ICF 122 - Fundamentos de Programación
Ingeniería Civil Informática

Unidad 4: Funciones

Facultad de Ingeniería

Universidad Andrés Bello

- Son especies de subrutinas o procedimientos.
- Se agrupa una serie de instrucciones, de tal manera de utilizarlas en otra parte del programa.
- El uso de funciones permite **modularizar** el programa, facilitando el desarrollo y la depuración (revisión) del mismo.
- Cada función, al igual que las variables, deben ser declaradas, y luego definir las instrucciones que contiene.
- Declaración:

```
tipo_retorno nombre_funcion(parametros)
{
 instrucciones;
}
```

```c
#include <stdio.h>

int potencia(int base, int n){
  int i,p;
  p = 1;
  for (i = 1;i <= n; i++)
    p = p*base;
  return p;
}

void main(){
  int i;
  for (i = 0; i < 10; i++){
    printf("%8d %8d %8d\n",i,potencia(2,i),potencia(-3,i));
  }
}
```

```c
#include <stdio.h>

float precio(float base, float impuesto){
  float calculo;
  calculo = base + (base * impuesto);
  return calculo;
}

void main(){
  float importe = 2.5;
  float tasa = 0.07;

  printf("El precio a pagar es: %.2f\n", precio(importe, tasa));
}
```

Funciones

Sin retorno

```c
#include <stdio.h>

void primera(){
  printf("Llamada a la primera funcion\n");
}

void segunda(){
  printf("Llamada a la segunda funcion\n");
}

void main(){
  printf("La primera funcion llamada, main\n");
  primera();
  segunda();
  printf("Final de la funcion main\n");
}
```

- Recordemos que las variables se clasifican según su tipo: `int`, `float`, `char`, entre otros.
- También pueden clasificarse según su **ámbito**: globales o locales.
- Una **variable local** es cualquier variable declarada dentro de una función, por tanto, su ámbito está confinado a dicha función.
- Esta situación permite que existan variables con el mismo nombre en diferentes funciones y que no mantengan ninguna relación entre sí.
- Cualquier variable declarada dentro de una función se considera como una variable automática (`auto`).

- Una variable se considera **automática**, porque, cuando se accede a la función, el espacio de memoria asignado a la variable se realiza de forma automática, y luego es liberado cuando se sale de la función.
- Una variable automática no conserva un valor entre dos llamadas sucesivas a la misma función.
- Para garantizar el contenido de las variables automáticas, se deben inicializar al entrar a la función.
- La utilización de la palabra reservada `auto` es opcional:

```
auto int contador;
```

equivale a

```
int contador;
```

```c
#include <stdio.h>

void imprimeValor(){
  int contador = 5;
  printf("El valor de contador es: %d\n", contador);
}

void main(){
  int contador = 0;
  contador++;
  printf("El valor de contador es: %d\n", contador);
  imprimeValor();
  printf("Ahora el valor de contador es: %d\n", contador);
}
```

- El ámbito de una **variable global** se extiende desde que es definida hasta el final del programa.
- La utilización de variables globales proporciona un mecanismo de intercambio de información entre funciones sin necesidad de utilizar argumentos.
- Estas variables mantienen el valor que se le ha asignado dentro de su ámbito, incluso después de finalizar las funciones que modifican dicho valor.
- Hay que tener cuidado con el uso de variables globales, ya que son útiles en algunas situaciones, en otras, puede ser muy complejo de interpretar el programa.
- Cuando se define una variable global, automáticamente se reserva un espacio de memoria.
- Si no se asigna un valor inicial a la variable global, automáticamente se le asigna valor 0.

```c
#include <stdio.h>

int variable;

void unaFuncion(){
  printf("En la funcion unaFuncion, variable es: %d\n", variable);
}

void otraFuncion(){
  variable++;
  printf("En la funcion otraFuncion, variable es: %d\n",variable);
}

void main(){
  variable = 9;
  printf("El valor de variable es: %d\n", variable);
  unaFuncion();
  otraFuncion();
  printf("Ahora el valor de variable es: %d\n", variable);
}
```

- Una **variable estática** (`static`), es una variable que puede ser tanto local como global.
- Una variable estática local, al igual que una automática, está únicamente asociada a la función en la que se declara, pero su existencia es permanente.
- Su contenido no se borra al finalizar la función, sino que mantiene su valor hasta el final del programa.

```c
#include <stdio.h>

void imprimeValor(){
  static int contador = 0;
  printf("El valor de contador es: %d\n", contador);
  contador++;
}

void main(){
  imprimeValor();
  imprimeValor();
  imprimeValor();
  imprimeValor();
}
```

Funciones

- Escriba la función `invertir_digitos(n)` que reciba como argumento un número entero `n` y entregue como resultado el número `n` con los dígitos en orden inverso:

```
Ingrese numero: 142
241
```

- Utilice la función anterior para escribir un programa que indique si el número ingresado es palíndromo o no:

```
Ingrese numero: 81418
Es palindromo
```

Funciones

- Escriba la función `perimetro(r)` que reciba como argumento el radio `r` de una circunferencia y entregue como resultado su perímetro:

```
Ingrese radio: 5
Perimetro: 31.415
```

- Escriba la función `area(r)` que reciba como argumento el radio `r` de una circunferencia y entregue como resultado su área:

```
Ingrese radio: 5
Area: 78.5375
```