Διάλεξη 12η: Δείκτες, μέρος 2

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Εισαγωγή στην Επιστήμη Υπολογιστών

Βασίζεται σε διαφάνειες του Κ. Παναγιωτάκη

Μετατροπή Τύπων και Δείκτες

- Ο τύπος του αντικειμένου στο οποίο δείχνει ο δείκτης χρειάζεται για την σωστή αριθμητική δεικτών
  - Όταν ένας δείκτης σε `int` προχωρά 4 θέσεις, περνάει `4 * sizeof(int)` bytes
- Όλοι οι δείκτες απαιτούν την ίδια μνήμη για την αποθήκευσή τους
  - Μια διεύθυνση μνήμης χρειάζεται τον ίδιο χώρο, άσχετα αν δείχνε σε `int` ή `double`
- Η μετατροπή τύπων από ένα τύπο δείκτη σε άλλο τύπο δείκτη δεν αλλάζει τα περιεχόμενα του δείκτη

Παράδειγμα: casting

ptrcast.c

```
#include<stdio.h>
int main(void) {
  int i = 0, *iptr = &i;
  float f = 3.14159, *fptr = &f;

  printf("i = %7d, *iptr = %7d\n", i, *iptr);
  printf("f = %7.5f, *fptr = %7.5f\n", f, *fptr);
  i = (int) f;
  printf("i = %7d, *iptr = %7d\n", i, *iptr);
  printf("f = %7.5f, *fptr = %7.5f\n", f, *fptr);
  iptr = (int *) fptr;
  printf("i = %7d, *iptr = %7d\n", i, *iptr);
  printf("f = %7.5f, *fptr = %7.5f\n", f, *fptr);
}
```

Έξοδος

```
i =       0, *iptr =       0
f = 3.14159, *fptr = 3.14159
i =       3, *iptr =       3
f = 3.14159, *fptr = 3.14159
i =       3, *iptr = 1078530000
f = 3.14159, *fptr = 3.14159
```

Παράδειγμα: sizeof

types.c

```
#include<stdio.h>
int main(void)
{
  int      i = 0,          *iptr  = &i;
  double   d = 0.0,        *dptr = &d;
  float    f = 0.0f,       *fptr  = &f;
  float    *fptrarr[10], farr[10];

  printf("Size of i       = %zd\n", sizeof(i));
  printf("Size of iptr    = %zd\n", sizeof(iptr));
  printf("Size of d       = %zd\n", sizeof(d));
  printf("Size of dptr    = %zd\n", sizeof(dptr));
  printf("Size of f       = %zd\n", sizeof(f));
  printf("Size of fptr    = %zd\n", sizeof(fptr));
  printf("Size of fptrarr = %zd\n", sizeof(fptrarr));
  printf("Size of farr    = %zd\n", sizeof(farr));
  return 0;
}
```

Έξοδος 32bit CPU

```
Size of i       = 4
Size of iptr    = 4
Size of d       = 8
Size of dptr    = 4
Size of f       = 4
Size of fptr    = 4
Size of fptrarr = 40
Size of farr    = 40
```

Έξοδος 64bit CPU

```
Size of i       = 4
Size of iptr    = 8
Size of d       = 8
Size of dptr    = 8
Size of f       = 4
Size of fptr    = 8
Size of fptrarr = 80
Size of farr    = 40
```

Σταθεροί δείκτες (const)

- Προσδιορισμός σταθεράς: `const`
  - Προσδιορίζει τύπους
  - Σημαίνει ότι η μεταβλητή με αυτόν τον τύπο δεν αλλάζει
  - Χρησιμοποιείται σε παραμέτρους που δεν αλλάζει η συνάρτηση
  - Αν υπάρχει εντολή που αλλάζει μια `const` μεταβλητή θα προκαλέσει λάθος μεταγλώττισης
- Σταθεροί δείκτες (`const` pointers)
  - Δείχνουν σε μια σταθερή διεύθυνση μνήμης
  - Πρέπει να αρχικοποιούνται στη δήλωση

Παράδειγμα

```
/* Can write to memory, cannot change pointer */
int *const const_ptr = &x;

/* Cannot write to memory, can change pointer */
const int * ptr_to_const = &x;

/* Cannot write to memory, cannot change pointer */
const int *const const_ptr_to_const = &x;
```

Παράδειγμα: const

const-error.c

```
1   #include<stdio.h>
2   int main(void)
3   {
4     int x, y;
5     int *const ptr = &x;
6
7     *ptr = 42;
8     ptr = &y;
9     return 0;
10  }
```

Έξοδος gcc100

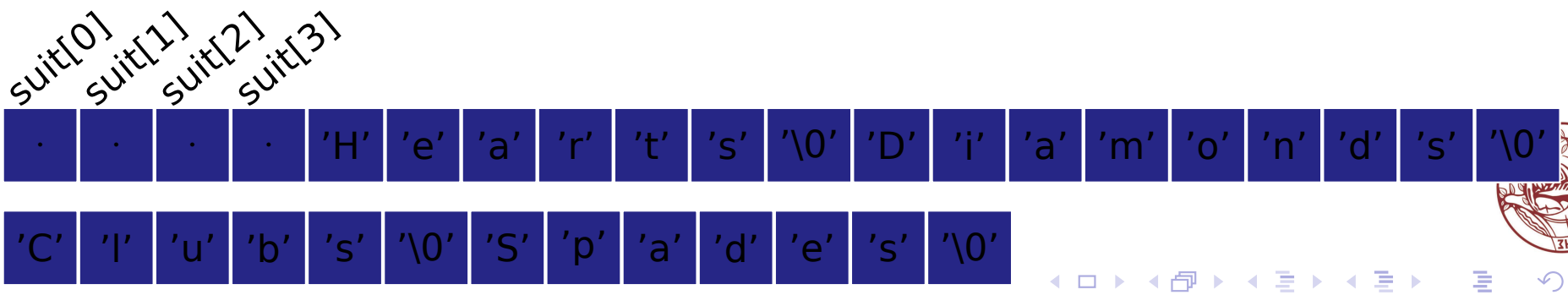
```
const-error.c: In function ‘main’:
const-error.c:8:3: error: assignment of read-only variable ‘ptr’
```

Πίνακες Δεικτών

- Ένας πίνακας μπορεί να περιέχει δείκτες
- Παράδειγμα: Πίνακας από strings

```
char *suit[4] = { "Hearts", "Diamonds", "Clubs", "Spades" };
```

  - Ο πίνακας `suit` περιέχει 4 δείκτες: στον πρώτο χαρακτήρα της κάθε λέξης
  - Κάθε στοιχείο του πίνακα `suit` είναι δείκτης σε χαρακτήρα
  - Τα strings δεν αποθηκεύονται στον πίνακα, μόνο οι δείκτες αποθηκεύονται στον πίνακα
  - Ο πίνακας έχει σταθερό μέγεθος: `4 * sizeof(char *)`
  - Το κάθε string μπορεί να έχει άλλο μέγεθος

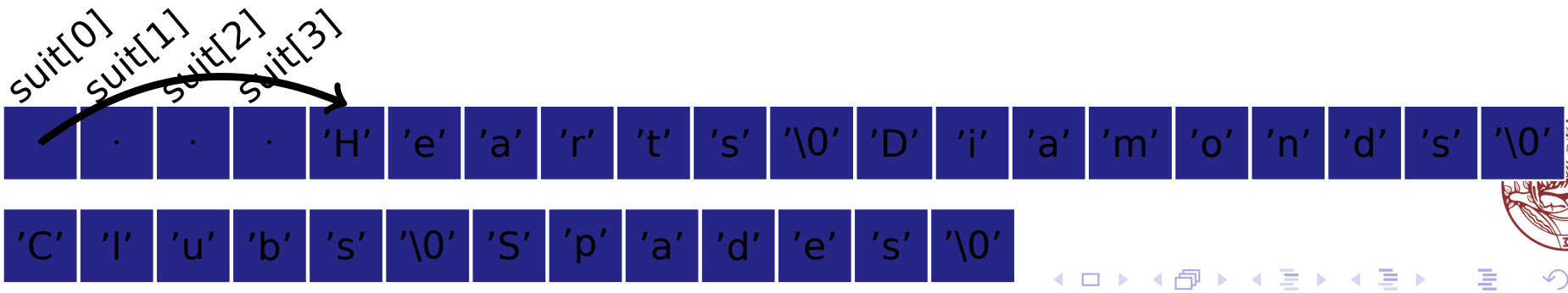


Πίνακες Δεικτών

- Ένας πίνακας μπορεί να περιέχει δείκτες
- Παράδειγμα: Πίνακας από strings

```
char *suit[4] = { "Hearts", "Diamonds", "Clubs", "Spades" };
```

  - Ο πίνακας `suit` περιέχει 4 δείκτες: στον πρώτο χαρακτήρα της κάθε λέξης
  - Κάθε στοιχείο του πίνακα `suit` είναι δείκτης σε χαρακτήρα
  - Τα strings δεν αποθηκεύονται στον πίνακα, μόνο οι δείκτες αποθηκεύονται στον πίνακα
  - Ο πίνακας έχει σταθερό μέγεθος: `4 * sizeof(char *)`
  - Το κάθε string μπορεί να έχει άλλο μέγεθος

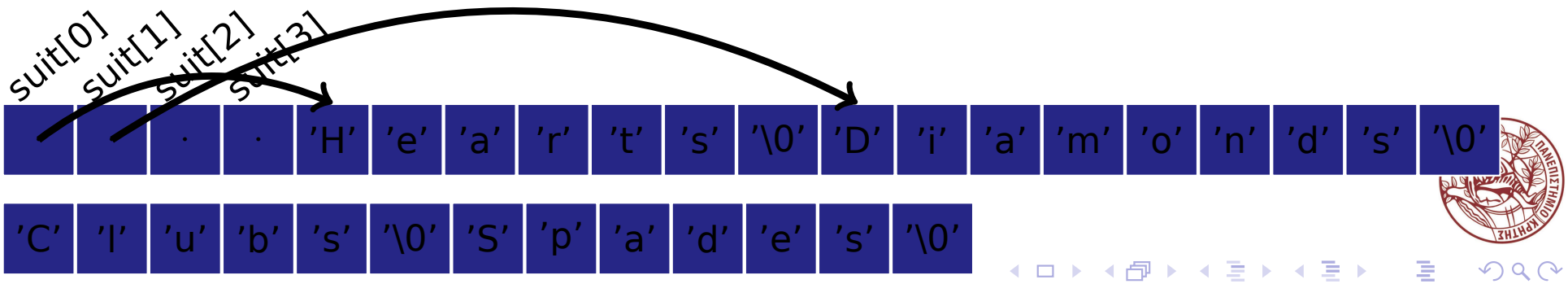


Πίνακες Δεικτών

- Ένας πίνακας μπορεί να περιέχει δείκτες
- Παράδειγμα: Πίνακας από strings

```
char *suit[4] = { "Hearts", "Diamonds", "Clubs", "Spades" };
```

  - Ο πίνακας `suit` περιέχει 4 δείκτες: στον πρώτο χαρακτήρα της κάθε λέξης
  - Κάθε στοιχείο του πίνακα `suit` είναι δείκτης σε χαρακτήρα
  - Τα strings δεν αποθηκεύονται στον πίνακα, μόνο οι δείκτες αποθηκεύονται στον πίνακα
  - Ο πίνακας έχει σταθερό μέγεθος: `4 * sizeof(char *)`
  - Το κάθε string μπορεί να έχει άλλο μέγεθος

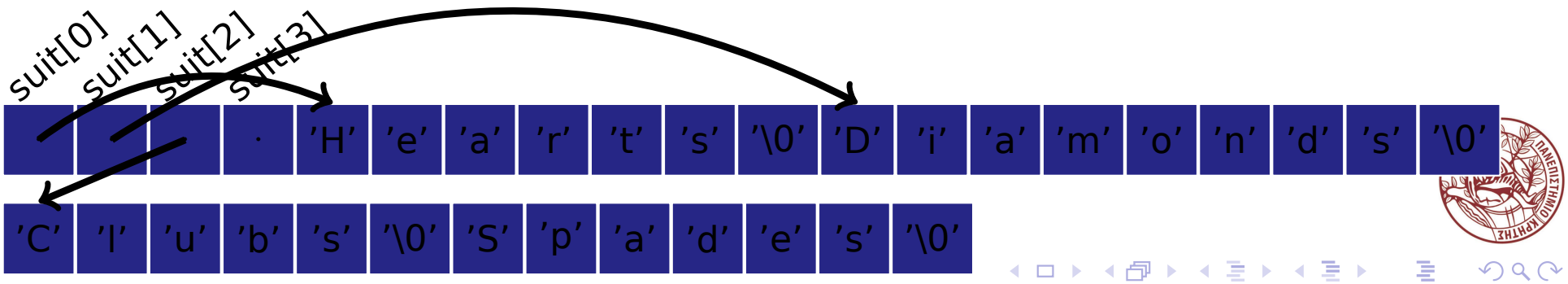


Πίνακες Δεικτών

- Ένας πίνακας μπορεί να περιέχει δείκτες
- Παράδειγμα: Πίνακας από strings

```
char *suit[4] = { "Hearts", "Diamonds", "Clubs", "Spades" };
```

  - Ο πίνακας `suit` περιέχει 4 δείκτες: στον πρώτο χαρακτήρα της κάθε λέξης
  - Κάθε στοιχείο του πίνακα `suit` είναι δείκτης σε χαρακτήρα
  - Τα strings δεν αποθηκεύονται στον πίνακα, μόνο οι δείκτες αποθηκεύονται στον πίνακα
  - Ο πίνακας έχει σταθερό μέγεθος: `4 * sizeof(char *)`
  - Το κάθε string μπορεί να έχει άλλο μέγεθος

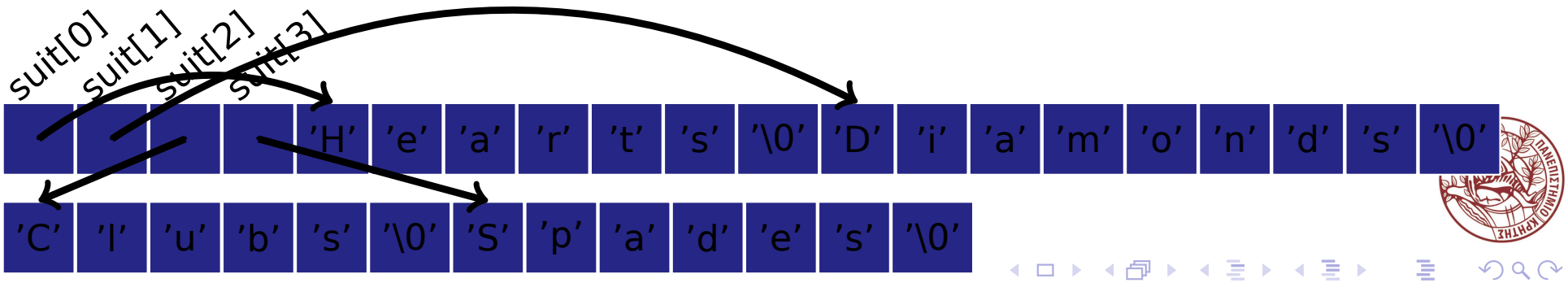


Πίνακες Δεικτών

- Ένας πίνακας μπορεί να περιέχει δείκτες
- Παράδειγμα: Πίνακας από strings

```
char *suit[4] = { "Hearts", "Diamonds", "Clubs", "Spades" };
```

  - Ο πίνακας `suit` περιέχει 4 δείκτες: στον πρώτο χαρακτήρα της κάθε λέξης
  - Κάθε στοιχείο του πίνακα `suit` είναι δείκτης σε χαρακτήρα
  - Τα strings δεν αποθηκεύονται στον πίνακα, μόνο οι δείκτες αποθηκεύονται στον πίνακα
  - Ο πίνακας έχει σταθερό μέγεθος: `4 * sizeof(char *)`
  - Το κάθε string μπορεί να έχει άλλο μέγεθος



Παράδειγμα: Τραπουλόχαρτα

cards.c

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

/* function declarations */
void shuffle(int[][13]);
void deal(int [][13],
          const char *const [],
          const char *const []);

/* function definitions */
int main(void)
{
  const char *const suit[4] =
    { "Hearts", "Diamonds",
      "Clubs", "Spades" };
  const char *const face[13] =
    { "Ace", "Deuce", "Three", "Four",
      "Five", "Six", "Seven", "Eight",
      "Nine", "Ten", "Jack", "Queen",
      "King" };
  /* Ο πίνακας που αναπαριστά την τράπουλα
   * Κάθε γραμμή είναι ένα χρώμα
   * Κάθε στήλη είναι ένα νούμερο
   * Τα περιεχόμενα του πίνακα είναι θέσεις
   * Π.χ., αν η θέση deck[3][0] περιέχει 10,
   * τότε  ο Άσσος Κούπα είναι το
   * 10-ο χαρτί στη σειρά.
   */
  int deck[4][13] = {0};

  srand(time(0));
  shuffle(deck);
  deal(deck, face, suit);
  printf("\n");
  return 0;
}
```

Παράδειγμα: Τραπουλόχαρτα (2)

```
/* Συνάρτηση shuffle
 * Παίρνει ένα πίνακα που
 * αναπαριστά μια τράπουλα.
 * Βάζει το κάθε χαρτί σε μια
 * τυχαία θέση από 1 μέχρι 52.
 */
void shuffle(int deck[][13])
{
  int row, col, pos;

  /* for each of the 4*13 = 52
   * positions in the deck of cards */
  for(pos = 1; pos <= 52; pos++) {
    /* find a random card that
     * is not yet in the deck */
    do {
      row = rand() % 4;
      col = rand() % 13;
    } while(deck[row][col] != 0);
    /* put the card in the next position */
    deck[row][col] = pos;
  }
}
```

```
/* Συνάρτηση deal
 * Παίρνει έναν πίνακα με την τράπουλα, τα
 * ονόματα των χρωμάτων και τα ονόματα
 * των χαρτιών
 * Τυπώνει τα τραπουλόχαρτα με τη σειρά
 * που έχουν στην τράπουλα.
 */
void deal( int deck[][13],
           const char *const face[],
           const char *const suit[] ) {
  int card, row, col;

  for(card = 1; card <= 52; card++ ) {
    for(row = 0; row <= 3; row++) {
      for(col = 0; col <= 12; col++) {
        if(deck[row][col] == card) {
          printf( "%5s of %-8s%c",
                  face[col], suit[row],
                  (card % 3) == 0 ? '\n' : '\t' );
        }
      } }
  }
}
```

Παράδειγμα: Τραπουλόχαρτα (2)

Έξοδος

```
 Four of Hearts      King of Clubs      Seven of Diamonds
  Ten of Diamonds     Ace of Diamonds     Nine of Spades
  Ace of Clubs       King of Hearts     Eight of Diamonds
  Six of Spades      Jack of Clubs      Seven of Spades
Three of Diamonds    Jack of Diamonds   Seven of Clubs
  Six of Clubs        Ten of Hearts     Eight of Spades
 Jack of Spades     Deuce of Diamonds    King of Spades
Queen of Clubs      Deuce of Hearts     Three of Spades
 Five of Spades      Nine of Hearts       Ten of Clubs
  Six of Diamonds    Five of Clubs       Nine of Clubs
 Five of Diamonds    Four of Spades     Deuce of Spades
Three of Hearts      Four of Diamonds    Jack of Hearts
Eight of Clubs     Queen of Spades       Nine of Diamonds
 Four of Clubs       Five of Hearts     Queen of Hearts
Deuce of Clubs     Three of Clubs         Ace of Hearts
  Ace of Spades    Seven of Hearts       King of Diamonds
Eight of Hearts      Six of Hearts      Queen of Diamonds
  Ten of Spades
```

Δείκτες σε Συναρτήσεις

- Δείκτης σε συνάρτηση (function pointer)
  - Περιέχει τη διεύθυνση των εντολών της συνάρτησης
  - Όπως ένας δείκτης δείχνει στο πρώτο στοιχείο ενός πίνακα
- Πού χρειάζονται
  - Πολλές φορές δεν ξέρουμε ποιά συνάρτηση θέλουμε να καλέσουμε μέχρι να τρέξουμε το πρόγραμμα
    - Π.χ., αν η συνάρτηση που πρέπει να καλέσουμε καθορίζεται από την είσοδο
  - Πολλές φορές χρησιμοποιούμε συναρτήσεις βιβλιοθήκης οι οποίες πρέπει να καλούν διαφορετικές υποσυναρτήσεις ανάλογα με την είσοδο που τους δίνουμε
    - Π.χ., μια συνάρτηση minimum χρησιμοποιεί άλλη σύγκριση για strings και άλλη σύγκριση για αριθμούς
    - Τους δίνουμε ως παράμετρο τη συνάρτηση που θα καλέσουν ώστε να μη χρειάζεται να τις ξαναμεταγλωττίσουμε
  - Απαραίτητες πολλές φορές για κατανοητό, ευανάγνωστο ή γρήγορο κώδικα
    - Βοηθούν να αποφύγουμε τις πολλές `if` και `switch`

Δείκτες σε συναρτήσεις (2)

- Χρήσεις των δεικτών σε συναρτήσεις
  - Περνώνται ως ορίσματα συναρτήσεων
  - Αποθηκεύονται σε μεταβλητές και πίνακες
- Διαφορές με τους υπόλοιπους δείκτες
  - Μπορούμε να καλέσουμε τη συνάρτηση που δείχνει ένας δείκτης

```
(*function_pointer)(argument1, argument2)
```

    - Καλείται η συνάρτηση που δείχνει ο δείκτης με αυτά τα ορίσματα
  - Αναθέτουμε τη διεύθυνση μιας συνάρτησης (όχι μιας μεταβλητής)

```
function_pointer = &main;
```

Δήλωση δεικτών σε συναρτήσεις

Παραδείγματα δήλωσης

```
int (*funPtr1)(int, int);
int *funPtr2(int, int);
int (*funPtr3[10])(int, int);
```

- Η `funPtr1` είναι μια μεταβλητή με τύπο δείκτη σε συνάρτηση που παίρνει δύο ορίσματα τύπου `int` και επιστρέφει `int`
- Η `funPtr2` είναι μια *συνάρτηση* που παίρνει δύο ορίσματα `int` και επιστρέφει ένα δείκτη σε `int`
- Η `funPtr3` είναι ένας πίνακας που περιέχει 10 δείκτες: κάθε ένας δείχνει σε μια συνάρτηση που παίρνει δύο ορίσματα `int` και επιστρέφει `int`

Παράδειγμα

fun-ptr.c

```
#include<stdio.h>
int f(int a, int b)
{
  return a * b;
}

int main(void)
{
  int (*funptr)(int, int);

  funptr = &f;
  printf("Calling f(3,5) returns %d\n", (*funptr)(3, 5));
  return 0;
}
```

Παράδειγμα: Bubblesort

- Η bubblesort που είδαμε ταξινομεί ακέραιους (`int`) σε αύξουσα σειρά χρησιμοποιώντας τον τελεστή `<` για να τους συγκρίνει μεταξύ τους
- Πώς θα αλλάξει ώστε να παίρνει ως όρισμα μια συνάρτηση σύγκρισης και να καλεί αυτή;
- Πώς θα την καλέσουμε για να κάνει αύξουσα ή φθίνουσα ταξινόμηση;

Παράδειγμα: Bubblesort (2)

bubblesort2.c

```
#include<stdio.h>
#define SIZE 10
void bubble(int [], const int, int(*)(int, int));
int less(int, int);
int more(int, int);
void swap(int *, int *);

int main(void)
{
  int order, i, a[SIZE] =
    { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
  int (*comparison)(int, int);
  char *order_name;

  printf("Press 1 to use ascending order\n"
         "Press 2 to use descending order: ");
  do {
    order = getchar();
  } while (order != '1' && order != '2');

  if(order == '1') {
    comparison = &more;
    order_name = "ascending";
  } else {
    comparison = &less;
    order_name = "descending";
  }

  printf("Data in original order:\n");
  for(i = 0; i < SIZE; i++) {
    printf("%5d", a[i]);
  }
  bubble(a, SIZE, comparison);

  printf("\nData in %s order:\n",
         order_name);
  for(i = 0; i < SIZE; i++) {
    printf("%5d", a[i]);
  }
  printf("\n");
  return 0;
}
```

Παράδειγμα: Bubblesort (3)

bubblesort2.c

```
void bubble(int work[], const int size,
            int (*compare)(int, int))
{
  int pass, i;

  for(pass = 1; pass < size; pass++) {
    for(i = 0; i < size - 1; i++) {
      if( (*compare)(work[i], work[i + 1]) ) {
        swap(&work[i], &work[i + 1]);
      }
    }
  }
}

void swap(int *p1, int *p2)
{
  int hold = *p1;
  *p1 = *p2;
  *p2 = hold;
}
```

```
int less(int a, int b)
{
  return a < b;
}

int more(int a, int b)
{
  return a > b;
}
```

Παράδειγμα: Bubblesort (4)

Έξοδος (1)

```
Press 1 to use ascending order
Press 2 to use descending order: 1
Data in original order:
    2    6    4    8   10   12   89   68   45   37
Data in ascending order:
    2    4    6    8   10   12   37   45   68   89
```

Έξοδος (2)

```
Press 1 to use ascending order
Press 2 to use descending order: 2
Data in original order:
    2    6    4    8   10   12   89   68   45   37
Data in descending order:
   89   68   45   37   12   10    8    6    4    2
```