توابع ۲

مبانی برنامه‌نویسی

(۱۳۹۹-۱۳-۱۱)

جلسه‌ی نوزدهم

دانشگاه شهید بهشتی

پاییز ۱۳۹۲

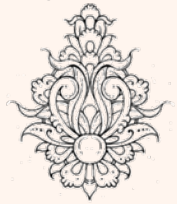
دانشکده‌ی مهندسی برق و کامپیوتر

احمد محمودی ازناوه

http://faculties.sbu.ac.ir/~a_mahmoudi/

فهرست مطالب

- مروری بر جلسه‌ی پیش
- توابع تودرتو
- آرگومان و پارامتر
- شناسه‌ی تابع



چگونگی کارکرد

- ابتدا تابع main اجرا می‌شود.
- دو مقدار ۵ و ۳ در متغیرهای a و b کپی شده و به تابع ارسال می‌گردند.
- مقدار مورد نظر در تابع محاسبه شده و مقدار ۸ باز می‌گردد.
- متغیرهای a و b تنها در محدوده‌ی تابع معتبر هستند.

```
int addition (int a, int b)

z = addition (  5  ,  3  );
```

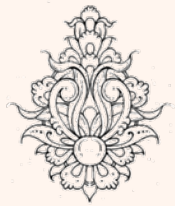
مثال

```
// function example
#include <iostream>
using namespace std;

int subtraction (int a, int b)
{
  int r;
  r=a-b;
  return (r);
}

int main ()
{
  int x=5, y=3, z;
  z = subtraction (7,2);
  cout << "The first result is " << z << '\n';
  cout << "The second result is " << subtraction (7,2) << '\n';
  cout << "The third result is " << subtraction (x,y) << '\n';
  z= 4 + subtraction (x,y);
  cout << "The fourth result is " << z << '\n';
  return 0;
}
```

```
The first result is 5
The second result is 5
The third result is 2
The fourth result is 6
```



فراخوانی تودرتو

```
void deeper()
 {
    cout << "I am now inside the function deeper.\n";
 }
void deep()
 {
     cout << "I am now inside the function deep.\n";
     deeper(); // Call function deeper
     cout << "Now I am back in deep.\n";
 }

 int main()
 {
     cout << "I am starting in function main.\n"
     deep(); // Call function deep
     cout << "Back in function main again.\n";
     return 0;
 }
```

```
I am starting in function main.
I am now inside the function deep.
I am now inside the function deeper.
Now I am back in deep.
Back in function main again.
```

```
void deeper() {
   cout<< "I am now inside the function deeper.\n";
}
```

```
void deep() {
   cout<< "I am  now inside the function deep.\n";
   deeoer();
   cout<<"Now I am back in deep.\n"
}
```

```
int main() {
   cout<<"I am starting in function main.\n";
   deep();
   cout<<"Back in function main again.\n";
   return 0;
}
```

مثال

```
int max(int x,int y)
{ // returns larger of the two given integers:
if (x < y) return y;
else return x;
}
int main()
{ // tests the max() function:
int m,n;
    do
    {
       cout<<"Enter 2 number (0 for quit):";
       cin >> m >> n;
       cout << "\tmax(" << m << "," << n << ") = " << max(m,n) << endl;
    }
    while (m != 0);
}
```

return برای تابع مانند break است برای حلقه یعنی هر زمان به return برسد از تابع بیرون می‌پرد. گر چه در بیشتر مواقع در پایان تابع است اما در هرجای دیگر نیز می‌تواند بیاید.

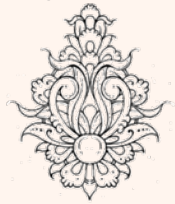
```
Enter 2 number (0 for quit):6 7
        max(6,7) = 7
Enter 2 number (0 for quit):89 4
        max(89,4) = 89
Enter 2 number (0 for quit):0 4
        max(0,4) = 4
```

امضای تابع!

- قبل از این‌که کامپایلر تابعی را فراخوانی کند، نیاز دارد چیزهایی راجع به تابع بداند:
  - نام تابع
  - تعداد و نوع آرگومان‌های ورودی
  - نوع آرگومان خروجی
- برای این منظور دو رویکرد وجود دارد:
  - اعلان تابع (declaration یا function prototype)

Function prototypes are also known as *function declarations*.

  - قرار دادن بدنه‌ی تابع یا (function definition)



اعلان‌ها و تعاریف

- **راه‌های تعریف یک تابع:**
  - توابع قبل از تابع main() به طور کامل با بدنه مربوطه آورده شوند.
  - راه دیگر، این گونه است که ابتدا تابع ***اعلان*** شود، سپس متن برنامه‌ی اصلی main() بیاید، پس از آن بدنه‌ی کامل تابع قرار بگیرد. (معمولا از این شیوه استفاده می‌شود)

اعلان تابع یا همان Prototype یک تابع، برای شناساندن تابع است که تنها شامل نام تابع و نوع آرگومان‌های ارسالی و نوع بازگشتی به همراه « ; » است.

مثال

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

int sub (int a,int b)
{
  int z;
  z=a-b;
  return z;
}

int main ()
{
  int x=10, y=3;
  cout << x<<"-"<<y<<"="<<sub(x,y)<< endl;
  return 0;
}
```

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

int sub (int ,int );
int main ()
{
  int x=10, y=3;
  cout << x<<"-"<<y<<"="<<sub(x,y)<< endl;
  return 0;
}
int sub (int a,int b)
{
  int z;
  z=a-b;
  return z;
}
```

آرگومان‌ها و پارامترها

- مقادیری که به تابع ارسال می‌شوند، **آرگومان** و متغیرهایی که مقادیر مذکور را دریافت می‌نمایند **پارامتر** نامیده می‌شوند.

```
#include <iostream>
using namespace std;
 void displayValue(int);
int main()
{
cout << "I am passing 5 to displayValue.\n";
displayValue(5);
cout << "Now I am back in main.\n";
return 0;
}
void displayValue(int num)
{
cout << "The value is " << num << endl;
}
```

```
I am passing 5 to displayValue.
The value is 5
Now I am back in main.
```

آرگومان ارسالی

پارامتر تابع

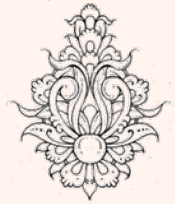
function prototype

```
void displayValue(int);
void displayValue(int num);
```

هر دو نمونه برای تعریف **prototype** صحیح است هر چند تنها مشخص کردن نوع کافیست و در مورد دوم کامپایلر **num** را در نظر نمی‌گیرد.

```
displayValue(5);                    // function call

void displayValue(int num)          // function header
{
     cout << "The value is " << num << endl;
}
```

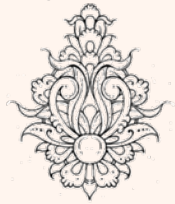


مثال

- تابعی بنویسید که دو عدد a و p را از ورودی خوانده و a را به توان p برساند.

```
#include <iostream>
using namespace std;
int power(int ,int );
int main()
{ int a,p;
cout<<"enter two number:";
cin>> a >> p;
cout <<"a^p=" <<power(a,p)<<endl;
    }
int power(int k,int l)
    {
    int i,n=1;
    for(i=1;i<=l;i++)
       n=n*k;
    return n;
    }
```

```
enter two number:2 5
a^p=32
```



مثال

- تابعی بنویسید که عددی را از ورودی گرفته فاکتوریل عدد را محاسبه نماید.

```
long fact(int n)
{ // returns n! = n*(n-1)*(n-2)*...*(2)(1)
if (n < 0) return 0;
int f = 1;
while (n > 1)
      f *= n--;
return f;
}
```

```
0 1 1 2 6 24 120
```

```
int main()
{ // tests the factorial() function:
for (int i=-1; i < 6; i++)
     cout << " " << fact(i);
cout << endl;
}
```

مثال

- تابعی بنویسید که فرمول جایگشت را پیاده سازی نماید:

$$P(n,k) = \frac{n!}{(n-k)!}$$

$$P(5,2) = \frac{5!}{(5-2)!} = \frac{5!}{3!} = \frac{120}{6} = 20$$

```
long fact(int n)
{ // returns n! = n*(n-1)*(n-2)*...*(2)(1)
  if (n < 0) return 0;
  int f = 1;
    while (n > 1)
         f *= n--;
    return f;
}
```

```
long perm(int n,int k)
{ // returns P(n,k), the number of permutations of k from n:
    if (n < 0 || k < 0 || k > n) return 0;
    return fact(n)/fact(n-k);
}
```

```
int main()
{ // tests the perm() function:
for (int i = -1; i < 8; i++)
  {
    for (int j=-1; j <= i+1; j++)
         cout << " " << perm(i,j);
    cout << endl;
  }
}
```

```
0 0
0 1 0
0 1 1 0
0 1 2 2 0
0 1 3 6 6 0
0 1 4 12 24 24 0
0 1 5 20 60 120 120 0
0 1 6 30 120 360 720 720 0
0 1 7 42 210 840 2520 5040 5040 0
```

- خروجی برنامه‌ی زیر چیست؟

```
#include <iostream>
using namespace std;
int main()
{
    double num=0.1;
    while((num+=0.1)!=1)
    {
        num+=0.1;
    }
    cout<<--num;
}
```