مبانی برنامه‌نویسی
(۱۳۹۹-۱۳-۱۱)
جلسه‌ی بیست‌ویکم

توابع ۴
تابع بازگشتی

دانشگاه شهید بهشتی
پاییز ۱۳۹۹
دانشکده‌ی مهندسی برق و کامپیوتر
احمد محمودی ازناوه

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

فهرست مطالب

- مروری بر جلسه‌ی پیش
- ارسال از طریق ارجاع
  - ارجاع ثابت

متغیر ایستا (مثال)

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

// Function prototype
void showStatic();
int main()
 {
 for (int count = 0; count < 5; count++)
 showStatic();
 return 0;
 }

 void showStatic()
 {
 static int statNum; // Static local variable

 cout << "statNum is " << statNum << endl;
 statNum++;
 }
```

```
statNum is 0
statNum is 1
statNum is 2
statNum is 3
statNum is 4
```

مثال

```
// Function prototype with default arguments
 void displayStars(int = 10, int = 1);

int main()
 {
 displayStars();
 cout << endl;
 displayStars(5);
 cout << endl;
 displayStars(7, 3);
 return 0;
 }
 void displayStars(int cols, int rows)
 {
  for (int down = 0; down < rows; down++)
    {
       for (int across = 0; across < cols; across++)
        cout << '*';
      cout << endl;
    }
 }
```

Uses default values of 10 and 1 for cols and rows

Uses 5 for cols and default value of 1 for rows

Uses 7 for cols and 3 for rows (no default values)

مثال

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

int operate (int a, int b)
{
  return (a*b);
}

float operate (float a, float b)
{
  return (a/b);
}

int main ()
{
  int x=5,y=2;
  float n=5.0,m=2.0;
  cout << operate (x,y);
  cout << "\n";
  cout << operate (n,m);
  cout << "\n";
  return 0;
}
```

```
10
2.5
```

C++ C99

تابع inline

- در فراخوانی تابع همواره مراحلی باید انجام شود که هم زمان‌بر است و هم به حافظه‌ی بیشتری جهت فراخوانی نیاز دارد.
- مراحلی چون ارسال آرگومان‌های ورودی، در نظر گرفتن فضا برای متغیرهای محلی و... وجود دارد.
- در برخی موارد بهتر است برای به دست آوردن بازده بیشتر از انجام چنین مراحلی جلوگیری شود.
- در این گونه موارد با قرار دادن کلمه‌ی کلیدی inline، کامپایلر بدنه‌ی تابع را به جای فراخوانی، در برنامه‌ی اصلی قرار می‌دهد.

ارسال با ارجاع (مثال)

```
void doubleNum(int&);
void getNum(int&);
 int main()
 {
 int value=0;
 getNum(value);
 doubleNum(value);
 cout << "That value doubled is " << value << endl;
 return 0;
 }
```

```
Enter a number: 45
That value doubled is 90
```

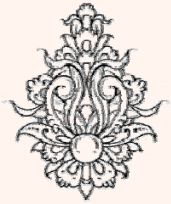
```
void getNum(int& userNum)
{
cout << "Enter a number: ";
cin >> userNum;
}
void doubleNum (int& refVar)
{
refVar *= 2;
}
```

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



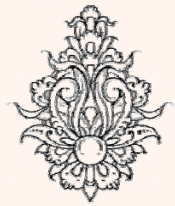
تنها در مواردی که نیاز است از ارسال به طریق ارجاع استفاده کنید

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

  - هنگامی‌که مقدار یک متغیر نمی‌باید در جریان فراخوانی تابع تغییر کند ارسال از طریق مقدار است.
  - هنگامی‌که خروجی تنها یک متغیر است از طریق return بازگشت داده می‌شود و ارسال از طریق مقدار است.



ارسال با ارجاع (نکات – ادامه...)

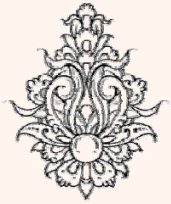
- چه زمان ارسال از طریق ارجاع داشته باشیم چه زمان از طریق مقدار؟
  - هنگامی‌که بیش از یک مقدار بناست تغییر داده شود (چون تنها یک مقدار بازگشتی می‌توان داشت)، ارسال از طریق ارجاع صورت می‌گیرد.
  - هنگامی‌که داده‌ی اصلی مقدار بزرگی است، کپی آن فرآیندی منطقی نیست، ارسال به شیوه‌ی ارجاع صورت می‌گیرد.
  - هنگامی‌که ماهیت تابع تغییر آرگومان‌های ورودی باشد.

تابع Swap

ارسال با ارجاع(مثال- تغییر روی آرگومان‌های ورودی)

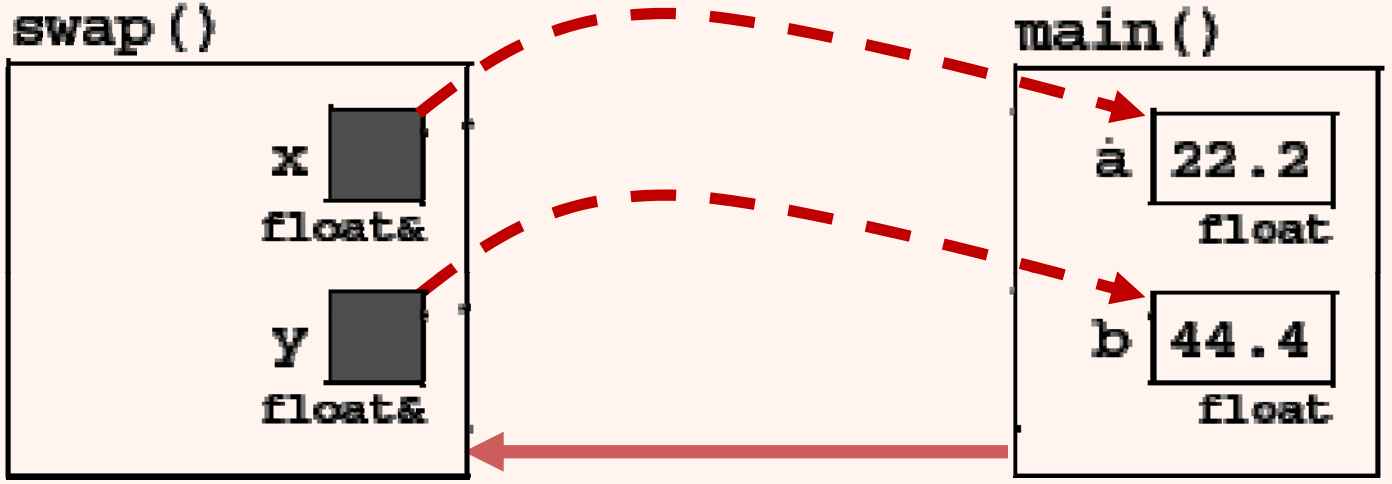
```
void swap(float&,float&);

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;
}
```

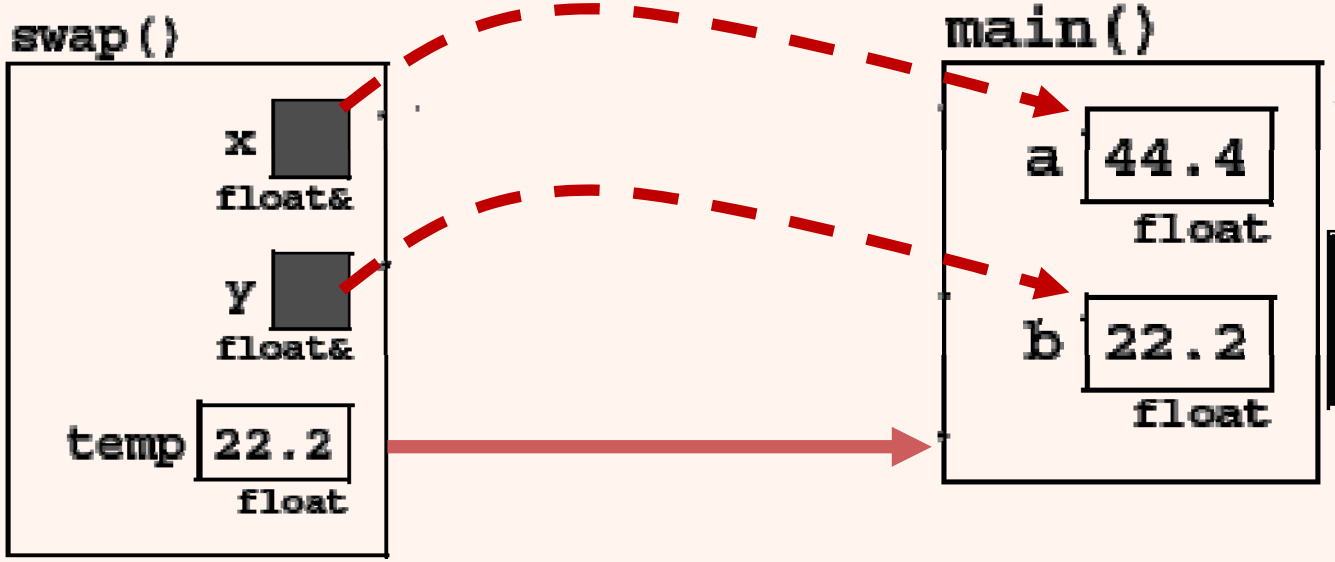
```
a = 22.2, b = 44.4
a = 44.4, b = 22.2
```



(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;
}
```





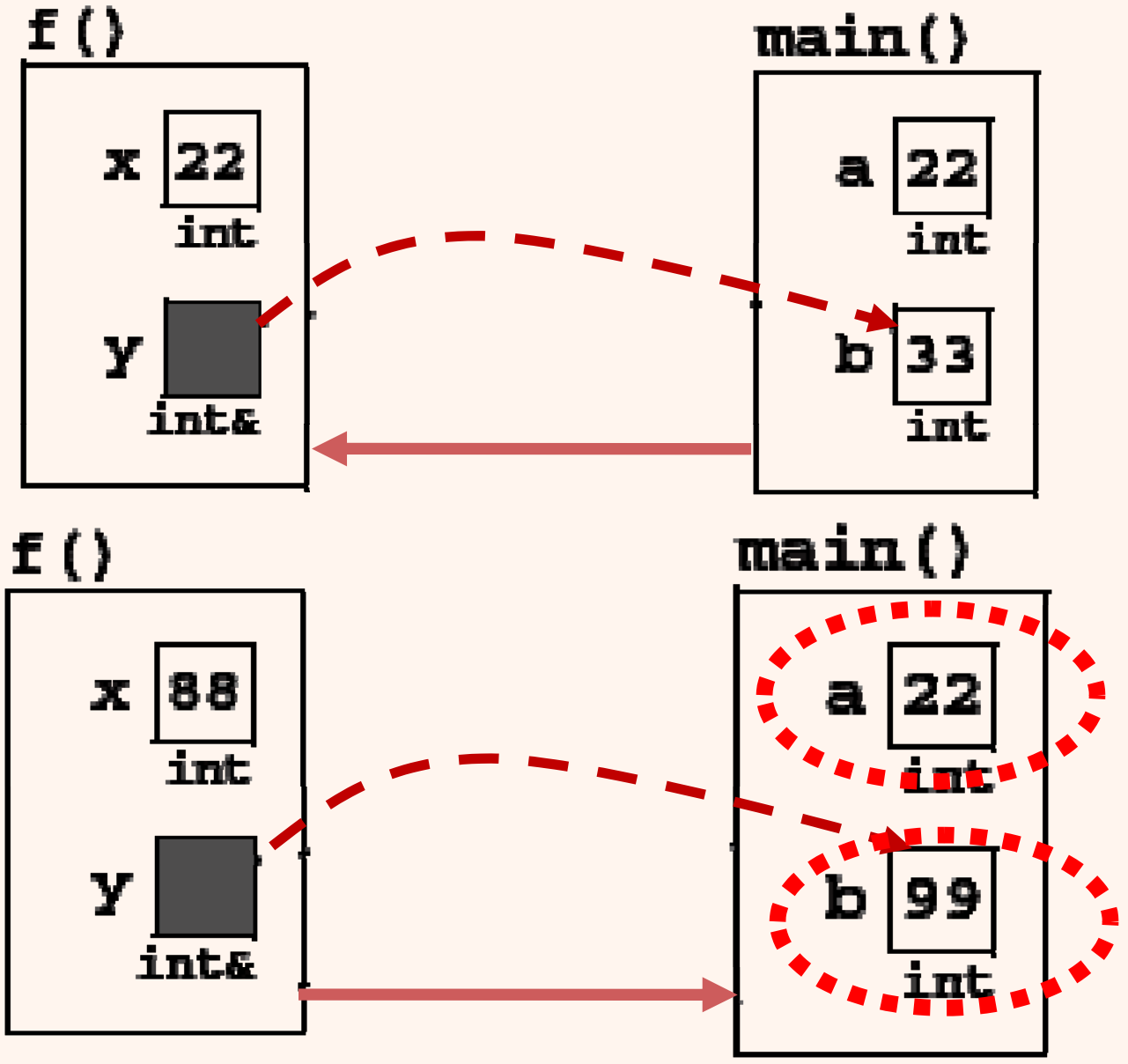
ارسال با ارجاع (مثال- تغییر روی آرگومان‌های ورودی)

```
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;
}
```



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;
 }
```

```
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;

 }
```

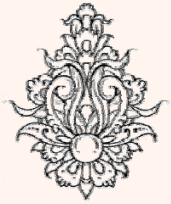
مثال

```
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
econd integer: 89
nput numbers:34 and 89
nput 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 هنگامی‌که 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 ارسالی حجیم است.
- در این صورت است که از کپی کردن آرگومان جلوگیری می‌شود و برنامه بازده بیشتری خواهد داشت.

ممکن است تابع آرگومان را ناخواسته 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

exit function

تابع خروج

- استفاده از تابع exit سبب می‌شود که از برنامه خارج شویم.
- هنگامی‌که 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;
 }

 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";
 }
```

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

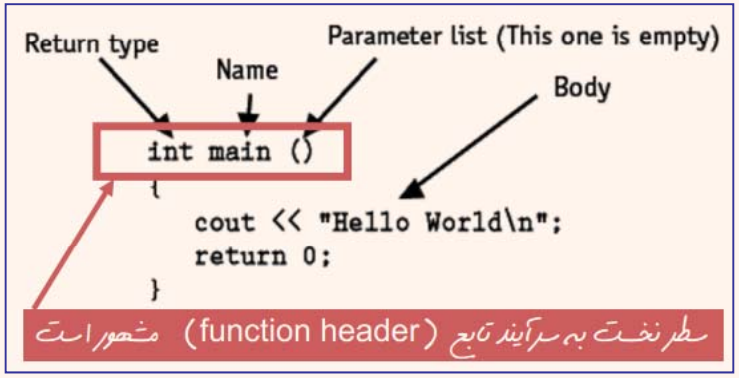
توابع بازگشتی

توابع بازگشتی

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

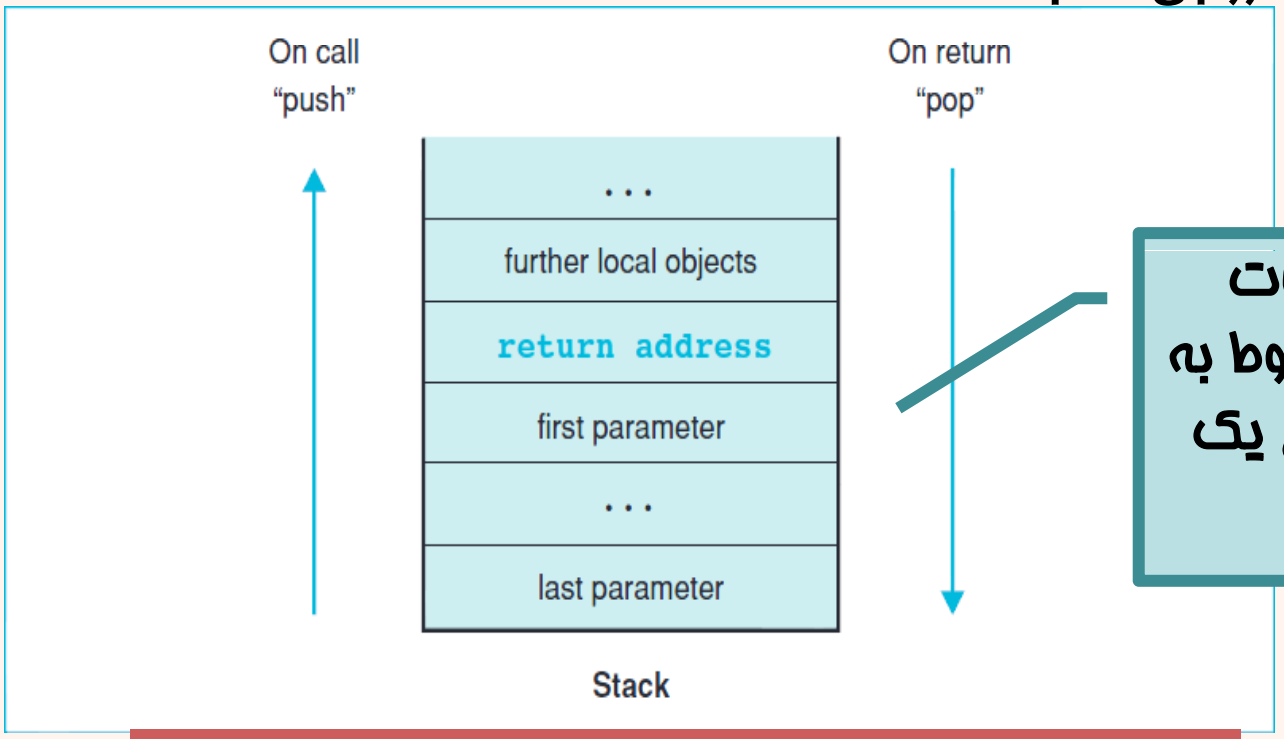
مروری بر تابع

- 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 وسیله‌ی توابع نیز، قابل حل بوده است.



فراخوانی تابع

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

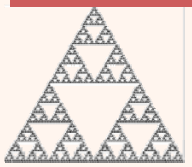


محتویات پشته مربوط به فراخوانی یک تابع

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.

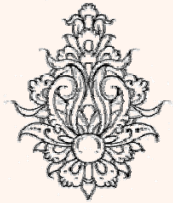
A recursive function is one that calls itself.



توابع بازگشتی

- 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 فرامی‌خواند:

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

- آیا مشکلی در تابع می‌بینید؟

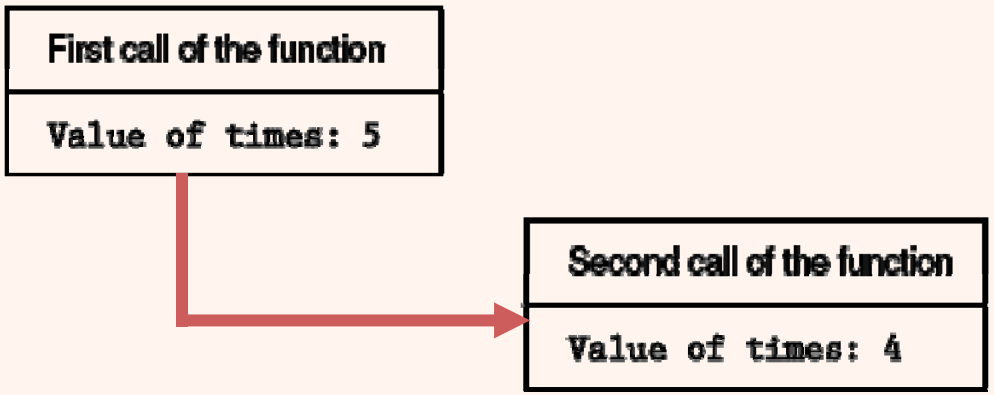
هیچ راه‌کاری برای توقف تابع تعبیه نشده است.

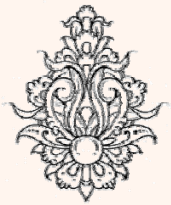
تابع مانند حلقه‌ی بی‌پایان عمل می‌کند.

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);
     }
}
```

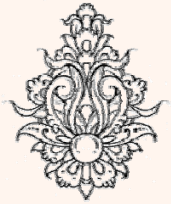




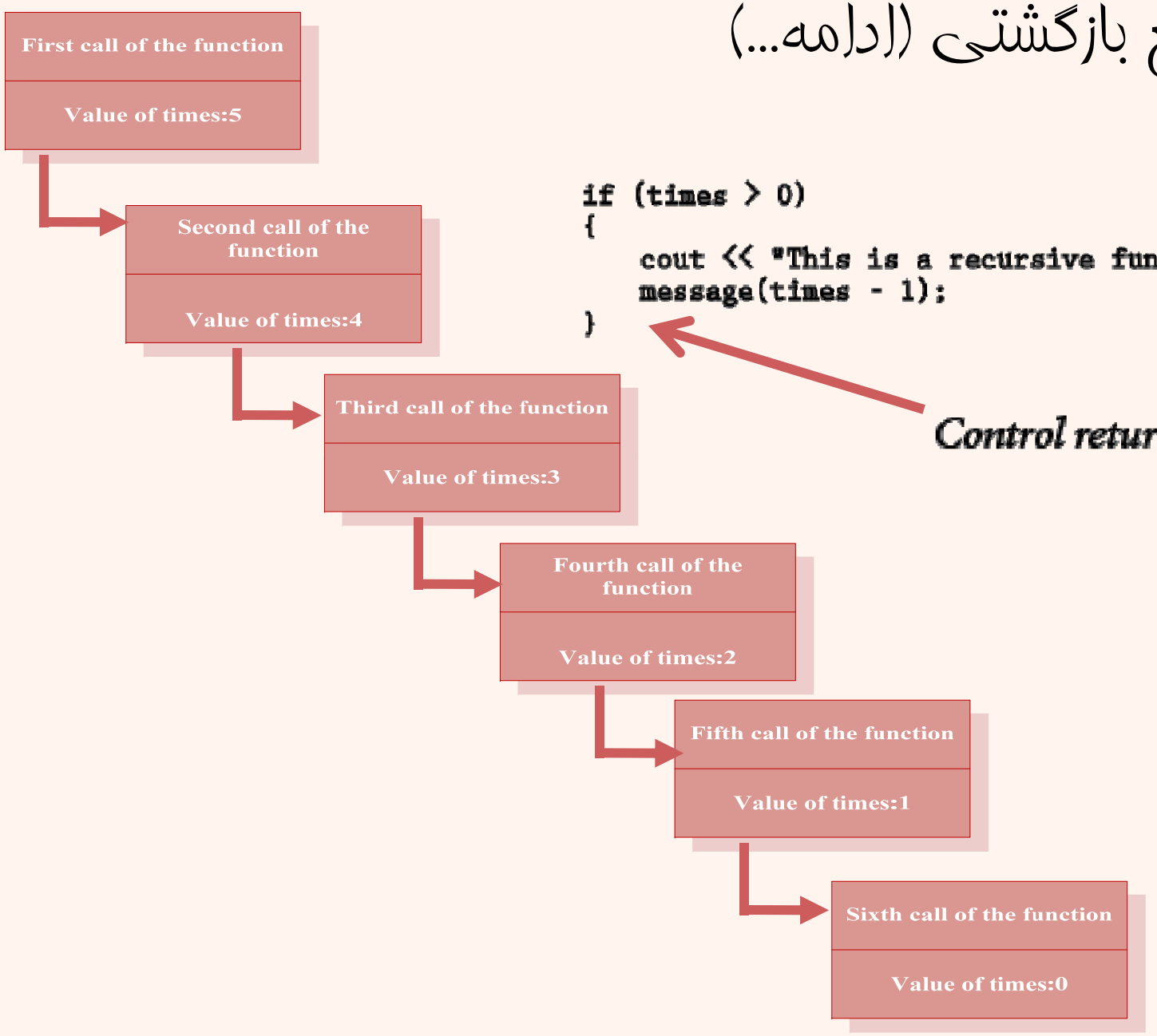
مثال

```
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.
```



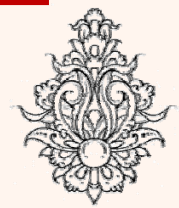
توابع بازگشتی (ادامه...)



مثال

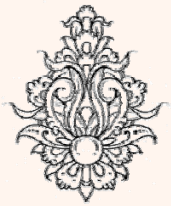
```
void message(int);
int main(){
    message(5);
    return 0;
}
void message(int times){
    if (times > 0){
        cout << "This is a recursive function\t"<<times<<endl;
        message(times - 1);
    }
}
```

```
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...)

```
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.
```

```
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";
 }
```

تمرین

- خروجی برنامه‌ی زیر چیست؟

```
void showMe(int arg);
int main()
{
    int num = 0;
    showMe(num);
    return 0;
}
void showMe(int arg)
{
    if (arg < 10)
        showMe(++arg);
    else
        cout << arg << endl;
}
```

```
10
```