Tema3programacion.pdf

Julieta_G_V

Programación I

1º Grado en Ciberseguridad e Inteligencia Artificial

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Tema 3 programación

Subprogramas

Un subprograma define un bloque de código independiente, que puede ser ejecutado múltiples veces y aplicado a diferentes valores. Básicamente, se divide un problema en subproblemas y se le asigna un subprograma a cada subproblema para resolverlo de forma independiente.

La definición de un subprograma debe aparecer antes de su invocación. La cabecera de un subprograma especifica el tipo de valor devuelto (int, double...), el nombre del subprograma (menor, mayor...) y los parámetros (parámetros formales); y el cuerpo del subprograma especifica las acciones que resuelven el subproblema.

Cada subprograma define sus propias variables locales.

- **Funciones:** calculan un único valor a partir de la información de entrada y el valor devuelto viene dado por "return". El cuerpo de una función debe tener una única sentencia return y será la última sentencia del cuerpo de la función.

```
//ejemplo de subprograma en el que se calcula el menor de 2
números (función)

int menor(int x, int y)
{   int m=x;
        if(m>y){
                m=y;
        }
        return m;
}
```

- **Procedimientos:** consisten en el procesamiento general de la información, no devuelven ningún valor. La cabecera especifica que no se devuelven valores (void), el nombre del procedimiento y los parámetros; y el cuerpo no debe tener ninguna sentencia return.

  Las variables se utilizan con punteros y en "scanf" no se hace uso de "&".

```
//ejemplo de subprograma que lee un número de teclado (proc
edimiento)

void leer(int* x, int* y)
{
    printf("Introduzca dos números: ");
    scanf(" %d %d", x, y); // !!!NO SE PONE &
}

void ordenar(int* x, int* y)
{
    if(*x>*y){
            int z=*x;
            *x=*y;
            *y=z;
    }
}
```

Cuando el procedimiento es un “int main()”, el return es opcional.

```
int main()
{
    int num1, num2;
    leer(&num1);
    leer(&num2);
    ordenar(&num1, &num2);
    mostrar(num1);
    mostrar(num2);
}
```

Paso por valor y punteros

La transferencia de información entre subprogramas se implementa mediante el paso por valor y los punteros.

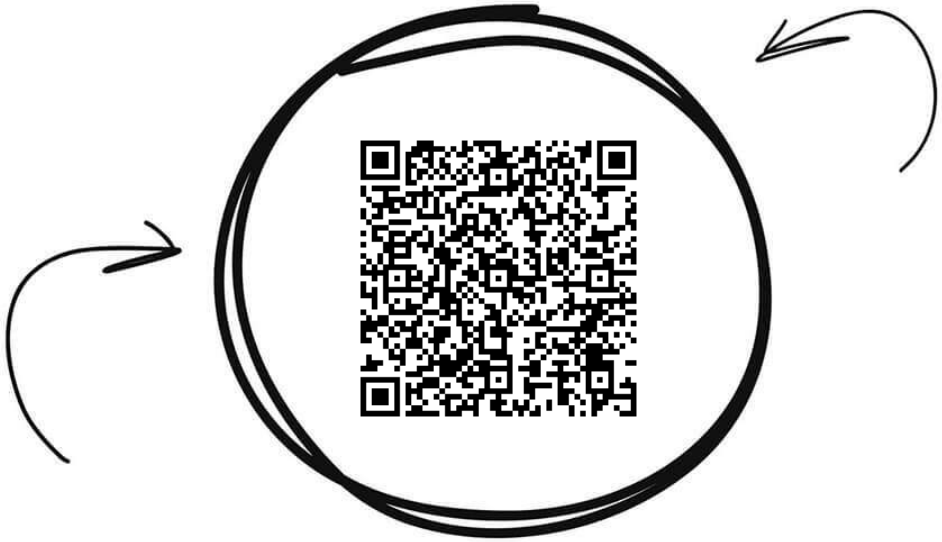
- **Paso por valor (int x):** el parámetro actúa como una nueva variable independiente. El parámetro es una variable distinta del argumento y la modificación de la variable del parámetro no afecta al argumento.

¿DÍA DE CLASES infinitas?

WRIGLEY'S Orbit

masca y fluye

Programación I



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

```
int valor_absoluto(int x)
{
        if(x<0){
                x=-x;
        }
        return x;
}
int main()
{
        int a=-3;
        int b=valor_absoluto(a);
}
```

- **Punteros (int* x):** es una variable que almacena la dirección de memoria de otra variable. Los punteros se utilizan para acceder a una variable a través de su dirección de memoria en lugar de su nombre. El parámetro se inicializa con la dirección de la variable situada como argumento. La modificación del valor de la variable del parámetro también modifica el valor de la variable del argumento.

```
void incremento(int* p) //declaración de la variable punter
o
{
        *p=*p+1;
}
int main()
{
        int a=4; //declaración de variable de tipo entero
        incremento(&a); //asignación de la dirección de mem
oria de a.
}
```

Por lo tanto, la dirección de memoria de una variable (&a) es un puntero (*p).

Definiciones importantes

Un parámetro es una variable que se utiliza dentro de la definición de una función, mientras que un argumento es el valor que se asigna a esa variable

cuando se llama a la función. Los parámetros se utilizan para permitir que una función acepte diferentes valores de entrada, lo que hace que la función sea más flexible y reutilizable.

Por ejemplo, en la siguiente función:

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

`a` y `b` son los parámetros de la función `suma`. Cuando se llama a la función `suma` con los argumentos `3` y `4`, se asignan los valores `3` y `4` a `a` y `b`, respectivamente.