ALGORITMOS Y ESTRUCTURA DE DATOS

Nota para el alumno

Esta guía de ejercicios ha sido desarrollada por los profesores de la cátedra de Algoritmos y Estructura de Datos de la Universidad Tecnológica Nacional, Facultad Regional Buenos Aires, República Argentina.

La presente no está totalmente depurada pudiendo contener errores de cualquier tipo: sintácticos, lógicos, de redacción, de código, etcétera.

Por lo expuesto, se agradecerá que en caso de detectar cualquiera de estos errores lo informe por mail a ejercicios.aye@gmail.com indicando además del error detectado, el número de ejercicio y el número de versión de la guía, indicado a continuación.

Versión

Algoritmos y estructura de datos - Modulo II (guia de ejercicios) v2013_2.1

Módulo II, guía de ejercicios

Ejercicio 1

Dado el siguiente fragmento de código que declara una estructura, represéntela gráficamente e indique la nomenclatura que permite acceder a cada uno de sus campos.

```
struct Alumno
{
   int legajo;
   string nombre;
   double nota;
};
```

Ejercicio 2

Dado el siguiente fragmento de código que declara estructuras anidadas, represente gráficamente la estructura principal e indique la nomenclagura que permite acceder a cada uno de sus campos.

```
struct Calificaciones
{
   int parcial1;
   int parcial2;
   int final;
}
```

```
struct Alumno
{
   int legajo;
   string nombre;
   Calificaciones calif;
};
```

Ejercicio 3

Idem anterior.

```
struct Direccion
{
   string calle;
   int numero;
   int piso
   string dto;
   string codigoPostal;
};

struct Fecha
{
   int dia;
   int mes;
   int anio;
};

struct Documento
{
   string tipo;
   string numero;
};

struct DatosPersonales
{
   Direccion direccion;
   Fecha fechaNacimiento;
   Documento documento;
   string telefono;
   string nacionalidad;
};

struct Alumno
{
   int legajo;
   Fecha fechaIngreso
   DatosPersonales datosPersonale;
};
```

Ejercicio 4

Desarrolle y pruebe la siguiente función que asigna en `a` los primeros `n` números primos. Se asegura que el *array* `a` tendrá una capacidad mayor o igual a `n`.

```
void obtenerPrimos(int a[], int n);
```

Ejercicio 5

Desarrolle y pruebe la siguiente función que asigna en `a` los primeros `n` términos de la sucesión de Fibonacci. Se asegura que el *array* `a` tendrá una capacidad mayor o igual a `n`.

```
void obtenerFibonacci(int a[], int n);
```

Ejercicio 6

Desarrolle y pruebe las siguientes funciones que proveen información sobre el mayor y el menor elemento contenido dentro de un *array*.

Función: `getMax`.

Retorna la posicion de la primer ocurrencia del elemento de mayor valor del *array* `a`.

```
int getMax(int a[], int len);
```

Función: `getMin`.

Retorna la posicion de la primer ocurrencia del elemento de menor valor de `a`.

```
int getMin(int a[], int len);
```

Función: `getMaxMin`.

Asigna en `ret` (que es un `int[2]`) las posiciones de la primer ocurrencia de los elementos de mayor y menor valor respectivamente contenidos en el array `a`.

```
void getMaxMin(int a[], int len,int ret[]);
```

Función: `getMaximos`.

Asigna en `ret` las posiciones de todas las ocurrencias del elemento de mayor valor contenido en `a`. Se asegura que no serán más de 10. Retorna la cantidad de ocurrencias detectadas de dicho valor.

```
int getMaximos(int a[], int len, int ret[]);
```

Función: `getMinimos`.

Idem anterior pero con las posiciones del elemento de menor valor contenido en `a`.

```
int getMinimos(int a[], int len, int ret[]);
```

Ejercicio 7

Desarrolle y pruebe las siguientes funciones que realizan operaciones sobre los elementos de un *array*.

Función: `sumatoria`.

Retorna la sumatoria de los valores contenidos en el *array* `a`.

```
long sumatoria(int a[], int len);
```

Función: `productoria`.

Retorna la productoria de los valores contenidos en el *array* `a`.

```
long productoria(int a[], int len);
```

Ejercicio 8

Desarrolle y pruebe las siguientes funciones que seleccionan valores de dos *arrays*.

Función: `intercala1`.

Asigna en `c` los valores intercalados de `a` y `b`. Ambos *arrays* tienen la misma longitud.

```
void intercala1(int a[], int b[], int len, int c[]);
```

Función: `intercala2`.

Asigna en `c` **los valores intercalados de** `a` **y** `b`.

```
void intercala2(int a[], int lenA, int b[], int lenB, int c[]);
```

Función: `interseccion`.

Asigna en `c` **los valores contenidos en** `a` **y en** `b`. **Retorna la cantidad de valores asignados.**

```
int interseccion(int a[], int lenA, int b[], int lenB, int c[]);
```

Función: `exclusion`.

Asigna en `c` **los valores contenidos en** `a` **o en** `b`, **pero no en ambos. Retorna la cantidad de valores asignados**

```
int exlusion(int a[], int lenA, int b[], int lenB, int c[]);
```

Ejercicio 9

Desarrolle y pruebe la función `invertirArray` **que invierte los valores de** `a` **con las restricciones que se detallan en cada caso.**

- **Utilizando dos subíndices.**
- **Utilizando un único subíndice.**

```
void invertirArray(int a[], int len);
```

Ejercicio 10

Desarrolle y pruebe las siguientes funciones que permiten realizar operaciones aritméticas entre números muy grandes representados en *arrays* de enteros; cada elemento del *array* corresponde a un dígito del número que representa. Ambos *arrays* tienen la misma longitud; los números representados pueden contener ceros a la izquierda.

Función: `suma`.

Asigna en `res` **la suma de los valores que representan** `a` **y** `b`.

```
void suma(int a[], int b[], int len, int res[]);
```

Función: `resta`.

Asigna en `res` **la resta de los valores que representan** `a` **y** `b`. **Aceptaremos que** `a>=b`;

```
void resta(int a[], int b[], int len, int res[],);
```

Rediseñe y desarrolle las funciones `suma` **y** `resta` **de forma tal que permitan operar con valores negativos.**

Ejercicio 11

Desarrolle y pruebe los *template* `maxGenerico` **y** `minGenerico` **que retornan respectivamente la posición de la primer ocurrencia del máximo y mínimo valor contenido en un *array* de tipo** `T`.

```
template <typename T> int maxGenerico(T a[], int len, int (*compara)(T,T));
template <typename T> int minGenerico(T a[], int len, int (*compara)(T,T));
```

Ejercicio 12

Desarrolle y pruebe las siguientes funciónes:

Función: `ordenado`.

Indica si un *array* de enteros está ordenado ascendentemente.

```
bool ordenado(int a[], int len);
```

Función: `ordenadoGenerico`.

Indica si un *array* está ordenado según el criterio que determina la función `comparar`.

```
template <typename T> bool ordenadoGenerico(T a[],int len,int (*comparar)(T,T);
```

Ejercicio 13

Dada la siguiente estructura:

```
struct Cliente
{
   int codigo;
   string nombre;
   double saldo;
};
```

Desarrolle un programa que, utilizando la función `ordenadoGenerico` del ejercicio anterior, permita determinar si un *array* de `Cliente` está ordenado ascendentemente por código, por nombre o por saldo.

Ejercicio 14

Dado un conjunto de registros cuya estructura se detalla a continuación:

```
struct Producto
{
   int codigo;
   string descripcion;
   double precio;
};
```

Se pide imprimirlos ordenados ascendentemente por `codigo`, por `descripcion` y finalmente decrecientemente por `precio`.

Ejercicio 15

Se arrojan 2 dados 50 veces. Escriba un programa que permita ingresar los valores de cada una de estas tiradas y, luego, muestre una lista indicando cuántas veces se dio cada uno de los resultados posibles.

2 salio: 99 veces
3 salio: 99 veces
:
12 salio: 99 veces

Ejercicio 16

En un club social se abrieron las inscripciones de los socios para los diferentes deportes, que están codificados de 1 a 20; y de cada inscripción se conoce número de socio y código de deporte.

Las inscripciones finalizan con un número de socio igual a 0. Se pide informar:

- La cantidad de inscriptos a cada deporte.
- El código de deporte con la mayor cantidad de inscriptos.

Ejercicio 17

Idem anterior pero considerando que los, a lo sumo, 20 deportes están codificados con 5 caracteres.

Ejercicio 18

Se dispone de un conjunto de registros que corresponden a las ventas del mes de un comercio. Cada registro contiene la siguiente información:

- `codCliente`, (8 dígitos)
- `codArticulo`, (*XX999999*)
- `cantidad`,
- `dia`.

Se pide informar:

1 - Total de unidades vendidas por cada artículo

2 - Total de compras por cada cliente.

3 - Si considera que los datos proporcionados no son suficientes, indique qué datos faltan y defínalos.

Ejercicio 19

Se cuenta con un conjunto de registros que representan productos, codificados de 1 a 100, sin ningún orden y con el siguiente diseño:

- `codigoProducto` (de 1 a 100)
- `precioUnitario` (`float`)

Se dispone también de un conjunto registros que corresponden a los pedidos que realizaron los clientes; de cada uno de estos pedidos se conoce `codigoProducto` y `cantidad` de unidades pedidas (`int`).

El ingreso de datos de ambos conjuntos de registros finalizará al ingresar un código de producto negativo.

Se pide:

- Informar, por cada pedido, código de producto, unidades, precio unitario y monto a abonar.
- Informar los códigos de los productos que no recibieron ningún pedido.
- Emitir un listado indicando por cada producto el total facturado.

Ejercicio 20

Idem anterior pero considere que los productos se codifican con un número entero de 8 dígitos. Se sabe que, a lo sumo, existen 100 productos diferentes.

Ejercicio 21

Se dispone de un conjunto registros que representan los cursos de una cátedra universitaria. De cada uno de ellos se conoce el `codigoCurso` (4 caracteres) y `cantAlumnos`. Además, por cada curso se ingresarán los legajos y las notas (de 1 a 10) de cada uno de los alumnos.

Se pide:

- Informar por cada curso, cuantos alumnos obtuvieron cada una de las notas posibles.
- Informar también el código el porcentaje de aprobados y reprobados de cada curso.

Caso A: Los registros se ingresan ordenados por `codigoCurso`. Caso B: El ingreso no tiene sin ningún orden.

Ejercicio 22

Una empresa de aviación que realiza 500 vuelos semanales a distintos puntos del país requiere desarrollar un programa que agilice la venta de pasajes.

Para ello cuenta con el archivo `VUELOS.DAT`, con un registro por cada uno de los 500 vuelos, sin ningún orden, con el siguiente diseño:

```
struct Vuelo
{
   string codigoVuelo; // 6 caracteres
   int cantPasajesDisponibles; // 3 digitos
};
```

Además se dispone del archivo `COMPRADORES.DAT` con los datos de los compradores interesados en adquirir pasajes. Los registros de este archivo tienen la siguiente estructura:

```
struct Comprador
{
   string codigoVuelo;
   int cantPasajesSolicitados;
   long dniSolicitante; // 8 digitos
   string apeYNomSolicitante; // 25 caracteres
};
```

Se pide:

1 - Emitir el siguiente listado para aquellos compradores que efectivamente podrán adquirir sus pasajes.

DNI	Nombre	Cantidad Pasajes	Código Vuelo
99999999	XXXXXXXXXXX	999	XXXXXXXXXX
99999999	XXXXXXXXXXX	999	XXXXXXXXXX
99999999	XXXXXXXXXXX	999	XXXXXXXXXX

2 - Emitir el siguiente listado ordenado por código de vuelo.

Código Vuelo	Pasajes Disponibles	Pasajes No Vendidos
XXXXXX	999	999
XXXXXX	999	999
XXXXXX	999	999

NOTA: Solo se venderán pasajes a aquellos compradores interesados a los que se les pueda vender la totalidad de los pasajes que solicitan. De lo contrario, esa cantidad de pasajes se imputará como "pasajes no vendidos".

Ejercicio 23

Una empresa que distribuye mercadería hacia distintas localidades del interior cuenta con los siguientes archivos: `DESTINOS.DAT`, con información sobre la distancia hacia cada uno de los destinos y la siguiente estructura:

```
struct Destino
{
   int nroDestino;
   int distanciaEnKM;
};
```

Además se dispone del archivo `VIAJES.DAT` que tiene la siguiente estructura:

```
struct Viaje
{
   string patenteCamion;
   int nroDestino;
   string codChofer; // 3 caracteres
};
```

NOTA: Se sabe que la empresa dispone de, a lo sumo, 50 camiones y no más de 100 choferes.

Se pide informar:

1. Cantidad de viajes realizados a cada destino.
2. Código del chofer con menor cantidad de km recorridos.
3. Patente de los camiones que viajaron a cada destino.

Ejercicio 24

Desarrolle y pruebe las siguientes funciones.

Función: `tablaDel`.

Completa a `t` con los valores de la tabla de multiplicar de `n`. Por ejemplo: `t[0]=n*1`, `t[1]=n*2`,...

```
void tablaDel(int n, int t[]);
```

Función: `tablas`.

Completa a `t` con los valores de todas las tablas de multiplicar.

```
void tablas(int t[10][10]);
```

Ejercicio 25

Invocar a la función `tablas` del ejercicio anterior y desarrollar los siguientes items:

- Mostrar su contenido recorriéndola por filas.
- Mostrar su contenido recorriéndola por columnas.
- Informar la sumatoria de sus elementos y su promedio.

Ejercicio 26

Desarrolle y pruebe la siguiente función.

Función: `sumaFila`.

Completa a `s` con la sumatoria de los valores de las filas de `t`, que es una matriz cuadrada cuya capacidad máxima es: 50.

```
void sumaFila(int t[50][50], int s[], int len);
```

Ejercicio 27

Desarrolle y pruebe la siguiente función.

Función: `procesaFila`.

Completa a `s` aplicando el proceso `procesar` sobre cada una de las filas de `t`, que es una matriz cuadrada cuya capacidad máxima es: 50.

```
void procesaFila(int t[50][50], int s[], int len, int (*procesar)(int,int));
```

Ejercicio 28

Diseñe, desarrolle y pruebe una función que realice el producto de dos matrices.

Ejercicio 29

Una fábrica de calzados elabora 7 modelos en 5 colores diferentes. Las ventas realizadas de cada modelo y cada color están registradas en un archivo llamado: `VENTAS.DAT`:

```
struct Venta
{
   int modelo
   int color;
   int cantidadVendida;
};
```

Se pide emitir un listado que informe los totales vendidos de cada uno de los modelos por cada uno de los colores. Informar también los totales por cada modelo, los totales por color y el total general.

Ejercicio 30

Un comercio textil, vende sus prendas en diferentes talles. Para realizar la facturación dispone de los siguientes archivos:

Un archivo PRENDAS.DAT con los precios de cada prenda/talle que comercializa, con la siguiente estructura.

```
struct Prenda
{
   int codPrenda; // de 1 a 100
   int talle;     // de 1 a 5
   double precio;
};
```

Las ventas que se han realizado y para las cuales hay que emitir las facturas correspondientes están registradas en el archivo VENTAS.DAT que tiene la siguiente estructura.

```
struct Venta
{
   string nombreCliente
   string fechaCompra;
   int codPrenda;
   int talle;
   int unidades;
};
```

Se pide:

1 - Desarrollar un programa que imprima la siguiente factura para cada venta, considerando un descuento del 10% cuando se lleven 3 o más unidades. Se debe considerar el IVA al 21%.

2 - Al finalizar el proceso emitir el siguiente listado, ordenado por prenda y talle.

Prenda: 999

Talle	**Unidades**
999	999
999	999
Total:	999

: : :

Total General: 999