UNIVERSIDAD DE MURCIA

Fundamentos de Informática
9. Funciones

Fundamentos de Informática
Grado en Ingeniería Química

2

Contenidos

- ***Funciones***
- ***Prototipos***
- ***Parámetros***
- ***Variables locales y globales***
- ***Funciones de biblioteca***

3

Funciones

Funciones

4

Empezando con C (estructura general)

- Un programa en C está formado por una o más funciones, más una cabecera al principio del archivo.
- ***main()*** es una función obligatoria en todo programa C.
- Una función es un grupo de instrucciones que realizan una o más acciones.
- Las funciones requieren los paréntesis porque pueden llevar cosas dentro de ellos.
- Las llaves **{ }** encierran el cuerpo de las funciones
- Las funciones ayudan a modularizar el programa, dividiéndolo en partes pequeñas.

```
cabecera…

funcion1( … )
{
  . . .
 }

funcion2( … )
{
  . . .
 }

. . .

main()
{
  . . .
}
```

Funciones

```
main()
{
      int a,b;
      float c;

      scanf("%d %d",&a,&b);

      c = media (a, b);

      printf("Media: %f\n",c);getchar();
}
```

• Al igual que existen ***printf*** o ***sqrt***, puedes usar tus propias funciones, como por ejemplo ***media()***.

• Desde **`main()`** se llama a la función **`media(),`** enviando como parámetros **`a`** y **`b`**; la función devuelve un valor que metemos en la variable **`c`**.

Funciones

• La función ***media()*** hay que ***definirla*** antes de ser nombrada.

• ***media()*** recibe dos ***int*** y devuelve un ***float***.

• Las variables de una función no ***"se ven"*** en las otras.

```
#include <stdio.h>

float media (int n1, int n2)
{
      float m;

      m=(n1+n2)/2.0;
      return m;
}

main() {
      int a,b; float c;
      scanf("%d %d",&a,&b);

      c=media(a,b);

      printf("Media: %f\n",c);getchar();
}
```

Funciones

Tipo de resultado

Debe ser un tipo de dato estándar.
Si es **int** no es necesario escribirlo
Puede ser **void** (nada)

Nombre de la función

Parámetros

Puede no llevar

```
float   media (int a, int b)
{
      float m;

      m = (a+b) / 2.0;
      return m;
}
```

Valor devuelto

Puede no devolver nada (), incluso no ponerse. Puede ser una expresión

Funciones

```
main() {
      int a,b; float c;

      ImprimirCabeceras(a,b);
      AjustarMargenes();
      Acabar(c);
}
```

• Las funciones pueden no devolver datos, y se usan para modularizar el programa

• Las funciones pueden llamar a otras funciones

```
float func1 (int a, int b)
{
      float m,x;

      m=(a+b)/2.0;
      x=ajusta(m)
      return x;
}
```

9

Funciones

Prototipos

10

Prototipos

Esto produce un error de compilación (en el mejor de los casos).

El compilador pasa por ***media()*** en ***main()*** antes que por la definición, y define ***media*** como una funcion ***int*** (valor por defecto).

Luego se encuentra con una definición de ***media*** como ***float***, y produce un conflicto de tipos.

También podría ocurrir que diese el error "***media() no está declarado***"

¡ Error !

```
#include <stdio.h>
main()
{
        int a,b;

        scanf("%d %d",&a,&b);
        printf("Media: %f\n", media(a,b));
        getchar();
}

float media (int a, int b)
{
        float m;
        m=(a+b)/2.0;
        return (m);
}
```

11

Funciones

Prototipos

Solución 1: definir las funciones antes de que sean llamadas.

¡Ojo! Cuando hay muchas funciones que se llaman entre sí, hay que poner cuidado en el orden de definición.

```
#include <stdio.h>

float media (int a, int b)
{
        float m;
        m=(a+b)/2.0;
        return (m);
}

main()
{
        int a,b;

        scanf("%d %d",&a,&b);
        printf("Media: %f\n", media(a,b));
        getchar();
}
```

12

Funciones

Prototipos

Solución 2: usar prototipos.
Las funciones se ***declaran*** y despues de ***definen***.

Declaracion de la función

Definición de la función

```
#include <stdio.h>

float media (int a, int b);

main()
{       int a,b;
        scanf("%d %d",&a,&b);
        printf("Media: %f\n", media(a,b));
        getchar();
}

float media (int a, int b)
{       float m;
        m=(a+b)/2.0;
        return (m);
}
```

13

Funciones

Prototipos

Primero includes y defines

Luego prototipos (declaraciones)

main()

Definición de funciones

```
#include <stdio.h>

float media (int a, int b);
int suma ( int a, int b);
void empezar (void);

main()
{       int a,b;
        scanf("%d %d",&a,&b);
        printf("Media: %f\n", media(a,b));
        getchar();
}

float media (int a, int b)
{return ((a+b)/2.0);}

int suma (int a, int b)
{return (a+b);}

…
….
```

14

Funciones

Parámetros

15

Funciones

Parámetros

Puedo usar y modificar la variable que he pasado como parámetro a una función.

Sin embargo, en la función que llama, no se modifica el valor de la variable

A esto se le llama
paso de parámetros por valor

```
funcion (int a)
{
      float m,h;
      printf("2:%d",a);
      m=a/2.0;
      a=a+10;
      h=a;
      printf("3:%d",a);
}

main()
{
      int a=10;
      printf("1:%d",a);
      funcion (a);
      printf("4:%d",a);
}
```

1: 10
2: 10
3: 20
4: 10

16

Funciones

Parámetros

Si necesito que el valor de la variable se modifique permanentemente, debo realizar el ***paso de parámetros por referencia.***

En la llamada pongo el operador ***&*** *(dirección de memoria)*

En la declaración de la función, y en la propia definición de la función, debo usar ******* *(el puntero de la variable)*

1:10 2: 10 3: 20 4: **20**

```
funcion (int *a)
{
      float m,h;
      printf("2:%d",*a);
      m=*a/2.0;
      *a=*a+10;
      h=*a;
      printf("3:%d",*a);
}

main()
{
      int a=10;
      printf("1:%d",a);
      funcion (&a);
      printf("4:%d",a);
}
```

17

Funciones

Parámetros

Si se usa el paso por referencia, por claridad, se puede declarar una variable auxiliar, y copiar el parámetro en ella a la entrada y a la salida de la función.

```
funcion (int *pa)
{
      float m,h;

      int a;
      a=*pa;

      printf("2:%d",a);
      m=a/2.0;
      a=a+10;
      h=a;
      printf("3:%d",a);

      *pa=a;
}
```

```
main()
{
      int a=10;
      printf("1:%d",a);
      funcion (&a);
      printf("4:%d",a);
}
```

1:10 2: 10 3: 20 4: 20

18

Funciones

Parámetros: Arrays *(paso de una celda)*

A una función se le pasa con normalidad ***una casilla*** de un array; paso por valor.

```
#include <stdio.h>

float media (int a, int b);

main()
{       int a[4]={1,3,4,5};

       printf("Media: %f\n", media(a[0],a[1]) );
       getchar();
}

float media (int v1, int v2)
{       float m;
       m=(v1+v2)/2.0;
       return m;
}
```

Pero si paso el array completo la cosa cambia:

¡ Los arrays se pasan siempre por referencia ! . . /. .

19

Funciones

Parámetros: Arrays *(paso por referencia)*

Hace la media de un vector

Al pasarle **a** *estoy pasando la dirección de memoria de inicio del array. Esto es paso por referencia*

Esta función además de hacer la media, pone el array a cero, para que main() lo reciba así.

```
#include <stdio.h>
#define MAX 4

float media (float v[MAX]);

main()
{       float a[MAX]={1,3,4,5};

        printf("Media: %f\n", media(a) );
}

float media (float v[MAX] )
{       int i; float m=0;

        for (i=0;i<MAX;i++)
                {m=m+v[ i ]; v[ i ]=0;}
        m=m/MAX;
        return m;
}
```

20

Funciones

Parámetros: Arrays *(mejor paso por referencia)*

Esta forma es más elegante. La función queda lista para ser utilizada en cualquier otro programa.

Está permitido no dimensionar el array en el parámetro. Tomará la dimensión que reciba.

(solo para unidimensionales)

Se pasa el tamaño como parámetro, para poder controlar el array.

```
#include <stdio.h>

float media (float v[ ], int lon);

main()
{       float a[4]={1,3,4,5};

        printf("Media: %f\n", media(a,4) );
}

float media (float v[ ], int lon )
{       int i; float m=0;

        for (i=0; i < lon; i++)
                {m=m+v[ i ]; v[ i ]=0;}
        m=m/lon;
        return m;
}
```

Parámetros: Arrays *(paso con valor de control)*

En esta otra forma un valor de control indica que es el fin del array.

Ya no hace falta pasar el tamaño como parámetro

*Itera hasta encontrar la marca **-1** de fin de array.*

Es necesario que haya un valor que solo sirva para hacer de marca.

```
#include <stdio.h>

float media (float a[ ]);

main()
{       float a[5]={1,3,4,5,-1};

        printf("Media: %f\n", media(a) );
}

float media (float a[ ])
{       int i=0; float m=0;

        while(a[ i ] != -1)
                {m=m+a[ i ]; a[ i ]=0;i++}
        m=m/lon;
        return m;
}
```


Funciones

Variables globales y locales

23

Funciones

Variables globales y locales

```
#include <stdio.h>

funcion1 (int a)
{       int i,j;
        sentencias…
}

funcion2 ()
{       int i,j,k;
        sentencias…
}

main()
{       int a,i,j;
        sentencias…
}
```

Las variables declaradas dentro de una función solo son ***visibles*** en esa función.

Además, cuando la función acaba, sus valores se pierden *(salvo que sean parámetros por referencia o que sean devueltos por la funcion)*

Son variables locales

(a la función en la que están)

Puedo poner variables I,j,k en varias funciones, y sus valores no se mezclan.

24

Funciones

Variables globales y locales

Se pueden definir **variables globales**, que son visibles en todo el programa y no pierden sus valores durante toda la ejecución.

Es recomendable que sean las mínimas

Las funciones dejan de ser exportables a otros programas si usan variables globales.

A veces son muy útiles y prácticas

```
#include <stdio.h>

int mes, h;
float cantidad;

funcion1 (int a)
{       int i,j;
        sentencias…
        mes=…
}
funcion2 ()
{       int i,j,k;
        i=….* mes
        sentencias…
}
main()
{       int a,b,j;
        mes=…
        sentencias…
}
```

25

Funciones

Variables globales y locales

¡Ojo! Si se definen variables locales con el mismo identificador que variables globales, el compilador no da errores, pero el programa no hará lo que quieres.

```
#include <stdio.h>

int i , j;

funcion1 (int a)
{   int i , k;
    sentencias…
    for ( i=0 …
}

main()
{   int a,b,j;
    i=10;
    funcion1(b);
    …
}
```

26

Funciones

Variables globales y locales

Primero includes y defines

Luego prototipos (declaraciones)
También #include "declaraciones.h"

Variables globales

main()

Definición de funciones

Este es el orden más correcto en un programa

```
#include <stdio.h>

float media (int a, int b);
int suma (int a, int b);
void empezar (void);

int mes, h;
float cantidad;

main()
{       int a,b;
        scanf("%d %d",&a,&b);
        printf("Media: %f\n", media(a,b));
        getchar();
}

float media (int a, int b)
{return ((a+b)/2.0);}

int suma (int a, int b)
{return (a+b);}

…
….
```

27

Funciones

Funciones de biblioteca (librerías)

28

Funciones

Funciones de librería

C tiene un conjunto de funciones predefinidas. Son las funciones de biblioteca o librerías

En la fase de linkado las funciones requeridas se unen a nuestro programa ejecutable.

Debemos incluir el archivo de cabecera en el que está declarada la función predefinida que necesitemos.

stdio.h	time.h
stdlib.h	math.h
stdef.h	string.h

El programador puede escribir sus propias funciones, de forma que queden compiladas para usar como funciones predefinidas.

29

¿Hay más cosas en C?

30

Sí, hay más:

Acceso a archivos completo
Variables externas, estáticas, de archivo
Tipos enumerados, uniones, campos de bits
Punteros y sus estructuras de datos
Asignación dinámica de memoria
Estructuras
Operadores de bits
Compilación separada
Generación de librerías
…

31

Funciones

- - - - - -