Programación Estructurada en C++

Juan Espejo[1]

23 de septiembre de 2019

[1]Escuela Profesional de Matemática, Universidad Nacional de Ingeniería, R1-325, Av. Túpac Amaru s/n, Rímac, Lima 25, Perú, e-mail: jespejod@uni.edu.pe

1.1. Punteros

Ejercicio 1.1. El siguiente programa compila y corre sin errores. Indicar qué es lo que escribe y justifique su respuesta.

```
1  #include <stdio.h>
2
3  void acumular(int * x, int * y);
4
5  void main(void)
6  {
7          int x = 20;
8          int y = 10;
9
10         acumular(&x, &y);
11         printf("\n x: %d, y: %d\n", x, y);
12         acumular(&y, &x);
13         printf(" x: %d, y: %d\n\n", x, y);
14 }
15
16 void acumular(int * x, int * y)
17 {
18         *x = *y + *x;
19 }
```

Ejercicio 1.2. Indique lo que se mostrará en la pantalla al ejecutar cada programa y explique por qué.

1. ```
   #include<stdio.h>

   int main()
   {
       int i=3, *j, k;
       j = &i;
       printf(" %d\n", i**j*i++*j);
       return 0;
   }
   ```

2. ```
   #include<stdio.h>

   int main()
   {
   ```

```
    int x = 30, *y, *z;
    y = &x;
    z = y;
    *y++ = *z++;
    x++;
    printf(" x = %d, y = %d, z = %d\n", x, y, z);
    return 0;
}
```

3.
```
#include<stdio.h>

int main()
{
    int ***r, **q, *p, i = 8;
    p = &i;
    q = &p;
    r = &q;
    printf(" %d, %d, %d\n", *p, **q, ***r);
    return 0;
}
```

Ejercicio 1.3. Almacene los valores: $2, 3, \ldots, 2017$ en un arreglo de enteros del tipo `short`. Luego, utilizando la criba de Eratóstenes, imprima los primos encontrados en el arreglo y la dirección de memoria en la que se encuentra. Por ejemplo:

```
   2  está en la dirección 0x7ffed9b65530
   3  está en la dirección 0x7ffed9b65532
   5  está en la dirección 0x7ffed9b65536
      ⋮
2017  está en la dirección 0x7ffed9b664ee
```

Sugerencia: utilizar el siguiente pseudocódigo.

```
Sea A el arreglo conformado por los enteros 2, 3, 4, ..., 2017,
indexados con 0, 1, 2, ..., 2015, respectivamente.

Para i = 2, 3, 4, ..., con i*i sin excer a 2017:
   Si A[i-2] es mayor que 0:
       Para j = i*i, i*i + i, i*i + 2*i, ..., con j sin exceder a 2017:
           A[j-2] = 0

Salida: Todos los A[i] mayores que 0.
```

Ejercicio 1.4. Sobre el siguiente programa:

```c
#include <stdio.h>
#define Arreglo 3
#define Longitud 5

void imprimir_arreglo_1(int (*ptr)[Longitud]);
void imprimir_arreglo_2(int (*ptr)[Longitud], int n);

void main()
{
        int k;
        int multi[Arreglo][Longitud] =
                {
                        { 1, 2, 3, 4, 5 },
                        { 6, 7, 8, 9, 10  },
                        { 11, 12, 13, 14, 15 }
                };

        int (*puntero)[Longitud];
        puntero = multi;

        for (k = 0; k < Arreglo; k++)
                imprimir_arreglo_1(puntero++);

        imprimir_arreglo_2(multi, Arreglo);
}

void imprimir_arreglo_1(int (*ptr)[Longitud])
{
        int *p, cont;
```

```
30          p = (int *)ptr;
31          for (cont = 0; cont < Longitud; cont++)
32                  printf(" %d\n", *p++);
33  }
34
35  void imprimir_arreglo_2(int (*ptr)[Longitud], int N)
36  {
37          int *p, cont;
38          p = (int *)ptr;
39          for (cont = 0; cont < (N * Longitud); cont++)
40                  printf(" %d\n", *p++);
41  }
```

1. ¿Qué se declara en la línea 18?
2. Las sentencias en las líneas 30 y 38 son las mismas. ¿Cuál es la razón de ser de estas sentencias?
3. Explique lo que hace cada una de las 3 funciones definidas arriba.

Ejercicio 1.5. Implemente el siguiente prototipo de función:

```
int verificar (int * dado, int n);
```

el cual retorna `1` si en un arreglo de 10 elementos apuntado por `dado`, contiene el valor de `n` (ver prototipo arriba); caso contrario, retorna `0`. En la función principal, se debe definir un arreglo de 10 enteros y asignar a sus elementos valores aleatorios en el intervalo $[11; 32]$ y asignar aleatoriamente un valor para `n` en el mismo intervalo. Finalmente, verificar si `n` está en dicho arreglo o no, mostrando un mensaje en la pantalla.

Ejercicio 1.6. Implemente el siguiente prototipo de función:

```
void intercambio (int * x, int * y);
```

el cual intercambia los valores almacenados en las direcciones `x` e `y`. En la función principal, se debe definir un arreglo de 10 enteros y asignar a sus elementos valores aleatorios en el intervalo $[0; 20]$; finalmente se debe mostrar dicho arreglo y el arreglo que resulta de ordenar de menor a mayor el arreglo original, utilizando la función `intercambio`.

Ejercicio 1.7. Cree un programa que genere aleatoriamente un arreglo de 10 números enteros del intervalo $[0, 20]$. Luego, muestre dicho arreglo, ordene dicho arreglo de manera creciente utilizando punteros y finalemte muestre el arreglo ordenado.

Ejercicio 1.8. ¿Qué se imprime y por qué?. Corregir si es necesario.

```c
#include <stdio.h>

int main()
{
    char *pc = NULL;
    int *pi = NULL;
    double *pd =NULL;
    printf("\n%d %d %d\n%d %d %d\n\n", (int)(pc +1), (int)(pi +1),
    (int)(pd +1), (int)(pc +3), (int)(pi +5), (int)(pd +7));
}
```

Ejercicio 1.9. Si las siguientes expresiones son verdaderas:

```c
sizeof(short) == 2
sizeof(int) == 4
sizeof(float) == 4
sizeof(double) == 8
```

Y si hemos declarado el arreglo:

```c
int arr[5] = {0, 0, 0, 0, 0};
```

Explique mediante un dibujo donde las siguientes sentencias almacenan la información (tanto el lugar como la cantidad de bytes escritos.) Asuma que todas las sentencias se ejecutaron exitosamente.

1. `arr[9] = 7;`
2. `arr[-4] = 1;`
3. `( (short*)arr )[7] = 128;`
4. `( (double*)arr )[2] = 3.14;`
5. `( (char*)&arr[1] )[6] = 'A';`
6. `( (float*)( &( (short*)&arr[3] )[-3] ) )[0] = 7.5;`