en este fragmento (en caso de error, indicar cuál)?:

```
p1->datos = 12;
p2->datos = 34;
p1->sig = p2;
printf(" %d ",p2->datos);
printf(" %d",p2->sig->datos);
```

9. ¿Cuál es la salida correcta del siguiente programa?:

```
1           void ejemplo_punteros()
2           {  int *p, *q, *r;
3
4             p = malloc(sizeof(int));
5             *p = 10;
6             q = p;
7             *q = *q + 1;
8             free(q);
9             q = NULL;
10            q = malloc(sizeof(int));
11            *q = 20;
12            printf("valor de *q = %i \n", *q);
13            printf("valor de *p = %i \n", *p);
14          }
```

a) Se produce un error de ejecución en la línea 13.

b) Se imprime: `valor de *p = 10`

c) Se imprime: `valor de *p = 11`

d) Se imprime: `valor de *q = 20`
`valor de *p = 20`