Διαδικαστικός Προγραμματισμός

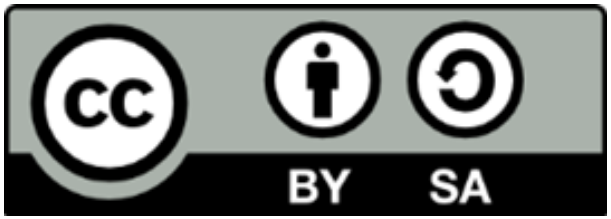
Ενότητα 7: Δείκτες

Καθηγήτρια Μαρία Σατρατζέμη

Τμήμα Εφαρμοσμένης Πληροφορικής

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 Μακεδονίας» 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.

Σκοποί ενότητας

- Να κατανοήσετε τη δυνατότητα χρήσης διευθύνσεων ως τιμών δεδομένων.
- Να μπορείτε να χρησιμοποιείτε τους τελεστές δεικτών & και *.
- Να μπορείτε να κάνετε κλήσεις κατ' αναφορά ώστε να είναι δυνατή η κοινή χρήση δεδομένων μεταξύ μιας συνάρτησης και του καλούντος της.

Ο σκοπός των δεικτών

Ένας **δείκτης** είναι ένα στοιχείο δεδομένων του οποίου η τιμή είναι η διεύθυνση μιας θέσης μνήμης στην οποία (θέση μνήμης) είναι αποθηκευμένη κάποια άλλη τιμή.

- Οι δείκτες σας επιτρέπουν να αναφέρεστε σε μεγάλες δομές δεδομένων με συμπαγή τρόπο  χρησιμοποιούμε τη διεύθυνση ως συντόμευση για την πλήρη τιμή

- Οι δείκτες κάνουν εφικτή την κοινή χρήση δεδομένων μεταξύ διαφορετικών τμημάτων ενός προγράμματος.

- Οι δείκτες κάνουν δυνατή τη δέσμευση νέων χώρων μνήμης κατά τη διάρκεια της εκτέλεσης ενός προγράμματος.

Η έννοια της “αριστερής τιμής”

Στη C, οποιαδήποτε παράσταση αναφέρεται σε κάποια εσωτερική θέση της μνήμης στην οποία είναι δυνατή η αποθήκευση δεδομένων ονομάζεται **“αριστερή τιμή”** (lvalue).

Οι “αριστερές τιμές” μπορούν να εμφανίζονται στην αριστερή πλευρά μιας εντολής ανάθεσης της C.

Παραδείγματα αριστερών τιμών:

- Οι απλές μεταβλητές: x = 10;
- Οι παραστάσεις επιλογής σε ένα πίνακα: intarray[2] = 20;

Παραδείγματα μη αριστερών τιμών:

- Σταθερές
- Αριθμητικές παραστάσεις

Βασικές αρχές για τις “αριστερές τιμές”

- Κάθε “αριστερή τιμή” είναι αποθηκευμένη κάπου στη μνήμη και, κατά συνέπεια, **έχει μια διεύθυνση**.
- Μετά τη δήλωση μιας “αριστερής τιμής”, **η διεύθυνσή της δεν αλλάζει ποτέ**, παρόλο που τα περιεχόμενά της αλλάζουν.
- Ανάλογα με τον τύπο των δεδομένων που περιέχουν, διαφορετικές “αριστερές τιμές” **απαιτούν διαφορετικές ποσότητες μνήμης**.
- **Η διεύθυνση μιας “αριστερής τιμής” αποτελεί και η ίδια στοιχείο δεδομένων**, και μπορείτε να τη χειριστείτε και να την αποθηκεύσετε στη μνήμη.

Χειρισμός δεικτών στη C

Σε μια μεταβλητή δείκτη μπορούμε να αποθηκεύουμε τιμές διευθύνσεων ή μπορούμε να μεταβιβάζουμε μεταβλητές δείκτη ως παραμέτρους σε διευθύνσεις.

Η δήλωση μιας μεταβλητής δείκτη στη C έχει την εξής μορφή:

τύπος_βάσης *μεταβλητή_δείκτη;

όπου:

τύπος_βασης είναι ο τύπος της τιμής στην οποία δείχνει ο δείκτης

μεταβλητή_δείκτη είναι η μεταβλητή που δηλώνεται

Παραδείγματα:

int ***p1**, p2;

η μεταβλητή **p1** δηλώνεται ως **μεταβλητή δείκτη** ή ως δείκτης προς έναν ακέραιο

η μεταβλητή p2 δηλώνεται ως ακέραια μεταβλητή

Βασικές πράξεις δεικτών (1)

Στη C ορίζονται δύο τελεστές για το χειρισμό τιμών δεικτών:

& Διεύθυνση του

***** Τιμή στην οποία δείχνει

Βασικές πράξεις δεικτών (2)

- Ο τελεστής **&** δέχεται ως **τελεστέο** μια παράσταση η οποία αντιστοιχεί σε κάποια τιμή που είναι αποθηκευμένη στη μνήμη, συνήθως μια **μεταβλητή** ή **αναφορά σε πίνακα**.

- Ο τελεστέος γράφεται μετά το σύμβολο **&** και πρέπει να είναι κάποια “αριστερή τιμή”.

- Ο τελεστής **& επιστρέφει τη διεύθυνση της μνήμης** στην οποία είναι αποθηκευμένη η συγκεκριμένη “αριστερή τιμή”.

Βασικές πράξεις δεικτών (3)

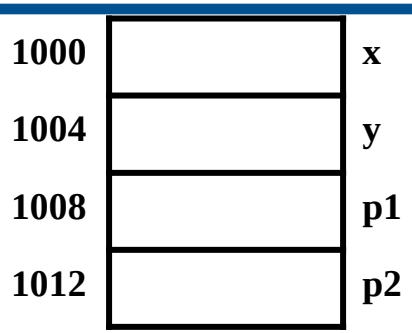
- Ο τελεστής * δέχεται μια τιμή οποιουδήποτε τύπου δείκτη και επιστρέφει την “αριστερή τιμή” στην οποία δείχνει.

- Αυτή η πράξη ονομάζεται **αποαναφοροποίηση** το δείκτη.

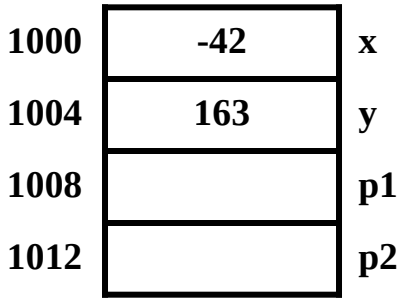
- Η πράξη * παράγει μια “αριστερή τιμή”, γεγονός που σημαίνει ότι σε έναν “αποαναφοροποιημένο δείκτη” μπορείτε να αναθέσετε μια τιμή.

Βασικές πράξεις δεικτών – παράδειγμα (1)

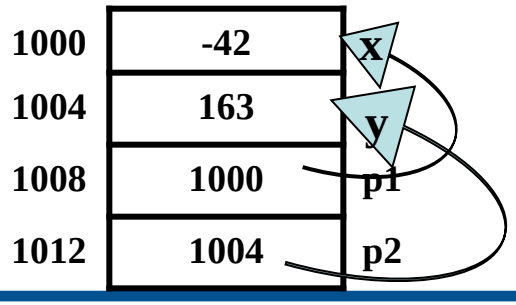
```
int x, y;
int *p1, *p2;
```




```
x = -42;
y = 163;
```



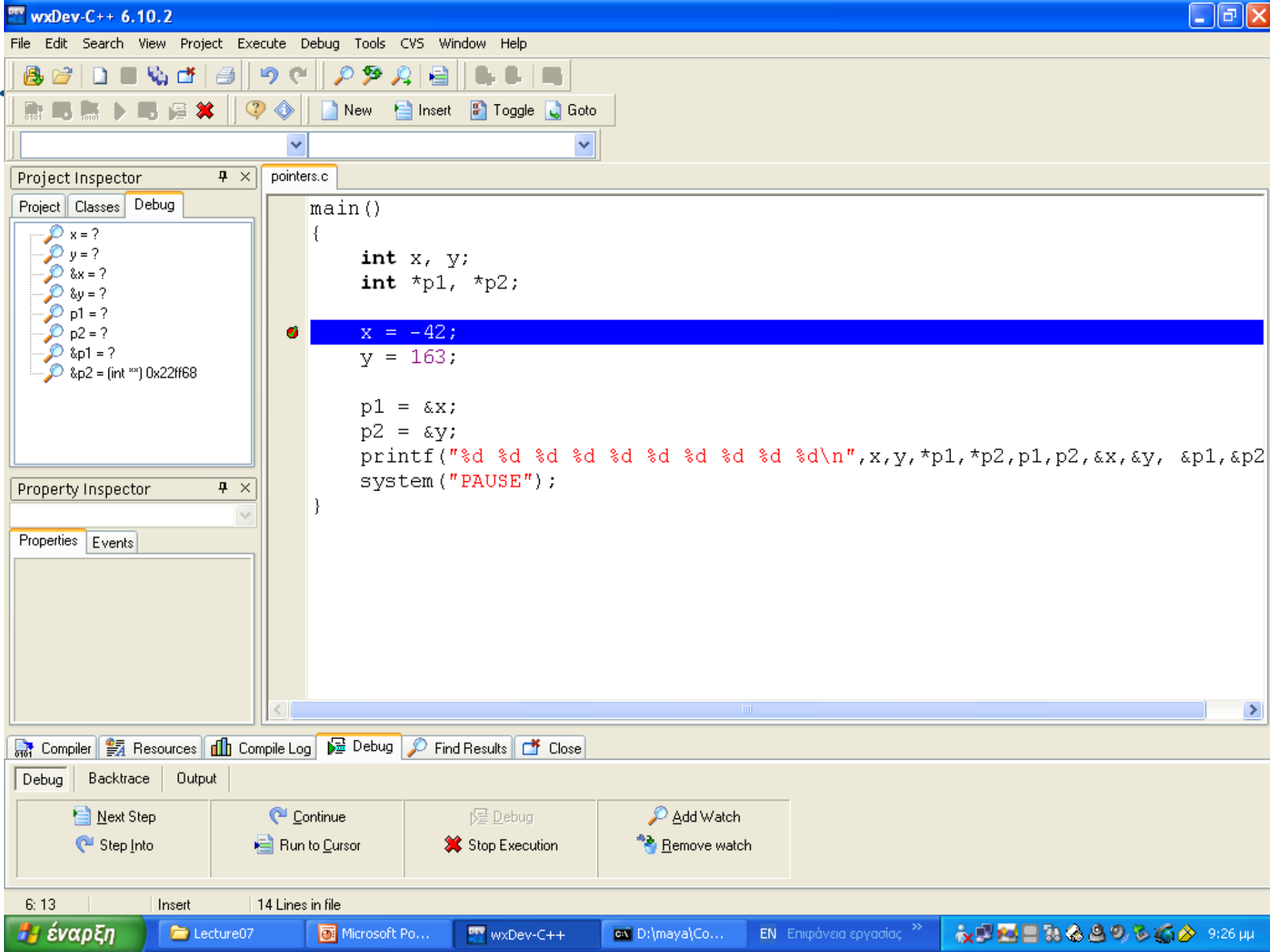

```
p1 = &x;
p2 = &y;
```

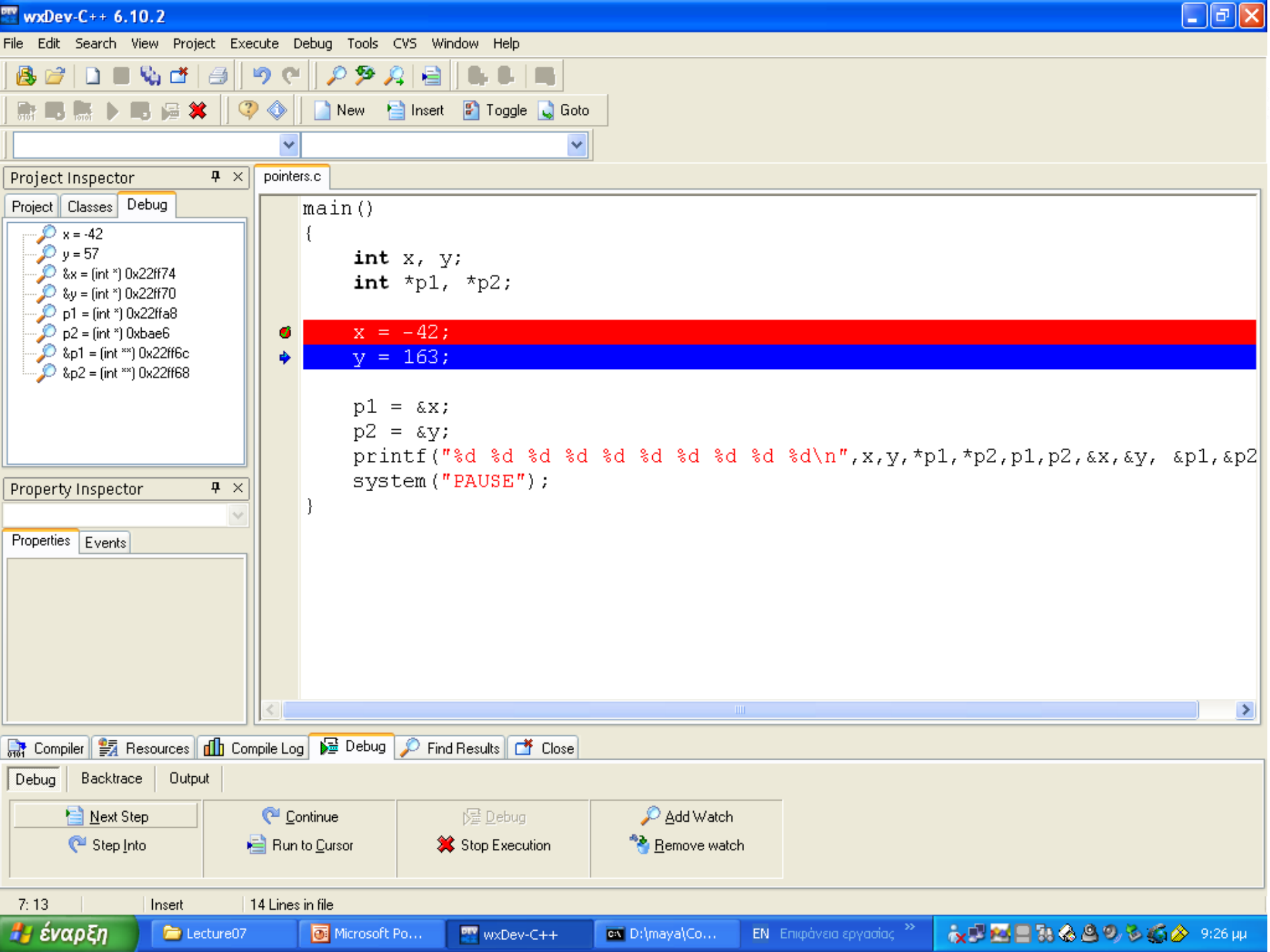


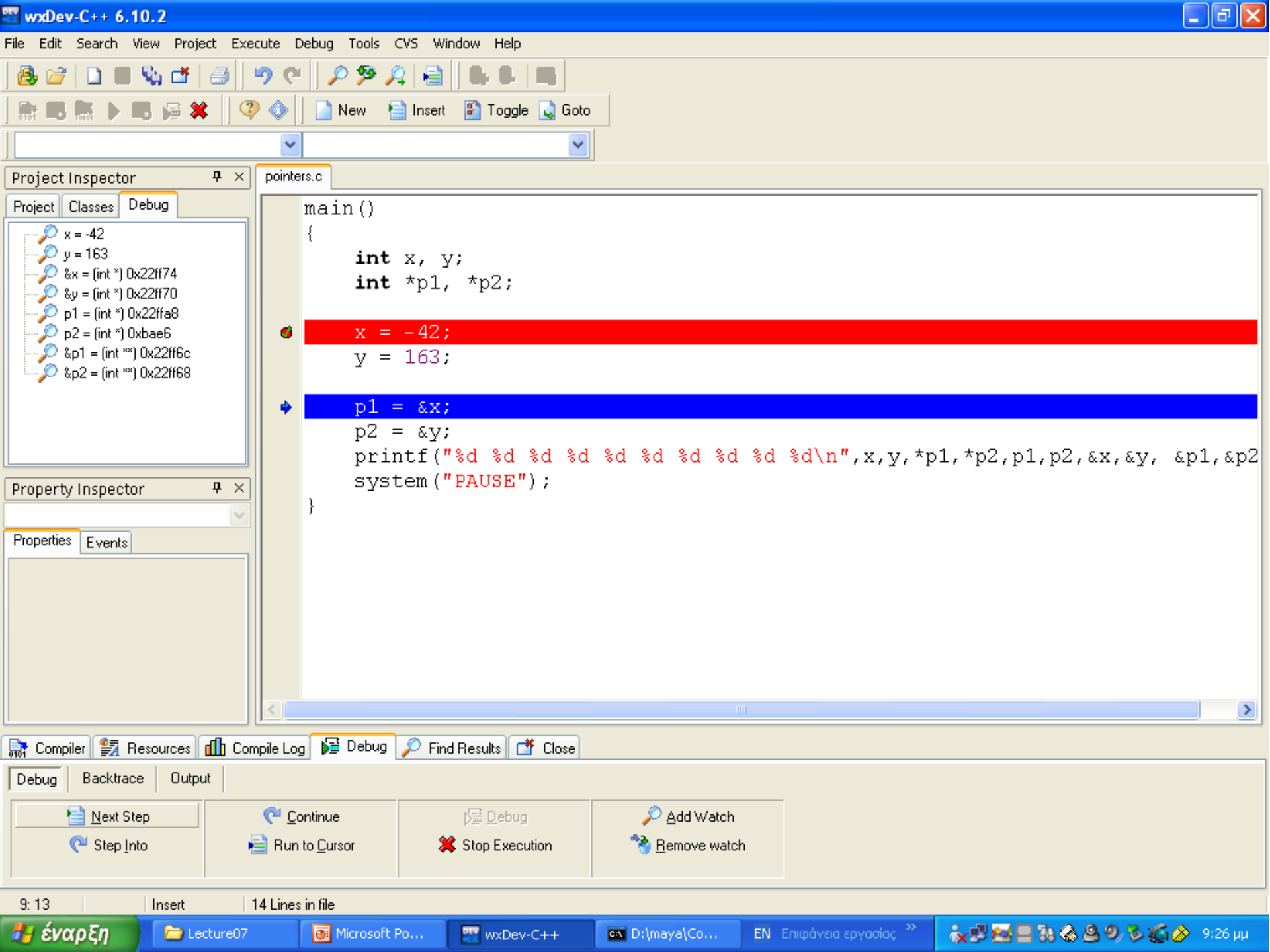
Ο τελεστής **&** δέχεται ως τελεστέο μια παράσταση η οποία αντιστοιχεί σε κάποια τιμή που είναι αποθηκευμένη στη μνήμη, συνήθως μια μεταβλητή ή αναφορά σε πίνακα.

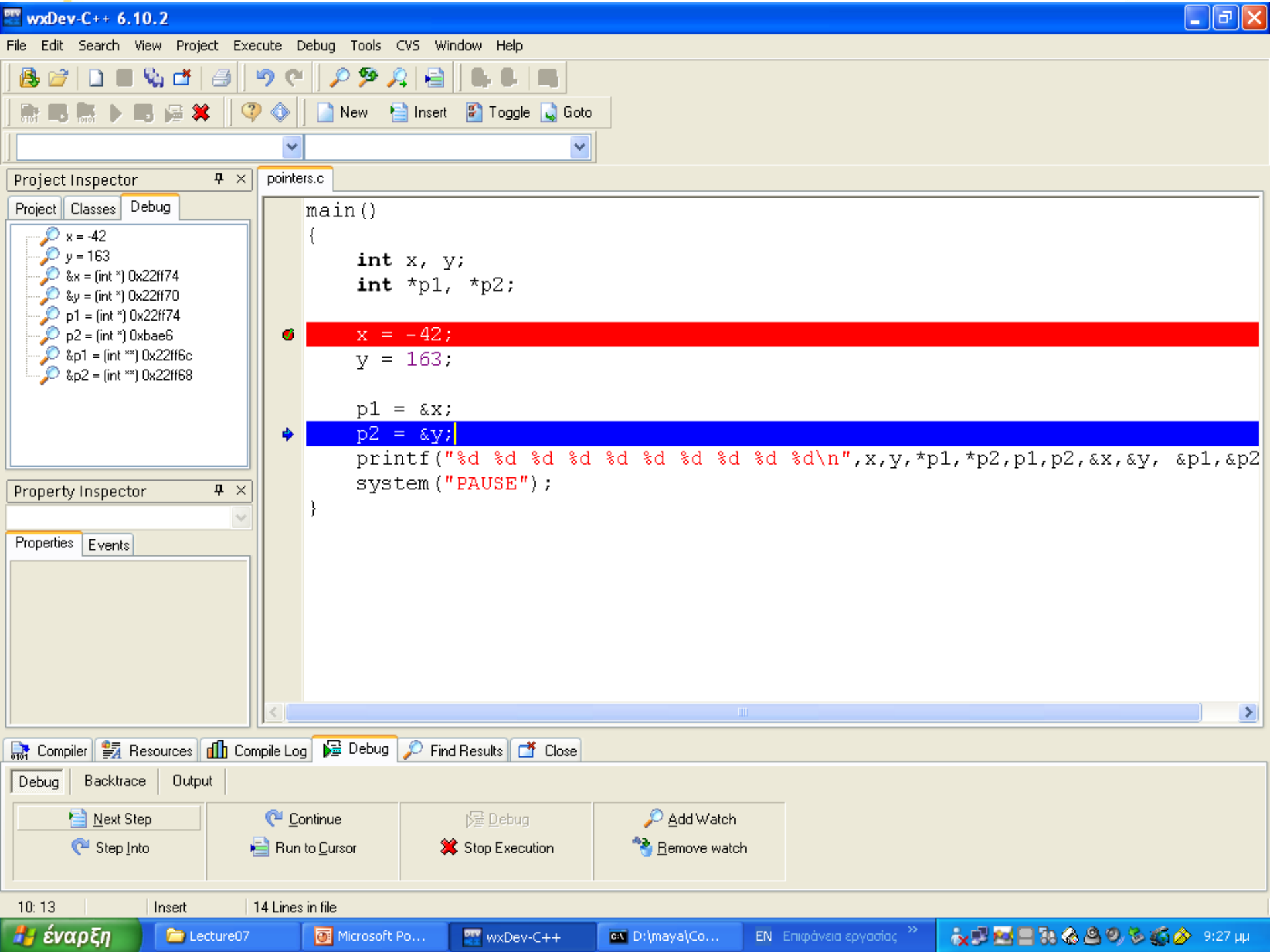
Ο τελεστέος γράφεται μετά το σύμβολο **&** και πρέπει να είναι κάποια “αριστερή τιμή”.

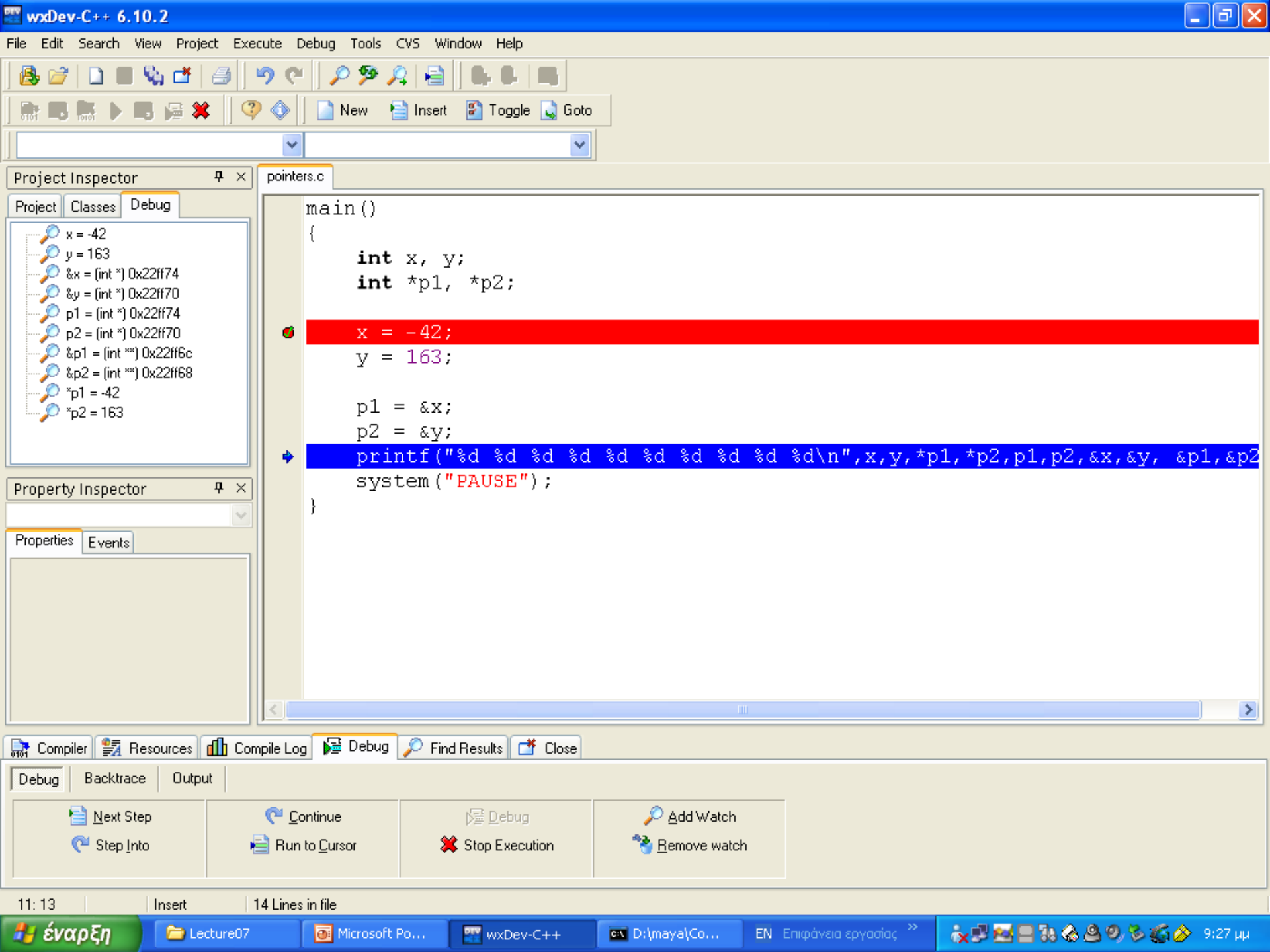
Ο τελεστής **&** επιστρέφει τη διεύθυνση της μνήμης στην οποία είναι αποθηκευμένη η συγκεκριμένη “αριστερή τιμή”.


wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = ?
y = ?
&x = ?
&y = ?
p1 = ?
p2 = ?
&p1 = ?
&p2 = (int **) 0x22ff68
Property Inspector
Properties Events
pointers.c
main()
{
int x, y;
int *p1, *p2;
x = -42;
y = 163;
p1 = &x;
p2 = &y;
printf("%d %d %d %d %d %d %d %d %d %d\n",x,y,*p1,*p2,p1,p2,&x,&y, &p1,&p2
system("PAUSE");
}
Compiler Resources Compile Log Debug Find Results Close
Debug Backtrace Output
Next Step
Step Into
Continue
Run to Cursor
Debug
Stop Execution
Add Watch
Remove watch
6: 13
Insert
14 Lines in file
έναρξη
Lecture07
Microsoft Po...
wxDev-C++
D:\maya\Co...
EN Επιφάνεια εργασίας
9:26 μμ


wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = -42
y = 57
&x = (int *) 0x22ff74
&y = (int *) 0x22ff70
p1 = (int *) 0x22ffa8
p2 = (int *) 0xbae6
&p1 = (int **) 0x22ff6c
&p2 = (int **) 0x22ff68
Property Inspector
Properties Events
pointers.c
main()
{
int x, y;
int *p1, *p2;
x = -42;
y = 163;
p1 = &x;
p2 = &y;
printf("%d %d %d %d %d %d %d %d %d %d\n",x,y,*p1,*p2,p1,p2,&x,&y, &p1,&p2
system("PAUSE");
}
Compiler Resources Compile Log Debug Find Results Close
Debug Backtrace Output
Next Step
Step Into
Continue
Run to Cursor
Debug
Stop Execution
Add Watch
Remove watch
7: 13
Insert
14 Lines in file
έναρξη
Lecture07
Microsoft Po...
wxDev-C++
D:\maya\Co...
EN Επιφάνεια εργασίας
9:26 μμ

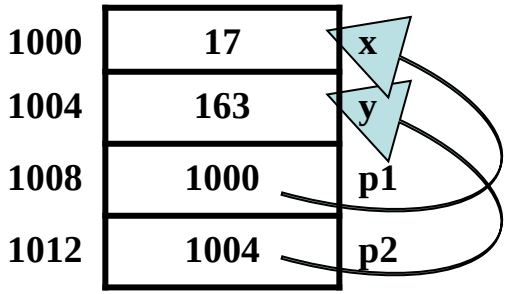

wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = -42
y = 163
&x = (int *) 0x22ff74
&y = (int *) 0x22ff70
p1 = (int *) 0x22ffa8
p2 = (int *) 0xbae6
&p1 = (int **) 0x22ff6c
&p2 = (int **) 0x22ff68
Property Inspector
Properties Events
pointers.c
main()
{
int x, y;
int *p1, *p2;
x = -42;
y = 163;
p1 = &x;
p2 = &y;
printf("%d %d %d %d %d %d %d %d %d %d\n",x,y,*p1,*p2,p1,p2,&x,&y, &p1,&p2
system("PAUSE");
}
Compiler Resources Compile Log Debug Find Results Close
Debug Backtrace Output
Next Step
Step Into
Continue
Run to Cursor
Debug
Stop Execution
Add Watch
Remove watch
9: 13
Insert
14 Lines in file
έναρξη
Lecture07
Microsoft Po...
wxDev-C++
D:\maya\Co...
EN Επιφάνεια εργασίας
9:26 μμ


wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = -42
y = 163
&x = (int *) 0x22ff74
&y = (int *) 0x22ff70
p1 = (int *) 0x22ff74
p2 = (int *) 0xbae6
&p1 = (int **) 0x22ff6c
&p2 = (int **) 0x22ff68
Property Inspector
Properties Events
pointers.c
main()
{
int x, y;
int *p1, *p2;
x = -42;
y = 163;
p1 = &x;
p2 = &y;
printf("%d %d %d %d %d %d %d %d %d %d\n",x,y,*p1,*p2,p1,p2,&x,&y, &p1,&p2
system("PAUSE");
}
Compiler Resources Compile Log Debug Find Results Close
Debug Backtrace Output
Next Step
Step Into
Continue
Run to Cursor
Debug
Stop Execution
Add Watch
Remove watch
10: 13
Insert
14 Lines in file
έναρξη
Lecture07
Microsoft Po...
wxDev-C++
D:\maya\Co...
EN Επιφάνεια εργασίας
9:27 μμ

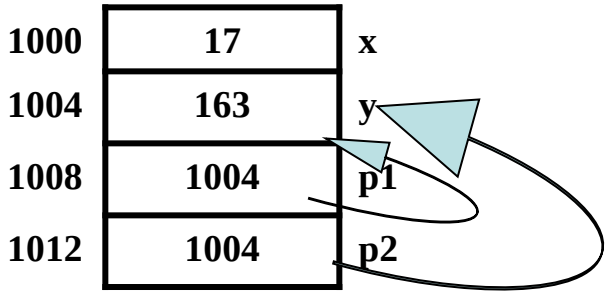

wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = -42
y = 163
&x = (int *) 0x22ff74
&y = (int *) 0x22ff70
p1 = (int *) 0x22ff74
p2 = (int *) 0x22ff70
&p1 = (int **) 0x22ff6c
&p2 = (int **) 0x22ff68
*p1 = -42
*p2 = 163
Property Inspector
Properties Events
pointers.c
main()
{
int x, y;
int *p1, *p2;
x = -42;
y = 163;
p1 = &x;
p2 = &y;
printf("%d %d %d %d %d %d %d %d %d %d\n",x,y,*p1,*p2,p1,p2,&x,&y, &p1,&p2
system("PAUSE");
}
Compiler Resources Compile Log Debug Find Results Close
Debug Backtrace Output
Next Step
Step Into
Continue
Run to Cursor
Debug
Stop Execution
Add Watch
Remove watch
11: 13
Insert
14 Lines in file
έναρξη
Lecture07
Microsoft Po...
wxDev-C++
D:\maya\Co...
EN Επιφάνεια εργασίας
9:27 μμ

Βασικές πράξεις δεικτών - παράδειγμα (2)

```
*p1 = 17;
```




```
p1 = p2;
```



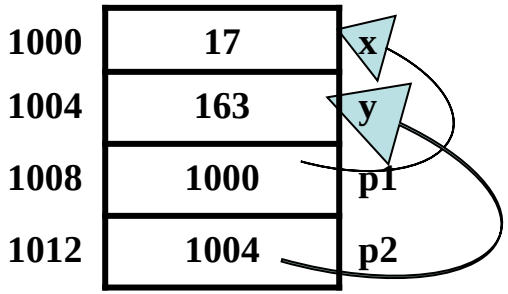
Ο τελεστής ***** δέχεται μια τιμή οποιουδήποτε τύπου δείκτη και επιστρέφει την “αριστερή τιμή” στην οποία δείχνει.

Αυτή η πράξη ονομάζεται **αποαναφοροποίηση** το δείκτη.

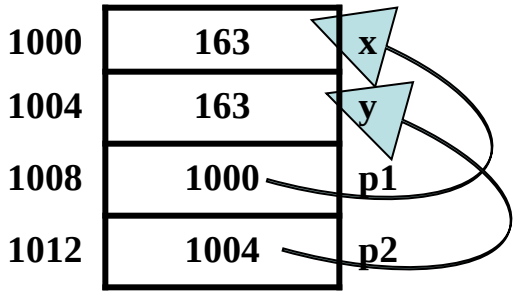
Η πράξη ***** παράγει μια “αριστερή τιμή”, γεγονός που σημαίνει ότι σε έναν “αποαναφοροποιημένο δείκτη” μπορείτε να αναθέσετε μια τιμή.

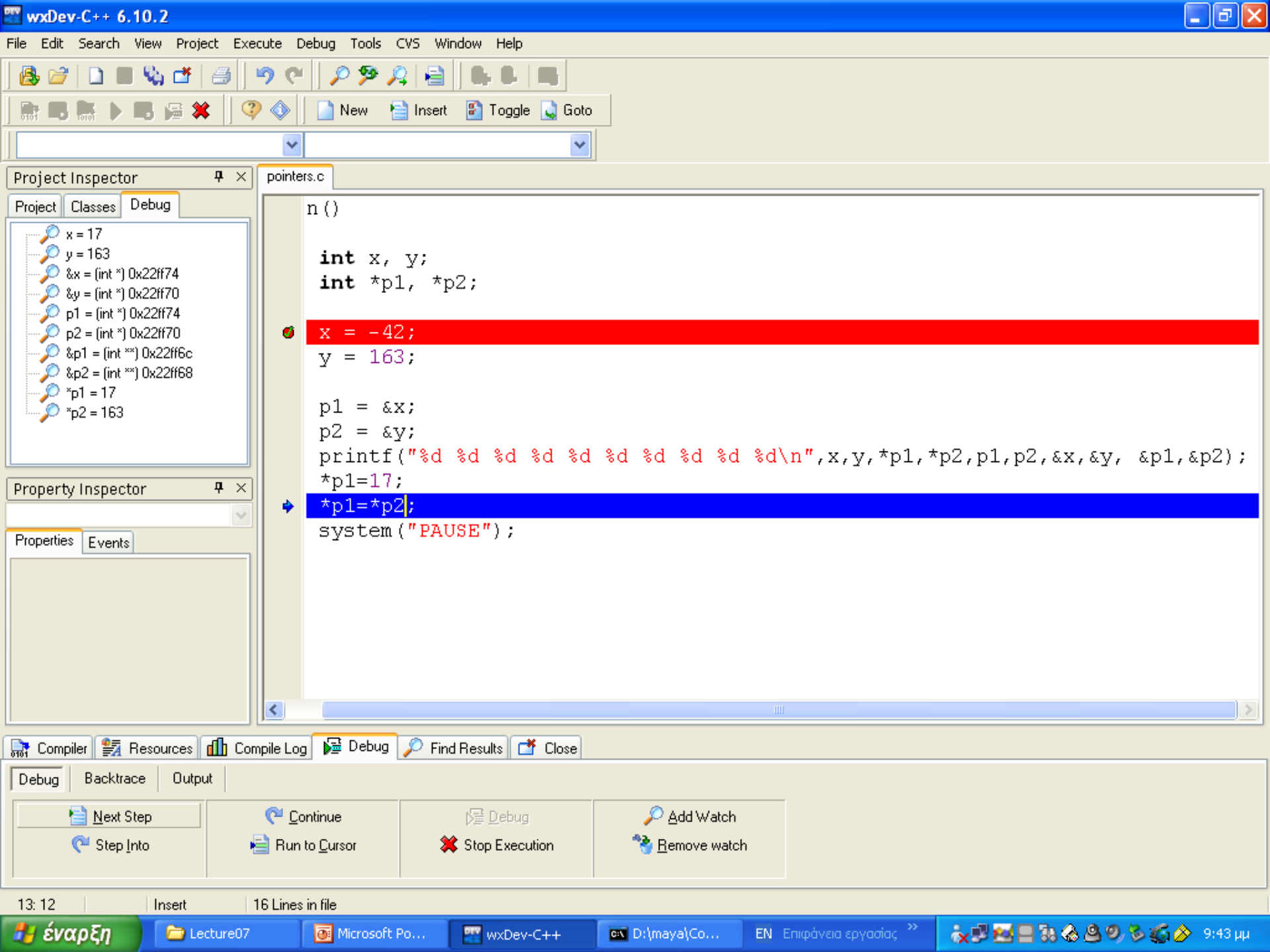
Βασικές πράξεις δεικτών - παράδειγμα (3)

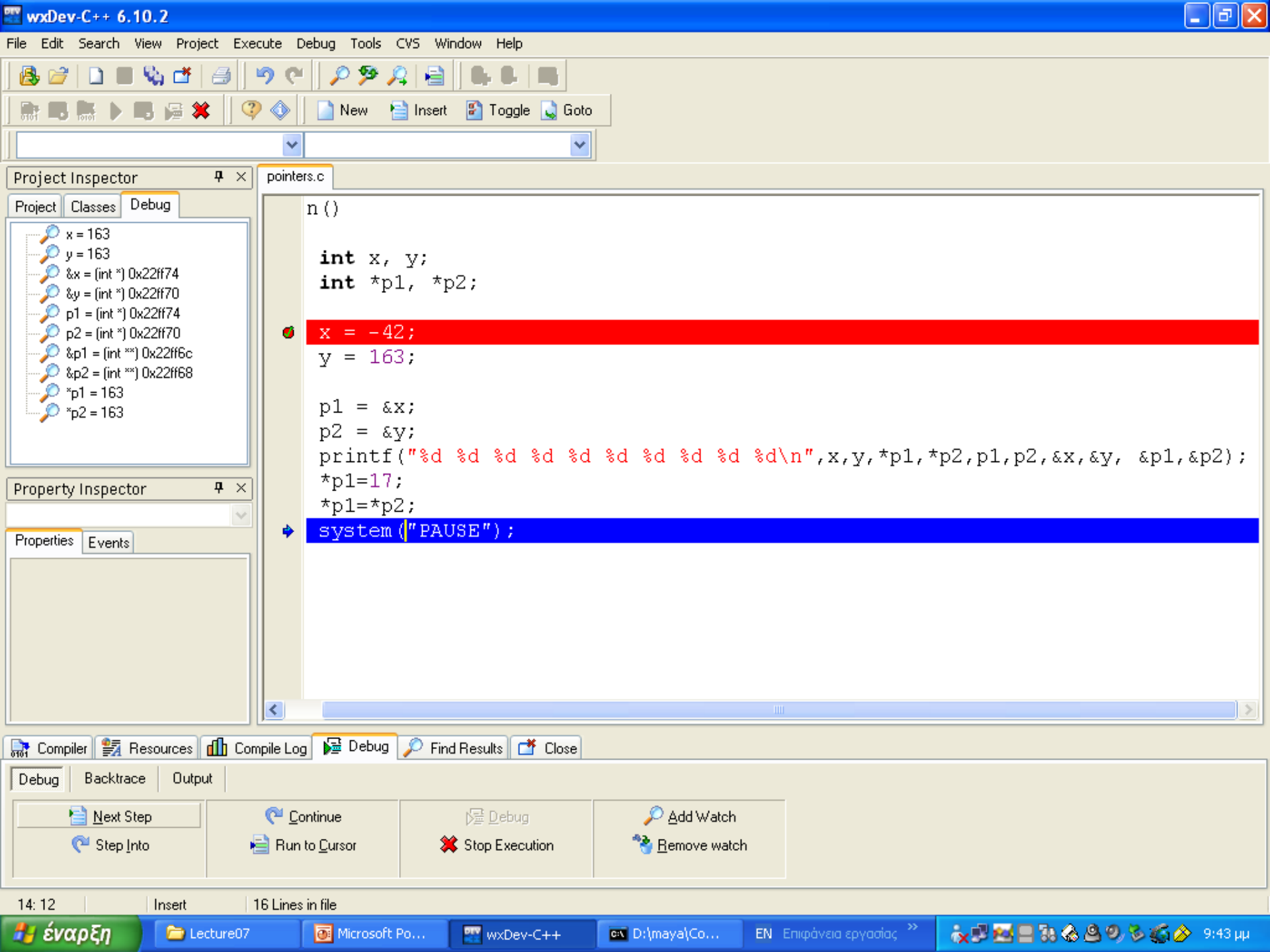
```
*p1 = 17;
```




```
*p1 = *p2;
```




wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = 17
y = 163
&x = (int *) 0x22ff74
&y = (int *) 0x22ff70
p1 = (int *) 0x22ff74
p2 = (int *) 0x22ff70
&p1 = (int **) 0x22ff6c
&p2 = (int **) 0x22ff68
*p1 = 17
*p2 = 163
Property Inspector
Properties Events
pointers.c
n()
int x, y;
int *p1, *p2;
x = -42;
y = 163;
p1 = &x;
p2 = &y;
printf("%d %d %d %d %d %d %d %d %d %d\n",x,y,*p1,*p2,p1,p2,&x,&y, &p1,&p2);
*p1=17;
*p1=*p2;
system("PAUSE");
Compiler Resources Compile Log Debug Find Results Close
Debug Backtrace Output
Next Step
Step Into
Continue
Run to Cursor
Debug
Stop Execution
Add Watch
Remove watch
13: 12
Insert
16 Lines in file
έναρξη
Lecture07
Microsoft Po...
wxDev-C++
D:\maya\Co...
EN Επιφάνεια εργασίας
9:43 μμ


wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = 163
y = 163
&x = (int *) 0x22ff74
&y = (int *) 0x22ff70
p1 = (int *) 0x22ff74
p2 = (int *) 0x22ff70
&p1 = (int **) 0x22ff6c
&p2 = (int **) 0x22ff68
*p1 = 163
*p2 = 163
Property Inspector
Properties Events
pointers.c
n()
int x, y;
int *p1, *p2;
x = -42;
y = 163;
p1 = &x;
p2 = &y;
printf("%d %d %d %d %d %d %d %d %d %d\n",x,y,*p1,*p2,p1,p2,&x,&y, &p1,&p2);
*p1=17;
*p1=*p2;
system("PAUSE");
Compiler Resources Compile Log Debug Find Results Close
Debug Backtrace Output
Next Step
Step Into
Continue
Run to Cursor
Debug
Stop Execution
Add Watch
Remove watch
14: 12
Insert
16 Lines in file
έναρξη
Lecture07
Microsoft Po...
wxDev-C++
D:\maya\Co...
EN Επιφάνεια εργασίας
9:43 μμ

wxDev-C++ 6.10.2

File Edit Search View Project Execute Debug Tools CVS Window Help

New Insert Toggle Goto

Project Inspector

Project Classes Debug

Property Inspector

Properties Events

pointers.c

```
1  main()
2  {
3      int x, y;
4      int *p1, *p2;
5
6      x = -42;
7      y = 163;
8
9      p1 = &x;
10     p2 = &y;
11     p1=x;
12     printf("%d %d %d %d %d %d %d %d %d %d\n",x,y,*p1,*p2,p1,p2,&x,&y, &p1,&
13     *p1=17;
14     *p1=*p2;
15     system("PAUSE");
16 }
17
```

Compiler Resources Compile Log Debug Find Results Close

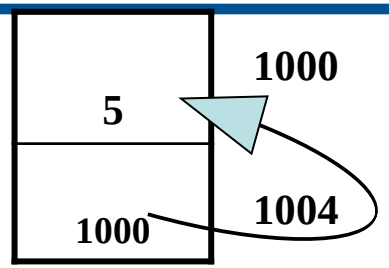
Line	File	Message
	D:\maya\Courses\C\C_ΕΠΛ\Lecture...	In function `main':
11	D:\maya\Courses\C\C_ΕΠΛ\Lecture...	[Warning] assignment makes pointer from integer without a cast

11: 1 Insert 17 Lines in file

έναρξη Lecture07 Microsoft PowerPoint... wxDev-C++ EN Επιφάνεια εργασίας 9:56 μμ

Βασικές πράξεις δεικτών

```
int x, *p;
p = &x;
```



Τι υποδεικνύουν τα σύμβολα x, &x, p, &p, *p

- ✓ Το **x** υποδεικνύει το περιεχόμενο (τη τιμή) της μεταβλητής (μνήμης) x.

  Η τιμή αυτή είναι ένας ακέραιος αριθμός (**στο πχ, η τιμή του x είναι 5**)

- ✓ Το **&x** υποδεικνύει τη διεύθυνση της μεταβλητής (μνήμης) x
  (**στο πχ, η τιμή του &x είναι 1000**)

- ✓ Το **p** (μεταβλητή δείκτης) υποδεικνύει το περιεχόμενο (τη τιμή) της μεταβλητής (μνήμης) p, που είναι μια διεύθυνση μνήμης, και σ' αυτή τη διεύθυνση μνήμης μπορεί να αποθηκευθεί μόνο ακέραιου τύπου αριθμός (**στο πχ, η τιμή του p είναι 1000**)

- ✓ Το **&p** υποδεικνύει τη διεύθυνση της μεταβλητής (μνήμης) p
  (**στο πχ, η τιμή του &p είναι 1004**)

- ✓ Το ***p** υποδεικνύει τη τιμή (περιεχόμενο) που βρίσκεται στη θέση μνήμης στην οποία δείχνει η μεταβλητή (δείκτης) p
  (**στο πχ, η τιμή του *p είναι 5**)

Ο ειδικός δείκτης NULL

- Η ειδική σταθερά **NULL** υποδηλώνει ότι μια μεταβλητή δείκτη δεν δείχνει σε έγκυρα δεδομένα.

- Όταν μια μεταβλητή δείκτη έχει την τιμή **NULL**, δεν πρέπει να την αποαναφοροποιήσετε με τον τελεστή *.

Μεταβίβαση παραμέτρων κατ' αναφορά (1)

- Στη C κάθε φορά που μεταβιβάζετε μια απλή μεταβλητή από μια συνάρτηση σε κάποια άλλη, η συνάρτηση προορισμού δέχεται ένα αντίγραφο της τιμής του ορίσματος.
- Η ανάθεση μιας νέας τιμής στην παράμετρο, στα πλαίσια της συνάρτησης, αλλάζει την τιμή του τοπικού αντιγράφου, αλλά δεν έχει καμία επίδραση στο όρισμα με το οποίο καλείται.
- Αυτό δεν συμβαίνει αν μεταβιβάσετε στη συνάρτηση ένα δείκτη προς τη μεταβλητή και όχι την ίδια τη μεταβλητή.
- Στη C πρέπει να δηλώνετε ρητά ότι θέλετε να επιτρέψετε τέτοιου είδους αλλαγές δηλώνοντας την τιμή της παραμέτρου ως τύπου δείκτη και μεταβιβάζοντας διευθύνσεις ως ορίσματα.

Μεταβίβαση παραμέτρων κατ' αναφορά (2)

//setToZeroErr.c

```
#include <stdio.h>
#include "genlib.h"

void setToZero(int ip);

main() {
   int x;
   setToZero(x);
   printf("x=%d\n",x);
   system("pause");
}

void setToZero(int ip)
{
    ip = 0;
}
```

Μεταβιβάζω τη τιμή της x ως όρισμα

//setToZero.c

```
#include <stdio.h>
#include "genlib.h"

void setToZero(int *ip);

main() {
   int x;
   setToZero(&x);
   printf("x=%d\n",x);
   system("pause");
}

void setToZero(int *ip)
{
    *ip = 0;
}
```

Μεταβιβάζω τη διεύθυνση της x ως όρισμα

Η τιμή της παραμέτρου είναι τύπου δείκτη

Μεταβίβαση παραμέτρων κατ' αναφορά (3)

```
1. //setToZero.c
2.
3. #include <stdio.h>
4. #include "genlib.h"
5.
6. void setToZero(int *ip);
7.
8. main() {
9.       int x;
10.      setToZero(&x);
11.      printf("x=%d\n",x);
12.      system("pause");
13. }
14.
15. void setToZero(int *ip)
16. {
17.     *ip = 0;
18. }
```

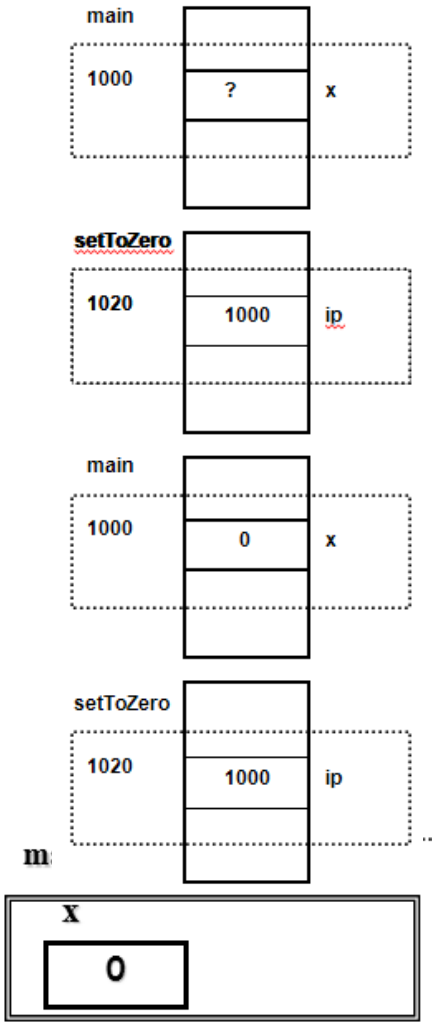
Μεταβιβάζω τη διεύθυνση της x ως όρισμα

Η τιμή της παραμέτρου είναι τύπου δείκτη

Όταν η main καλεί τη setZero με την εντολή setToZero(&x); Δημιουργείται ένα νέο πλαίσιο για τη συνάρτηση

Η εντολή *ip = 0; Μηδενίζει τη τιμή της μνήμης x στην οποία δείχνει η μεταβλητή ip

Όταν επιστρέψει η συνάρτηση setToZero η αλλαγή που έγινε στη διεύθυνση 1000 θα εξακολουθεί να ισχύει

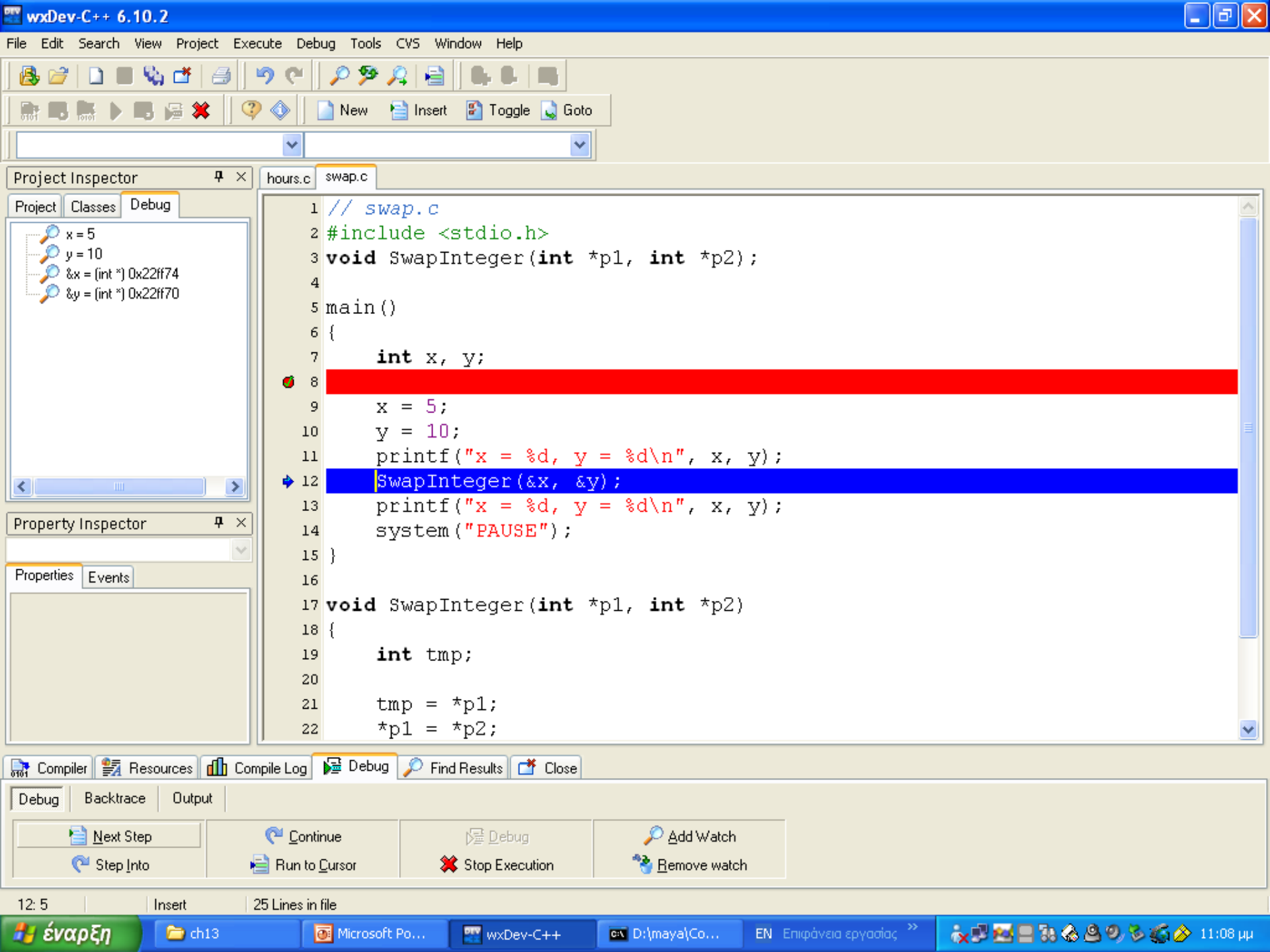


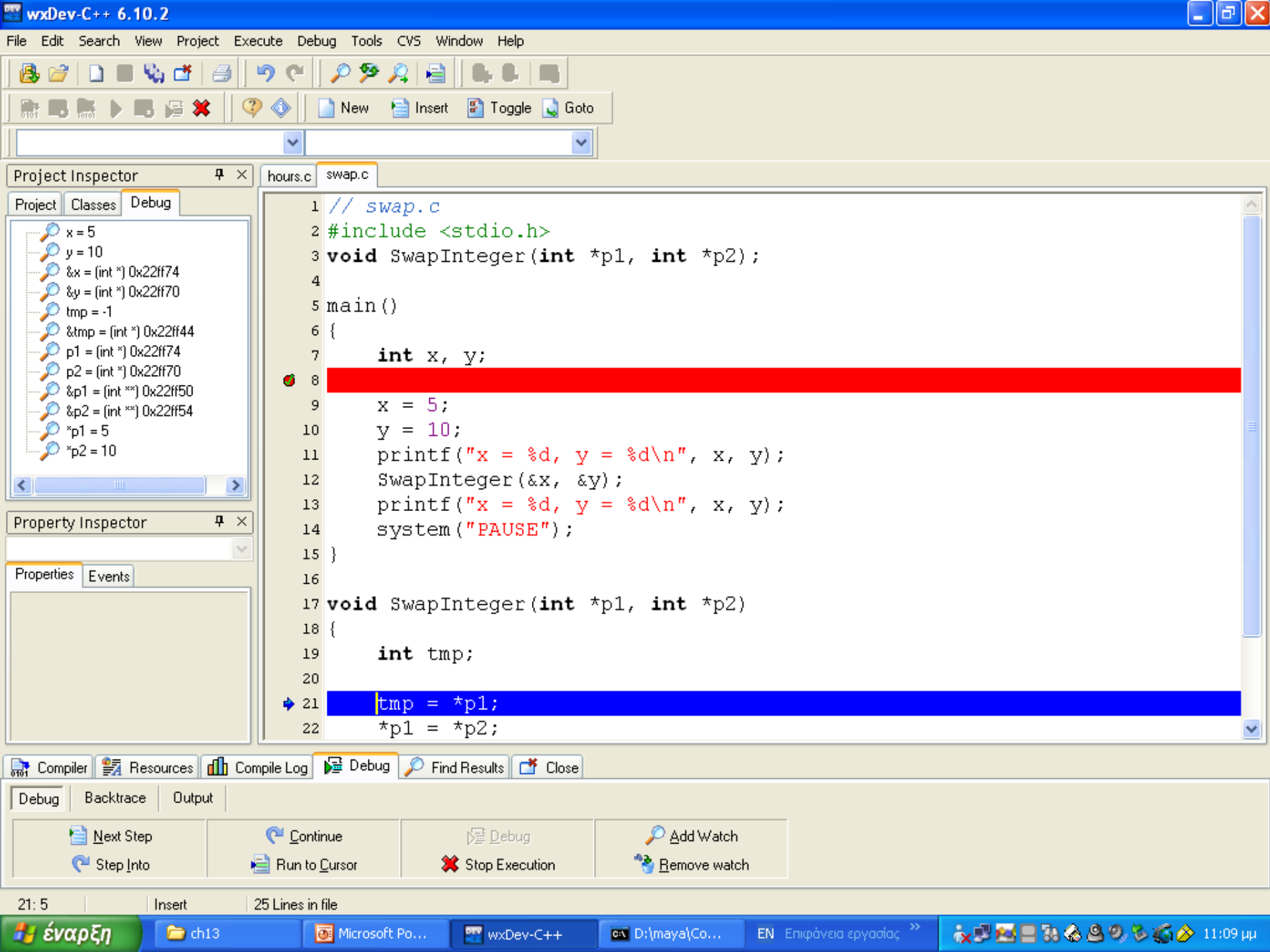
Μεταβίβαση παραμέτρων κατ' αναφορά (4)

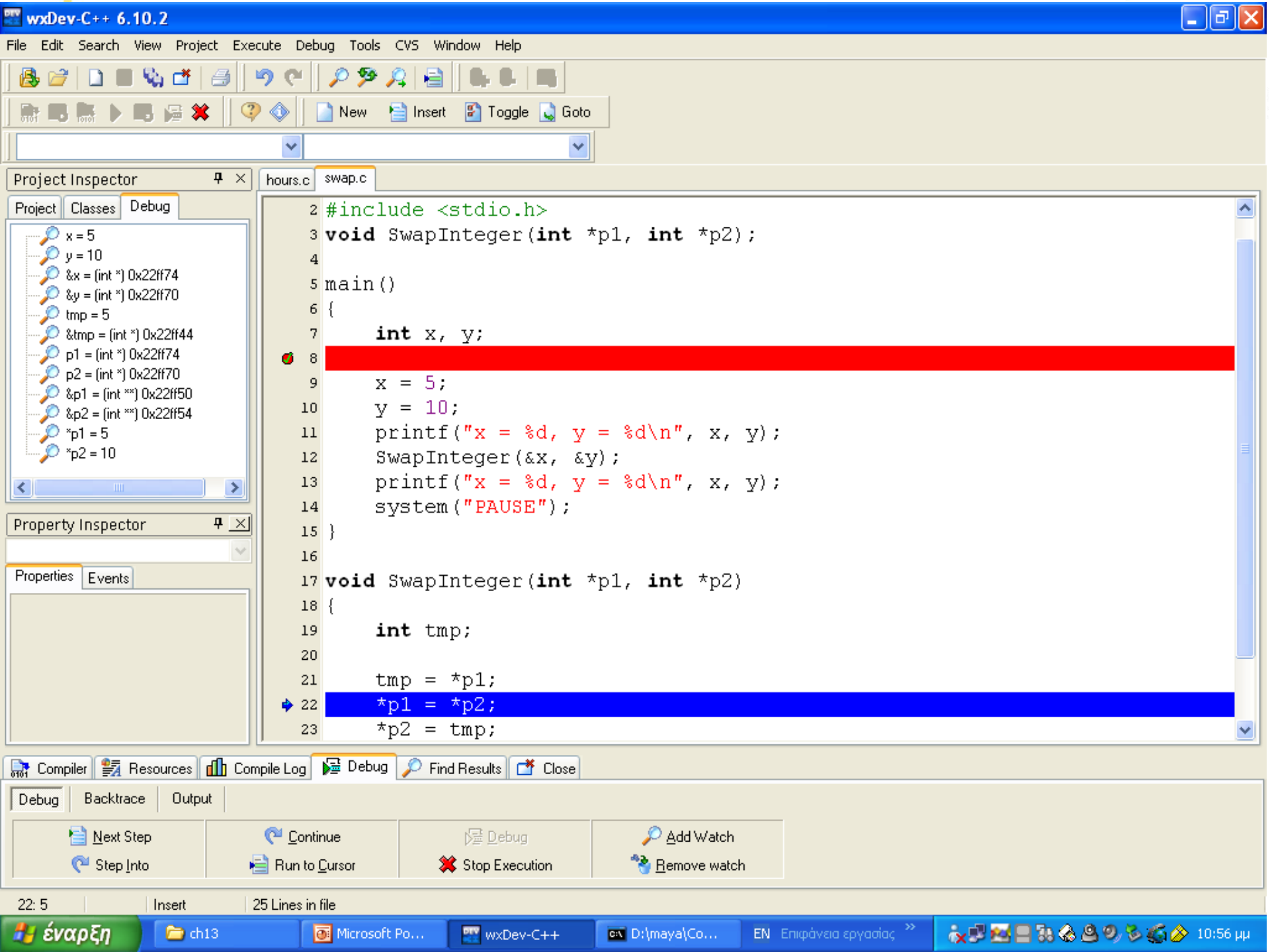
```
#include <stdio.h>
void SwapInteger(int *p1, int *p2);

main()
{
        int x, y;
        x = 5;
        y = 10;
        printf("x = %d, y = %d\n", x, y);
        SwapInteger(&x, &y);
        printf("x = %d, y = %d\n", x, y);
}

void SwapInteger(int *p1, int *p2)
{
        int tmp;
        tmp = *p1;
        *p1 = *p2;
        *p2 = tmp;
}
```

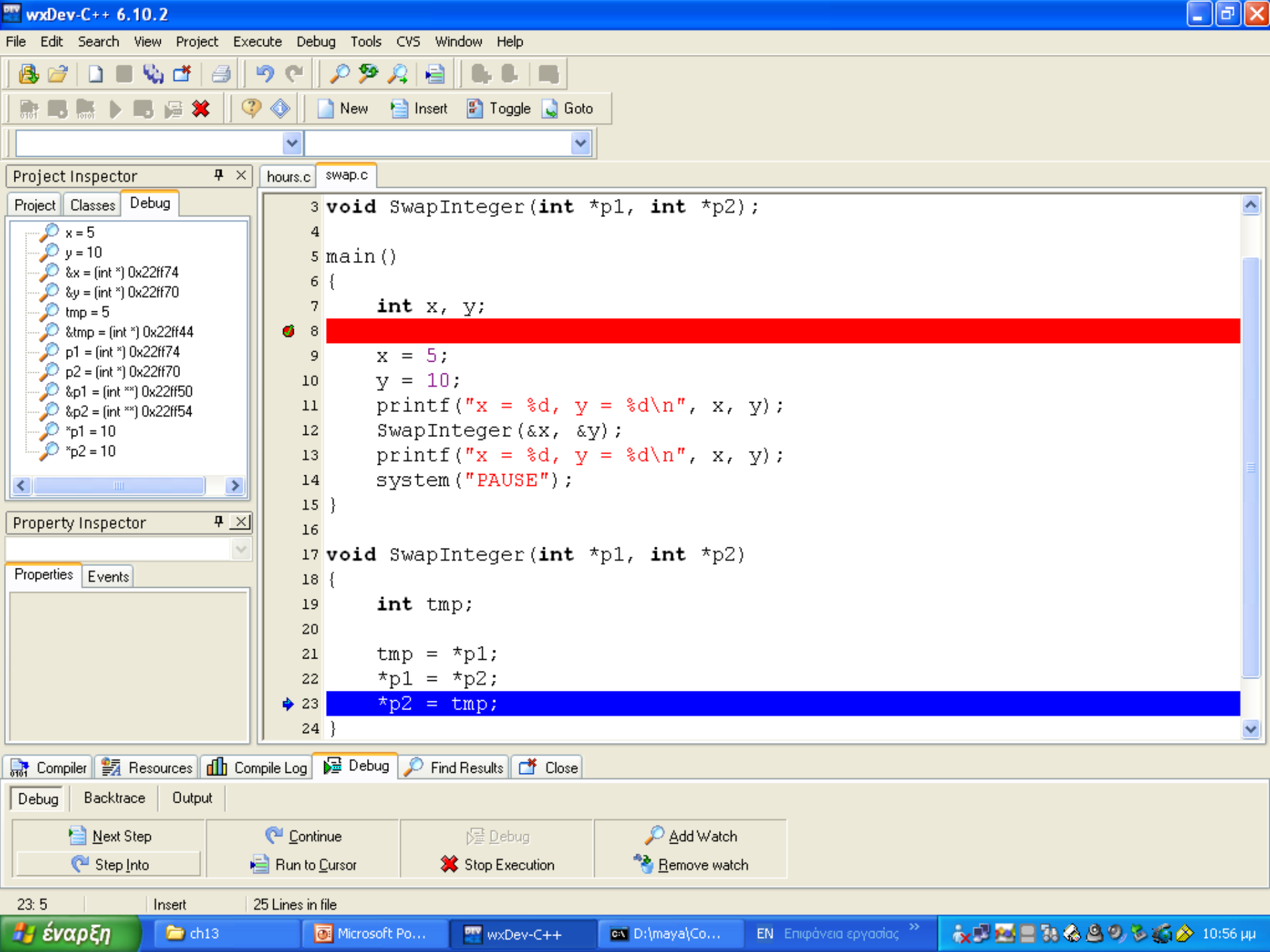
Μεταβιβάζω τη διεύθυνση της x και τη διεύθυνση της y ως ορίσματα

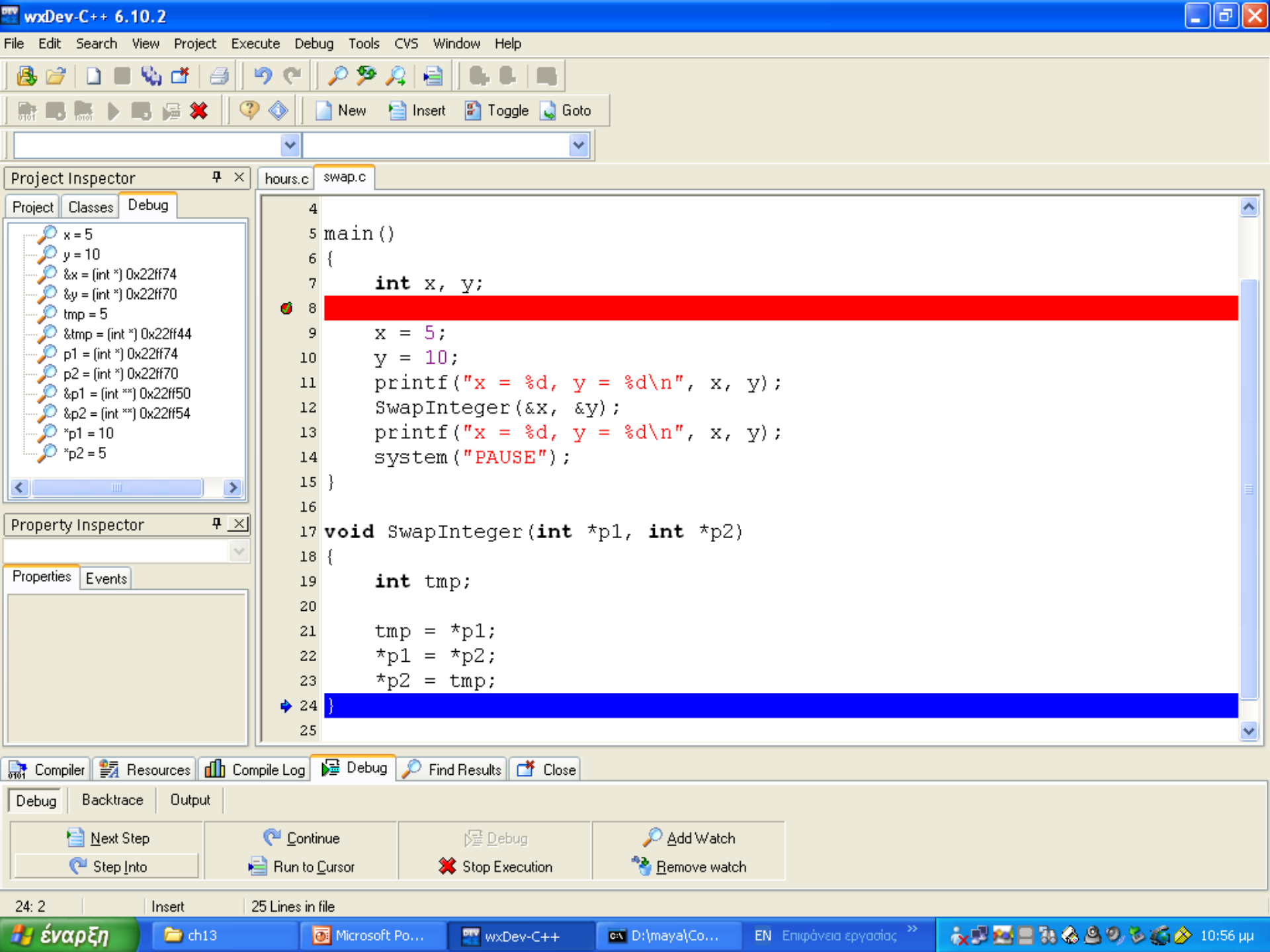
Η τιμή της παραμέτρου είναι τύπου δείκτη

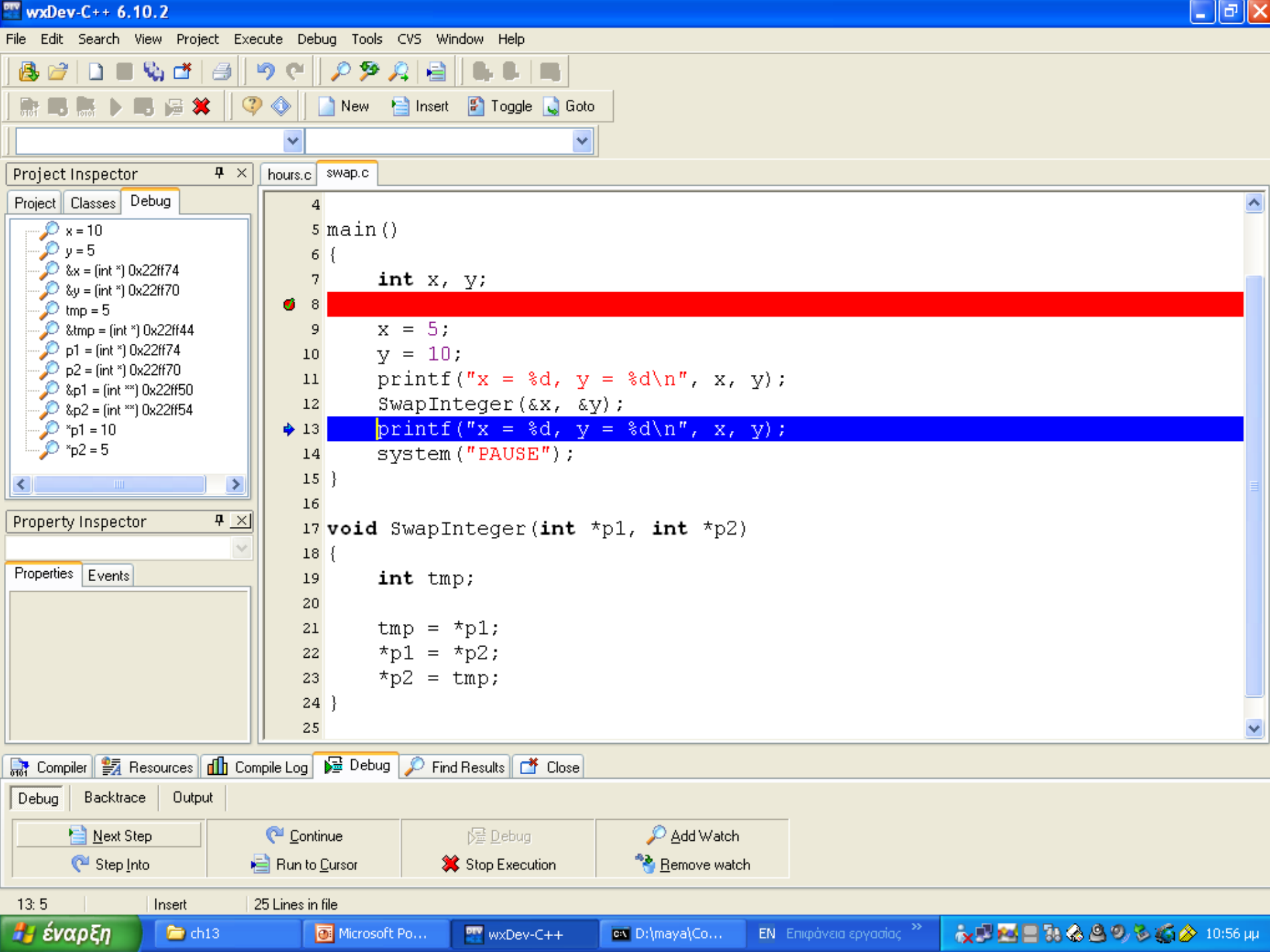

wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = 5
y = 10
&x = (int *) 0x22ff74
&y = (int *) 0x22ff70
Property Inspector
Properties Events
hours.c swap.c
1 // swap.c
2 #include <stdio.h>
3 void SwapInteger(int *p1, int *p2);
4
5 main()
6 {
7 int x, y;
8
9 x = 5;
10 y = 10;
11 printf("x = %d, y = %d\n", x, y);
12 SwapInteger(&x, &y);
13 printf("x = %d, y = %d\n", x, y);
14 system("PAUSE");
15 }
16
17 void SwapInteger(int *p1, int *p2)
18 {
19 int tmp;
20
21 tmp = *p1;
22 *p1 = *p2;
Compiler Resources Compile Log Debug Find Results Close
Debug Backtrace Output
Next Step
Step Into
Continue
Run to Cursor
Debug
Stop Execution
Add Watch
Remove watch
12: 5
Insert
25 Lines in file
έναρξη
ch13
Microsoft Po...
wxDev-C++
D:\maya\Co...
EN Επιφάνεια εργασίας
11:08 μμ

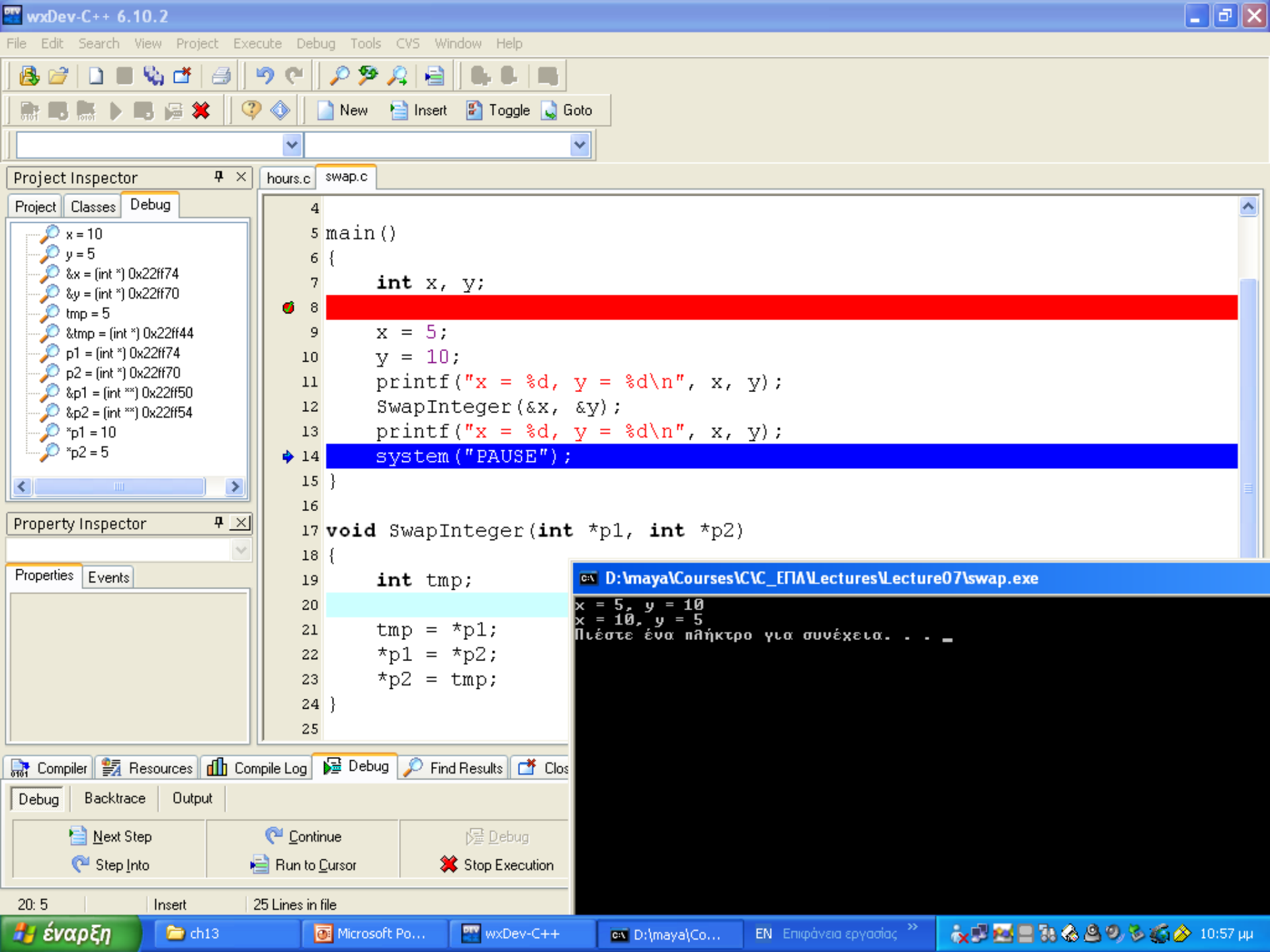

wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = 5
y = 10
&x = (int *) 0x22ff74
&y = (int *) 0x22ff70
tmp = -1
&tmp = (int *) 0x22ff44
p1 = (int *) 0x22ff74
p2 = (int *) 0x22ff70
&p1 = (int **) 0x22ff50
&p2 = (int **) 0x22ff54
*p1 = 5
*p2 = 10
Property Inspector
Properties Events
hours.c swap.c
1 // swap.c
2 #include <stdio.h>
3 void SwapInteger(int *p1, int *p2);
4
5 main()
6 {
7 int x, y;
8
9 x = 5;
10 y = 10;
11 printf("x = %d, y = %d\n", x, y);
12 SwapInteger(&x, &y);
13 printf("x = %d, y = %d\n", x, y);
14 system("PAUSE");
15 }
16
17 void SwapInteger(int *p1, int *p2)
18 {
19 int tmp;
20
21 tmp = *p1;
22 *p1 = *p2;
Compiler Resources Compile Log Debug Find Results Close
Debug Backtrace Output
Next Step Continue Debug Add Watch
Step Into Run to Cursor Stop Execution Remove watch
21: 5 Insert 25 Lines in file
έναρξη ch13 Microsoft Po... wxDev-C++ D:\maya\Co... EN Επιφάνεια εργασίας 11:09 μμ


wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = 5
y = 10
&x = (int *) 0x22ff74
&y = (int *) 0x22ff70
tmp = 5
&tmp = (int *) 0x22ff44
p1 = (int *) 0x22ff74
p2 = (int *) 0x22ff70
&p1 = (int **) 0x22ff50
&p2 = (int **) 0x22ff54
*p1 = 5
*p2 = 10
Property Inspector
Properties Events
hours.c swap.c
2 #include <stdio.h>
3 void SwapInteger(int *p1, int *p2);
4
5 main()
6 {
7 int x, y;
8
9 x = 5;
10 y = 10;
11 printf("x = %d, y = %d\n", x, y);
12 SwapInteger(&x, &y);
13 printf("x = %d, y = %d\n", x, y);
14 system("PAUSE");
15 }
16
17 void SwapInteger(int *p1, int *p2)
18 {
19 int tmp;
20
21 tmp = *p1;
22 *p1 = *p2;
23 *p2 = tmp;
Compiler Resources Compile Log Debug Find Results Close
Debug Backtrace Output
Next Step Continue Debug Add Watch
Step Into Run to Cursor Stop Execution Remove watch
22: 5 Insert 25 Lines in file
έναρξη ch13 Microsoft Po... wxDev-C++ D:\maya\Co... EN Επιφάνεια εργασίας 10:56 μμ


wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = 5
y = 10
&x = (int *) 0x22ff74
&y = (int *) 0x22ff70
tmp = 5
&tmp = (int *) 0x22ff44
p1 = (int *) 0x22ff74
p2 = (int *) 0x22ff70
&p1 = (int **) 0x22ff50
&p2 = (int **) 0x22ff54
*p1 = 10
*p2 = 10
Property Inspector
Properties Events
hours.c swap.c
3 void SwapInteger(int *p1, int *p2);
4
5 main()
6 {
7 int x, y;
8
9 x = 5;
10 y = 10;
11 printf("x = %d, y = %d\n", x, y);
12 SwapInteger(&x, &y);
13 printf("x = %d, y = %d\n", x, y);
14 system("PAUSE");
15 }
16
17 void SwapInteger(int *p1, int *p2)
18 {
19 int tmp;
20
21 tmp = *p1;
22 *p1 = *p2;
23 *p2 = tmp;
24 }
Compiler Resources Compile Log Debug Find Results Close
Debug Backtrace Output
Next Step
Step Into
Continue
Run to Cursor
Debug
Stop Execution
Add Watch
Remove watch
23: 5 Insert 25 Lines in file
έναρξη
ch13
Microsoft Po...
wxDev-C++
D:\maya\Co...
EN Επιφάνεια εργασίας
10:56 μμ


wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = 5
y = 10
&x = (int *) 0x22ff74
&y = (int *) 0x22ff70
tmp = 5
&tmp = (int *) 0x22ff44
p1 = (int *) 0x22ff74
p2 = (int *) 0x22ff70
&p1 = (int **) 0x22ff50
&p2 = (int **) 0x22ff54
*p1 = 10
*p2 = 5
Property Inspector
Properties Events
hours.c swap.c
4
5 main()
6 {
7 int x, y;
8
9 x = 5;
10 y = 10;
11 printf("x = %d, y = %d\n", x, y);
12 SwapInteger(&x, &y);
13 printf("x = %d, y = %d\n", x, y);
14 system("PAUSE");
15 }
16
17 void SwapInteger(int *p1, int *p2)
18 {
19 int tmp;
20
21 tmp = *p1;
22 *p1 = *p2;
23 *p2 = tmp;
24 }
25
Compiler Resources Compile Log Debug Find Results Close
Debug Backtrace Output
Next Step Continue Debug Add Watch
Step Into Run to Cursor Stop Execution Remove watch
24: 2 Insert 25 Lines in file
έναρξη ch13 Microsoft Po... wxDev-C++ D:\maya\Co... EN Επιφάνεια εργασίας 10:56 μμ


wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = 10
y = 5
&x = (int *) 0x22ff74
&y = (int *) 0x22ff70
tmp = 5
&tmp = (int *) 0x22ff44
p1 = (int *) 0x22ff74
p2 = (int *) 0x22ff70
&p1 = (int **) 0x22ff50
&p2 = (int **) 0x22ff54
*p1 = 10
*p2 = 5
Property Inspector
Properties Events
hours.c swap.c
4
5 main()
6 {
7 int x, y;
8
9 x = 5;
10 y = 10;
11 printf("x = %d, y = %d\n", x, y);
12 SwapInteger(&x, &y);
13 printf("x = %d, y = %d\n", x, y);
14 system("PAUSE");
15 }
16
17 void SwapInteger(int *p1, int *p2)
18 {
19 int tmp;
20
21 tmp = *p1;
22 *p1 = *p2;
23 *p2 = tmp;
24 }
25
Compiler Resources Compile Log Debug Find Results Close
Debug Backtrace Output
Next Step
Step Into
Continue
Run to Cursor
Debug
Stop Execution
Add Watch
Remove watch
13: 5
Insert
25 Lines in file
έναρξη
ch13
Microsoft Po...
wxDev-C++
D:\maya\Co...
EN Επιφάνεια εργασίας
10:56 μμ


wxDev-C++ 6.10.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Inspector
Project Classes Debug
x = 10
y = 5
&x = (int *) 0x22ff74
&y = (int *) 0x22ff70
tmp = 5
&tmp = (int *) 0x22ff44
p1 = (int *) 0x22ff74
p2 = (int *) 0x22ff70
&p1 = (int **) 0x22ff50
&p2 = (int **) 0x22ff54
*p1 = 10
*p2 = 5
Property Inspector
Properties Events
hours.c swap.c
4
5 main()
6 {
7 int x, y;
8
9 x = 5;
10 y = 10;
11 printf("x = %d, y = %d\n", x, y);
12 SwapInteger(&x, &y);
13 printf("x = %d, y = %d\n", x, y);
14 system("PAUSE");
15 }
16
17 void SwapInteger(int *p1, int *p2)
18 {
19 int tmp;
20
21 tmp = *p1;
22 *p1 = *p2;
23 *p2 = tmp;
24 }
25
D:\maya\Courses\C\C_ΕΠΛ\Lectures\Lecture07\swap.exe
x = 5, y = 10
x = 10, y = 5
Πιέστε ένα πλήκτρο για συνέχεια. . .
Compiler Resources Compile Log Debug Find Results Clos
Debug Backtrace Output
Next Step Continue Debug
Step Into Run to Cursor Stop Execution
20: 5 Insert 25 Lines in file
έναρξη ch13 Microsoft Po... wxDev-C++ D:\maya\Co... EN Επιφάνεια εργασίας 10:57 μμ

Μεταβίβαση παραμέτρων κατ' αναφορά (5)

```
#include <stdio.h>
#include "genlib.h"
#include "simpio.h"
#define MinutesPerHour 60
static void ConvertTimeToHM(int time, int *pHours, int *pMinutes);
main()
{
   int time, hours, minutes;
   printf("Test program to convert time values\n");
   printf("Enter a time duration in minutes: ");
   time = GetInteger();
   ConvertTimeToHM(time, &hours, &minutes);
   printf("HH:MM format: %d:%02d\n", hours, minutes);
}
static void ConvertTimeToHM(int time, int *pHours, int *pMinutes)
{
   *pHours = time / MinutesPerHour;
   *pMinutes = time % MinutesPerHour;
}
```

Μεταβίβαση παραμέτρων κατ' αναφορά (6)

- Όταν μια συνάρτηση πρέπει να επιστρέψει στο καλούν πρόγραμμα περισσότερες από μια τιμές τότε χρειάζεται να χρησιμοποιηθεί κλήση κατ' αναφορά.
- Από μια συνάρτηση ένα αποτέλεσμα μπορεί να επιστραφεί, εκεί απ' όπου κλήθηκε η συνάρτηση ως τιμή της ίδιας της συνάρτησης.
- Αν χρειάζεται να επιστρέψετε από μια συνάρτηση περισσότερα από ένα αποτελέσματα δεν μπορείτε να χρησιμοποιήσετε την τιμή επιστροφής της συνάρτησης.
- Η προσέγγιση που χρησιμοποιείται για να λύσουμε το πρόβλημα είναι η μετατροπή της συνάρτησης σε διαδικασία και η μεταβίβαση τιμών από και προς αυτή μέσω της λίστας των ορισμάτων της (μεταβίβαση κατ' αναφορά).
- Οπότε αντί να μεταβιβάσετε στις συναρτήσεις ως ορίσματα τις τιμές των μεταβλητών μεταβιβάζετε ένα δείκτη προς κάθε μεταβλητή, δηλαδή τη διεύθυνση της μεταβλητής.

Αριθμητική δεικτών (1)

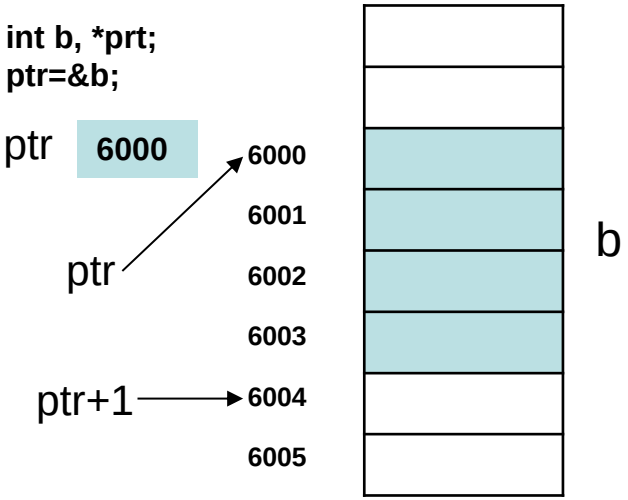
Ανακεφαλαίωση

- Οι δείκτες είναι μεταβλητές που περιέχουν διευθύνσεις μνήμης, συνήθως διευθύνσεις άλλων μεταβλητών
- Το μέγεθος μιας μεταβλητής δείκτη είναι συγκεκριμένο και ανεξάρτητο από τον τύπο με τον οποίο δηλώθηκε
- Ο τελεστής & αποδίδει τη διεύθυνση μιας θέσης μνήμης
- Ο τελεστής αναφοράς * χρησιμοποιείται σε συνδυασμό με μια μεταβλητή δείκτη για να αναφερθούμε στη μεταβλητή στην οποία δείχνει ο δείκτης

Αριθμητική δεικτών (2)

Μια μεταβλητή δείκτη δείχνει σε μεταβλητή συγκεκριμένου τύπου. Αν αυξήσουμε τη μεταβλητή δείκτη κατά ένα, τότε αυτή θα πάρει τέτοια τιμή ώστε να δείχνει στην επόμενη διεύθυνση του ιδίου τύπου

```
int b, *prt;
ptr=&b;          //η μεταβλητή ptr θα πάρει τη τιμή 6000
ptr = ptr + 1; //η μεταβλητή ptr θα πάρει τη τιμή 6004
++ptr;        //η μεταβλητή ptr θα πάρει τη τιμή 6008
ptr = ptr+3;   //η μεταβλητή ptr θα πάρει τη τιμή 6020
```



Αν αυξήσουμε μια μεταβλητή δείκτη πχ κατά 1 τότε η διεύθυνση που περιέχει :

Θα αυξηθεί κατά 4 όταν δείχνει σε μεταβλητές τύπου int

Θα αυξηθεί κατά 4 όταν δείχνει σε μεταβλητές τύπου float

Θα αυξηθεί κατά 8 όταν δείχνει σε μεταβλητές τύπου double

Για να εμφανίσετε μια διεύθυνση σε δεκαεξαδική μορφή χρησιμοποιήστε %p στη μορφοποίηση της printf, πχ printf(“pointer ptr: **%p**”,ptr);

Αριθμητική δεικτών (3)

- ✓ Η παράσταση *ptr++ μπορεί να θεωρηθεί ισοδύναμη με

  (*ptr)++ ή με *(ptr++)

- ✓ Στη C οι μονομελείς τελεστές (όπως ++, - -, *) αποτιμώνται από δεξιά προς τα αριστερά.
- ✓ Άρα, οι τελεστές ++ και -- έχουν μεγαλύτερη προτεραιότητα από τον τελεστή αναφοράς *.
- ✓ Συνεπώς η ερμηνεία της παράστασης *ptr++ είναι ισοδύναμη με *(ptr++).
- ✓ Άρα η παράσταση *ptr++ αυξάνει κατά 1 την τιμή της μεταβλητής ptr (σύμφωνα με την αριθμητική δεικτών), οπότε ο δείκτης ptr δείχνει σε νέα διεύθυνση μνήμης!
- ✓ Η παράσταση (*ptr)++ αυξάνει κατά 1 τη τιμή της μεταβλητής που δείχνει ο ptr.

```
1.  int a, *p;
2.  p=&a;
3.  a=44;
4.  *p=9;
5.  (*p)++;
6.  p++;
7.  *p++=100;
```

Η θέση μνήμης με διεύθυνση 2008 δεν έχει καμία σχέση με τη μεταβλητή a και προφανώς μ' αυτό τον τρόπο θα καταστραφούν άλλα δεδομένα του προγράμματος ή άλλων εφαρμογών. Να είστε προσεκτικοί με τους δείκτες!

100

2008 ←

p	a
2000	
2000	44
2000	9
2000	10
2004	10
2008	10

Δείκτες και Πίνακες (1)

Στη C υφίσταται πολύ στενή σχέση μεταξύ δεικτών και πινάκων.

Η δήλωση ενός πίνακα: double list[3];

δεσμεύει τρεις διαδοχικές θέσεις μνήμης. Κάθε ένα από τα τρία στοιχεία του πίνακα έχει μια διεύθυνση, την οποία μπορεί κανείς να μάθει χρησιμοποιώντας τον τελεστή &.

Για παράδειγμα η διεύθυνση του δεύτερου στοιχείο είναι &list[1];

Στη C ισχύει ότι (αριθμητική δεικτών):

Αν p είναι δείκτης προς το αρχικό στοιχείο ενός πίνακα arr και k ακέραιος,

τότε η παράσταση p+k είναι επίσης δείκτης που δείχνει στο k στοιχείο του πίνακα,

δηλ: p + k ισούται εξ'ορισμού με &arr[k];

Δείκτες και Πίνακες (2)

Για το κάτωθι τμήμα κώδικα:

```
int A[5] = { 3, 2, 1, 5, 4};      //Δήλωση πίνακα
int* p;                                  //Δήλωση δείκτη
p = &A[0];                               //ανάθεση αρχικής τιμής στο p
printf("%d\n", *(p+2));                  //ισοδύναμο με
                                  //printf("%d\n", A[2]);
```

θα εκτυπωθεί το περιεχόμενο του 3ου στοιχείου, A[2]

Ένα από τα ασυνήθιστα χαρακτηριστικά της C είναι ότι **το όνομα ενός πίνακα είναι ταυτόσημο με τη διεύθυνση του πρώτου στοιχείου του.**

Δηλ. για το παραπάνω τμήμα κώδικα

A ταυτίζεται με &A[0]

Όταν μεταβιβάζουμε σε μία συνάρτηση ως όρισμα έναν πίνακα γράφουμε **απλώς το όνομα του πίνακα**, δηλαδή **μεταβιβάζουμε τη διεύθυνση του πρώτου στοιχείου του.**

Δείκτες και Πίνακες (3)

- Αφού κατά την μεταβίβαση ενός πίνακα σε συνάρτηση μεταβιβάζεται ουσιαστικά η διεύθυνση του πρώτου στοιχείου του, η αντίστοιχη παράμετρος της συνάρτησης μπορεί να είναι δείκτης.
- Τα δύο προγράμματα είναι ισοδύναμα.

```
void setToZero(int Array[], int size);

main() {
   int i;
   int A[5] = {3, 2, 1, 5, 4};
   setToZero(A, 5);
   for(i=0; i<5; i++)
      printf("%d\n", A[i]);
}

void setToZero(int Array[], int size) {
   int i;
   for(i=0; i<size; i++)
      Array[i] = 0;

}
```

```
void setToZero(int* p, int size);

main() {
   int i;
   int A[5] = {3, 2, 1, 5, 4};
   setToZero(A, 5);
   for(i=0; i<5; i++)
      printf("%d\n", A[i]);
}

void setToZero(int* p, int size) {
   int i;
   for(i=0; i<size; i++)
      *(p+i) = 0;

}
```

Δυναμική Κατανομή Μνήμης (1)

1. Όταν δηλώνετε μια καθολική μεταβλητή ο μεταγλωττιστής κατανέμει χώρο μνήμης γι' αυτή τη μεταβλητή ο οποίος παραμένει διαθέσιμος σε όλη τη διάρκεια εκτέλεσης του προγράμματος. Αυτό το στυλ κατανομής μνήμης ονομάζεται **στατική κατανομή (static allocation).**
2. Όταν δηλώνετε μια τοπική μεταβλητή στο εσωτερικό μιας συνάρτησης ο χώρος μνήμης γι' αυτή τη μεταβλητή κατανέμεται στη στοίβα του συστήματος. Η κλήση της συνάρτησης έχει ως αποτέλεσμα να εκχωρηθεί μνήμη στη μεταβλητή που αποδεσμεύεται μόλις επιστρέψει η συνάρτηση. Αυτό το στυλ κατανομής μνήμης ονομάζεται **αυτόματη κατανομή (automatic allocation).**
3. Υπάρχει και 3ος τρόπος κατανομής μνήμης ο οποίος επιτρέπει να αποκτάται πρόσβαση σε νέα μνήμη όταν χρειάζεται και την απελευθερώνετε ρητώς όταν δε τη χρειάζεστε. Η διεργασία της απόκτησης νέων χώρων μνήμης κατά τη διάρκεια εκτέλεσης του προγράμματος ονομάζεται δυναμική **κατανομή μνήμης (dynamic allocation).**

Δυναμική Κατανομή Μνήμης (2)

Η συνάρτηση **malloc(nbytes)**, όπου nbytes πλήθος από bytes,

Κατανέμει ένα τμήμα μνήμης ώστε να φιλοξενήσει nbytes πληροφορία και επιστρέφει έναν δείκτη προς τη 1η διεύθυνση αυτού του τμήματος μνήμης. Ο τύπος του δείκτη που επιστρέφει η malloc είναι ένας γενικός τύπος δείκτη και θα πρέπει να γίνει μετατροπή στο τύπο δείκτη που θέλουμε.

Π.χ. Αν θέλουμε έναν δυναμικό πίνακα ακεραίων 10 θέσεων

```
int *arr;                                   // δήλωση δείκτη

arr=(int *)malloc(10*sizeof(int));          //κατανομή μνήμης
```

Ή

```
arr= =(int *)malloc(n*m*sizeof(int));  //τα n, m θα έχουν λάβει προηγουμένως τιμές
```

Με τον ίδιο τρόπο κατανέμω μνήμη όπως και αν χρησιμοποιήσω τον πίνακα (είτε ως μονοδιάστατο είτε ως δισδιάστατο).

Δυναμική Κατανομή Μνήμης (3)

```
// Dymanic.c

#include <stdio.h>
#include <stdlib.h>
#include "genlib.h"
#include "simpio.h"

#define elements 10

main()
{
    int b[10];
    int *a;
    int i,n;

    printf("n?= ");
     n=GetInteger();
     a=(int *)malloc(n*sizeof(int));
     printf("dwse times gia ton pinaka a\n");
     for (i=0;i<n;i++)
         a[i]=GetInteger();
    printf("dwse times gia ton pinaka b\n");
     for (i=0;i<elements;i++)
         b[i]=GetInteger();
}
```

Τέλος Ενότητας