Metodología de la Programación

Tema 0. Paso de parámetros por referencia y Registros (Structs)

Departamento de Ciencias de la Computación e Inteligencia Artificial

ETSIIT Universidad de Granada

Curso 2015-16

Contenido del tema

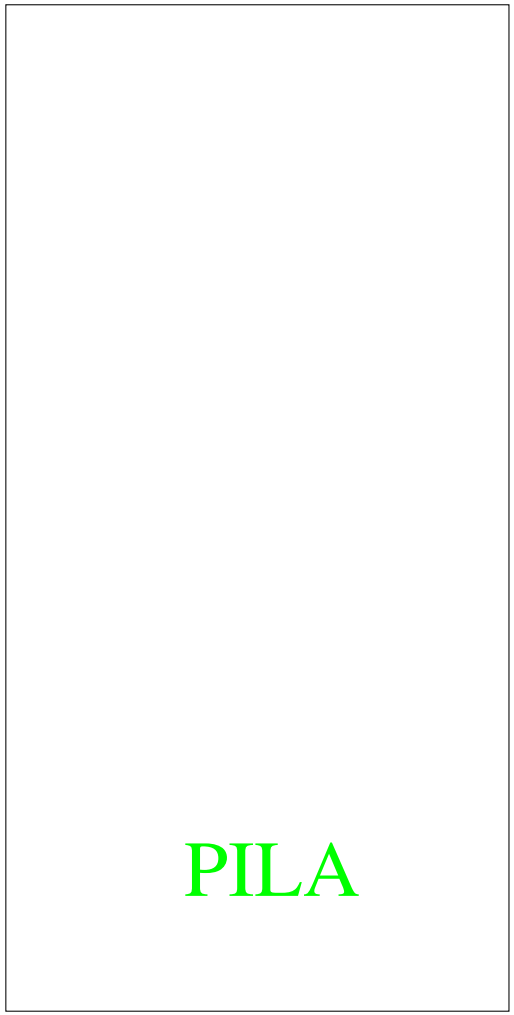
1. Paso de parámetros a funciones
2. Paso de parámetros por referencia
3. Paso de parámetros y devolución
4. Registros (Structs)
   - Definición y sintaxis
5. Operaciones con estructuras
   - Declaración de variables
   - Inicialización
   - Operadores de acceso a miembros de la estructura
   - Operación de asignación
   - Operaciones de entrada y salida
6. Paso de estructuras a funciones
7. Estructuras de estructuras

Contenido del tema

Ejercicio:

Construir un módulo que intercambie el valor de dos variables

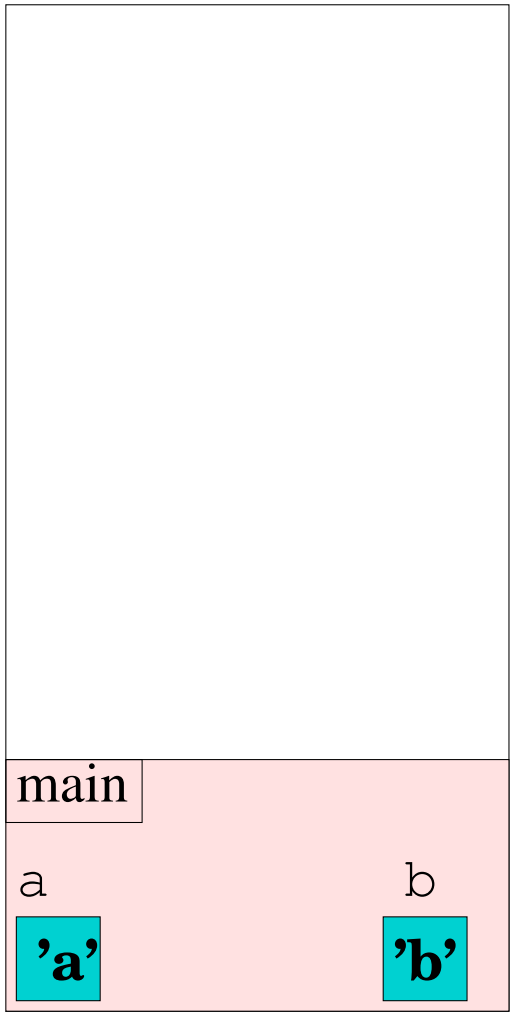
```
1  #include <iostream>
2  using namespace std;
3
4  void Swap(char c1, char c2){
5    char aux=c1;
6    c1=c2;
7    c2=aux;
8  }
9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Ejercicio:

Construir un módulo que intercambie el valor de dos variables

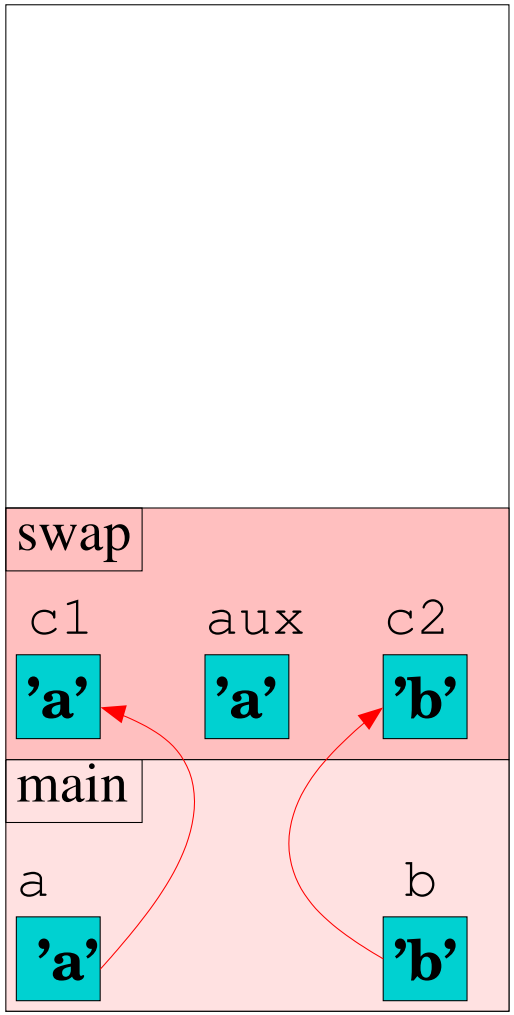
```
1  #include <iostream>
2  using namespace std;
3
4  void Swap(char c1, char c2){
5    char aux=c1;
6    c1=c2;
7    c2=aux;
8  }
9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Ejercicio:

Construir un módulo que intercambie el valor de dos variables

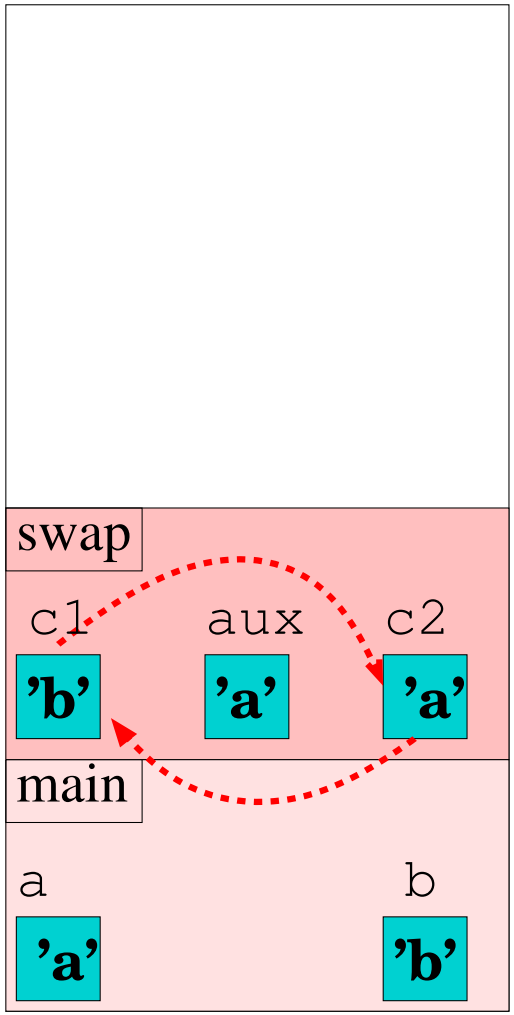
```
1  #include <iostream>
2  using namespace std;
3
4  void Swap(char c1, char c2){
5    char aux=c1;
6    c1=c2;
7    c2=aux;
8  }
9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Ejercicio:

Construir un módulo que intercambie el valor de dos variables

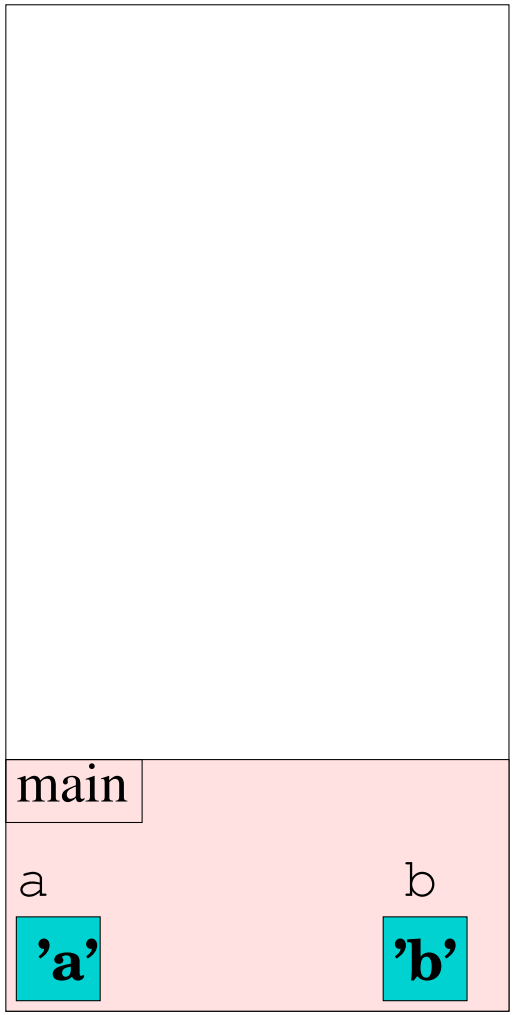
```
1  #include <iostream>
2  using namespace std;
3
4  void Swap(char c1, char c2){
5    char aux=c1;
6    c1=c2;
7    c2=aux;
8  }
9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Ejercicio:

Construir un módulo que intercambie el valor de dos variables

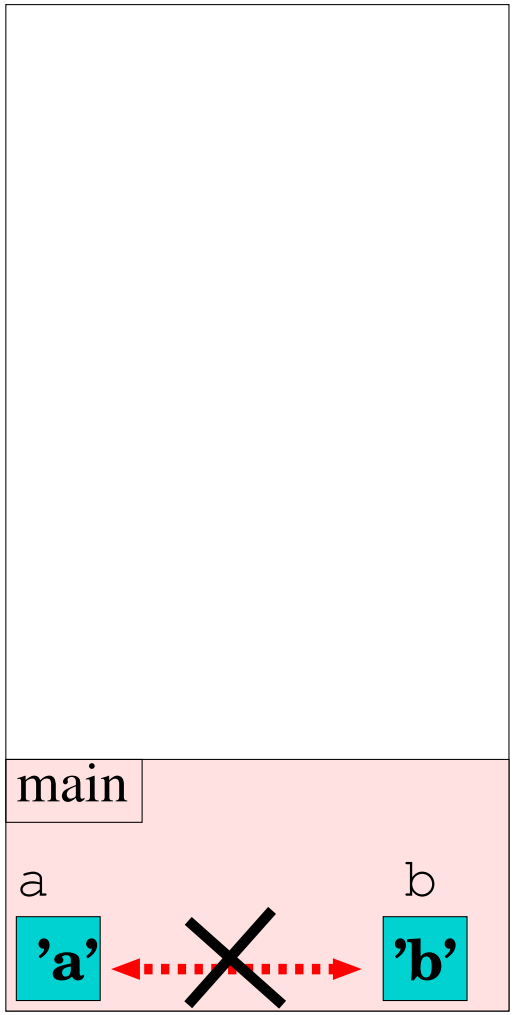
```
1  #include <iostream>
2  using namespace std;
3
4  void Swap(char c1, char c2){
5    char aux=c1;
6    c1=c2;
7    c2=aux;
8  }
9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Ejercicio:

Construir un módulo que intercambie el valor de dos variables

```
1  #include <iostream>
2  using namespace std;
3
4  void Swap(char c1, char c2){
5    char aux=c1;
6    c1=c2;
7    c2=aux;
8  }
9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Ejercicio:

Construir un módulo que intercambie el valor de dos variables

Análisis

- Los valores de las variables a y b no se han modificado.

Ejercicio:

Construir un módulo que intercambie el valor de dos variables

Análisis

- Los valores de las variables a y b no se han modificado.
- Los que se intercambiaron fueron sus copias c1 y c2.

Ejercicio:

Construir un módulo que intercambie el valor de dos variables

Análisis

- Los valores de las variables `a` y `b` no se han modificado.
- Los que se intercambiaron fueron sus copias `c1` y `c2`.
- El problema es que se necesita extender el ámbito de `a` y `b` para que sean manipulables en el entorno de `Swap`.

Ejercicio:

Construir un módulo que intercambie el valor de dos variables

Análisis

- Los valores de las variables `a` y `b` no se han modificado.
- Los que se intercambiaron fueron sus copias `c1` y `c2`.
- El problema es que se necesita extender el ámbito de `a` y `b` para que sean manipulables en el entorno de `Swap`.

Solución

Paso de parámetros por referencia

Ejercicio:

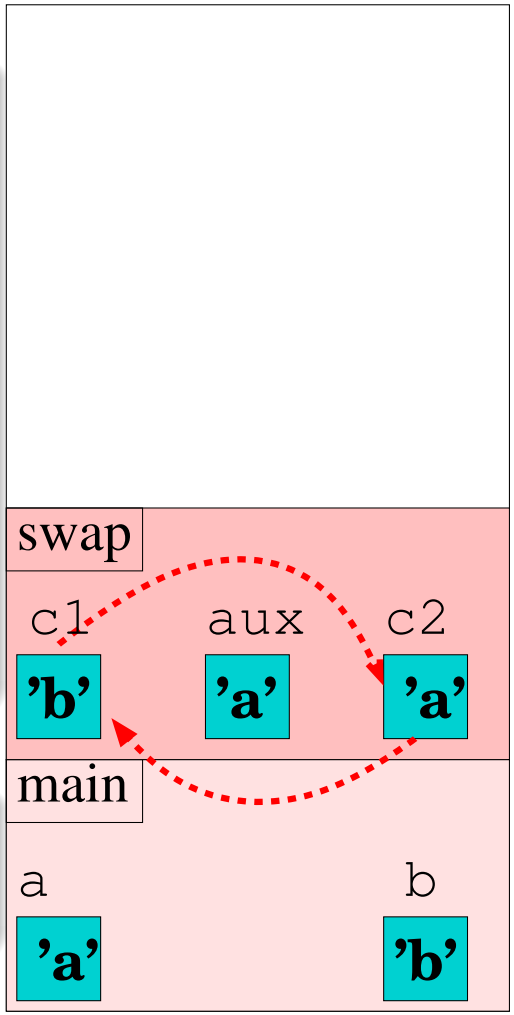
Construir un módulo que intercambie el valor de dos variables

Análisis

- Los valores de las variables a y b no se han modificado.
- Los que se intercambiaron fueron sus copias c1 y c2.
- El problema es que se necesita extender el ámbito de a y b para que sean manipulables en el entorno de Swap.

Solución

Paso de parámetros por referencia



Contenido del tema

Paso de parámetros

Por valor o copia

- Es el paso de argumentos por defecto.

Paso de parámetros

Por valor o copia

- Es el paso de argumentos por defecto.
- Durante la llamada se realiza una copia del parámetro actual en el parámetro formal.

Paso de parámetros

Por valor o copia

- Es el paso de argumentos por defecto.
- Durante la llamada se realiza una copia del parámetro actual en el parámetro formal.
- De esta forma, el módulo invocado trabaja con una copia y no con el valor original.

Paso de parámetros

Por valor o copia

- Es el paso de argumentos por defecto.
- Durante la llamada se realiza una copia del parámetro actual en el parámetro formal.
- De esta forma, el módulo invocado trabaja con una copia y no con el valor original.

Para resolver el problema del ejercicio anterior tenemos que trabajar con los datos originales y no con las copias.

Paso de parámetros

Por referencia o variable

- No realiza una copia del parámetro actual en el formal, sino un vínculo entre ellos, de tal forma que una modificación en el parámetro formal, conlleva la misma modificación en el parámetro actual.

Paso de parámetros

Por referencia o variable

- No realiza una copia del parámetro actual en el formal, sino un vínculo entre ellos, de tal forma que una modificación en el parámetro formal, conlleva la misma modificación en el parámetro actual.
- Se usa **&** entre el tipo y el identificador del argumento para indicar que el paso se realiza por referencia.

Paso de parámetros

Por referencia o variable

- No realiza una copia del parámetro actual en el formal, sino un vínculo entre ellos, de tal forma que una modificación en el parámetro formal, conlleva la misma modificación en el parámetro actual.
- Se usa **&** entre el tipo y el identificador del argumento para indicar que el paso se realiza por referencia.

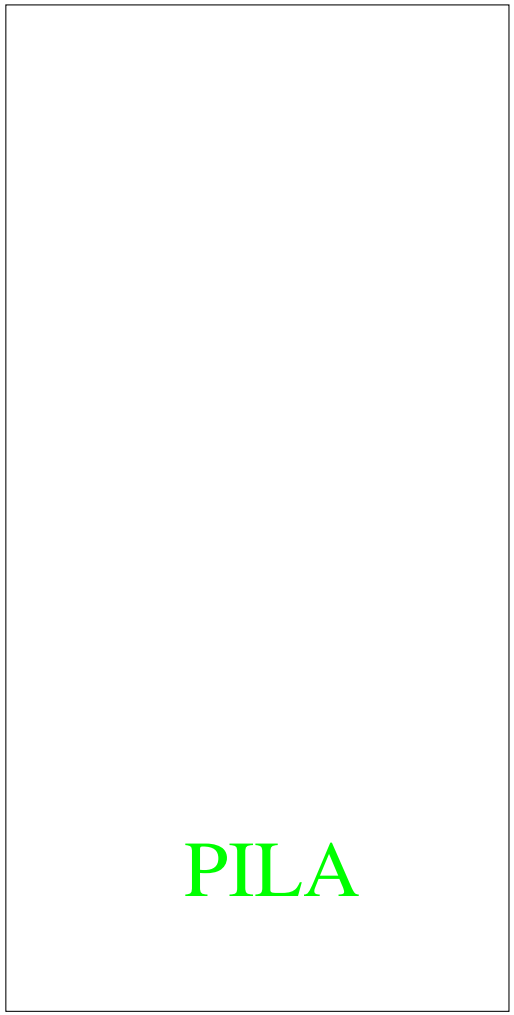
Ejemplos

```
1 void Swap (char &c1, char &c2);
2 void Division (int dividendo, int divisor,
3                int &coc, int &resto);
4 void ElegirOpcion (char &opcion);
```

Solución del ejercicio

Construir un módulo que intercambie el valor de dos variables

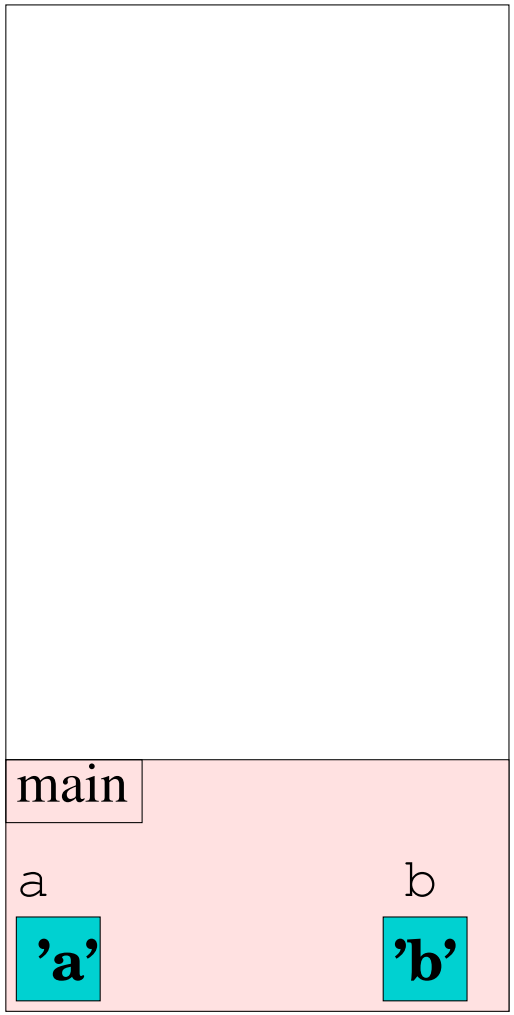
```
 1 #include <iostream>
 2 using namespace std;
 3
 4 void Swap(char &c1, char &c2){
 5   char aux=c1;
 6   c1=c2;
 7   c2=aux;
 8 }
 9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Solución del ejercicio

Construir un módulo que intercambie el valor de dos variables

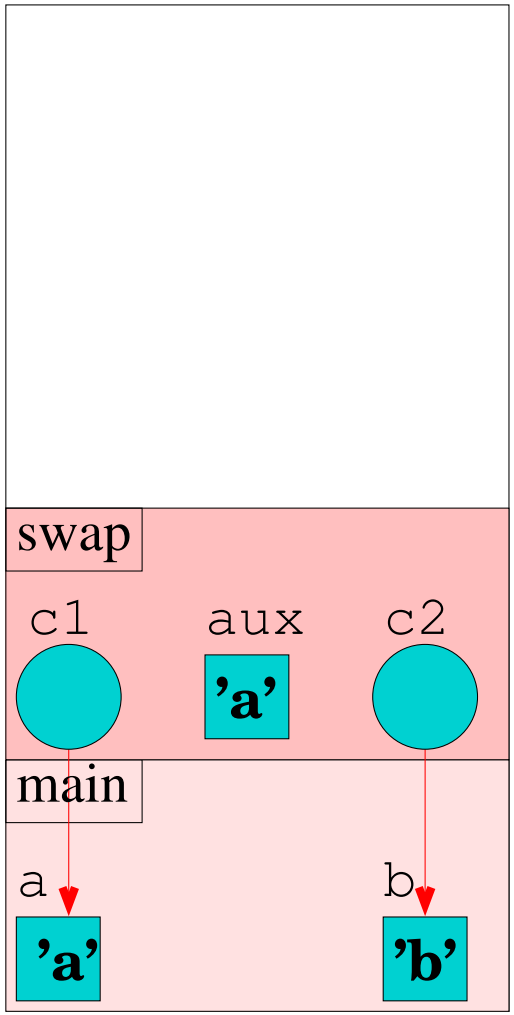
```
 1 #include <iostream>
 2 using namespace std;
 3
 4 void Swap(char &c1, char &c2){
 5   char aux=c1;
 6   c1=c2;
 7   c2=aux;
 8 }
 9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Solución del ejercicio

Construir un módulo que intercambie el valor de dos variables

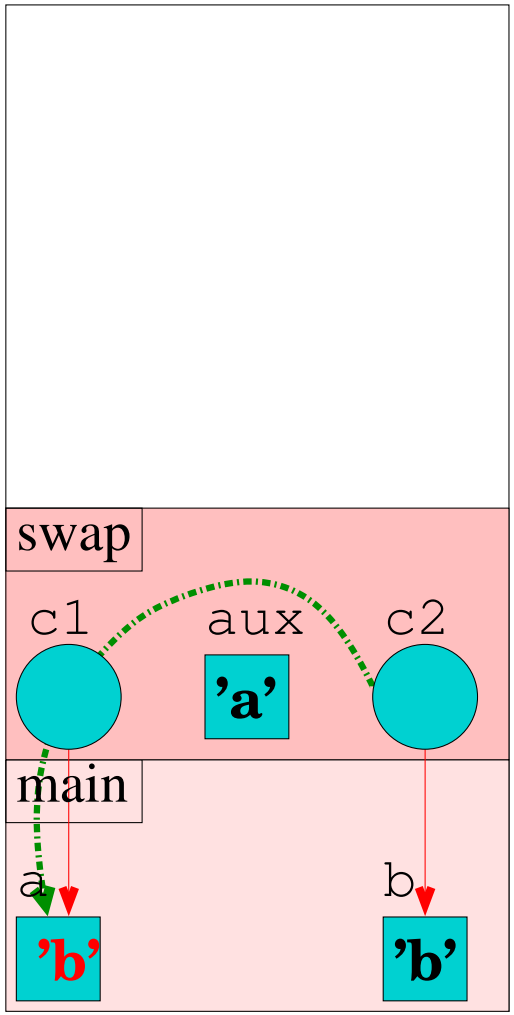
```
1  #include <iostream>
2  using namespace std;
3
4  void Swap(char &c1, char &c2){
5    char aux=c1;
6    c1=c2;
7    c2=aux;
8  }
9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Solución del ejercicio

Construir un módulo que intercambie el valor de dos variables

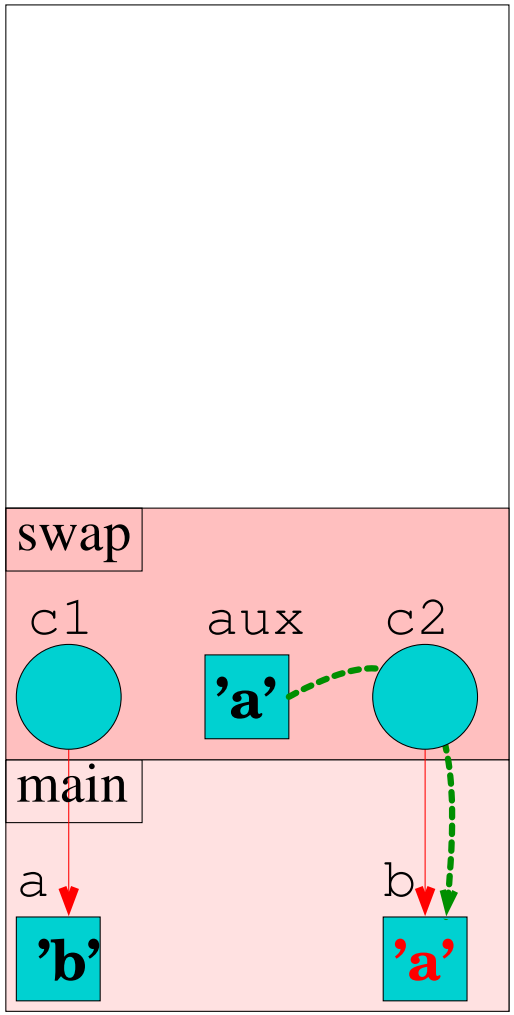
```
1  #include <iostream>
2  using namespace std;
3
4  void Swap(char &c1, char &c2){
5    char aux=c1;
6    c1=c2;
7    c2=aux;
8  }
9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Solución del ejercicio

Construir un módulo que intercambie el valor de dos variables

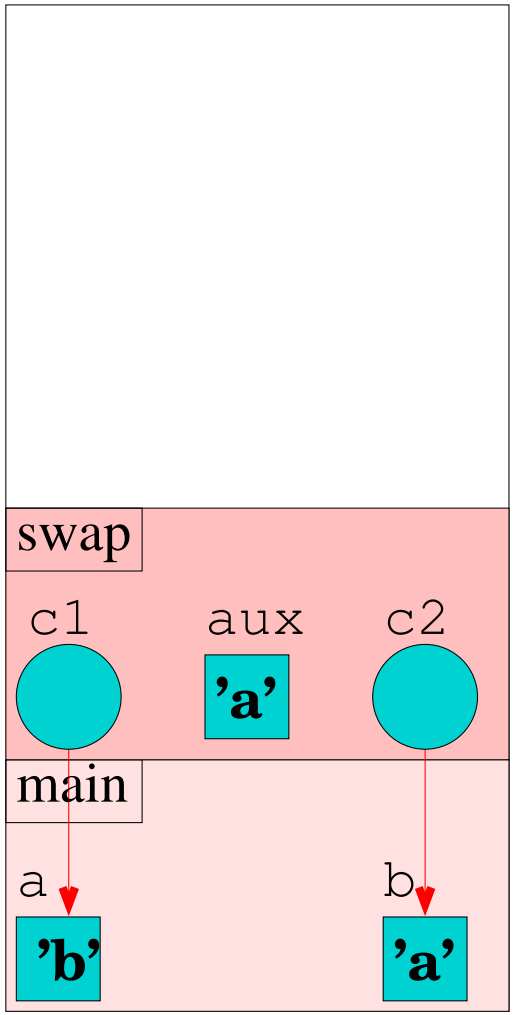
```
1  #include <iostream>
2  using namespace std;
3
4  void Swap(char &c1, char &c2){
5    char aux=c1;
6    c1=c2;
7    c2=aux;
8  }
9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Solución del ejercicio

Construir un módulo que intercambie el valor de dos variables

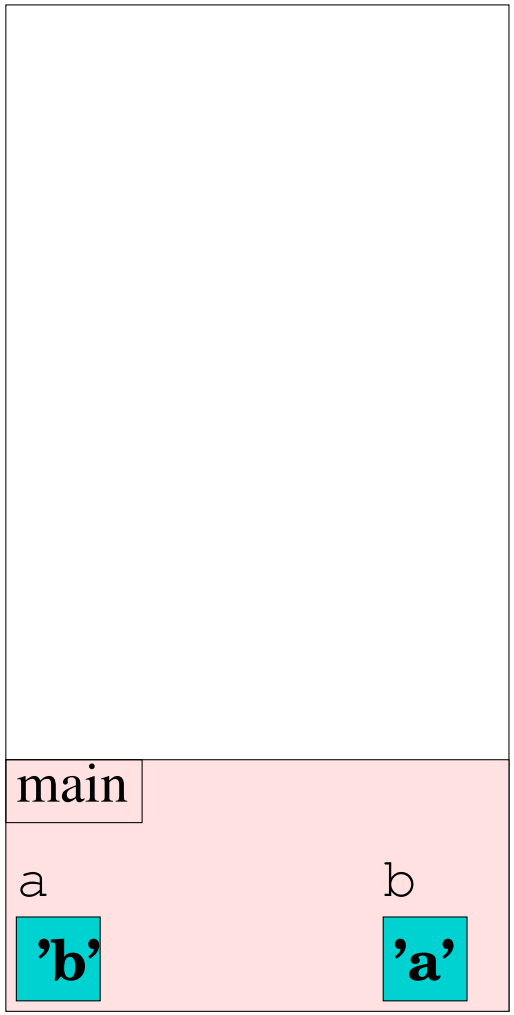
```
1  #include <iostream>
2  using namespace std;
3
4  void Swap(char &c1, char &c2){
5    char aux=c1;
6    c1=c2;
7    c2=aux;
8  }
9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Solución del ejercicio

Construir un módulo que intercambie el valor de dos variables

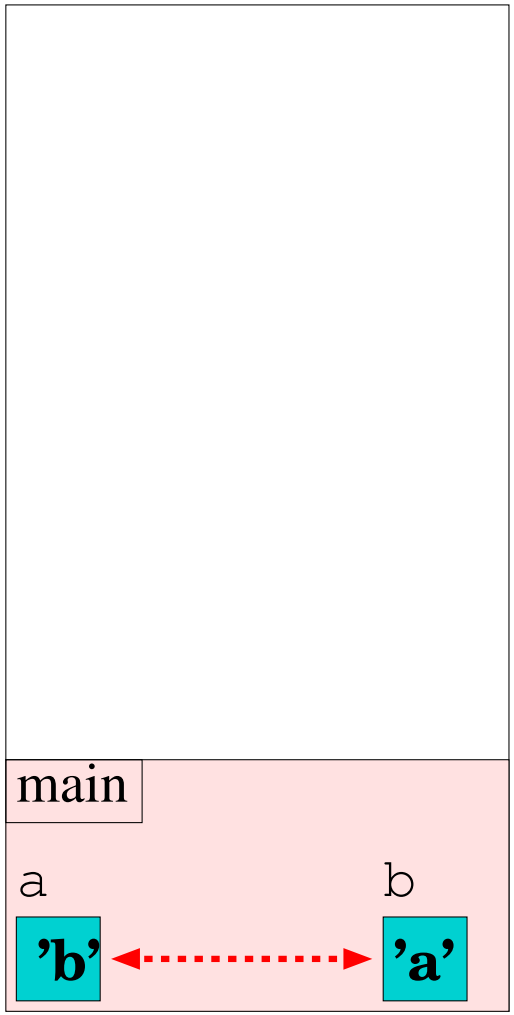
```
 1 #include <iostream>
 2 using namespace std;
 3
 4 void Swap(char &c1, char &c2){
 5   char aux=c1;
 6   c1=c2;
 7   c2=aux;
 8 }
 9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Solución del ejercicio

Construir un módulo que intercambie el valor de dos variables

```
 1 #include <iostream>
 2 using namespace std;
 3
 4 void Swap(char &c1, char &c2){
 5   char aux=c1;
 6   c1=c2;
 7   c2=aux;
 8 }
 9
10 int main(){
11   char a='a', b='b';
12
13   Swap(a,b);
14   cout << "a=" << a
15        << " y b=" << b << endl;
16 }
```



Contenido del tema

Valor/Referencia versus Entrada/Salida

Valor/Referencia versus Entrada/Salida

La identificación de la información que aporta un argumento en una función nos indica la forma en la que debe ser pasado dicho argumento.

Valor/Referencia versus Entrada/Salida

La identificación de la información que aporta un argumento en una función nos indica la forma en la que debe ser pasado dicho argumento.

- Si el argumento es usado como vehículo para obtener la solución, entonces nos encontramos ante un **parámetro de entrada**.

Valor/Referencia versus Entrada/Salida

La identificación de la información que aporta un argumento en una función nos indica la forma en la que debe ser pasado dicho argumento.

- Si el argumento es usado como vehículo para obtener la solución, entonces nos encontramos ante un **parámetro de entrada**.

PASO POR VALOR

Valor/Referencia versus Entrada/Salida

La identificación de la información que aporta un argumento en una función nos indica la forma en la que debe ser pasado dicho argumento.

- Si el argumento es usado como vehículo para obtener la solución, entonces nos encontramos ante un **parámetro de entrada**.

PASO POR VALOR

- Si el argumento es usado para almacenar la solución o parte de ella, entonces nos encontramos ante un **parámetro de salida**.

Valor/Referencia versus Entrada/Salida

La identificación de la información que aporta un argumento en una función nos indica la forma en la que debe ser pasado dicho argumento.

- Si el argumento es usado como vehículo para obtener la solución, entonces nos encontramos ante un **parámetro de entrada**.

PASO POR VALOR

- Si el argumento es usado para almacenar la solución o parte de ella, entonces nos encontramos ante un **parámetro de salida**.
- Si el argumento es tanto vehículo para obtener la solución como parte de la misma, entonces nos encontramos con un **parámetro de entrada/salida**.

Valor/Referencia versus Entrada/Salida

La identificación de la información que aporta un argumento en una función nos indica la forma en la que debe ser pasado dicho argumento.

- Si el argumento es usado como vehículo para obtener la solución, entonces nos encontramos ante un **parámetro de entrada**.

PASO POR VALOR

- Si el argumento es usado para almacenar la solución o parte de ella, entonces nos encontramos ante un **parámetro de salida**.
- Si el argumento es tanto vehículo para obtener la solución como parte de la misma, entonces nos encontramos con un **parámetro de entrada/salida**.

PASO POR REFERENCIA

Ejercicio: Escribir el prototipo de:

a) Determinar si un número es primo.

Ejercicio: Escribir el prototipo de:

a) Determinar si un número es primo.

```
bool Es_Primo (int);
```

Ejercicio: Escribir el prototipo de:

a) Determinar si un número es primo.

b) Calcular el número de primos existentes en un intervalo de valores.

Ejercicio: Escribir el prototipo de:

a) Determinar si un número es primo.

b) Calcular el número de primos existentes en un intervalo de valores.

```
int Num_Primos (int, int);
```

Ejercicio: Escribir el prototipo de:

a) Determinar si un número es primo.

b) Calcular el número de primos existentes en un intervalo de valores.

c) Calcular el máximo y el mínimo de una secuencia de valores reales introducidos por la entrada estándar (los valores se leen dentro de la función).

Ejercicio: Escribir el prototipo de:

a) Determinar si un número es primo.

b) Calcular el número de primos existentes en un intervalo de valores.

c) Calcular el máximo y el mínimo de una secuencia de valores reales introducidos por la entrada estándar (los valores se leen dentro de la función).

```
void MaxAndMix (double &, double &);
```

Ejercicio: Escribir el prototipo de:

a) Determinar si un número es primo.

b) Calcular el número de primos existentes en un intervalo de valores.

c) Calcular el máximo y el mínimo de una secuencia de valores reales introducidos por la entrada estándar (los valores se leen dentro de la función).

d) Escribir por la salida estándar un menú.

Ejercicio: Escribir el prototipo de:

a) Determinar si un número es primo.

b) Calcular el número de primos existentes en un intervalo de valores.

c) Calcular el máximo y el mínimo de una secuencia de valores reales introducidos por la entrada estándar (los valores se leen dentro de la función).

d) Escribir por la salida estándar un menú.

```
void Menu ();
```

Ejercicio: Escribir el prototipo de:

a) Determinar si un número es primo.

b) Calcular el número de primos existentes en un intervalo de valores.

c) Calcular el máximo y el mínimo de una secuencia de valores reales introducidos por la entrada estándar (los valores se leen dentro de la función).

d) Escribir por la salida estándar un menú.

e) Calcular la suma de dos números complejos. Supóngase que representamos un número complejo usando dos números reales.

Ejercicio: Escribir el prototipo de:

a) Determinar si un número es primo.

b) Calcular el número de primos existentes en un intervalo de valores.

c) Calcular el máximo y el mínimo de una secuencia de valores reales introducidos por la entrada estándar (los valores se leen dentro de la función).

d) Escribir por la salida estándar un menú.

e) Calcular la suma de dos números complejos. Supóngase que representamos un número complejo usando dos números reales.

```
void SumaComplejos (double c1r, double c1i,
double c2r, double c2i, double &c3r, double &c3i);
```

Ejercicio: Escribir el prototipo de:

a) Determinar si un número es primo.

b) Calcular el número de primos existentes en un intervalo de valores.

c) Calcular el máximo y el mínimo de una secuencia de valores reales introducidos por la entrada estándar (los valores se leen dentro de la función).

d) Escribir por la salida estándar un menú.

e) Calcular la suma de dos números complejos. Supóngase que representamos un número complejo usando dos números reales.

f) Calcular la derivada de un polinomio de grado 3.

Ejercicio: Escribir el prototipo de:

a) Determinar si un número es primo.

b) Calcular el número de primos existentes en un intervalo de valores.

c) Calcular el máximo y el mínimo de una secuencia de valores reales introducidos por la entrada estándar (los valores se leen dentro de la función).

d) Escribir por la salida estándar un menú.

e) Calcular la suma de dos números complejos. Supóngase que representamos un número complejo usando dos números reales.

f) Calcular la derivada de un polinomio de grado 3.

```
void DerivadaPol (double a, double b, double c,
double d, double &ad, double &bd, double &cd);
```

Ejercicio: Escribir el prototipo de:

a) Determinar si un número es primo.

b) Calcular el número de primos existentes en un intervalo de valores.

c) Calcular el máximo y el mínimo de una secuencia de valores reales introducidos por la entrada estándar (los valores se leen dentro de la función).

d) Escribir por la salida estándar un menú.

e) Calcular la suma de dos números complejos. Supóngase que representamos un número complejo usando dos números reales.

f) Calcular la derivada de un polinomio de grado 3.

Valor/Referencia versus Entrada/Salida

Debes tener en cuenta que

cuando el paso de parámetros es por valor, el argumento actual puede ser una expresión, una constante o una variable.

Valor/Referencia versus Entrada/Salida

Debes tener en cuenta que

cuando el paso de parámetros es por valor, el argumento actual puede ser una expresión, una constante o una variable.

Sin embargo

cuando el paso de parámetros es por referencia, el argumento actual debe ser obligatoriamente una variable.

Contenido del tema

Definición y sintaxis

Definición

Las estructuras o registros son tipos de dato compuestos heterogéneos que se definen a partir de elementos de otros tipos.

Definición y sintaxis

Definición

Las estructuras o registros son tipos de dato compuestos heterogéneos que se definen a partir de elementos de otros tipos.

Sintaxis

```
struct <NombreEstructura> {
  <tipo1> <miembro1>;
  <tipo2> <miembro2>;
  ...     ...
  <tipon> <miembron>;
};
```

La estructura tiene un nombre, `<NombreEstructura>` , que es el nombre del tipo de dato. Cada miembro (campo) tiene un nombre asociado, `<miembroX>` , que nos permitirá referenciarlo.

Ejemplos

- Definición de un punto en el plano (con coordenadas x, y).

```
struct Punto{
  double x;
  double y;
};
```

- Información sobre un alumno (NIF, nombre, curso, grupo, calificaciones parciales).

```
struct Alumno{
  string NIF;
  string nombre;
  int curso;
  char grupo;
  double notas[3];
};
```

Structs y class

- Los `struct` son herramientas muy similares a las clases. Pueden contener:
  - Especificadores de acceso
  - Métodos miembro
  - Constructores y destructores
- Diferencia:
  - Los miembros de una estructura son por defecto `public` (`private` en `class`).
- Habitualmente usaremos `struct` en lugar de `class` cuando la clase es muy simple y no necesita métodos (comportamiento): serviría como forma de agrupar los datos que contiene.
- Los `struct` suelen usarse a menudo para ayudar a definir *estructuras de datos*.

Contenido del tema

Declaración de variables

```
struct Alumno{
  string NIF;
  string nombre;
  int curso;
  char grupo;
  double notas[3];
};
```

Muy importante: La declaración sólo implica la creación de un tipo, no de variables. Una vez declarado nuestro nuevo tipo mediante structs, podremos usarlo de la misma forma que el resto de tipos.

Ejemplo

```
Alumno ahora, arrayAlumnos[10], matrizAlumnos[5][7];
```

Inicialización

Una estructura se puede inicializar en la declaración usando la misma notación que para inicializar arrays.

Ejemplo

```
1  struct Punto{
2    double x;
3    double y;
4  };
5  struct Alumno{
6    string NIF;
7    string nombre;
8    int curso;
9    char grupo;
10   double notas[3];
11 };
12 Punto origen = {0.0, 0.0};
13 Alumno estudiante = {"12345678Z", "Juan Sevilla", 1, 'B',
14                      {0.0,0.0,0.0} };
```

Operadores de acceso a valores miembros de la estructura

Los miembros de una estructura se acceden mediante:

- El operador punto (.)
  Accede a un miembro a través del nombre de la variable del tipo de la estructura.

```
Punto punto={7.5, 2.3};

cout<<punto.x<<", "<<punto.y;
```

Importante:

En general, `<variable>.<miembro>` es una variable y se comporta como cualquier variable.

- El operador flecha (->)
  El operador flecha accede a un miembro a través de la dirección de memoria de una variable del tipo de la estructura.
  Lo estudiaremos más adelante.

Asignación campo a campo

- La asignación se puede realizar de forma individual sobre cada uno de los miembros de la estructura, combinando las operaciones de acceso con el operador de asignación

  `<OperacionAcceso> = <expresion>;`

Ejemplo

```
struct Alumno{
  string NIF;
  string nombre;
  int curso;
  char grupo;
  double notas[3];
};
Alumno alumno;
alumno.NIF="26262727T";
...
alumno.notas[0]=7.2;
...
```

Asignación completa

- La asignación se puede realizar de forma completa, asignando a una estructura otra estructura del mismo tipo.

Ejemplo

```
struct Alumno{
  string NIF;
  string nombre;
  int curso;
  char grupo;
  double notas[3];
};
Alumno estudiante1 = {"12345678Z", "Juan Sevilla", 1, 'B',
                      {7.2,5.3,3.7} };
Alumno estudiante2;
estudiante2=estudiante1;
```

Operaciones de entrada y salida

En principio, se deben realizar individualmente sobre cada valor miembro de la estructura, y consiste en combinar las operaciones de entrada y salida con las operaciones de acceso a valores miembro.

```
Punto punto;
...
cout<< "Introduce coordenada x: ";
cin>>punto.x;
cout<<Introduce coordenada y: ";
cin>>punto.y;

cout<<"El punto introducido es: "<<punto.x<<
                               ", "<<punto.y<<endl;
```

Contenido del tema

Paso de estructuras a funciones

Las estructuras se comportan en C++ como si fueran tipos de datos básicos cuando se utilizan como argumento de las funciones:

- Se pueden pasar **por valor**:

```
double calculaDistancia(Punto punto1,Punto punto2);
```

- Se pueden pasar **por referencia**:

```
void leerPunto(Punto &punto);
```

- Se pueden pasar **por referencia constante**:

```
double calculaDistancia(const Punto &punto1,const Punto
&punto2);
```

- Las funciones pueden **devolver estructuras**:

```
Punto puntoMedio(const Punto &punto1,const Punto &punto2);
```

Diferencia entre paso por referencia constante y paso por valor

- Paso por valor:

```
void imprimePunto(Punto punto){
    ....
    punto.x=5.2; //Permitido: se modifica punto que es una copia
```

- Paso por referencia constante:

```
void imprimePunto(const Punto &punto){
    ....
    punto.x=5.2; //NO Permitido
```

El paso por referencia constante, pasa una referencia sobre el dato con el que se quiere trabajar (con lo que evitamos una copia que puede ocupar mucha memoria) pero lo protege para que no se pueda modificar el dato original.

Contenido del tema

Estructuras de estructuras

- Como caso particular, un valor miembro de una estructura puede ser a su vez otra estructura.

```
struct Punto{
  double x;
  double y;
};
struct Circulo{
  Punto centro;
  double radio;
};
Circulo circulo1={{5.0,4.0},10.0};
Circulo circulo2;
circulo2.centro.x=7.2;
circulo2.centro.y=5.2;
circulo2.radio=3.0;
```