به نام یزدان پاک

موضوع : توابع و متغیر ها در C++

گردآوری : فرهاد بیگی راد

استاد راهنما : استاد علیاری

in

توابع (Functions) در C++

هربرنامه از بخشهای کوچکتری به نام تابع تشکیل میشود.هر تابع ، کار واحدی انجام میدهد .
هر تابع باید یک نام اختصاصی داشته باشد که بتوان آن را فراخوانی کرد.
چگونگی تعریف یک تابع را باهم مرور میکنیم :

```
type name(parameter1, parameter2, ...)
{
     statements
}
```

type	نوع داده ای هر تابع عبارت است از نوعی که بازگردانده خواهد شد.
name	نام تابع را تعیین میکند .
parameter	پارامتر های ورودی تابع را مشخص میکند
statement	دستورات اجرایی تابع است.

۱. توابع هم میتوانند پارامتر دریافت کنند و هم پارامتر ورودی نداشته باشند.
۲. توابع هم میتوانند نوع خاصی از داده هارا بازگردانند و هم میتوانند فقط عملی خاص را انجام داده و عبارت بازگشتی نداشته باشند.

ما تنها دو حالت را باهم مرور میکنیم که اولی داده بازگشتی دارد و دومی بدون عبارت بازگشتی ست.

مثال یکم :

```
1  // function example
2  #include <iostream>
3  using namespace std;
4
5  int addition (int a, int b)
6  {
7    int r;
8    r=a+b;
9    return r;
10 }
11
12 int main ()
13 {
14   int z;
15   z = addition (5,3);
16   cout << "The result is " << z;
17 }
```

```
The result is 8
```

توضیح برنامه :

- این برنامه به دو بخش main و addition تقسیم شده است.البته هنوز هم طبق قوانین برنامه با اجرای main اجرا میگردد و تابع addition به طور غیر مستقیم و از طریق فراخوانی در تابع main به اجرا در می آید.
- در زمان فراخوانی هر تابع و در صورتی که پارامتر ورودی داشته باشد باید به همان تعداد ، پارامتر به آن ارسال کنیم.
- نوع داده ای پارامتر ارسالی باید با نوع داده ای پارامتر های تابع یکسان باشد .
- ترتیب در ارسال پارامتر ها مهم است.
- متغیر های a و b به عنوان متغیر محلی در تابع شناخته میشود.
- در خطوط بعدی که عمل جمع کردن و انتساب به متغیر r انجام میگردد.
- همانطور که میبینید نوع داده ای باز گردانده شده با نوع خود تابع یکی ست.
- در خطوط بعدی مقدار بازگردانده شده به متغیر z انتساب میابد.و در آخر نیز در خروجی چاپ میشود.

```
int addition (int a, int b)
                  ↑     ↑
z = addition (  5   ,   3  );
```

```
int addition (int a, int b)
  ↓
z = addition (  5   ,   3  );
```

مثال دوم :

نوع داده ای که در توابع بدون مقدار بازگشتی ، به جای type ذکر میگردد ، عبارت void است.
void به معنای پوچ و خالی ست.

دراینجا این عبارت نشانگر این است که این تابع مقداری به عنوان خروجی یا بازگرداننده ندارد..

```
1  // void function example
2  #include <iostream>
3  using namespace std;
4
5  void printmessage ()
6  {
7    cout << "I'm a function!";
8  }
9
10 int main ()
11 {
12   printmessage ();
13 }
```

```
I'm a function!
```

لازم به ذکر است void را میتوان به عنوان پارامتر نیز معرفی کرد. در زبان C++ نوشتن این عبارت اختیاری ست ولی در زبان C الزامی ست.

```
void printmessage(void)
{
      cout << "I'm a function!";
}
```

❖ اعلام تابع

در C++ ب طور کلی هر متغیر ابتدا باید تعریف گردد و بعد مورد استفاده قرار گیرد.به عنوان مثال ابتدا متغیر x تعریف میشود و بعد مورد استفاده قرار میگیرد.

```
int x;
```

همین مطلب در مورد توابع نیز صدق میکند.توابع را قبل از معرفی نمیتوان فراخوانی کرد.

- اگر تابعی بعد از main تعریف شود کامپایل نمیشود مگر در حالتی خاص.

همیشه کامپایلر C++ در ابتدا توابع را در حد جزئیات کوتاه (نوع بازگشتی و نوع و تعداد پارامتر) بررسی میکند.و در زمان اجرا به بررسی کامل میپردازد.

حال ما برای اینکه توابعمان را بعد از main تعریف کنیم ، جزئیات تابع مورد نظرمان را قبل از main ذکر میکنیم و به نوعی فقط اعلام میکنیم که چینین تابعی با نام مشخص و ویژگی های مشخص وجود دارد.

چگونگی اعلام یک تابع

```
type name(parameters…);
```

- در این اعلام حتی نام پارامتر ها نیز اهمیتی ندارد و ذکر آن اختیاری ست.
- نکته مهم و حائز اهمیت این است که بعد از دستور اعلام اولیه ی تابع ، باید علامت سمی کولن(;) گذاشته شود.

مثالی را بررسی میکنیم :

```
1   // declaring functions prototypes
2   #include <iostream>
3   using namespace std;
4
5   void odd (int x);
6   void even (int);
7
8   int main()
9   {
10    int i;
11    do {
12      cout << "Please, enter number (0 to
13  exit): ";
14      cin >> i;
15      odd (i);
16    } while (i!=0);
17    return 0;
18  }
19
20  void odd (int x)
21  {
22    if ((x%2)!=0) cout << "It is odd.\n";
23    else even (x);
24  }
25
26  void even (int x)
27  {
28    if ((x%2)==0) cout << "It is even.\n";
29    else odd (x);
    }
```

```
Please, enter number (0 to
exit): 9
It is odd.
Please, enter number (0 to
exit): 6
It is even.
Please, enter number (0 to
exit): 1030
It is even.
Please, enter number (0 to
exit): 0
It is even.
```

در محل مشخص شده میبینید که در اعلام یکی از توابع نام پارامترها وارد نشده.اما هر دو تابع به درستی فراخوانی میشوند.پس نتیجه میشود که کامپایلر در ابتدا توجهی به نام پارامتر ها ندارد و وارد کردن آنها اختیاری ست.

توضیحات:

- همانطور که میبینید تابع odd و even بعد از main تعریف شده اند.اما به علت اعلام اولیه ، و شناسایی کامپایلر ، در main استفاده شده اند.
- روند کار بدین صورت است که حلقه do/while تازمانیکه کلید صفر فشرده نشده ، دستورات را اجرا میکند.تابع <<cin که متغیر i را مقدار دهی میکند و در مرحله بعداین مقدار به تابع odd که فرد بودن عدد را بررسی میکند ارسال میشود که درصورت فرد بودن پیغام مربوطه چاپ میگردد و در غیر اینصورت نیز تابع even را فراخوانی میکنید.

❖ مقادیر پیشفرض در پارامترها

در C++ میتوان ورود برخی پارامتر هارا اختیاری اعلام نمود.به عنوان مثال تابعی که دارای سه پارامتر است را با وارد کردن دوپارامتر فراخوانی کرد.

مثالی با هم ببینیم:

```
1   // default values in functions
2   #include <iostream>
3   using namespace std;
4
5   int divide (int a, int b=2)
6   {
7     int r;
8     r=a/b;
9     return (r);
10  }
11
12  int main ()
13  {
14    cout << divide (12) << '\n';
15    cout << divide (20,4) << '\n';
16    return 0;
17  }
```

```
6
5
```

همانطور که میبینید تابع divide دوبار در main فراخوانی شده.در اولی :

```
divide (12)
```

در این تابع تنها یک مقدار ارسال میگردد.در صورتی که divide دو پارامتر دارد.اما پارامتر دومی دارای مقدار پیشفرض (b=2) است.پس نتیجه برابر ۶ میشود.

اما در فراخوانی دوم هردو مقدار وارد شده که در این صورت 20/4 برابر ۵ میشود.

❖ توابع بازگشت (Recursivity) :

یک خاصیت برای توابع است که توابع را قادر میسازد که خود را فراخوانی کنند.

این ویژگی در بعضی عملیات مانند مرتب کردن و یا محاسبه فاکتوریل کاربرد دارد.

```
1   // factorial calculator
2   #include <iostream>
3   using namespace std;
4
5   long factorial (long a)
6   {
7     if (a > 1)
8      return (a * factorial (a-1));
9     else
10     return 1;
11  }
12
13  int main ()
14  {
15    long number = 9;
16    cout << number << "! = " << factorial (number);
17    return 0;
18  }
```

```
9! = 362880
```

میبینید که تابع factorial مانند یک حلقه تکرار شونده while عمل کرده و تا زمانیکه شرط a>1 برقرار است ، تابع factorial اجرا میگردد.

انواع متغیر در C++ از نظر سطح دسترسی

همانطور که قبلا در تعریف متغیر داشتیم به خانه هایی از حافظه کامپیوتری که میتونند اطلاعاتی را درخود نگهداری کنند متغیر گفته میشود.

متغیر دارای ویژگی هایی ست :

- نام : نامگذاری متغیر در هر زبان برنامه نویسی قوانین خاص خود را دارد.

بر فرض مثال در C++ نام متغیر نباید با اعداد شروع شود و در نامگذاری محدودیتی در طول نام وجود ندارد.همچنین C++ به بزرگی و کوچکی حروف لاتین حساس است(CaseSensitve) و نباید از کلمات کلیدی باشد.

- اندازه : هر متغیر بسته به نوعی که دارد اندازه معینی از حافظه را اشغال میکند.به جدول توجه نمایید :

Name	Description	Size*	Range*
char	کاراکتر یا اعداد صحیح کوتاه	1byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	اعداد صحیح کوتاه	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	اعداد صحیح	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	درست یا غلط	1byte	true or false
float	اعداد اعشاری کوتاه.	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	اعداد اعشاری بلند	8bytes	+/- 1.7e +/- 308 (~15 digits)

توابع از نظر سطح دسترسی به دو دسته تقسیم میشوند:

۱. متغیر های محلی Local
۲. متغیر های سراسری Global

این دو نوع از نظر ماهیتی هیچ تفاوتی با هم ندارند به جز در سطح دسترسی :

متغیر های محلی (Local) :

۱. این داده ها در داخل یک تابع پنهان هستند.
۲. تنها در همان تابع قابل دسترسی و تغییرند.

متغیر های سرسری (Global):

۱. در همه توابع قابل دسترسی هستند.
۲. در خارج از تابع خاص تعریف میشنود.

```
1   #include<iostream>
2   #include<conio.h>
3
4   using namespace std;
5   // Global Variable
6   int iGlobal = 5;
7   int Rusult();
8   int main()
9   {
10          //Loval Variable
11          int iMain = 10;
12          cout << "Global :" << iGlobal;
13          cout << endl << "Local :" << iMain;;
14          _getch();
15  }
16  int Result()
17  {
18          //      Local Variables]
19          int iResult = 2;
20          return iResult;
21  }
22
23
```

```
Global :5
Local :10
```

علوم رایانه هیچگاه شخصی را تبدیل به یک برنامه نویس خوب نمی‌کنند . همانطور که مطالعه در مورد رنگ‌ها و قلم‌ها شما را تبدیل به یک نقاش خوب نمی‌کند.

(Eric Raymond)

اردیبهشت ۹۴