ฟังก์ชันในภาษา C

ในภาษา C มีฟังก์ชันต่างๆ มากมายที่เราสามารถใช้ได้ เพื่อทำหน้าที่ต่างๆ ที่ต้องการ ในการใช้ฟังก์ชันโดยปกติต้อง include header file ของฟังก์ชันนั้นมาด้วย ตารางข้างล่างเป็นตัวอย่างฟังก์ชันบางส่วนในภาษา C

|  |  |  |
| --- | --- | --- |
| **Function name** | **Library** | **Usage** |
| scanf | stdio.h | getting keyboard input |
| printf | stdio.h | display text to screen |
| getch | conio.h | get character input |
| sqrt | math.h | get square root of number |
| floor | math.h | floor floating point of number to int |
| strlen | stdio.h | get characters' length |

ตัวอย่างการใช้ฟังก์ชันที่มากับภาษา C

sqrt(25); //5

floor(1.34); //1

การประกาศฟังก์ชัน

เนื้อหาที่เราจะเน้นในบทนี้คือฟังก์ชันที่ผู้ใช้สร้างขึ้นเอง ในภาษา C นั้น คุณสามารถสร้างฟังก์ชันของคุณเอง เพื่อที่จะให้ทำงานตามที่ต้องการได้ แนวคิดของฟังก์ชันก็คือนำโค้ดกลับมาใช้ใหม่โดยที่ไม่ต้องเขียนโปรแกรมใหม่ทั้งหมด ก่อนที่เราจะใช้ฟังก์ชันมันจำเป็นต้องถูกประกาศก่อน ซึ่งรูปแบบการประกาศฟังก์ชันในภาษา C:

type name (parameter1, parameter2, ...)

{

statements

}

**type** เป็นประเภทของฟังก์ชันที่ต้องการประกาศ ซึ่งขึ้นกับค่าผลลัพธ์ของฟังก์ชันเป็นอะไร คุณสามารถประกาศ function type ได้เหมือนกับ primitive datatype เช่น int, float, char และอื่นๆ สำหรับฟังก์ชันที่ไม่มีการส่งค่ากลับจะมี type เป็น void

**name** เป็นชื่อของฟังก์ชันที่คุณต้องการสร้าง ในการตั้งชื่อฟังก์ชันมันมีกฏเช่นเดียวกันเหมือนกับการตั้งชื่อตัวแปร ชื่อของฟังก์ชันเป็นสิ่งที่เราจะใช้เมื่อต้องการใช้งานฟังก์ชัน

**parameters** เป็นตัวแปรที่ส่งเข้ามาในฟังก์ชัน พารามิเตอร์เป็นทางเลือกซึ่งสามารถมีหรือไม่ก็ได้

**statement** เป็นคำสั่งของโปรแกรมเพื่อที่จะให้ฟังก์ชันทำงานและได้ผลลัพธ์ที่ต้องการ

มาดูตัวอย่างของฟังก์ชันหา factorial ของตัวเลขที่เราจะสร้างขึ้นเอง

ฟังก์ชันพารามิเตอร์

ฟังก์ชันพารามิเตอร์เป็นพารามิเตอร์ที่ส่งเข้ามาในฟังก์ชัน โดยจะเป็นลำดับของตัวแปรที่ส่งเข้ามา พารามิเตอร์จะมีความสัมพันธ์กับอาร์กิวเมนต์ของฟังก์ชัน จากตัวอย่างก่อนหน้า ในฟังก์ชันหา factorial นั้นมีพารามิเตอร์คือ n มาดูตัวอย่างของฟังก์ชันพรารามิเตอร์

ฟังก์ชันอาร์กิวเมนต์

ฟังก์ชันอาร์กิวเมนต์ เป็นค่าของตัวแปรที่จะส่งเข้าไปในฟังก์ชัน ซึ่งจะต้องสอดคล้องกับฟังก์ชันพารามิเตอร์ ในตัวอย่างของคำสั่ง area(width, height) สังเกตุว่าฟังก์ชันอาร์กิวเมนต์จะมีสองตัว และประเภทของฟังก์ชันจะตรงกับลำดับของฟังก์ชันพารามิเตอร์ที่เราได้สร้างไว้

ค่า return ของฟังก์ชัน

ในการเขียนฟังก์ชันนั้น ส่วนมากจำเป็นต้องมีการส่งค่ากลับ โดยการใช้คำสั่ง return โดยในการประกาศฟังก์ชัน ประเภทของฟังก์ชันบ่งบอกถึงประเภทค่าที่ต้องการส่งกลับ จากในตัวอย่าง

float area(float, float);

สังเกตุว่า function type จะเป็น float นั่นหมายความว่าฟังก์ชันนี้จะมีค่าที่ส่งกลับเป็นเลขจำนวนจริง

Random Number Generator

rand () เป็นฟังก์ชันใน C standard library ที่ใช้ในการสร้างเลขสุ่มที่เป็นจำนวนเต็มที่อยู่ระหว่าง 0 และ RAND\_MAX ซึ่ง RAND\_MAX เป็นค่าคงที่ที่ถูกกำหนดไว้ให้แล้วใน <stdlib.h>

ในโปรแกรม เวลามีการเรียกใช้ rand () หลายๆ ครั้ง ค่าที่ฟังก์ชันนี้ส่งกลับมาจะได้ตัวเลขจำนวนเต็มที่แตกต่างกันออกไป

โดยปกติแล้ว ในการรันโปรแกรมในแต่ละครั้ง ชุดเลขสุ่มที่โปรแกรมได้รับจากฟังก์ชันจะเป็นชุดเลขสุ่มที่เหมือนกัน แต่ถ้าหากเราต้องการชุดเลขสุ่มที่แตกต่างกันออกไป เราจำเป็นต้องเรียก srand() หนึ่งครั้งเพื่อกำหนดค่า seed ให้กับ random number generator ก่อนที่เราจะเรียกใช้ rand() ต่อไป

**Global Variables**

* + ตัวแปรที่ถูกประกาศไว้นอกฟังก์ชันเป็นตัวแปร global ซึ่งสามารถถูกเข้าถึงได้จากฟังก์ชันต่างๆ รวมถึงฟังก์ชัน main ได้
  + หากตัวแปร local มีชื่อซ้ำกับตัวแปร global แล้ว การแก้ไขค่าในตัวแปร local จะไม่ได้มีผลกระทบกับค่าของตัวแปร global

**Storage Classes**

* + จากที่ผ่านมา นอกจากการที่ตัวแปรมี ชื่อ ประเภท และค่า ได้แล้ว ภาษา C ยังกำหนดให้ผู้ใช้สามารถกำหนด storage class ให้กับตัวแปร ใน 4 ลักษณะ ผ่านการใช้ keyword **auto**, **register**, **extern**, และ **static** เพื่อช่วยกำหนดช่วงชีวิตในการเก็บข้อมูลของตัวแปรใน memory
  + storage class 4 ประเภทนี้ สามารถแบ่งออกเป็น 2 storage duration ซึ่งคือ **automatic storage duration** และ **static storage duration**
  + **auto** และ **register** ใช้กำหนดตัวแปรที่เป็น **automatic storage duration** ซึ่งก็คือ ตัวแปรเหล่านี้จะถูกกำหนดขึ้นมาใน memory เมื่อโปรแกรมเข้าไปทำงานใน block ที่ถูกแปรเหล่านี้ถูกประกาศขึ้น และเมื่อโปรแกรมออกจาก block ตัวแปรเหล่านี้จะไม่อยู่ใน memory อีกต่อไป
  + ตัวแปร local เป็นตัวแปรที่อยู่ใน **automatic storage duration** โดย default ดังนั้นการใช้ keyword **auto** จึงไม่เป็นการจำเป็น ซึ่งเรามักจะไม่ค่อยเห็น **auto** ปรากฏอยู่ในโค้ดโปรแกรม
  + ส่วน **register** เป็นการแจ้งให้คอมไพเลอร์ได้ทราบว่าเราต้องการเก็บค่าตัวแปรนี้ไว้ใน register เพื่อความเร็วในการเข้าถึงตัวแปร ดังนั้นหากระบบมี register เหลือใช้ คอมไพเลอร์ก็จะจัดสรร register ให้กับโปรแกรมได้ใช้งาน