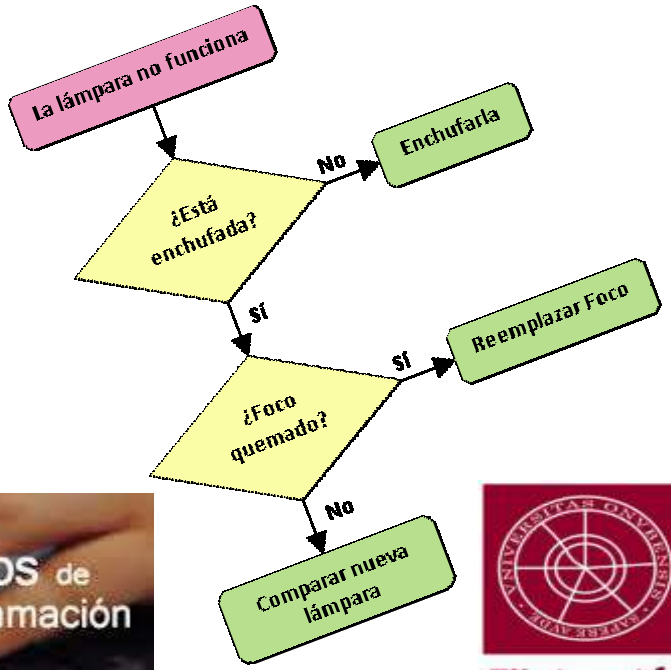
Tema 4.

Tipos Estructurados de Datos

Fundamentos de Programación
Grado en Ingeniería Informática

Profesores:
José Manuel Martín Ramos
Francisco Roche Beltrán.



ÍNDICE

1. Registros. Registros jerarquizados.

Definición: Tipo de dato estructurado estático en el que sus componentes (llamados campos) pueden ser de distinto tipo.

	Homogénea	Heterogénea
Estática	Tablas	Registros
Dinámica	Tablas dinámicas	Memoria Dinámica

Declaración Tipo de Datos…

```
struct nombre_registro  {
  tipo  nombre_campo1;
  tipo  nombre_campo2;
  tipo  nombre_campo3;
  } ;
```

Otra forma:

```
typedef  struct {
     tipo nombre_campo1;
     tipo nombre_campo2;
     tipo nombre_campo3;
     } nombre_registro;
//Los dos acaban con ;
```

```
//Declaración de tipos
struct alumno {
      int tf;
      int edad;
      float peso;
      char sexo;
      int DNI;
      char letranif;
      };


int main () {
  //Declaración de variables
  alumno a, b;
  ...
  return 0;
}
```

```
//Declaración de tipos
struct    persona {
     float     peso;
     double    altura;
     };
struct fecha {
     int dia;
     int mes;
     int anio;
     };


int main () {
  //Declaración de variables
  persona p1,p2;
  fecha  f;
  …
  return 0;
}
```

Operaciones con registros.

```
#include <iostream>
using namespace std;
struct punto {
  int x;
  int y;
  };
int main ( ){
  punto uno, dos;
  punto origen ={0,0};
  uno.x = 3;
  uno.y = 4;
  //cin >> dos;   Error
  cin >> dos.x;
  cin >> dos.y;
  origen = dos;
  //cout << origen;  Error
  cout << "Origen tiene de coordenada x : "
       << origen.x <<"\n";
  cout << "Origen tiene de coordenada y : "
       << origen.y;
  return 0;
}
```

Inicialización:

- En la declaración.
- Sentencia de asignación.
- Desde teclado.

Copia de registros completos.

Mostrar por pantalla campos de un registro

Registros jerarquizados.

```
#include <iostream>
using namespace std;
struct  fechas{
  int   dia;
  int   mes;
  int   anio;
  };
struct  InforPersona{
  float     peso;
  float     altura;
  fechas    nacimiento;
  };
int main () {
  InforPersona persona;
  cout << "Indica tu peso: ";
  cin >> persona.peso;
  cout << "Indica el dia del mes en que naciste: ";
  cin >> persona.nacimiento.dia;
  cout << "Indica el numero del mes en que naciste: ";
  cin >> persona.nacimiento.mes;
  persona.nacimiento.anio = 1995;
  cout << " Naciste el dia " << persona.nacimiento.dia << " mes "
    << persona.nacimiento.mes << " a" << char(164) <<"o "
    << persona.nacimiento.anio << " y tu peso es : " << persona.peso << " Kg.";
  return 0;
}
```

2. Tablas. Tablas de registros.

Definición: Tipo de dato estructurado estático en el que sus componentes (llamados elementos) tienen que ser del mismo tipo.

	Homogénea	**Heterogénea**
Estática	Tablas	Registros
Dinámica	Tablas dinámicas	Memoria Dinámica

Sintaxis declaración de una tabla (array, vector):

Tipo_cada_elemento nombre_variable [número_de_elementos];

```
#include <iostream>
using namespace std;
int main () {
  float notas[10];
  char palabra[20];
  double coleccion[700];
 ….
 return 0;
}
```

Acceso a los elementos de una tabla.

```
int main () {
    int iva[45];
    //El primer elemento tiene como índice 0 y el último el 44
    iva[0] = 22;
    iva[44] = 8;
    iva[45] = 13;    //Error
    …
}
```

El número de dimensiones de una tabla es el número de índices que tiene:

```
int tab[10];                        //Una dimensión
char palabras[10][20];              //Dos dimensiones…
float  almacen[10][15][10][13];     //Cuatro dimensiones…
```

Operaciones con tablas:

```
#include <iostream>
using namespace std;
int main (){
  int tab1[5];
  int tab2[3][3];
  int tab3[5] = { 2, 4, 6, 8, 10};
  tab1[0] = 3;
  tab2[1][0] = 5;
  cin >> tab3[0];
  //tab1 = tab3;   ERROR
  for (int i=0; i < 5; i++)
    tab1[i]= tab3[i];
  //cin >> tab3    ERROR
  for (int i=0; i < 5; i++)
    cin >> tab3[i];
  for (int i=0; i < 3; i++)
    for (int j=0; j < 3; j++)
        cin >> tab2[i][j];
  cout << tab3[0];
  cout << tab2[0][0];
  //cout << tab3; ERROR
  cout << "Muestro la tabla 1: ";
  for (int i=0; i < 5; i++)
    cout << tab3[i] << " ";
  cout << "\nMuestro la tabla 2: ";
  for (int i=0; i < 3; i++)
    for (int j=0; j < 3; j++)
        cout << " " << tab2[i][j];
  return 0;
}
```

Inicialización:

- En la declaración.
- Sentencia de asignación.
- Desde teclado.

Copiar una tabla en otra.

Leer desde teclado una tabla

Mostrar por pantalla un elemento.

Mostrar una tabla por pantalla,

Tablas de registros.

```
#include <iostream>
using namespace std;
struct  Tpersona{
  float     peso;
  float     altura;
  int       dni;
  };
int main () {
  Tpersona persona[3];
  for (int i=0; i <3; i++){ //Relleno la tabla con valores
    cout << "Introduce el peso de la persona " << i <<": ";
    cin >> persona[i].peso;
    cout << "Introduce la altura de la persona " << i << ": ";
    cin >> persona[i].altura;
    cout << "Introduce el dni (sin letra) de la persona " << i << ": ";
    cin >> persona[i].dni;
  }
  for (int i=0; i <3; i++){ //Muestro la tabla por pantalla
    cout << "\nPersona n." << i << ":\n";
    cout << "\tpeso: " << persona[i].peso;
    cout << "\taltura: " << persona[i].altura;
    cout << "\tdni: " << persona[i].dni;
  }
  return 0;
}
```

3. Esquemas de recorrido y búsqueda.

Esquema de recorrido

Cuando siempre debo tratar TODOS los elementos obligatoriamente

Supongamos que tenemos declarado:

int tabla1[6];

for (int i=0; i < 6, i++)

Tratamiento del elemento i;

Tratamiento final

Rellenar la tabla con valores leídos desde teclado:

```
for (int i=0; i <6; i++)
      cin >> tabla1[i];
```

Mostrar por pantalla todos los elementos de la tabla1:

```
for (int i=0; i <6; i++)
      cout << tabla1[i];
```

Rellenar la tabla con el valor de su índice:

```
for (int i=0; i <6; i++)
      tabla1[i] = i;
```

Esquema de búsqueda

Cuando NO siempre debo de tratar todos los elementos obligatoriamente.

Accedo al primer elemento.
No encontrado.
while ((no último) && (no encontrado))
** if (elementoi == buscado) encontrado**
** else accedo al siguiente elemento**
if (encontrado) tratar elemento_encontrado
else tratar ausencia_del_elemento

```
char letras[20];
//¿Está la letra a en la tabla?
i=0;
encontrado = false;
while ( (i<20) && (encontrado==false))
  if (letras[i]=='a') encontrado=true;
  else  i++;
if (encontrado) cout << "Esta la a";
else cout << "NO esta la a";
```

Indique si es recorrido o búsqueda	Esquema de recorrido	Esquema de búsqueda
¿Cuántos “Luis” hay en la tabla?		
¿Cuál es el valor máximo almacenado?		
¿Esta “Juan” en la tabla?		
¿Cuánto suman los valores positivos almacenados?		
¿En qué fila y columna está almacenado “Javier”?		
Cargar la tabla completa con valores leídos desde teclado		
¿Son idénticas dos tablas en sus elementos?		
Indique por pantalla si una matriz es simétrica		
Copiar una tabla en otra		

4. Cadenas.

Una cadena es una tabla de una dimensión de elementos de tipo char acabada con el código ASCII número 0.

```
char pal1[20];
```

Operaciones con cadenas:

```
cin  >> pal1;      //Por ejemplo Francisco
/*Si se pone por teclado Juan Luis, sólo tomará la primera
palabra, quedando pal1 con el contenido Juan*/
cout << pal1;    /*Mostraría por pantalla la tabla hasta el
código ASCII número 0 */
```

c1 y c2 deben de ser necesariamente tablas de una dimensión de tipo char. **#include <string.h>**

Nombre de la Función	***Actuación***
strcpy(c1,c2)	Copia c2 en c1
strcat(c1,c2)	Concatena c2 al final de c1
strlen(c1)	Devuelve la longitud de c1
strcmp(c1,c2)	Devuelve 0 si son iguales, menor que 0 si c1<c2 y mayor que 0 cuando c1>c2
strchr(c1,car)	Devuelve un índice a la primera ocurrencia de car en c1
strstr(c1,c2)	Devuelve un índice a la primera ocurrencia de c2 en c1

```
#include<iostream>
#include <string.h>
using namespace std;
#define M 3
typedef char cadena[30];
int main() {
    cadena tabla[M];
    cadena leida;
    bool encontrado;
    int i;
    for(i = 0; i < M; i++){
        cout << "\nIntroduce la palabra " << i <<"\n";
        cin >> tabla[i];
        }

    cout    << "Introduzca la palabra a buscar\n";
    cin >> leida;
    //Esquema de búsqueda de tabla[i] == leida !! OJO
    i = 0;
    encontrado = false;
    while ( (i < M)  &&  (!encontrado) ) {
        if (strcmp(tabla[i], leida) == 0)
                    encontrado = true;
        else i++;
        }
    if (encontrado) cout << "Se encontro en la posicion "
                         << i;
    else        cout << "No se encontro";
    return 0;
}
```

#include <string.h>

Utilización de typedef

Tabla de 3 cadenas, cada una de 30 char máximo.

Recorrido, rellenando la tabla de 3 palabras desde teclado.

Palabra a buscar

Búsqueda de la palabra en la tabla.
Utilización de strcmp

5. Objetos y Clases.

```
#include <iostream>
using namespace std;

class fraccion{
…
};




int main () {
  int a;
  a = 1;
  cout << a;
  return 0;
}
```

```
#include <iostream>
using namespace std;

class fraccion {

  Atributos y métodos privados de la clase

public:
  Métodos públicos de la clase

};

int main () {
…
}
```

Una class es un tipo estructurado de datos heterogéneo en donde a diferencia de los registros sus elementos son privados y tiene la posibilidad de declarar funciones (métodos en la terminología de POO) que trabajen con dicha información privada.

```cpp
#include<iostream>
#include <string.h>
using namespace std;
#define M 3
typedef char cadena[30];
class tpalabras {
    cadena tabla[M];
public:
 void leer();
 void buscar();
};
void tpalabras::leer() {
 //Esquema de recorrido
 for(int i = 0; i < M; i++){
   cout << "\nIntroduce la palabra " << i <<"\n";
   cin >> tabla[i];
   }
}
```

```cpp
void tpalabras::buscar(){
 cadena leida;
 bool encontrado;
 int i;
 cout << "Introduce la palabra a buscar\n";
 cin >> leida;
 //Esquema de búsqueda
 i = 0;
 encontrado = false;
 while ( (i < M)  &&  (!encontrado) ) {
   if (strcmp(tabla[i], leida) == 0)
              encontrado = true;
   else i++;
   }
 if (encontrado) cout << "Se encontro en"
                   <<" la posicion " << i;
 else       cout << "No se encontro";
}
int main() {
 tpalabras palabras;
 tpalabras palabrasb[2];
 palabras.leer();
 palabras.buscar();
 for (int i=0; i < 2; i++)
     palabrasb[i].leer();
 //...
 return 0;
}
```