Πανεπιστήμιο Αιγαίου

Μεθοδολογίες και Γλώσσες Προγραμματισμού Ι

Δείκτες

Εργίνα Καβαλλιεράτου (kavallieratou@aegean.gr)

Μόνιμη Επίκουρος Καθηγήτρια

Τμήμα Μηχανικών Πληροφοριακών & Επικοινωνιακών Συστημάτων

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αιγαίου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημερινό Μάθημα!

- ✓ Χρήση pointer
- ✓ Τελεστής *
- ✓ Τελεστής &
- ✓ Γενικοί δείκτες
- ✓ Ανάκληση
- ✓ Δέσμευση μνήμης
- ✓ new / delete
- ✓ Pointer σε αντικείμενο

Μνήμη μεταβλητών

- ✓ Κάθε μεταβλητή έχει διεύθυνση
- ✓ Δεν χρειάζεται να γνωρίζουμε τον αριθμό της διεύθυνσης αλλά πολλές φορές χρειάζεται να πούμε στο compiler πόση μνήμη θα χρειαστεί.

  Π.χ. ένας long integer χρειάζεται 4 bytes
- ✓ Ακόμα και αν δεν γνωρίζουμε τη διεύθυνση μπορούμε να την αποθηκεύσουμε σε ένα pointer

Χρήση pointer

int *pAge = 0;

- ✓ Ο pAge είναι δείκτης σε ακέραιο δηλαδή θα έχει τη διεύθυνση μιας θέσης μνήμης αρκετά μεγάλης για να αποθηκεύσει ακέραιο.
- ✓ Όταν η διεύθυνση ενός δείκτη είναι 0 κατά σύμβαση ο δείκτης είναι null.
- ✓ Καλό είναι οι δείκτες να αρχικοποιούνται κατά τη δήλωση τους:

```
unsigned short int howOld = 50;
unsigned short int * pAge = &howOld;
```

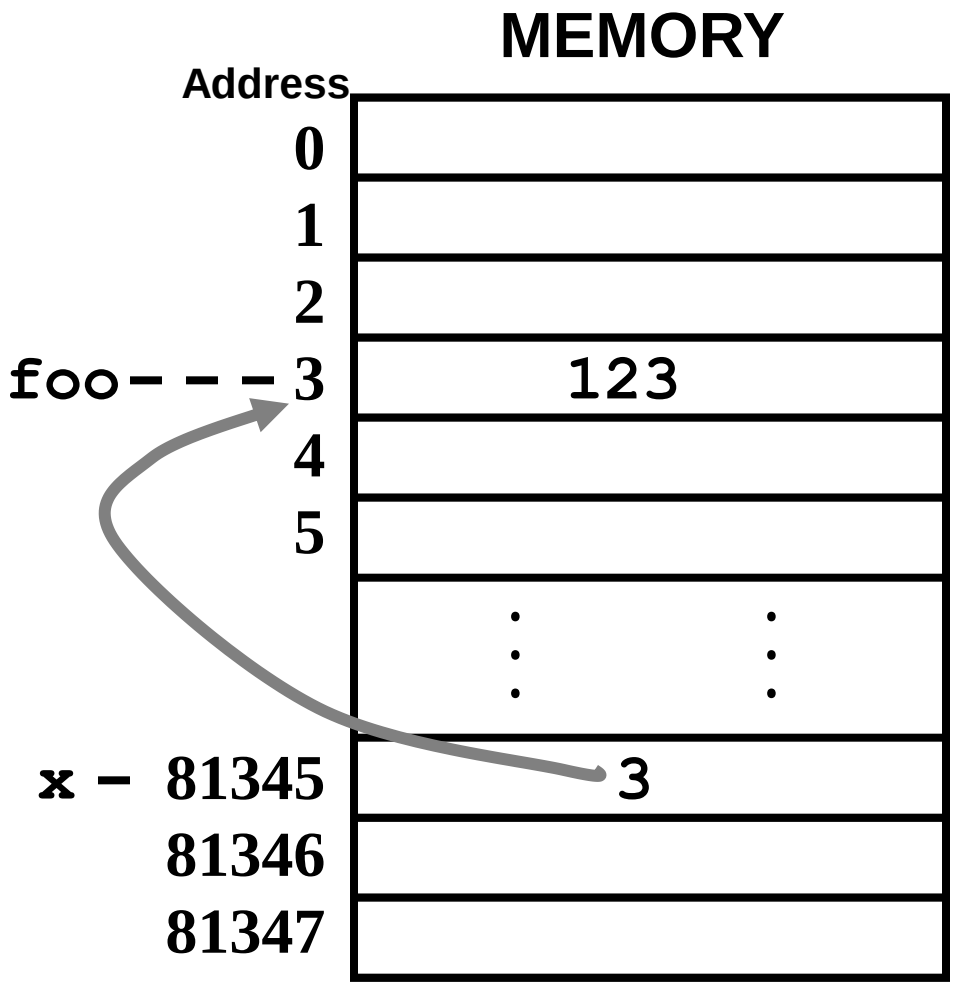
- ✓ Ο δείκτης παρέχει έμμεση πρόσβαση στη μεταβλητή, της οποίας τη διεύθυνση περιέχει:

```
unsigned short int yourAge;
yourAge = *pAge;
```

Δείκτες (Pointers)

```
int foo;
int *x;

foo = 123;
x = &foo;
```



Τελεστής *

- ✓Ο τελεστής * έχει δύο χρήσεις:

```
unsigned short * pAge = 0;

*pAge = 5;
```

Τελεστής &

- Για να πάρουμε τη διεύθυνση μιας μεταβλητής χρησιμοποιούμε τον τελεστή &

```
int a = 3, *ptr=0;
ptr = &a;
cout << "The address of variable a is " << (int)ptr << "
  and it contains " << *ptr << endl;
```

- Ο τύπος του δείκτη ορίζει το πλήθος των bytes που απαιτεί η μεταβλητή ΌΧΙ ο δείκτης.

Pointer – διεύθυνση –τιμή μεταβλητής

```
int theVariable = 5;
int * pPointer = &theVariable ;
```

Παράδειγμα

```
#include <iostream.h>:
 typedef unsigned short int USHORT;
 int main()  {
    USHORT myAge;         // a variable
    USHORT * pAge = 0;    // a pointer
    myAge = 5;
    cout << "myAge: " << myAge << "\n";
    pAge = &myAge;
    cout << "*pAge: " << *pAge << "\n\n";
     cout << "*pAge = 7\n";
    *pAge = 7;
    cout << "*pAge: " << *pAge << "\n";
    cout << "myAge: " << myAge << "\n\n";
    cout << "myAge = 9\n";
    myAge = 9;
    cout << "myAge: " << myAge << "\n";
    cout << "*pAge: " << *pAge << "\n";
  return 0; }
```

```
myAge: 5
*pAge: 5

*pAge = 7
*pAge: 7
myAge: 7

myAge = 9
myAge: 9
*pAge: 9
```

Γενικοί δείκτες: void*

- ✓ Δήλωση: **void * ptr;**
- ✓ Μπορεί να δείξει σε οποιοδήποτε τύπο

  void * ptr = GetAddress();

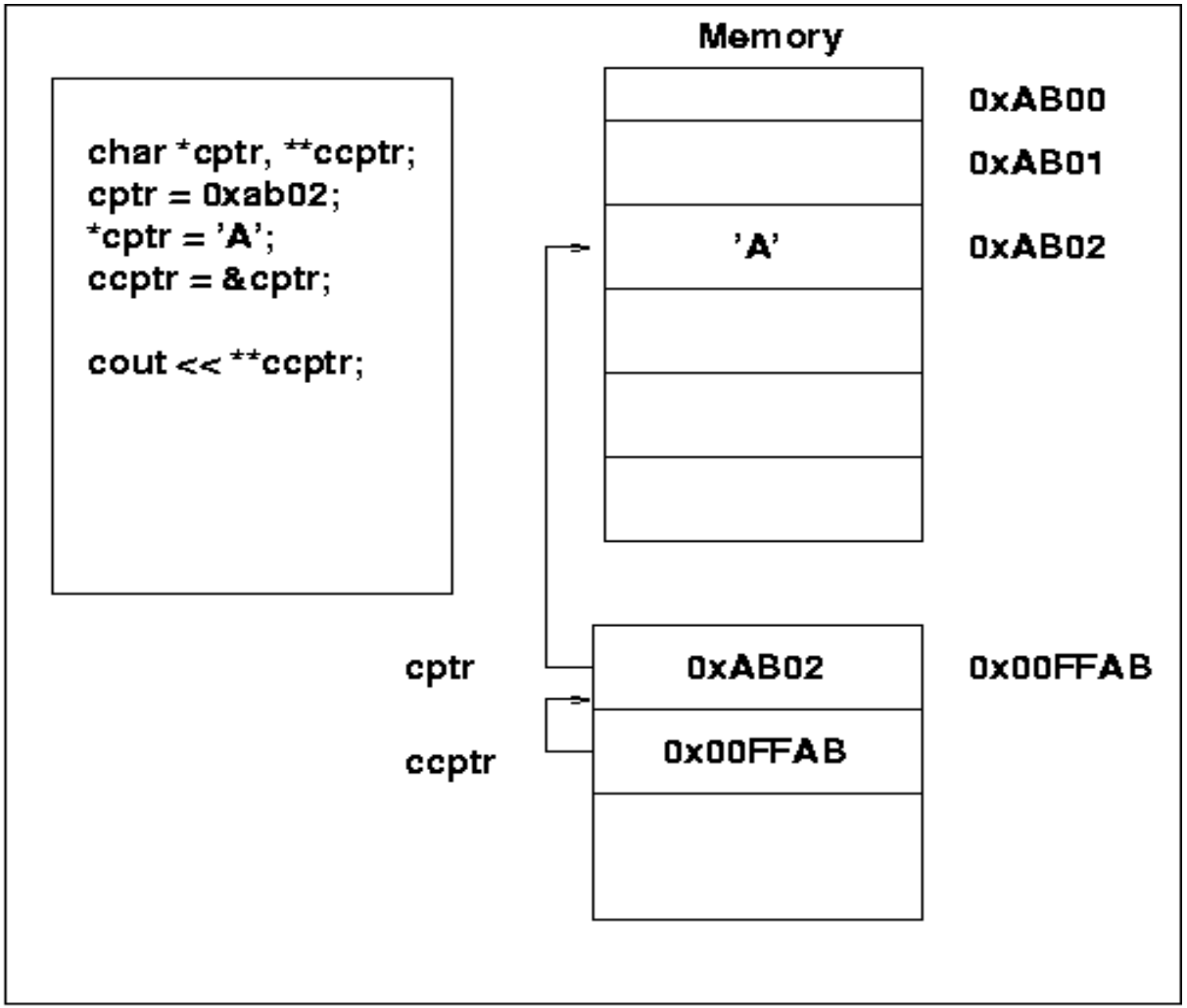
  float *fptr = (float*)ptr;
- ✓ Οι συναρτήσεις μπορούν να λάβουν και να επιστρέψουν δείκτες void
- ✓ ΟΛΟΙ οι δείκτες έχουν το ίδιο μέγεθος

Ανάκληση

- ✓ Μπορεί να χρησιμοποιηθεί δείκτης που δείχνει σε δείκτη
- ✓ Λέγεται Ανάκληση

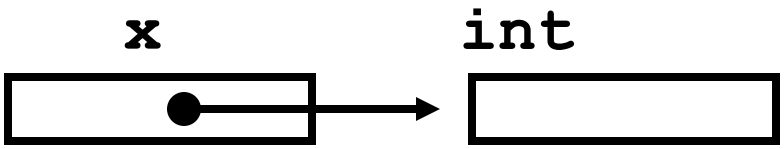
  char **ccptr; // pointer to char pointer

Ανάκληση

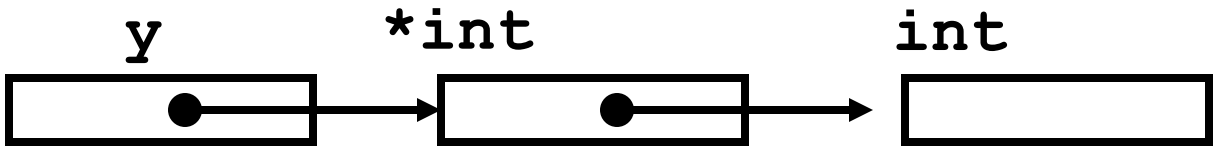

Memory
char *cptr, **ccptr;
cptr = 0xab02;
*cptr = 'A';
ccptr = &cptr;
cout << **ccptr;
0xAB00
0xAB01
'A'
0xAB02
cptr
ccptr
0xAB02
0x00FFAB
0x00FFAB

Ανάκληση

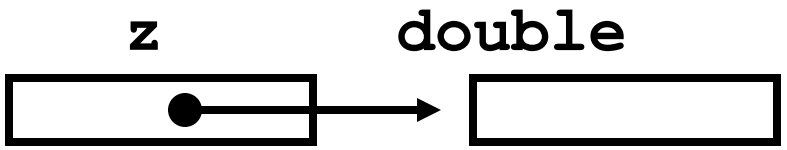
```
int *x;
```




```
int **y;
```




```
double *z;
```



Χρησιμότητα pointer

- ✓ Διαχείριση δεδομένων & μνήμης (δέσμευση μνήμης)
- ✓ Πρόσβαση στα δεδομένα-μέλη κλάσεων και συναρτήσεων
- ✓ Πέρασμα με αναφορά σε συναρτήσεις

Δέσμευση μνήμης

- ✓ Για να διαχειριστούμε σωστά τα δεδομένα και να δεσμεύσουμε μνήμη όπου χρειάζεται χρησιμοποιούμε την εντολή new:

```
unsigned short int * pPointer;
pPointer = new unsigned short int;
```

Ή

```
unsigned short int * pPointer = new unsigned short int;
```

- ✓ Το new ακολουθείται από τον τύπο του αντικειμένου που θέλουμε να δεσμεύσουμε
- ✓ Η διεύθυνση αποδίδεται σε ένα δείκτη
- ✓ Κάθε φορά που αιτούμαστε για μνήμη πρέπει να ελέγχουμε το δείκτη για null (αποτυχία στη δέσμευση)

delete

- ✓ Όταν δεν μας χρειάζεται η δεσμευμένη μνήμη την απελευθερώνουμε με delete:

  delete pPointer;

- ✓ Αν ξανακαλέσουμε το delete στον ίδιο δείκτη μπορεί να προκαλέσει crash
- ✓ Για τον ίδιο λόγο καλό είναι μετά το delete να αποδίδουμε στο δείκτη τιμή 0(null)
- ✓ Crash θα προκαλέσει και η προσπάθεια να αποδώσουμε εκ νέου μνήμη σε διαγραμμένο δείκτη.
- ✓ ΠΡΟΣΟΧΗ: το σωστό είναι για κάθε new να υπάρχει ένα delete

Παράδειγμα/1

```
#include <iostream.h>
int main()
{
   int localVariable = 5;
   int * pLocal= &localVariable;
   int * pHeap = new int;
   if (pHeap == NULL)
    {
        cout << "Error! No memory for pHeap!!";
        return 0;
    }
   *pHeap = 7;
   cout << "localVariable: " << localVariable << "\n";
   cout << "*pLocal: " << *pLocal << "\n";
```

localVariable: 5

*pLocal: 5

Παράδειγμα/2

```
    cout << "*pHeap: " << *pHeap << "\n";
    delete pHeap;
    pHeap = new int;
    if (pHeap == NULL)
    {
        cout << "Error! No memory for pHeap!!";
        return 0;
    }
    *pHeap = 9;
    cout << "*pHeap: " << *pHeap << "\n";
    delete pHeap;
    return 0;
}
```

```
localVariable: 5
*pLocal: 5
*pHeap: 7
*pHeap: 9
```

Pointer σε αντικείμενο

Cat *pCat = new Cat;

- ✓ Γίνεται δέσμευση για όσο χώρο απαιτεί η συγκεκριμένη κλάση

Παράδειγμα/1

```
#include <iostream.h>
class SimpleCat
{
public:
      SimpleCat();
        ~SimpleCat();
 private:
       int itsAge;
 };

   SimpleCat::SimpleCat()
   {
        cout << "Constructor called.\n";
        itsAge = 1;
   }
```

Παράδειγμα/2

```
SimpleCat::~SimpleCat()
    {
        cout << "Destructor called.\n";
    }
    int main()
    {
        cout << "SimpleCat Frisky...\n";
        SimpleCat Frisky;
        cout << "SimpleCat *pRags = new SimpleCat...\n";
        SimpleCat * pRags = new SimpleCat;
        cout << "delete pRags...\n";
        delete pRags;
        cout << "Exiting, watch Frisky go...\n";
    return 0;
}
```

```
SimpleCat Frisky...
Constructor called.
SimpleCat *pRags = new SimpleCat..
Constructor called.
delete pRags...
Destructor called.
Exiting, watch Frisky go...
Destructor called.
```

Παράδειγμα 1.1

```
#include <iostream>
using namespace std;
int main ()
{
  int firstvalue, secondvalue;
  int * mypointer;
  mypointer = &firstvalue;
  *mypointer = 10;
  mypointer = &secondvalue;
  *mypointer = 20;
  cout << "firstvalue is " << firstvalue << '\n';
  cout << "secondvalue is " << secondvalue << '\n';
  return 0;
}
```

Παράδειγμα 2.1

```
#include <iostream>
int main (){
  int firstvalue = 5, secondvalue = 15;
  int * p1, * p2;
  p1 = &firstvalue;
  p2 = &secondvalue;
  *p1 = 10;
  *p2 = *p1;
  p1 = p2;
  *p1 = 20;
  cout << "firstvalue is " << firstvalue << '\n';
  cout << "secondvalue is " << secondvalue << '\n';
  return 0;
}
```

Παράδειγμα 3.1

```
#include <iostream>
int main ()
{
  int numbers[5];
  int * p;
  p = numbers;  *p = 10;
  p++;  *p = 20;
  p = &numbers[2];  *p = 30;
  p = numbers + 3;  *p = 40;
  p = numbers;  *(p+4) = 50;
  for (int n=0; n<5; n++)
    cout << numbers[n] << ", ";
  return 0;
}
```