



# Αναφορές (references)

Εργίνα Καβαλλιεράτου

# Σημερινό Μάθημα

- ✓ Αναφορές (references)
- ✓ Που ορίζουμε αναφορά
- ✓ Αναφορές σε αντικείμενα
- ✓ Πέρασμα με αναφορά
- ✓ Πέρασμα πολλαπλών τιμών

# Αναφορές (references)

- ✓ Οι αναφορές έχουν τη δύναμη των δεικτών αλλά με πιο εύκολη σύνταξη
- ✓ Όταν δημιουργούμε μια αναφορά την αρχικοποιούμε με μία άλλη μεταβλητή (στόχο)

```
int &rSomeRef = someInt;
```

- ✓ Η αναφορά λειτουργεί ως εναλλακτικό όνομα για το στόχο και ότι κάνουμε στην αναφορά περνάει στο στόχο

# Παράδειγμα

```
#include <iostream.h>

int main() {
    int intOne;
    int &rSomeRef = intOne;    intOne = 5;
    cout << "intOne: " << intOne << endl;
    cout << "rSomeRef: " << rSomeRef << endl;
    rSomeRef = 7;
    cout << "intOne: " << intOne << endl;
    cout << "rSomeRef: " << rSomeRef << endl;
    return 0; }
```

```
intOne: 5
rSomeRef: 5
intOne: 7
rSomeRef: 7
```

# Παράδειγμα

```
#include <iostream.h>

int main() {
    int intOne;
    int &rSomeRef = intOne;
    intOne = 5;
    cout << "intOne: " << intOne << endl;
    cout << "rSomeRef: " << rSomeRef << endl;
    cout << "&intOne: " << &intOne << endl;
    cout << "&rSomeRef: " << &rSomeRef << endl;
    return 0; }
```

```
intOne: 5
rSomeRef: 5
&intOne: 0x3500
&rSomeRef: 0x3500
```

# Παράδειγμα/1

```
#include <iostream.h>

int main() {
    int intOne;
    int &rSomeRef = intOne;
    intOne = 5;
    cout << "intOne:\t" << intOne << endl;
    cout << "rSomeRef:\t" << rSomeRef << endl;
    cout << "&intOne:\t" << &intOne << endl;
    cout << "&rSomeRef:\t" << &rSomeRef << endl;
    int intTwo = 8;
```



## Παράδειγμα/2

```
rSomeRef = intTwo;  
cout << "\nintOne:\t" << intOne << endl;  
cout << "intTwo:\t" << intTwo << endl;  
cout << "rSomeRef:\t" << rSomeRef << endl;  
cout << "&intOne:\t" << &intOne << endl;  
cout << "&intTwo:\t" << &intTwo << endl;  
cout << "&rSomeRef:\t" << &rSomeRef;  
return 0; }
```

intOne:	5
rSomeRef:	5
&intOne:	0x213e
&rSomeRef:	0x213e
intOne:	8
intTwo:	8
rSomeRef:	8
&intOne:	0x213e
&intTwo:	0x2130
&rSomeRef:	0x213e

# Συμβουλές

- ✓ Αρχικοποιείται όλες τις αναφορές σε στόχους
- ✓ ΜΗΝ προσπαθήστε να ορίσετε νέο στόχο σε αναφορά
- ✓ Μην συγχέετε τον τελεστή διεύθυνσης με τον τελεστή δήλωσης αναφοράς



# Που ορίζουμε αναφορά

**Λάθος**

```
int & rIntRef = int;
```

**Σωστό**

```
int howBig = 200;  
int & rIntRef = howBig;
```

# Που ορίζουμε αναφορά

- Μπορούμε να ορίσουμε αναφορά σε αντικείμενο
- Δεν μπορούμε να ορίσουμε αναφορά σε κλάση
- Οι αναφορές πρέπει να αρχικοποιούνται κατά τη δήλωση

```
int hisAge;
```

```
int &rAge = hisAge;
```

```
CAT boots;
```

```
CAT &rCatRef = boots;
```

# Που ορίζουμε αναφορά

**Λάθος**

CAT & rCatRef = CAT;

**Σωστό**

CAT frisky;

CAT & rCatRef = frisky;

# Αναφορές σε αντικείμενα/1

```
#include <iostream.h>

class SimpleCat
{
public:
    SimpleCat (int age, int weight);
    ~SimpleCat() {}
    int GetAge() { return itsAge; }
    int GetWeight() { return itsWeight; }
private:
    int itsAge;      int itsWeight;  };
```

## Αναφορές σε αντικείμενα/2

```
SimpleCat::SimpleCat(int age, int weight) {  
    itsAge = age;  
    itsWeight = weight; }  
  
int main() {  
    SimpleCat Frisky(5,8);  
    SimpleCat & rCat = Frisky;  
    cout << "Frisky is: ";  
    cout << Frisky.GetAge() << " years old. \n";  
    cout << "And Frisky weighs: ";  
    cout << rCat.GetWeight() << " pounds. \n";  
    return 0; }
```

Frisky is: 5 years old.  
And Frisky weighs 8 pounds.

# Πέρασμα με αναφορά

- Το πέρασμα με αναφορά μπορεί να γίνει με τη χρήση δεικτών ή αναφορών
- Σε αυτή την περίπτωση αντί να περάσουμε αντίγραφο των δεδομένων στη συνάρτηση, περνάμε τα original δεδομένα.



# Πέρασμα με τιμή/1

```
#include <iostream.h>
```

```
void swap(int x, int y);
```

```
int main()
```

```
{
```

```
    int x = 5, y = 10;
```

```
    cout << "Main. Before swap, x: " << x << " y: " << y << "\n";
```

```
    swap(x,y);
```

```
    cout << "Main. After swap, x: " << x << " y: " << y << "\n";
```

```
    return 0;
```

```
}
```

Main. Before swap, x: 5 y: 10  
Swap. Before swap, x: 5 y: 10  
Swap. After swap, x: 10 y: 5  
Main. After swap, x: 5 y: 10

# Πέρασμα με αναφορά (δείκτες)/1

```
#include <iostream.h>
```

```
void swap(int *x, int *y);
```

```
int main()
```

```
{
```

```
    int x = 5, y = 10;
```

```
    cout << "Main. Before swap, x: " << x << " y: " << y << "\n";
```

```
    swap(&x,&y);
```

```
    cout << "Main. After swap, x: " << x << " y: " << y << "\n";
```

```
    return 0;
```

```
}
```

## Πέρασμα με αναφορά (δείκτες)/2

```
void swap (int *px, int *py)
```

```
{
```

```
    int temp;
```

```
    cout << "Swap. Before swap, *px: " << *px << " *py: " << *py << "\n";
```

```
    temp = *px;
```

```
    *px = *py;
```

```
    *py = temp;
```

```
    cout << "Swap. After swap, *px: " << *px << " *py: " << *py << "\n";
```

```
}
```

```
Main. Before swap, x: 5 y: 10  
Swap. Before swap, *px: 5 *py: 10  
Swap. After swap, *px: 10 *py: 5  
Main. After swap, x: 10 y: 5
```

# Πέρασμα με αναφορά (αναφορές)/1

```
#include <iostream.h>
```

```
void swap(int &rx, int &ry);
```

```
int main() {
```

```
    int x = 5, y = 10;
```

```
    cout << "Main. Before swap, x: " << x << " y: " << y << "\n";
```

```
    swap(x,y);
```

```
    cout << "Main. After swap, x: " << x << " y: " << y << "\n";
```

```
    return 0;
```

```
}
```

## Πέρασμα με αναφορά (αναφορές)/2

```
void swap (int &rx, int &ry)
```

```
{
```

```
    int temp;
```

```
    cout << "Swap. Before swap, rx: " << rx << " ry: " << ry << "\n";
```

```
    temp = rx;
```

```
    rx = ry;
```

```
    ry = temp;
```

```
    cout << "Swap. After swap, rx: " << rx << " ry: " << ry << "\n";
```

```
}
```

```
Main. Before swap, x:5 y: 10  
Swap. Before swap, rx:5 ry:10  
Swap. After swap, rx:10 ry:5  
Main. After swap, x:10, y:5
```

# Πέρασμα πολλαπλών τιμών με δείκτες/1

```
#include <iostream.h>
typedef unsigned short USHORT;
short Factor(USHORT, USHORT*, USHORT*);
int main() {
    USHORT number, squared, cubed;    short error;
    cout << "Enter a number (0 - 20): ";    cin >> number;
    error = Factor(number, &squared, &cubed);
    if (!error) {
        cout << "number: " << number << "\n";
        cout << "square: " << squared << "\n";
        cout << "cubed: " << cubed << "\n"; }
    else
        cout << "Error encountered!!\n";
    return 0;}
```



## Πέρασμα πολλαπλών τιμών με δείκτες/2

```
short Factor(USHORT n, USHORT *pSquared, USHORT *pCubed)
{
    short Value = 0;
    if (n > 20)
        Value = 1;
    else {
        *pSquared = n*n;
        *pCubed = n*n*n;
        Value = 0;    }
    return Value; }
```

```
Enter a number (0-20): 3
number: 3
square: 9
cubed: 27
```

# Πέρασμα πολλαπλών τιμών με αναφορές/1

```
#include <iostream.h>
typedef unsigned short USHORT;
enum ERR_CODE { SUCCESS, ERROR };
ERR_CODE Factor(USHORT, USHORT&, USHORT&);
int main() {
    USHORT number, squared, cubed;    ERR_CODE result;
    cout << "Enter a number (0 - 20): ";
    cin >> number;
    result = Factor(number, squared, cubed);
}
```

## Πέρασμα πολλαπλών τιμών με αναφορές/2

```
if (result == SUCCESS) {  
    cout << "number: " << number << "\n";  
    cout << "square: " << squared << "\n";  
    cout << "cubed: " << cubed << "\n"; }  
else  
    cout << "Error encountered!!\n";  
return 0 }
```

## Πέρασμα πολλαπλών τιμών με αναφορές/3

```
ERR_CODE Factor(USHORT n, USHORT &rSquared, USHORT &rCubed) {  
    if (n > 20)  
        return ERROR;  
    else {  
        rSquared = n*n;  
        rCubed = n*n*n;  
        return SUCCESS;  
    }  
}
```

# Δείκτες σε αντικείμενα/1

```
#include <iostream.h>

class SimpleCat {
public:
    SimpleCat ();
    SimpleCat(SimpleCat&);    // copy constructor
    ~SimpleCat();    };

SimpleCat::SimpleCat() {
    cout << "Simple Cat Constructor...\n"; }

SimpleCat::SimpleCat(SimpleCat&) {
    cout << "Simple Cat Copy Constructor...\n"; }

SimpleCat::~~SimpleCat() {
    cout << "Simple Cat Destructor...\n"; }
```

## Δείκτες σε αντικείμενα/2

```
SimpleCat FunctionOne (SimpleCat theCat);  
SimpleCat* FunctionTwo (SimpleCat *theCat);  
int main() {  
    cout << "Making a cat...\n";  
    SimpleCat Frisky;  
    cout << "Calling FunctionOne...\n";  
    FunctionOne(Frisky);  
    cout << "Calling FunctionTwo...\n";  
    FunctionTwo(&Frisky);  
    return 0; }
```

1: Making a cat...  
2: Simple Cat Constructor...  
3: Calling FunctionOne...  
4: Simple Cat Copy Constructor... 5:  
Function One. Returning...  
6: Simple Cat Copy Constructor... 7:  
Simple Cat Destructor...  
8: Simple Cat Destructor...  
9: Calling FunctionTwo...  
10: Function Two. Returning...  
11: Simple Cat Destructor...



## Δείκτες σε αντικείμενα/3

**SimpleCat FunctionOne(SimpleCat theCat)**

{

cout << "Function One. Returning...\n";

return theCat;

}

**SimpleCat\* FunctionTwo (SimpleCat \*theCat)**

{

cout << "Function Two. Returning...\n";

return theCat;

}

# Αναφορές σε αντικείμενα/1

```
#include <iostream.h>

class SimpleCat {
public:
    SimpleCat();
    SimpleCat(SimpleCat&);
    ~SimpleCat();
    int GetAge() const { return itsAge; }
    void SetAge(int age) { itsAge = age; }
private:
    int itsAge;  };
```

## Αναφορές σε αντικείμενα/2

```
SimpleCat::SimpleCat() {
```

```
    cout << "Simple Cat Constructor...\n";
```

```
    itsAge = 1;    }
```

```
SimpleCat::SimpleCat(SimpleCat&) {
```

```
    cout << "Simple Cat Copy Constructor...\n";    }
```

```
SimpleCat::~~SimpleCat() {
```

```
    cout << "Simple Cat Destructor...\n";    }
```

```
const    SimpleCat & FunctionTwo (const SimpleCat & theCat);
```

## Αναφορές σε αντικείμενα/3

```
int main() {  
    cout << "Making a cat...\n";  
    SimpleCat Frisky;  
    cout << "Frisky is " << Frisky.GetAge() << " years old\n";  
    int age = 5;  
    Frisky.SetAge(age);  
    cout << "Frisky is " << Frisky.GetAge() << " years old\n";  
    cout << "Calling FunctionTwo...\n";  
    FunctionTwo(Frisky);  
    cout << "Frisky is " << Frisky.GetAge() << " years old\n";  
    return 0; }
```

Making a cat...  
Simple Cat constructor...  
Frisky is 1 years old  
Frisky is 5 years old  
Calling FunctionTwo...  
FunctionTwo.  
Returning...  
Frisky is now 5 years old  
Frisky is 5 years old  
Simple Cat Destructor...

# Αναφορές σε αντικείμενα/1

**const SimpleCat & FunctionTwo (const SimpleCat & theCat)**

```
{  
    cout << "Function Two. Returning...\n";  
    cout << "Frisky is now " << theCat.GetAge();  
    cout << " years old \n";  
    // theCat.SetAge(8);  
    return theCat;  
}
```

Making a cat...  
Simple Cat constructor...  
Frisky is 1 years old  
Frisky is 5 years old  
Calling FunctionTwo...  
FunctionTwo.  
Returning...  
Frisky is now 5 years old  
Frisky is 5 years old  
Simple Cat Destructor...

???

```
class CAT {  
    public:  
        CAT(int age) { itsAge = age; }  
        ~CAT(){}  
        int GetAge() const { return itsAge;}  
    private:  
        int itsAge; };
```



???

```
CAT & MakeCat(int age);
```

```
int main() {
```

```
    int age = 7;
```

```
    CAT Boots = MakeCat(age);
```

```
    cout << "Boots is " << Boots.GetAge() << " years old\n"; }
```

```
CAT & MakeCat(int age) {
```

```
    CAT * pCat = new CAT(age);
```

```
    return *pCat; }
```