آشنایی با مفهوم تابع و برنامه‌نویسی ساخت‌یافته

درک رابطه‌ی پیش‌نمونه‌ی یک تابع و تعریف آن

نوشتن تابع‌ها در C++ و فراخوانی آن‌ها

تابع‌ها

عبداله جلیلیان

گروه آمار

دانشگاه رازی

چرا تابع‌ها؟

- برنامه‌هایی که برای حل بسیاری از مسائل واقعی به کار گرفته می‌شوند می‌توانند بسیار طولانی و پیچیده باشند.
- اگر بتوان برنامه را به اجزای کوچکتری نقسیم کرد که در نهایت مسئله‌ی مورد نظر را حل کنند، نوشتن برنامه و خطایابی آن بسیار ساده‌تر می‌شود.
- تابع‌ها به شما این امکان را می‌دهند تا برنامه را به بخش‌های مستقلی تقسیم کرد.
- پس از ایجاد دستورهای بدنه‌ی تابع، می‌توان تابع را به دفعات در مکان‌های مختلف برنامه فراخوانی کرد.
- در هر فراخوانی، تابع یا مقداری را به عنوان نتیجه برمی‌گرداند و یا پس از پایان کارش، کنترل اجرای برنامه را به مکانی که فراخوانده شده است بر می‌گرداند.
- از تابعی که از پیش تهیه شده است می‌توان برای برنامه‌های بعدی استفاده کرد.

- تا بدین جا با استفاده از چند تابع از پیش نوشته شده در کتابخانه‌ی استاندارد C++ آشنا شده‌ایم.
- حال می‌بینیم چطور می‌توان تابع‌های جدیدی نوشت و چطور آن‌ها را فراخوانی کرد.

ساختار تعریف یک تابع

- ساختار ایجاد یک تابع
- باید نوع برگشتی (خروجی) یک تابع را هنگام نعریف یک تابع مشخص کرد.
- پس از نوع برگشتی، باید نام تابع (یک شناسه‌ی معتبر) نوشته شود و پس از آن درون یک جفت پرانتز باز و بسته نوع و شناسه‌ی پارامترهای ورودی تابع باید مشخص شود.
- بدنه‌ی تابع درون یک جفت آکولاد نوشته می‌شود.

```
نوع خروجی تابع نام تابع(آرگومان‌های ورودی)
{
        دستورهای بدنه‌ی تابع
}
```

- مثال: تابع main

```
int main()
{
}
```

مثال ۱۸

```
#include <iostream>
using namespace std;

void printfun();

int main()
{
    printfun();
    return 0;
}

void printfun()
{
     cout << "Welcome to C++!" << endl;
}
```

شرح مثال ۱۸

- دستور

```
void printfun();
```

- پیش‌نمونه‌ی تابع printfun است.
- پیش‌نمونه‌ی یک تابع، اعلانی از تابع است که به کامپایلر نام تابع، نوع برگشتی آن و نوع پارامترهای آن را می‌گوید.
- پیش‌نمونه‌ی تابع printfun به کامپایلر اعلان می‌کند که تابع printfun نوع برگشتی ندارد (خروجی را برنمی‌گرداند) و به هیچ پارامتری برای انجام کارش نیاز ندارد.
- بخش

```
void printfun()
{
      cout << "Welcome to C++!" << endl;
}
```

- تابع printfun را تعریف می‌کند.
- خط اول تعریف هر تابع باید با پیش‌نمونه‌ی تابع سازگار باشد.

شرح مثال ۱۸

- در خط اول تعریف تابع، مانند پیش‌نمونه‌ی تابع، نوع برگشتی، نام تابع و لیست پارامترهای تابع نوشته می‌شود.
- پس از پیش‌نمونه باید از سمی‌کالن استفاده کرد، در صورتی که گذاشتن سمی‌کالن پس از خط اول تعریف تابع خطای دستوری است و باعث می‌شود بدنه‌ی بخش جداگانه‌ای در نظر گرفته شود.
- بدنه‌ی تابع `printfun` تنها شامل یک دستور چاپ در خروجی است. به همین خاطر نوع برگشتی `void` (بدون نوع برگشتی) تعیین شده است.
- تابع به هیچ پارامتری برای اجرای وظیفه‌اش نیاز ندارد، به همین خاطر لیست پارامترهای تابع خالی است.
- دستور

```
printfun();
```

در تابع main یک فراخوانی تابع است. فراخوانی تابع می‌تواند به تعداد دلخواه صورت گیرد.

مثال ۱۹

```
#include <iostream>
using namespace std;

void factorial(int);

int main()
{
    factorial(3);
    factorial(5);
    factorial(7);
    factorial(10);

    return 0;
}
```

مثال ۱۹

```
void factorial(int n)
{
    int res = 1;
    for (int i = 1; i <= n; i++)
       res *= i;
    cout << "factorial(" << n << ") = "
         << res << endl;
}
```

- در این مثال، تابع factorial نوع برگشتی void دارد (هیچ مقداری را برنمی‌گرداند). مقدار فاکتور آرگومان ورودی پس از محاسبه در بدنه‌ی تابع چاپ می‌شود.
- این تابع یک پارامتر از نوع int دارد که با دریافت آن، مقدار فاکتوریل آن را حساب و در خروجی چاپ می‌کند.

مثال ۲۰

- برنامه‌ای بنویسید که با دریافت سه عدد اعشاری، میانگین آن‌ها را محاسبه و در خروجی چاپ کند. از یک تابع برای این منظور استفاده کنید.

```
#include <iostream>
using namespace std;

void mean(double, double, double);

int main()
{
    double d1, d2, d3;
    cout << "Enter three decimal numbers: ";
    cin >> d1 >> d2 >> d3;

    mean(d1, d2, d3);
    return 0;
}
```

مثال ۲۰

```
void mean(double x, double y, double z)
{
    cout << "The mean is: "
         << (x + y + z)/3  << endl;
}
```

- در اینجا نیز تابع پس از محاسبه‌ی میانگین سه عدد، این مقدار را چاپ می‌کند و به مکان فراخوانی (در بدنه‌ی تابع `main`) برنمی‌گرداند. به همین خاطر نوع برگشتی تابع `void` است.

متغیرهای محلی و حوزه‌ی تعریف

- حوزه‌ی تعریف یک متغیر قسمت‌هایی از برنامه است که آن متغیر برای مقداردهی و شرکت در محاسبات در دسترس است.
- اگر حوزه‌ی تعریف متغیری تنها در یک بخش از برنامه (بدنه‌ی یک تابع، یک کلاس و یا یک حلقه) باشد، آن متغیر یک متغیر محلی گفته می‌شود.
- اگر حوزه‌ی تعریف متغیری در همه‌ی تابع‌ها (و کلاس‌های) برنامه باشد، یک متغیر کلی نامیده می‌شود.
- متغیرهایی که در بدنه‌ی یک تابع اعلان می‌شوند، متغیرهای محلی آن تابع می‌باشند و حوزه‌ی تعریف آن‌ها در همان تابع از بلافاصله بعد از اعلان تا پایان بدنه‌ی تابع است.
- متغیرهایی که درون یک ساختار کنترلی (حلقه، دستورهای یک یا چند انتخابی) اعلان می‌شوند متغیرهای محلی آن ساختار کنترلی هستند و حوزه‌ی تعریف آن‌ها تنها درون همان ساختار کنترلی می‌باشد.
- متغیرهایی که در ابتدای برنامه و خارج از تابع `main` اعلان می‌شوند متغیرهای کلی برنامه هستند و حوزه‌ی تعریف آن‌ها از بلافاصله بعد از اعلان تا پایان برنامه است و در همه‌ی تابع‌ها (و کلاس‌ها) قابل دسترسی هستند.

مثال ۲۱ (متغیر کلی)

```
#include <iostream>
using namespace std;

void mean(double, double, double);

int n = 3;

int main()
{
    double d1, d2, d3;
    cout << "Enter " << n
         << " decimal numbers:";
    cin >> d1 >> d2 >> d3;
    mean(d1, d2, d3);
    return 0;
}
```

مثال ۲۱ (متغیر کلی)

```
void mean(double x, double y, double z)
{
    double w;
     w = (x + y + z) / n;
      cout << "The mean is: " << w << endl;
}
```

- متغیر n در این برنامه یک متغیر کلی است و حوزه‌ی تعریف آن شامل هر دو تابع mian و mean است.
- متغیرهای d1، d2 و d3 متغیرهای محلی تابع main هستند. بنابراین در تابع mean در دسترس نیستند.
- متغیر w نیز متغیر محلی تابع mean است و در تابع main در دسترس نیست.