TP “0”: Práctica de repaso e introducción a C

Actividades de Aprendizaje

Para la resolución de esta práctica ponemos a tu disposición el archivo de encabezados TP0_Repaso.h de tal manera que solo te queda implementar las funciones que te pedimos y alguna forma de probar los algoritmos que escribiste.

Ejercicio 1

Escribir una función que dadas:

- las coordenadas `(xc, yc)` de un punto que consideramos centro de una circunferencia;
- el valor del radio de la circunferencia;
- las coordenadas `(xp, yp)` de un punto del plano;

Informe si el punto `(xp, yp)` está ubicado sobre la circunferencia, o dentro o fuera del círculo.

Nota: Se utilizará un enumerado Referencia que incluirá las opciones INTERNO, EXTERNO y EN_CIRCUNFERENCIA.

```
enum Referencia {
    INTERNO, // el punto está dentro del círculo
    EXTERNO, // el punto está fuera del círculo
    EN_CIRCUNFERENCIA // el punto está en la circunferencia
};

enum Referencia dondeEstaElPunto(int xc, int yc, int rc, int xp, int yp);
```

Casos de testeo

```
    dondeEstaElPunto(0, 0, 2, 1, 1 ); => INTERNO
    dondeEstaElPunto(0, 0, 1, 1, 1 ); => EXTERNO
    dondeEstaElPunto(0, 0, 1, 1, 0 ); => EN_CIRCUNFERENCIA
```

Ejercicio 2

Necesitamos una función que reciba un valor entero `n` y un dígito `d` (entero entre 0 y 9), y confirme que dicho dígito `d` está (o no) entre los del número `n`.

```
Boolean digitoEnNumero(long n, short d);
```

Casos de testeo

```
    digitoEnNumero(987, 7); => True
    digitoEnNumero(987, 6); => False
```

Ejercicio 3

Refactorizar el **Ejercicio 1** para que los parámetros recibidos sean:

```
enum Referencia dondeEstaElPuntoBis(Punto centroC, int rc, Punto p);
```

donde Punto se define así:

```
struct Punto {
    int x, y;
};
```

Ejercicio 4

Es necesario administrar los datos de los jugadores de un equipo de futbol (nombre, edad, partidosJugados).

Se pide escribir funciones que informen:

- Los nombres de los jugadores ordenados desde el que más partidos jugó hasta el que menos.
- Los nombres de los jugadores ordenados desde el más joven al más grande.
- La cantidad de partidos jugados en promedio dada una edad.

```
const TAMANO_STRING = 50;

struct Jugador {
    char[TAMANO_STRING] nombre;
    int edad, partidosJugados;
};

struct Jugador *jugadoresOrdenadosPorCantDePartidos(struct Jugador
equipo[]);
struct Jugador *jugadoresOrdenadosPorEdad(struct Jugador equipo[]);
float promedioDePartidosJugados(struct Jugador equipo[], int edad);
```

Casos de testeo

```
struct Jugador equipo[] = {
    {"Messi", 34, 800},
    {"Ronaldo", 37, 900},
    {"Neymar Jr.", 29, 500},
    {"Mbappe", 22, 200},
    {"Lewandowski", 33, 700},
    {"Salah", 29, 400},
    {"De Bruyne", 30, 600},
    {"van Dijk", 30, 400},
    {"Ramos", 35, 650},
    {"Neuer", 35, 800},
    {"Davies", 20, 100}
};
```

```
    Jugador *jugadoresOrdenadosPorCantDePartidos(struct Jugador equipo[]);
    Jugador *jugadoresOrdenadosPorEdad(struct Jugador equipo[]);
    float promedioDePartidosJugados(struct Jugador equipo[], int edad);
```

Ejercicio 5

Se quiere controlar el número de habitantes de un edificio con 8 pisos y 5 departamentos (A, B, C, D y E) en cada piso. Se pide escribir funciones que informen:

- el departamento que más habitantes tiene de todo el edificio.
- el piso que más habitantes tiene de todo el edificio.
- las viviendas vacías.
- el promedio de habitantes por departamento.

```
    const CANT_PISOS = 8;
    const CANT_DEPARTAMENTOS = 5;
    const CANT_VIVIENDAS = CANT_PISOS * CANT_DEPTOS;

    struct Vivienda {
        short piso;
        char depto;
    }

    int pisoConMasHabitantes(int edificio[CANT_PISOS][CANT_DEPARTAMENTOS]);
    int cantidadDeViviendasVacias(int
edificio[CANT_PISOS][CANT_DEPARTAMENTOS]);
    float promedioHabitantesPorVivienda(int
edificio[CANT_PISOS][CANT_DEPARTAMENTOS]);
    Vivienda viviendaConMasHabitantes(int
edificio[CANT_PISOS][CANT_DEPARTAMENTOS]);
```

Casos de testeo

```
    int edificio[PISOS][DEPARTAMENTOS] = {
        {3, 5, 2, 1, 0},
        {2, 4, 3, 0, 2},
        {1, 0, 1, 2, 1},
        {0, 1, 0, 0, 0},
        {1, 3, 2, 1, 1},
        {2, 2, 0, 1, 3},
        {1, 0, 0, 2, 1},
        {0, 0, 1, 1, 1}
    };

    pisoConMasHabitantes(edificio) => 2
    cantidadDeViviendasVacias(edificio) => 12
    promedioHabitantesPorVivienda(edificio) => 1.325
    viviendaConMasHabitantes(edificio) => 1-B
```

Ejercicio 6

Dado un archivo de texto que contiene datos de minutos trabajos por personal de una consultora de sistemas informáticos a diferentes clientes, se pide obtener los totales por cliente/año en forma de grilla como se muestra debajo.

El archivo contiene los siguientes datos: “Cliente Año Mes Minutos_Trabajados”. El separador de campos es un “Tabulador”.

A continuación se muestra un pequeño ejemplo. Si el tuviera el siguiente contenido:

Cliente Anio Mes Minutos_Trabajados
1000 2019 01 50
1001 2020 02 120
1001 2021 03 95
1001 2022 01 48

La tabla a producir por el programa debería ser:

Clientes / Años	**2019**	**2020**	**2021**	**2022**
1000	50			
1001		120	95	48

Se adjunta un TXT con estos datos según lo especificado a procesar con el nombre practica-0-ejercicio-11.txt que contiene aprox. unas 400.000 filas a procesar.

Pruebas de escritorio

Los programas siguientes son presentados sin el texto común `main() {...}`.

En cada uno de los siguientes casos, sin ejecutar el código, interpretar y justificar qué es lo que se imprime en la salida estándar.

Prueba 1

```
int i = 4, x = 5;
for (i = 0; i < 10; i++) {
    if ( i < x )
        printf("%d ", i);
    else
        printf("%d ", i-x);
}
```

Prueba 2

```
int array[10], i = 0;
while (i<10) {
    array[i] = i * i;
    i++;
}
do {
    printf("%d", array[--i]);
} while (i >= 0);
```

Prueba 3

```
int mi_funcion(int x) {
    x=x*5;
    return x;
}

void main(void){
    int y = 3, x = 4;
    printf(“La function devuelde %d”, mi_funcion(y));
    printf(“La variable vale %d”, x);
}
```

Prueba 4

```
int *punta, *puntb;
int x = 7;
int y = 5;
punta = &x;
*punta = 3;
puntb = &y;
*puntb = x;
x = 9;
printf("%d, %d",*puntb, *punta);
```

Prueba 5

```
int datos(int x, float y, char c)
{
    printf("%d %f %c", x, y, c);
    x = 8;
    y = 4.2;
    c = 'g';
    return x;
```

```
}

void main(void)
{
    int x = 9;
    float y = 44.6;
    char c = 'a';
    x = datos (x, y, c);
    printf("%d %d %f %c", x, y, c);
}
```

Prueba 6

```
struct medidas{
    int alto, ancho, largo;
};

void main(void){
    int i;
    struct medidas cajas[5];
    for(i=0;i<5;i++) {
        cajas[i].alto = 4;
        cajas[i].ancho = 2 * i;
        cajas[i].largo = i + 1;
    }
    for(i=0;i<5;i++) {
        printf("Medidas de cubilete Nº%d: %d alto, %d ancho, %d largo",
        cubiletes[i].alto, cubilietes[i].ancho, cubilietes[i].largo);
    }
}
```