UNIVERSIDAD DE LOS LLANOS

03 Funciones

Néstor Suat-Rojas. Ing. Msc (c)
nestor.suat@unillanos.edu.co

Escuela de Ingeniería
Facultad de Ciencias Básicas e Ingeniería

Introducción

- Una **función** es una secuencia de sentencias que realizan una operación y recibe un **nombre**.
- Se especifica el nombre y la **secuencia de sentencias**.
- Más adelante, se puede **llamar** a la función por ese nombre.

Introducción

```
pow(3,2);
```

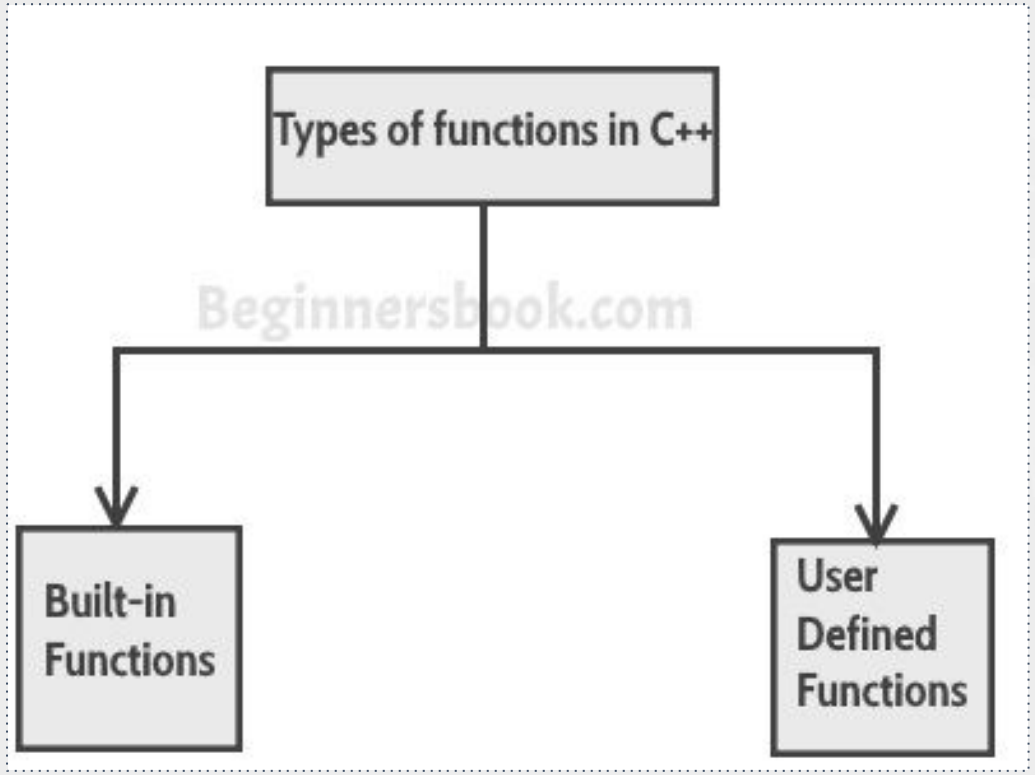
Podemos ver,

- El nombre de la función es `pow( )`
- La expresión entre paréntesis recibe el nombre de **argumento**,
- Una función **toma** (o recibe) un argumento y r**etorna** (o devuelve) un resultado.
- El resultado se llama **valor de retorno.**

UNIVERSIDAD DE LOS LLANOS

Introducción

- Tipos de funciones



UNIVERSIDAD DE LOS LLANOS

Built-in functions

C++ nos ofrece un conjunto de funciones para resolver problemas comunes y las podemos utilizar fácilmente.

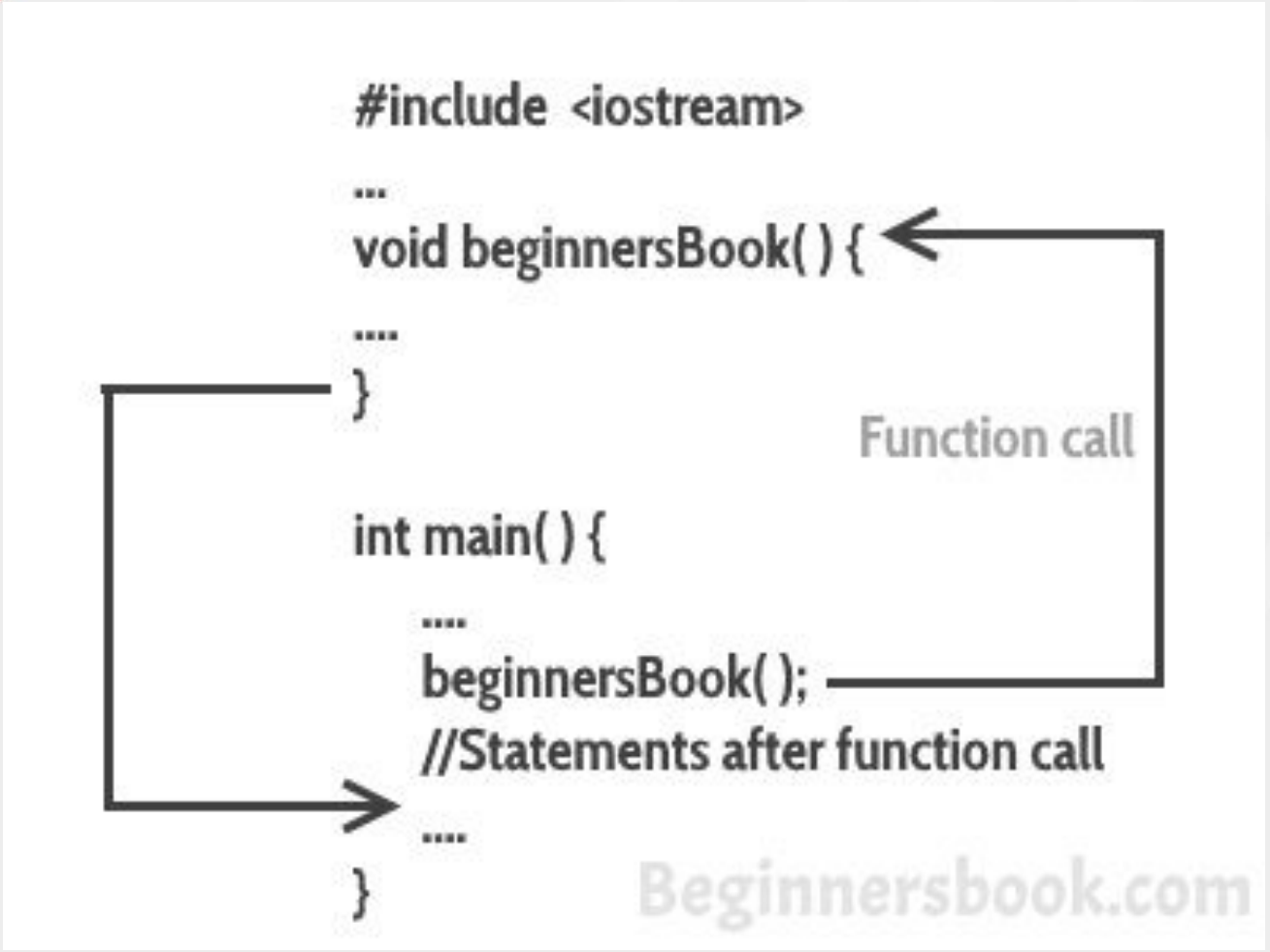
```
#include <iostream>
#include <cmath>
using namespace std;
int main(){
    /* Calling the built-in function
     * pow(x, y) which is x to the power y
     * We are directly calling this function
     */
    cout<<pow(2,5);
    return 0;
}
```

UNIVERSIDAD DE LOS LLANOS

Añadiendo funciones nuevas

- Las funciones usadas hasta ahora vienen incorporadas en C++, pero es posible **añadir nuevas funciones.**
- Una **definición de función** especifica el **nombre** de una función nueva y la **secuencia de sentencias** que se ejecutan cuando esa función es llamada.
- Una vez definida una función, se puede reutilizar una y otra vez a lo largo de todo el programa.

User-defined functions



UNIVERSIDAD DE LOS LLANOS

User-defined functions

```
void muestraEstribillo() {
    cout<<"'Soy un leñador, qué alegría.'"<<endl;
    cout<<"Duermo toda la noche y trabajo todo el día."<<endl;
}

int main(){
    muestraEstribillo();
    cout<<" ";
    muestraEstribillo();

    return 0;
}
```

UNIVERSIDAD DE LOS LLANOS

Parámetros y argumentos

Algunas de las funciones internas que hemos visto necesitan argumentos.

- `to_string()` → Recibe un argumento.
- `pow()` → Recibe dos: la base y la potencia.

Dentro de las funciones, los **argumentos** son asignados a **variables** llamadas **parámetros**.

UNIVERSIDAD DE LOS LLANOS

Parámetros y argumentos

```
//Definiendo la función sum
int sum(int a, int b) {
    int c = a+b;
    return c;
}

int main(){
    int x, y;
    cout<<"Ingrese primer numero: ";
    cin>> x;

    cout<<"Ingrese segundo numero: ";
    cin>>y;

    cout<<"La suma de los dos numeros es: "<<sum(x,y);
    return 0;
}
```

UNIVERSIDAD DE LOS LLANOS

Argumentos por defecto

En la definición de una función asignamos de una vez que parametros va recibir como argumento y su tipo. Otras veces no es obligatorio tener que pasar los parámetros, y en este caso se toman unos valores por defecto.

Argumentos por defecto

```
//Definiendo la función sum
int sum(int a=2, int b=3) {
    int c = a+b;
    return c;
}

int main(){
    int x, y;
    cout<<"Ingrese primer numero: ";
    cin>> x;

    cout<<"Ingrese segundo numero: ";
    cin>>y;

    cout<<"La suma de los dos numeros es: "<<sum();
    return 0;
}
```

UNIVERSIDAD DE LOS LLANOS

Funciones productivas y funciones estériles

- **fruitful functions:** o funciones **productivas**, son como las funciones matemáticas que producen resultados.
- **void functions:** o funciones **estériles** realizan una acción, pero no devuelven un valor.

Funciones productivas

Con las **funciones productivas** queremos hacer algo con el **resultado**.

Por ejemplo, asignar el resultado a una variable o usarlo como parte de una expresión:

```
int main(){
    int x = pow( x: 3, y: 2);
    cout<<x;
    return 0;
}
```

UNIVERSIDAD DE LOS LLANOS

Funciones estériles

Las **funciones estériles** pueden mostrar algo en la pantalla o tener cualquier otro efecto, **pero no devuelven un valo**r.

```
void muestraEstribillo() {
    cout<<"'Soy un leñador, qué alegría.'"<<endl;
    cout<<"Duermo toda la noche y trabajo todo el día."<<endl;
}

int main(){
    muestraEstribillo();
    cout<<" ";
    muestraEstribillo();

    return 0;
}
```

UNIVERSIDAD DE LOS LLANOS

Devolver varios valores de una función

En ocasiones es necesario devolver má de un valor en una función:

```cpp
#include <tuple>
#include <iostream>
#include <string>

using namespace std;
```

```cpp
tuple<int, string, double> f(){
    int i = 108 ;
    string s = "Some text";
    double d = .01 ;
    return { i,s,d };
}

int main() {
    auto[x, y, z] = f();
    cout<<x<<" "<<y<<" "<<z<<endl;
    return 0;
}
```

Funciones lambda

Una manera cómoda de definir un objeto de función anónimo (un cierre) justo en la ubicación donde se invoca o se pasa como argumento a una función.

```
int main(){

    auto f1 = [](int x, int y) { return x+y; };
    int n = [](int x, int y) { return x + y; }( x: 5, y: 4);
    cout<<f1( x: 2, y: 3)<<endl;
    cout<<n<<endl;

    return 0;
}
```

UNIVERSIDAD DE LOS LLANOS

Funciones recursivas

El proceso en el que una función se llama a sí misma se conoce como recursividad y la función correspondiente se llama función recursiva.

Funciones recursivas

```
int factorial(int num){
    if(num <= 1){
        return 1;
    }else{
        return num*factorial( num: num-1);
    }
}

int main() {

    int numero;
    cout<<"Ingrese un número entero: ";
    cin>>numero;
    cout<<"El factor del número ingresado es: "<<factorial(numero);
    return 0;
}
```

UNIVERSIDAD DE LOS LLANOS®

Funciones recursivas

Factorial function: f(n) = n*f(n-1)

Lets say we want to find out the factorial of 5 which means n =5

f(5) = 5* f(5-1) = 5* f(4)

5* 4* f(4-1) = 20* f(3)

20*3* f(3-1) = 60* f(2)

60* 2* f(2-1) = 120* f(1)

120*1* f(1-1) = 120*f(0)

120*1=120

Taller (Parte 2)

1. Escriba una función que recibe como argumento una cadena de texto y retorna la cantidad de espacios presente.

   ```
   Introduzca una cadena: Nadie existe para un proposito

       Resultado:
       Espacios: 4
   ```

2. Escriba una función que permita calcular la distancia euclidiana entre dos puntos 3-dimensionales. Como argumento recibe los dos puntos y retorna la distancia. La fórmula es la siguiente: Para dos puntos como p = (2,1,3) y q=(1,1,1) la distancia es 2.23.

En general, la distancia euclidiana entre los puntos $P = (p_1, p_2, \ldots, p_n)$ y $Q = (q_1, q_2, \ldots, q_n)$, del espacio euclídeo n-dimensional, se define como:

$$d_E(P,Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

Bibliografía

- BeginnersBook https://beginnersbook.com
- Microsoft https://docs.microsoft.com/es-es/cpp/cpp/lambda-expressions-in-cpp?view=msvc-160

UNIVERSIDAD DE LOS LLANOS

Gracias...

UNIVERSIDAD DE LOS LLANOS®