

توابع بازگشتی ۱

مبانی برنامه نویسی

(۱۳۹۱-۱۳۹۰-۱۱)

جلسه نوزدهم



دانشگاه شهید بهشتی

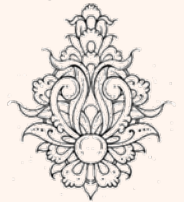
پاییز ۱۳۹۳

دانشکده مهندسی برق و کامپیوتر

احمد محمودی ازناوه

# فهرست مطالب

- ارسال از طریق مقدار و ارسال از طریق ارجاع
- ارجاع ثابت
- آشنایی با توابع بازگشتی
  - ساختار پیشته
  - چند مثال از توابع بازگشتی

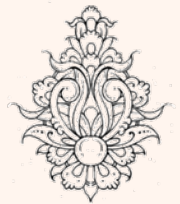


## ارسال با ارجاع (نکات)

- تنها «متغیرها» از طریق ارجاع می‌توانند ارسال گردند.
- ارسال عبارات و ثابت‌ها از طریق ارجاع برنامه را با خطا مواجه می‌کند.

```
doubleNum(5);           // Error  
doubleNum(value + 10); // Error
```

- به دلیل تخییر متغیر خارج از تابع هنگام ارسال از طریق ارجاع، این پتانسیل به برنامه داده می‌شود که به صورت سهوی مقادیر تخییر یابد.



## ارسال با ارجاع (نکات)

• چه زمان ارسال از طریق ارجاع داشته باشیم چه زمان از طریق مقدار؟

– هنگامی که آرگومان ورودی ثابت است، ارسال از طریق مقدار صورت می‌گیرد (تنها متغیرها از طریق ارجاع قابلیت ارسال دارند)

### ارسال از طریق مقدار

– هنگامی که مقدار یک متغیر نمی‌باید در جریان فراخوانی تابع تغییر کند ارسال از طریق مقدار است.

– هنگامی که فروجی تنها یک متغیر است از طریق return بازگشت داده می‌شود و ارسال از طریق مقدار است.



ارسال با ارجاع (نکات - ادامه...)

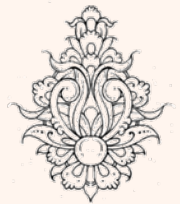
• چه زمان ارسال از طریق ارجاع داشته باشیم چه زمان از طریق مقدار؟

- هنگامی که بیش از یک مقدار بناست تخییر داده شود (چون تنها یک مقدار بازگشتی می توان داشت)، ارسال از طریق ارجاع صورت می گیرد.

- هنگامی که داده ی اصلی مقدار بزرگی است، کپی آن فرآیندی منطقی نیست، ارسال به شیوه ی ارجاع صورت می گیرد.

- هنگامی که ماهیت تابع تخییر آرگومان های ورودی باشد.

تابع *Swap*



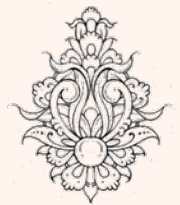
# ارسال با ارجاع (مثال - تغییر روی آرگومان‌های ورودی)

```
void swap(float&,float&);

int main()
{ // tests the swap() function:
  float a = 22.2,b = 44.4;
  cout << "a = " << a << ", b = " << b << endl;
  swap(a,b);
  cout << "a = " << a << ", b = " << b << endl;
}

void swap(float& x,float& y)
{ // exchanges the values of x and y:
  float temp = x;
  x = y;
  y = temp;
}
```

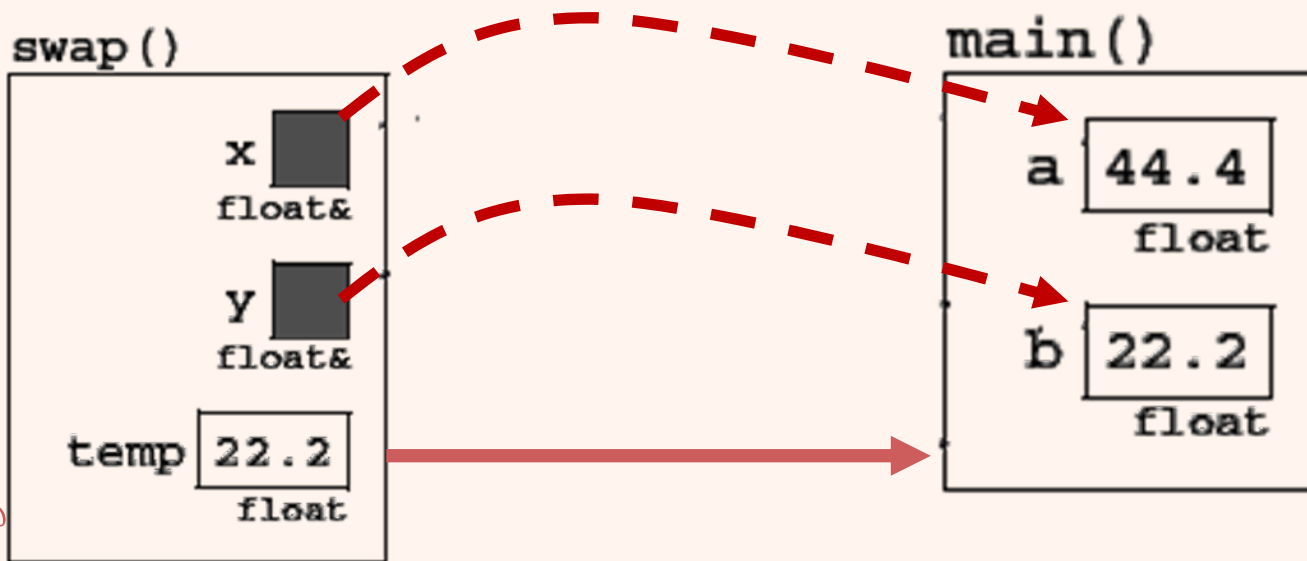
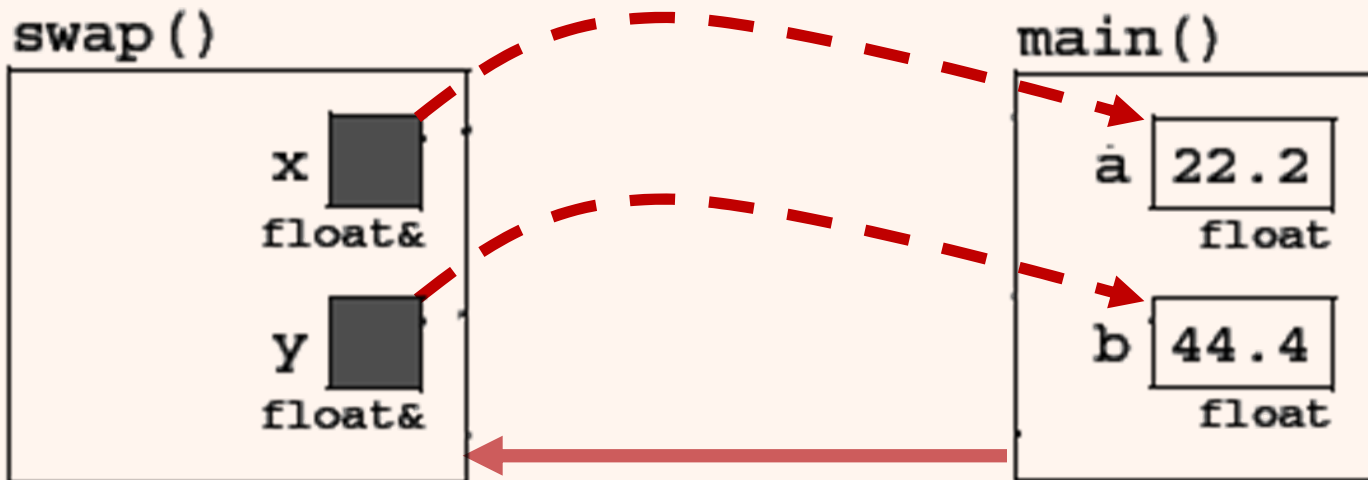
```
a = 22.2, b = 44.4
a = 44.4, b = 22.2
```



# تغییر روی آرگومان‌های ورودی

```
int main()
{ // tests the swap() function:
  float a = 22.2, b = 44.4;
  cout << "a = " << a << ", b = " << b << endl;
  swap(a,b);
  cout << "a = " << a << ", b = " << b << endl;
}

void swap(float& x, float& y)
{ // exchanges the values of x and y:
  float temp = x;
  x = y;
  y = temp;
}
```



دانشگاه  
تهران  
پهشتی

# ارسال با ارجاع (مثال - تغییر روی آرگومان‌های ورودی)

```
void f(int, int&);

int main()
{ // tests the f() function:
int a = 22, b = 33;
cout << "a = " << a << ", b = " << b << endl;
f(a, b);
cout << "a = " << a << ", b = " << b << endl;
f(2*a-3, b);
cout << "a = " << a << ", b = " << b << endl;
}

void f(int x, int& y)
{
x = 88;
y = 99;
}
```

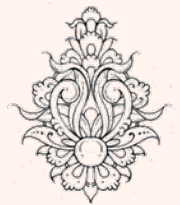
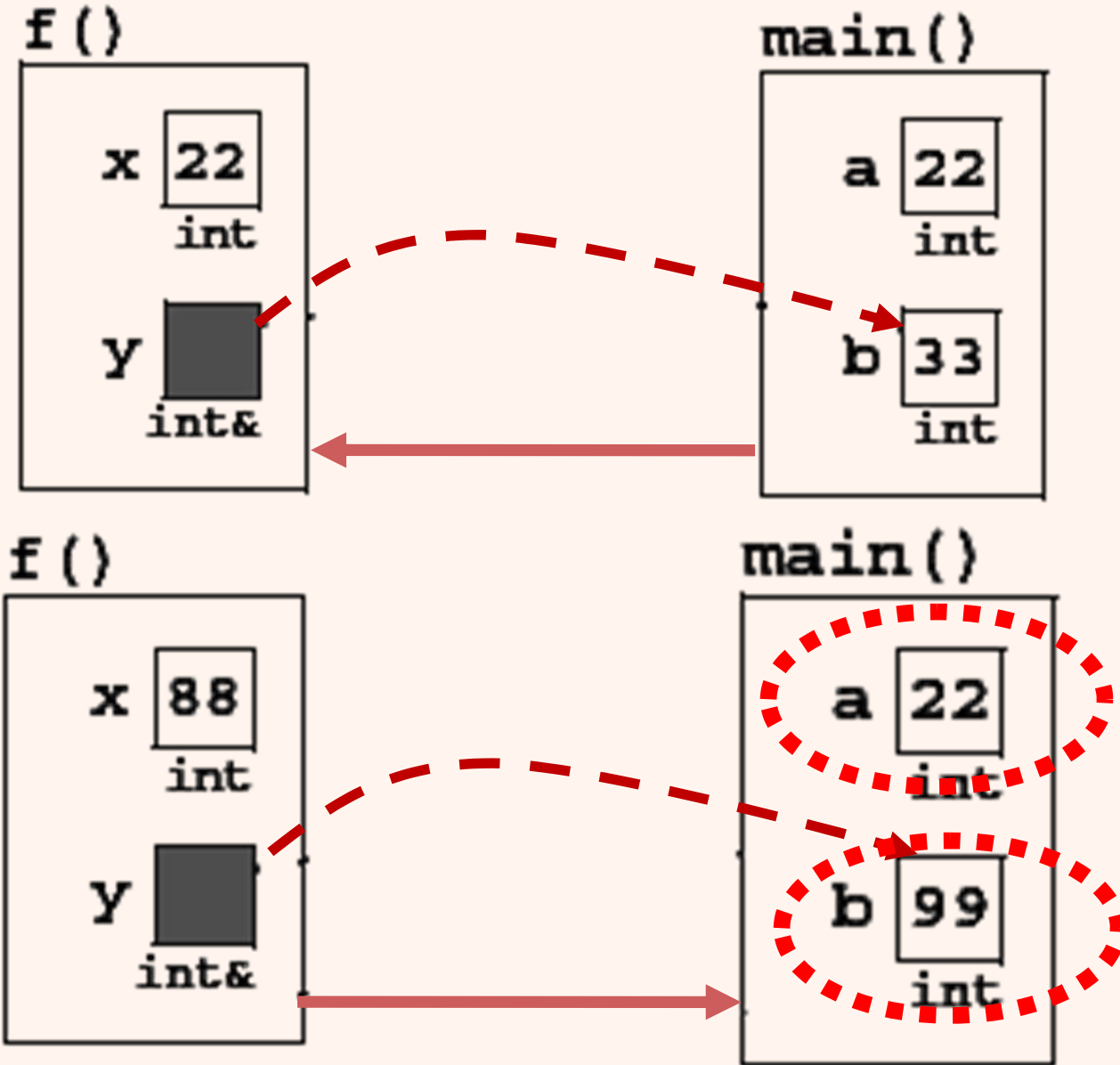
```
a = 22, b = 33
a = 22, b = 99
a = 22, b = 99
```





# تغییر روی آرگومان‌های ورودی

```
void f(int x, int& y)
{
x = 88;
y = 99;
}
```



```
void getNums (int&, int&);
```

```
void SwapNums (int&, int&)
```

```
int main()
```

```
{
int small, big;
```

```
getNums (small, big);
```

```
cout << "The two input numbers:"
```

```
<< small << " and " << big << endl;
```

```
SwapNums (small, big);
```

```
cout << "The two input numbers Swapped:"
```

```
<< small << " and " << big << endl;
```

```
return 0;
```

```
}
```

```
void getNums (int& input1, int& input2)
```

```
{
```

```
cout << "Enter an integer: ";
```

```
cin >> input1;
```

```
cout << "Enter a second integer: ";
```

```
cin >> input2;
```

```
}
```

```
Enter an integer: 34
```

```
Enter a second integer: 89
```

```
The two input numbers:34 and 89
```

```
The two input numbers Swapped:89 and 34
```

```
void SwapNums (int& num1, int& num2)
```

```
{
```

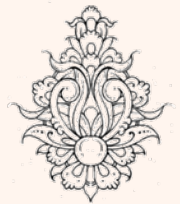
```
int temp;
```

```
temp = num1;
```

```
num1 = num2;
```

```
num2 = temp;
```

```
}
```



## مثال

```
void getNums (int&, int&);
```

```
void SwapNums (int, int);
```

```
int main()
```

```
{
```

```
int small, big;
```

```
getNums (small, big);
```

```
cout << "The two input numbers:"
```

```
<< small << " and " << big << endl;
```

```
SwapNums (small, big);
```

```
cout << "The two input numbers Swapped:"
```

```
<< small << " and " << big << endl;
```

```
return 0;
```

```
}
```

```
void getNums (int& input1, int& input2)
```

```
{
```

```
cout << "Enter an integer: ";
```

```
cin >> input1;
```

```
cout << "Enter a second integer: ";
```

```
cin >> input2;
```

```
}
```

```
integer: 34
```

```
second integer: 89
```

```
input numbers:34 and 89
```

```
input numbers Swapped:34 and 89
```

```
void SwapNums (int num1, int num2)
```

```
{
```

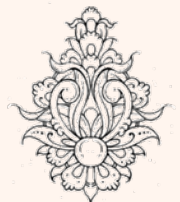
```
int temp;
```

```
temp = num1;
```

```
num1 = num2;
```

```
num2 = temp;
```

```
}
```



## ارسال با ارجاع (مثال - چند خروجی)

- استفاده از ارجاع هنگامی که بیش از یک خروجی در تابع نیاز است:

```
void computeCircle(double& area, double& circumference, double r)
{
    const double PI = 3.141592653589793;
    area = PI*r*r;
    circumference = 2*PI*r;
}

int main()
{
    double r,a,c;
    cout << "Enter radius: ";
    cin >> r;
    computeCircle(a, c, r);
    cout << "area = " << a << ", circumference = " << c << endl;
}
```

```
Enter radius: 3
area = 28.2743, circumference = 18.8496
```

## ارسال به وسیلهی ارجاع ثابت

- یکی از مواردی که ارسال به وسیلهی ارجاع پیشنهاد داده می‌شود زمانی است که آرگومان ارسال می‌شود.
- در این صورت است که از کپی کردن آرگومان جلوگیری می‌شود و برنامه بازده بیشتری خواهد داشت.

ممکن است تابع آرگومان را نافرماسته تغییر دهد

پیشنهاد

ارسال ارجاع ثابت



# مثال (ارسال به وسیلهی ارجاع ثابت)

```
void f(int, int&, const int&);
int main()
{ // tests the f() function:
  int a = 22, b = 33, c = 44;
  cout << "a = " << a << ", b = " << b << ", c = " << c << endl;
  f(a, b, c);
  cout << "a = " << a << ", b = " << b << ", c = " << c << endl;
  f(2*a-3, b, c);
  cout << "a = " << a << ", b = " << b << ", c = " << c << endl;
}
void f(int x, int& y, const int& z)
{
  x += z;
  y += z;
  cout << "x = " << x << ", y = " << y << ", z = " << z << endl;
}
```



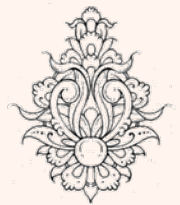
```
a = 22, b = 33, c = 44
x = 66, y = 77, z = 44
a = 22, b = 77, c = 44
x = 85, y = 121, z = 44
a = 22, b = 121, c = 44
```



## مثال (ارسال به وسیلهی ارجاع ثابت)

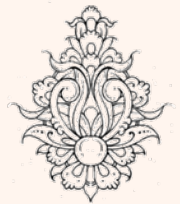
- اگر سعی کنید ارجاعی را که ثابت است در تابع تغییر دهید، با خطای کامپایلری مواجه می‌شوید.

```
int main()
{ // tests the f() function:
  int a = 22, b = 33, c = 44;
  cout << "a = " << a << ", b = " << b << ", c = " << c << endl;
  f(a,b,c);
  cout << "a = " << a << ", b = " << b << ", c = " << c << endl;
  f(2*a-3,b,c);
  cout << "a = " << a << ", b = " << b << ", c = " << c << endl;
}
void f(int x, int& y, const int& z)
{
  x += z;
  y += z;
  z++;
  cout << "x = " << x << ", y = " << y << ", z = " << z << endl;
}
```



**error C3892: 'z' : you cannot assign to a variable that is const**

- استفاده از تابع **exit** سبب می‌شود که از برنامه خارج شویم.
- هنگامی که در **main** به دستور **return** می‌رسیم برنامه پایان می‌یابد. استفاده از **return** در توابع دیگر سبب می‌شود کنترل برنامه به **main** (یا تابعی که تابع فعلی را صدا زده است) بازگردد.
- در بعضی موارد نیاز است در هر موقعیتی از برنامه خارج شویم.
- تابع **exit** از هر کجا فراخوانی شود، سبب خروج از برنامه می‌گردد.





# مثال (تابع خروج)

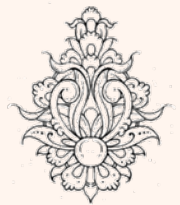
```
#include <iostream>
using namespace std;

// Function prototype
void someFunction();
```

```
int main()
{
    someFunction();
    return 0;
}
```

```
This program terminates with the exit function.
Bye!
```

```
void someFunction()
{
    cout << "This program terminates with the exit function.\n";
    cout << "Bye!\n";
    exit(0);
    cout << "This message will never be displayed\n";
    cout << "because the program has already terminated.\n";
}
```



# توابع بازگشتی

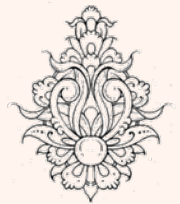
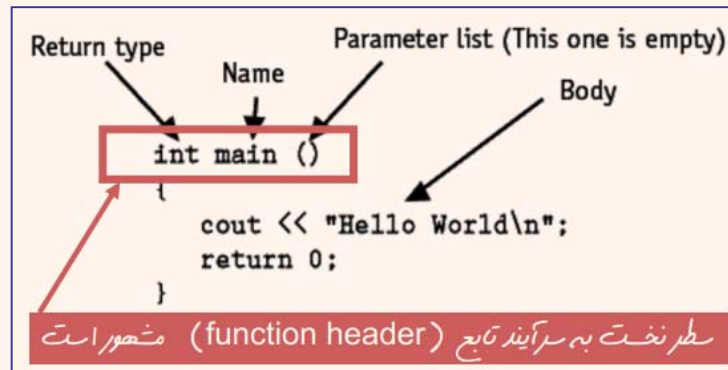


## توابع بازگشتی

- Overhead بالا به دلیل فراخوانی تابع
  - نگاه داشتن آدرس بازگشتی، ایجاد متغیرها
- مدیریت حافظه
  - در هر بار فراخوانی تابع مقداری حافظه اختصاص داده شده مصرف می‌شود.
- پیاده‌سازی با دستورات عمل‌های کمتر همراه با ظرافت و خوانایی بهتر برنامه

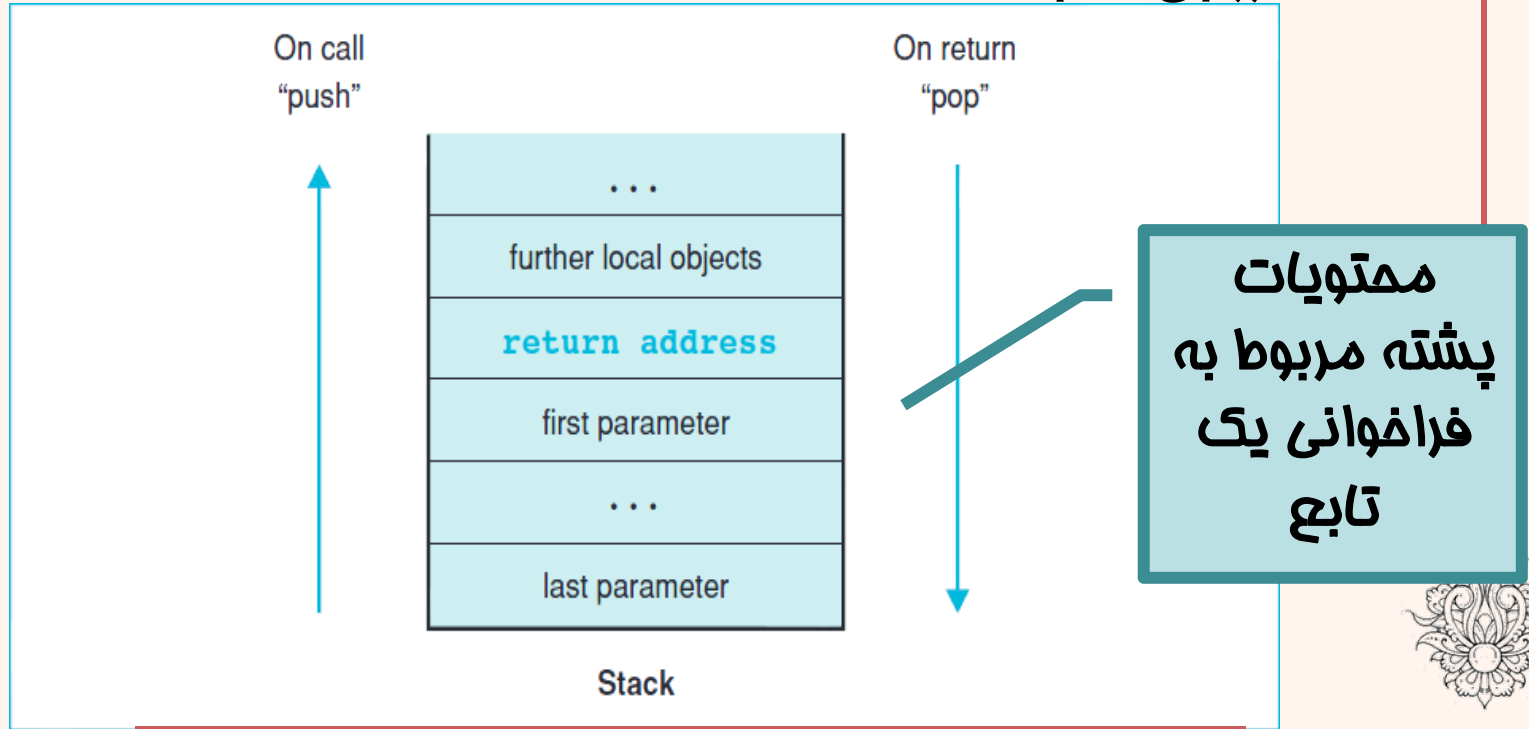
## مروری بر تابع

- در جلسات گذشته در مورد تابع و دلایل استفاده از آن صحبت کردیم.
- گفتیم که یک تابع به ازای هر ترتیب از آرگومان‌های ورودی حداکثر یک خروجی رسمی (**return value**) دارد؛
- با این حال تاکنون اجباری در استفاده از توابع برای حل مسائل نداشتیم و هر سوالی که در مورد آن صحبت کردیم، مستقل از پیاده‌سازی به وسیله توابع نیز، قابل حل بوده است.



# فراخوانی تابع

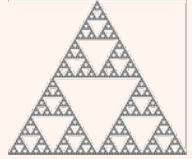
- پیش‌تر در مورد صدا کردن یک تابع، ارسال آرگومان و بازگرداندن خروجی صحبت شد.



A complete guide to programming in C++

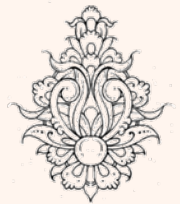
پشته (**stack**) یکی از انواع ساختمان داده‌ها است و برای ذخیره و بازیابی داده‌ها به کار می‌رود. شیوهی عملکرد پشته بر اساس سیاست **LIFO (Last in First Out)** است.





- ساختار توابع بازگشتی به گونه‌ای است که خود را فراخوانی می‌نمایند.
- فراخوانی می‌تواند به صورت مستقیم و یا غیرمستقیم از طریق توابع دیگر صورت پذیرد.

```
void recurse ( int count )  
{  
    cout<< count <<"\n";  
  
    recurse ( count + 1 );  
}  
  
int main()  
{  
    recurse ( 1 );  
}
```



توابع بازگشتی (ادامه...)

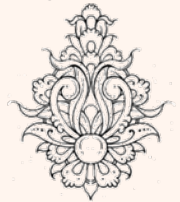
- تابع زیر در هر بار فراخوانی مجدداً خود را فرا می‌خواند:

```
void message() {  
    cout << "This is a recursive function.\n";  
    message();  
}
```

- آیا مشکلی در تابع می‌بینید؟

هیچ راه‌کاری برای توقف تابع تعبیه نشده است.

تابع مانند طقه‌ی بی‌پایان عمل می‌کند.



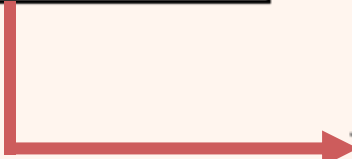
## توابع بازگشتی (ادامه...)

- برای کاربردی بودن تابع بازگشتی می‌باید به‌گونه‌ای راهی برای توقف تابع در نظر گرفت:

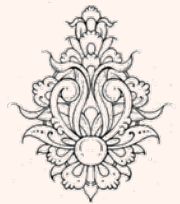
```
void message(int times){  
    if (times > 0){  
        cout << "This is a recursive function.\n";  
        message(times - 1);  
    }  
}
```



First call of the function
Value of times: 5

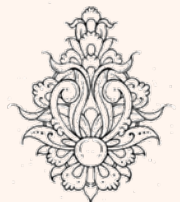


Second call of the function
Value of times: 4



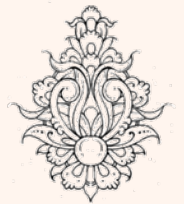
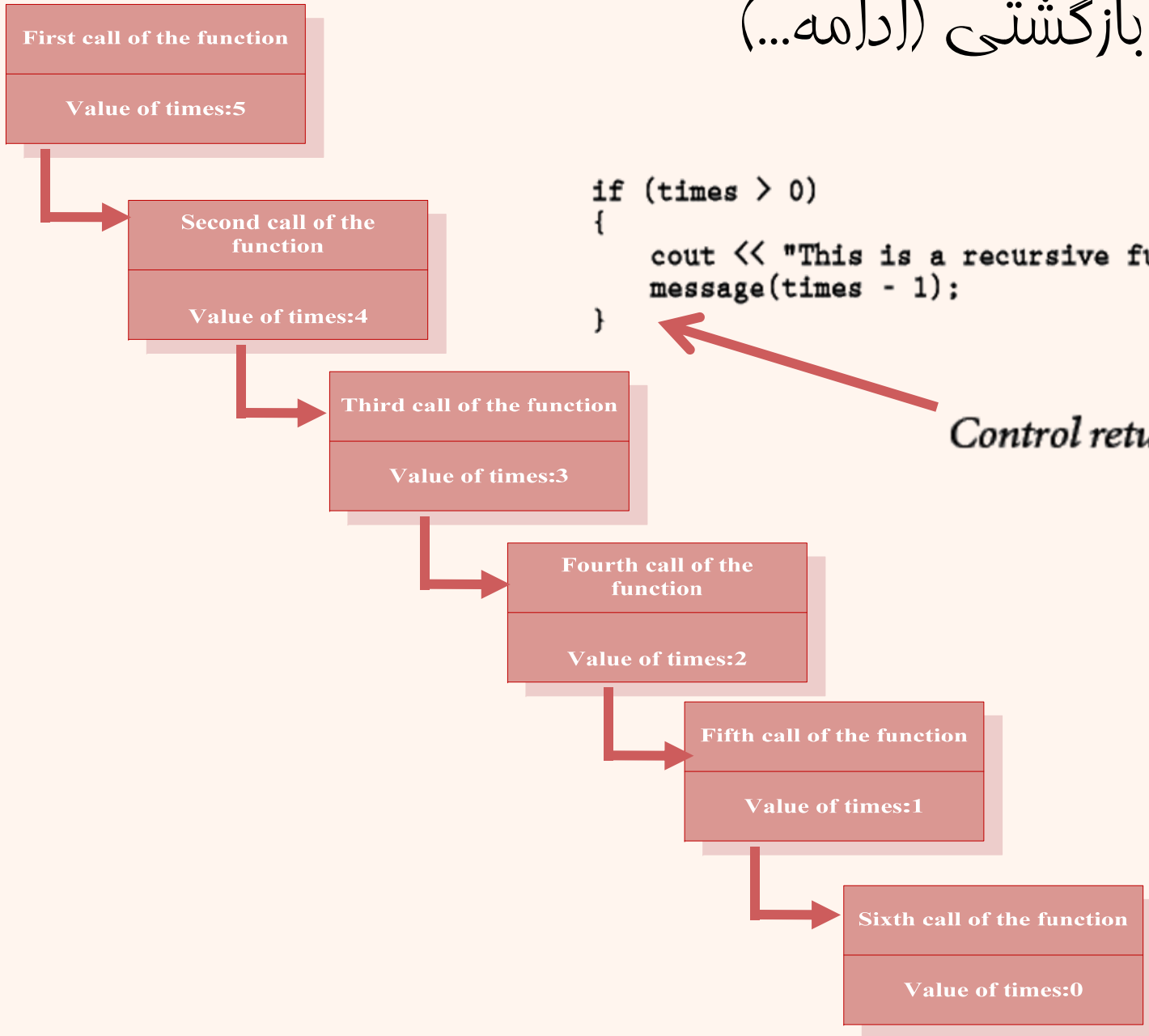
```
void message(int);  
int main() {  
    message(5);  
    return 0;  
}  
void message(int times) {  
    if (times > 0) {  
        cout << "This is a recursive function.\n";  
        message(times - 1);  
    }  
}
```

```
This is a recursive function.  
This is a recursive function.  
This is a recursive function.  
This is a recursive function.  
This is a recursive function.
```





# توابع بازگشتی (ادامه...)



```

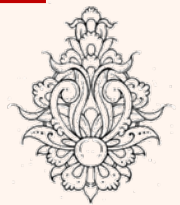
void message(int);
int main(){
    message(5);
    return 0;
}
void message(int times){
    if (times > 0){
        cout << "This is a recursive function\t"<<times<<endl;
        message(times - 1);
    }
}

```

```

This is a recursive function 5
This is a recursive function 4
This is a recursive function 3
This is a recursive function 2
This is a recursive function 1

```



## توابع بازگشتی (ادامه...)

- تابع بازگشتی با شکستن فرآیندی تکراری به حل مساله کمک می‌کند.
- اجرای فرآیند تا زمانی که به یک «base case» برسیم ادامه خواهد داشت.
- در مثال قبل base case زمانی است که پارامتر time برابر با صفر است.
- در این صورت فرآیند فراخوانی متوقف می‌شود.



## توابع بازگشتی (ادامه...)

```
void message(int);  
int main()  
{  
    message(5);  
    return 0;  
}
```

```
void message (int times)  
{  
    cout << "message called with " << times << " in times.\n";  
    if (times > 0)  
    {  
        cout << "This is a recursive function.\n";  
        message(times - 1);  
    }  
    cout << "message returning with " << times;  
    cout << " in times.\n";  
}
```

```
message called with 5 in times.  
This is a recursive function.  
message called with 4 in times.  
This is a recursive function.  
message called with 3 in times.  
This is a recursive function.  
message called with 2 in times.  
This is a recursive function.  
message called with 1 in times.  
This is a recursive function.  
message called with 0 in times.  
message returning with 0 in times.  
message returning with 1 in times.  
message returning with 2 in times.  
message returning with 3 in times.  
message returning with 4 in times.  
message returning with 5 in times.
```

