Urmia Branch
Islamic Azad University

توابع در برنامه نویسی

توابع، طراحی توابع و فراخوانی با مقدار و فراخوانی با ارجاع

استاد مربوطه : دکتر پیمان ایوبی

دانشگاه آزاد واحد ارومیه

ساختمان داده

آتیلا اصغری

موضوع

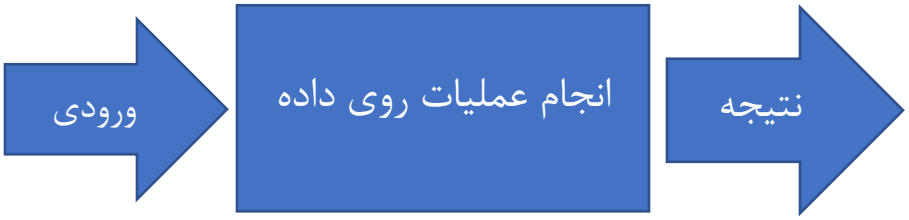
تحقیق در مورد توابع و طراحی توابع در برنامه نویسی و فراخوانی با مقدار (Call by value) و فراخوانی با ارجاع (Call by reference).

آتیلا اصغری

asghariatilla@gmail.com

توابع (Functions)

توابع بخشی از کد هستند که کار مشخصی را انجام می دهند. کارکرد تابع به این صورت می باشد که داده ای را از ورودی دریافت می کند و رو آن داده عملیات انجام می دهد و به ما نتیجه را برمی گرداند، برای درک بهتر می توان تابع را با دستگاه آبمیوه گیر مقایسه کرد که میوه (داده) را از ورودی دریافت می کند و روی آن عملیاتی انجام میدهد و آبمیوه (نتیجه) را به ما تحویل می دهد.



از توابع برای کپسوله سازی و عایت اصول (DRY = Don’t Repeat Yourself) استفاده میشود. برای مثال وقتی برنامه ای می نویسیم که در آن بخشی از کد تکرار می شود که در این صورت خوانایی برنامه کم می شود و حجم کدها زیاد میشود که باعث به وجود آمدن مشکلاتی در ادامه می شود برای حل این مشکل میتوان از توابع استفاده کرد به اینصورت که تابعی تعریف کنیم و آن کدی که قرار است تکرار شود را درون تابع قرار می دهیم و هر زمان که آن کد نیاز باشد فقط کافی است که تابع را فرا خوانی کنیم. تابع را می توان به هر تعداد که نیاز باشد فراخوانی کرد، توابع را می توان درون توابع دیگر نیز فراخوانی کرد، ولی نمی توانیم توابع تو در تو (nested functions) تعریف کنیم.

هنگامی که تابع ای فراخوانی می شود برنامه بخش فعلی کد را رها می کند و شروع به اجرای اولین خط کد درون تابع می کند، کنترل جریان این فرایند به صورت زیر می باشد :

1. برنامه به خطی از کد که تابع فراخوانی شده می رسد.
2. برنامه وارد تابع می شود و اولین خط کد تابع را اجرا می کند.
3. تمام دستورالعمل ها داخل تابع از بالا به پایین اجرا می شوند.
4. برنامه از تابع خارج می شود و به بخشی که تابع فراخوانی شده بود بر می گردد.
5. تمامی داده هایی که در تابع محسابه شده و **برگردانده** شده اند به جای تابع در کد اصلی استفاده می شوند.

دلایل استفاده از توابع

1. توابع به ما این قابلیت را می دهند که بتوانیم برنامه خود را به عنوان مجموعه ای از مراحل فرعی تصور کنیم (هر تابعی می تواند درون خود توابع دیگری داشته باشد .)
2. به ما اجازه می دهد کد را چندین بار استفاده کنیم به جای اینکه چندین بار کد تکراری بنویسیم

3. تابع به ما کمک می کند که فضای نام متغیر(Variable namespace) مرتبی و آشکاری داشته باشیم به زبانی دیگر وقتی در تابع 1 از متغیری به نام x استفاده می کنیم در تابع 2 نیز می توانیم متغیری با همین نام استفاده کنیم بدون اینکه باعث ایجاد تداخل شود.
4. توابع به ما این امکان را می دهند که بتوانیم بخش کوچکی از برنامه را به صورت ایزوله از بقیه برنامه تست کنیم.

طراحی توابع

مراحل نوشت تابع

1. هدف تابع را مشخص کنید.
2. داده هایی که قرار است از ورودی دریافت شوند را به صورت پارامتر تعریف کنید
3. داده هایی که داخل تابع برای انجام عملیات و رسیدن به هدف تابع نیاز است را تعریف کنید.
4. درمورد مجموعه مراحلی که برای دستیابی به هدف برنامه نیاز است تصمیم بگیرید (الگوریتم)

اجزای یک تابع

وقتی تابع ای را تعریف می کنیم باید ویژگی های زیر را توصیف کنیم.

1. نام : هدف تابع را توصیف میکند، بیشتر اوقات یک کلمه یا عبارت می باشد. برای مثال " user_login " یا " Login ".
2. ورودی ها : به ورودی ها پارامتر گفته می شود. داده های ضروری برای کارکردن تابع را توصیف می کند و به داده یک نام نمادین برای استفاده در تابع می دهد.
3. محاسبه : کد های این بخش در هر تابع متفاوت است.
4. خروجی : مقادیری که داخل تابع محاسبه شده و توسط متغیرهای خروجی برگردانده می شوند، بیشتر اوقات تابع یک خروجی دارد (ولی برخی اوقات ممکن است نداشته باشد یا بیشتر از یک عدد داشته باشد.)

فضای کاری تابع

هر تابع دارای فضای کار مختص خود می باشد. این بدان معنی می باشد که هر متغیری که داخل تابع تعریف شده فقط در هنگام اجرای تابع قابل استفاده هستند و سپس متغیرها از بین می روند.

داشتن فضای کار جداگانه برای هر تابع برای مهندسی نرم افزار مناسب بسیار مهم است. اگر متغیر های توابع در کل برنامه قابل دسترسی بودند آن وقت تغییر ناخواسته مقادیری که نباید تغییر یابد به راحتی اتفاق می افتاد. به علاوه به خاطر سپردن نام های متغیری که استفاده شده و پیدا کردن نام برای متغیر های جدیدی که برای اهدافی مشابه استفاده می شوند طاقت فرسا می شد.

از عوارض جانبی عدم دسترسی به متغیر های تابع خارج از آن این است که تنها راه دریافت اطلاعات خروجی تابع از طریق برگرداندن(returning) آن می باشد. در کنار این تابع فقط می تواند اطلاعاتی که با آن از طریق پارامتر پاس داده شده اند را مشاهده کند. پس تنها راه دادن اطلاعات به تابع از طریق پارامتر ها می باشد.

مثال تابع

تابع زیر عدد را گرفته و مکعب آن را محاسبه میکند:

تابع زیر به زبان C نوشته شده است.

Number cube

```c
#include <stdio.h>
#include <stdlib.h>

void noCube(int number){
    int numberCube;
    numberCube = number * number * number;

    printf("Cube of %d : %d", number, numberCube);
}

int main(){
    int userNumber;
    printf("Enter number to get cube of the number: ");
    scanf("%d", &userNumber);
    noCube(userNumber);

    return 0;
}
```

فراخوانی با مقدار و فراخوانی با ارجاع (call by value & call by reference)

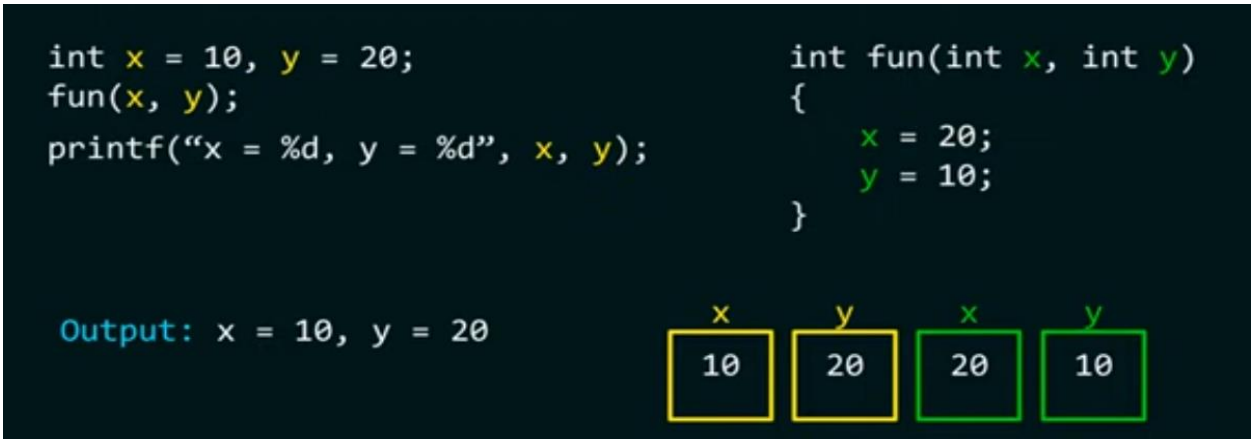
برای درک بهتر فراخوانی با مقدار و فراخوانی با ارجاع ابتدا باید با پارامتر های حقیقی و صوری آشنا باشیم.

پارامترهای حقیقی (Actual Parameters) : پارامترهایی که به تابع پاس داده می شوند.

پارامترهای صوری (Formal Parameters) : پارامترهایی که توسط تابع دریافت می شوند.

فراخوانی با مقدار (call by value)

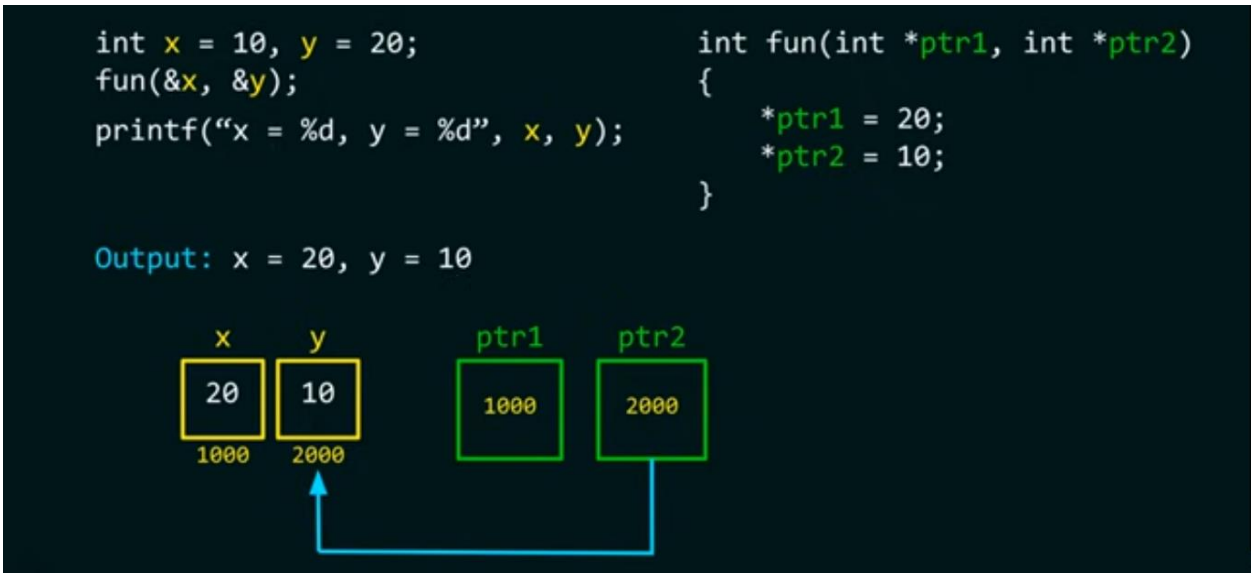
در فراخوانی با مقدار، مقادیر پارامتر حقیقی در پارامتر صوری کپی می شود و این دو پارامتر متفاوت مقادیر خود را در مکان های متفاوتی ذخیره می کنند.



همانطور که در مثال بالا دیده می شود در فراخوانی با مقدار تغییراتی که در متغیرهای داخل تابع انجام میشود اثری برروی پارامترهای حقیقی ندارد چون مکان آنها در حافظه متفاوت می باشد.

فراخوانی با ارجاع (call by reference)

در فراخوانی با ارجاع هردو پارامتر حقیقی (actual) و صوری (formal) به یک مکان حافظه اشاره می کنند. از این رو هر تغییری روی پارامترهای صوری به پارامترهای حقیقی منعکس خواهد شد.
همچنین در این حالت ما مقادیر را به تابع پاس نمی دهیم بلکه آدرس آن ها در حافظه را به تابع پاس می دهیم.



همانطور که در شکل بالا مشاهده می کنید برای پاس دادن پارامتر با ارجاع از & (امپرسند) استفاده می کنیم ولی در تابع برای ذخیره این مقدار نیاز به متغیر خاصی به نام **پوینتر(pointer)** استفاده می کنیم، که برای تعریف این نوع متغیر پیش از نام آن از کاراکتر " * " استفاده میکنیم (*numbers). به کاراکتر " * " dereference نیز گفته می شود. پوینترها متغییرهای خاصی هستند که آدرس حافظه درون خود نگهداری می کنند.

منابع :

https://www.cs.utah.edu/~germain/PPS/Topics/functions.html

https://www.youtube.com/watch?v=HEiPxjVR8CU&ab_channel=NesoAcademy

https://ray.so/ (برای نمایش کد در قالب عکس)