ΕΙΣΑΓΩΓΉ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΌ

- Ορίσματα γραμμής εντολών
- Δείκτες/pointer σε συναρτήσεις
- Αρχεία

Ενώσεις (Unions) -1

Μία ένωση (union) μπορεί να περιέχει αντικείμενα διαφορετικών τύπων και μεγεθών

π.χ. union u_tag {

```
int ival;
float fval;
char *sval;
} u;
```

Ενώσεις (Unions) -2

```
if (utype == INT)
 printf(“%d\n”, u.ival);
else if (utype==FLOAT)
  printf(“%f\n”, u.fval);
else if (utype==STRING)
  printf(“%s\n”, u.sval);
```

Στις ενώσεις επιτρέπεται απόδοση τιμής ή αντιγραφή σαν ενότητα, λήψη διεύθυνσης, προσπέλαση μέλους.

Η αρχική τιμή πρέπει να έχει τύπο ίδιο με αυτό του πρώτου της μέλους.

Ορίσματα Γραμμής Διαταγών (1)

```
#include <stdio.h>
main(int argc, char * argv[])
{
        int i;
        for (i=1; i<argc; i++)
        printf("%s%s", argv[i], (i<argc-1)? " " : " \n");
        return 0;
 }
```

Ορίσματα Γραμμής Διαταγών (2)

```
#include <stdio.h>
main(int argc, char * argv[])
{
        while (--argc >0)
        printf("%s%s", *++argv, (argc>1)?"  ": "\n ");
        return 0;
 }
```

Δείκτες σε Συναρτήσεις

Στην C η συνάρτηση δεν είναι μεταβλητή. Μπορούμε να ορίσουμε δείκτες σε συναρτήσεις, που μπορούν να αποδίδονται σαν τιμές, να τοποθετούνται σε πίνακες, να μεταβιβάζονται σε συναρτήσεις και να επιστρέφονται από αυτές.

Pointers to Functions στη C

- Τι είναι pointer σε συνάρτηση;
- Γιατί τους χρησιμοποιούμε;
- Βασική σύνταξη
- Παραδείγματα & συνηθισμένες χρήσεις (callbacks, πίνακες συναρτήσεων)

Δείκτες σε συναρτήσεις

Παράδειγμα: bubblesort

- Η συνάρτηση `bubble` παίρνει είσοδο δείκτη συνάρτησης
  - `bubble` καλεί αυτή τη βοηθητική συνάρτηση
  - Αυτό καθορίζει την αύξουσα ή φθίνουσα ταξινόμηση
- Το όρισμα στην ταξινόμηση φυσαλίδων για τον δείκτη συνάρτησης:

```
int ( *compare )( int a, int b )
```

λέει στο bubblesort να αναμένει έναν δείκτη σε μια συνάρτηση που παίρνει δύο ακέραιους και επιστρέφει έναν ακέραιο

Αν οι παρενθέσεις παρέλειπαν:

```
int *compare( int a, int b )
```

- Ορίζει μια συνάρτηση που λαμβάνει δύο ακέραιους αριθμούς και επιστρέφει έναν δείκτη σε έναν ακέραιο

Τι είναι pointer σε συνάρτηση;

- Στη C, οι συναρτήσεις έχουν διεύθυνση στη μνήμη, όπως και οι μεταβλητές.
- Ένας pointer to function είναι μια μεταβλητή που κρατάει τη διεύθυνση μιας συνάρτησης.
- Μέσω αυτού:
  - Μπορούμε να καλούμε διαφορετικές συναρτήσεις δυναμικά (run-time).
  - Μπορούμε να περνάμε «συμπεριφορά» ως όρισμα (callbacks).
  - Μπορούμε να φτιάχνουμε πίνακες από «στρατηγικές» (π.χ. διαφορετικοί αλγόριθμοι).

Βασική μορφή δήλωσης

- Γενική μορφή: return_type (*pointer_name)(parameter_types);
- Παράδειγμα:

```
int add(int a, int b)
 {
 return a + b;
}
int (*fp)(int, int);  // δήλωση pointer σε συνάρτηση
```

- fp είναι pointer σε συνάρτηση που παίρνει δύο int και επιστρέφει int.

Ανάθεση & κλήση

- Ανάθεση διεύθυνσης συνάρτησης:

fp = &add; // ή απλά fp = add;

Κλήση μέσω pointer:

int result1 = fp(2, 3); // έμμεση κλήση

int result2 = (*fp)(2, 3); // ισοδύναμο

- - Η έκφραση add «αποσυντίθεται» σε pointer στη συνάρτηση, οπότε το & είναι προαιρετικό.
- - Αντίστοιχα, το * στην κλήση είναι προαιρετικό.

Πλήρες απλό παράδειγμα

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

int main(void) {
int (*fp)(int, int);  // pointer σε συνάρτηση
 fp = add;            // ανάθεση
  int x = 5, y = 7;
int sum = fp(x, y);   // κλήση της add μέσω fp
   printf("Sum = %d\n", sum);     return 0;}
```

Pointers σε functions ως ορίσματα (callbacks)

Μπορούμε να περάσουμε pointers σε συνάρτηση ως παραμέτρους σε άλλες συναρτήσεις:

```
void apply_and_print(int (*op)(int, int), int a, int b) {
    int result = op(a, b);
    printf("Result = %d\n", result);
}
int add(int a, int b) { return a + b; }
int mul(int a, int b) { return a * b; }
int main(void) {
    apply_and_print(add, 3, 4); // χρησιμοποιεί add
    apply_and_print(mul, 3, 4); // χρησιμοποιεί mul
}
```

Δείκτες σε συναρτήσεις

Δείκτης στη συνάρτηση

- Περιέχει διεύθυνση συνάρτησης
- Παρόμοιο με το πώς το όνομα του πίνακα είναι η διεύθυνση του πρώτου στοιχείου ..
- Το όνομα της συνάρτησης είναι η αρχική διεύθυνση του κώδικα που ορίζει τη συνάρτηση

Οι δείκτες στη συνάρτηση μπορούν να:

- Περαστούν σε συναρτήσεις
- Αποθηκευτούν σε πίνακας
- Ανατεθούν σε άλλες δείκτες συνάρτησης.

Typedef για function pointers

```
typedef int (*operation)(int, int);
int add(int a, int b) { return a + b; }
int sub(int a, int b) { return a - b; }
int main(void) {
    operation op;
    op = add;
    printf("%d\n", op(10, 3)); // 13
    op = sub;
    printf("%d\n", op(10, 3)); // 7
}
```

Πίνακες από pointers σε συναρτήσεις

```c
#include <stdio.h>
int add(int a, int b) { return a + b; }
int sub(int a, int b) { return a - b; }
int mul(int a, int b) { return a * b; }
typedef int (*operation)(int, int);
int main(void) {
    operation ops[3] = { add, sub, mul };
    int a = 10, b = 5;
    printf("add: %d\n", ops[0](a, b));
    printf("sub: %d\n", ops[1](a, b));
    printf("mul: %d\n", ops[2](a, b));
    return 0;
}
```


- Χρήσιμο για μενού επιλογών ή πίνακες «στρατηγικών».

Function pointers σε struct

```c
typedef struct {
    const char *name;
    int (*op)(int, int);
} Command;
int add(int a, int b) { return a + b; }
int main(void) {
    Command c = {"add", add};
    printf("%s(2, 3) = %d\n", c.name, c.op(2, 3));
    return 0;
}
```

- Χρήσιμο για πίνακες από «εντολές» ή προσομοίωση OOP.

Σημειώσεις για συμβατότητα τύπων

- Ο τύπος του pointer σε συνάρτηση πρέπει να ταιριάζει ακριβώς με τον τύπο της συνάρτησης:
  - Τύποι παραμέτρων
  - Τύπος επιστροφής
  - const / volatile qualifiers στις παραμέτρους
- Διαφορετικά → undefined behavior.
- Απαγορεύεται η αριθμητική σε function pointers (π.χ. fp + 1).

Συνηθισμένα λάθη

- Ξεχνάμε παρενθέσεις στη δήλωση:

int *fp(int, int); // ΔΕΝ είναι pointer σε συνάρτηση, είναι δήλωση συνάρτησης

int (*fp)(int, int); // σωστό

- Παίρνουμε pointer σε συνάρτηση static από άλλο αρχείο (δεν είναι ορατή).
- Χρήση λάθος υπογραφής (π.χ. callback με λάθος παραμέτρους).

Mini calculator με function pointers

```
`#include <stdio.h>

typedef int (*operation)(int, int);

int add(int a, int b) { return a + b; }

int sub(int a, int b) { return a - b; }

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

int main(void) {
    operation ops[] = { add, sub, mul };
    char symbols[] = { '+', '-', '*' };
    int choice, a, b;
    printf("0: +, 1: -, 2: *\n");
    printf("Επιλογή: ");
    scanf("%d", &choice);
```

```
    printf("Δώσε δύο αριθμούς: ");
    scanf("%d %d", &a, &b);
if (choice < 0 || choice > 2) {
        printf("Άκυρη επιλογή\n");
        return 1;
    }
    int result = ops[choice](a, b);
    printf("%d %c %d = %d\n", a, symbols[choice], b, result);
    return 0;
}
```

Πότε αξίζει να τους χρησιμοποιήσω;

- - Όταν θέλω:
- - Callbacks (sort, events, signal handlers).
- - Πίνακες από λειτουργίες (π.χ. emulator, interpreters, state machines).
- - Pluggable πολιτικές/στρατηγικές (π.χ. διαφορετικοί αλγόριθμοι).
- - Σε απλό κώδικα ίσως να είναι περιττή πολυπλοκότητα.

- Function pointer = μεταβλητή που δείχνει σε συνάρτηση.
- Μορφή: return_type (*name)(params);
- Μπορούμε:
- Να τους περνάμε ως ορίσματα (callbacks).
- Να τους αποθηκεύουμε σε arrays & structs.
- Να χρησιμοποιούμε typedef για καθαρότερη σύνταξη.
- - Προσοχή σε:
- Συμβατότητα τύπων
- Σωστή δήλωση (παρενθέσεις!).

```c
#include <stdio.h>
#define N 10

/* Απλός αλγόριθμος διατάξεως με χρήση bubblesort
και με χρήση δείκτη σε συνάρτηση*/

void read(int ar[], int size);
void display(int ar[], int size);
void bubblesort(int ar[], int size, int (*comp)(int,
int));
int unsorted(int ar[], int size);
int afksousa(int x,int y);
int fthinousa(int x,int y);
int bsearch(int ar[], int left, int right, int element);
```

```c
int main()
{   int ar[N];
    int x;
    int result;
    read(ar, N);
    if (unsorted(ar,N)==1)  bubblesort(ar, N, afksousa);
    display(ar,N);
    printf("\n");
    printf("Give element: ");
    scanf("%d", &x);
    result=bsearch(ar, 0, N-1, x);
    if (result==-1) printf("The element does not exist.");
    else printf("It exists in %d position\n", result);
    bubblesort(ar, N, fthinousa);
    display(ar,N);
}
```

```c
void read(int ar[], int size)
{
    printf("Give elements: ");
    int i;
    for (i=0; i<size; i++) scanf("%d", &ar[i]);
}

void display(int ar[], int size)
{
    printf("The elements are: ");
    int i;
     for (i=0; i<size; i++) printf("%d ", ar[i]);
}
```

```
int afksousa(int x, int y)
{
   return x>y;
}

int fthinousa(int x, int y)
{
   return x<y;
}
```

```
void helper(int ar[], int up,  int (*comp)(int, int))
{
    int j;
```

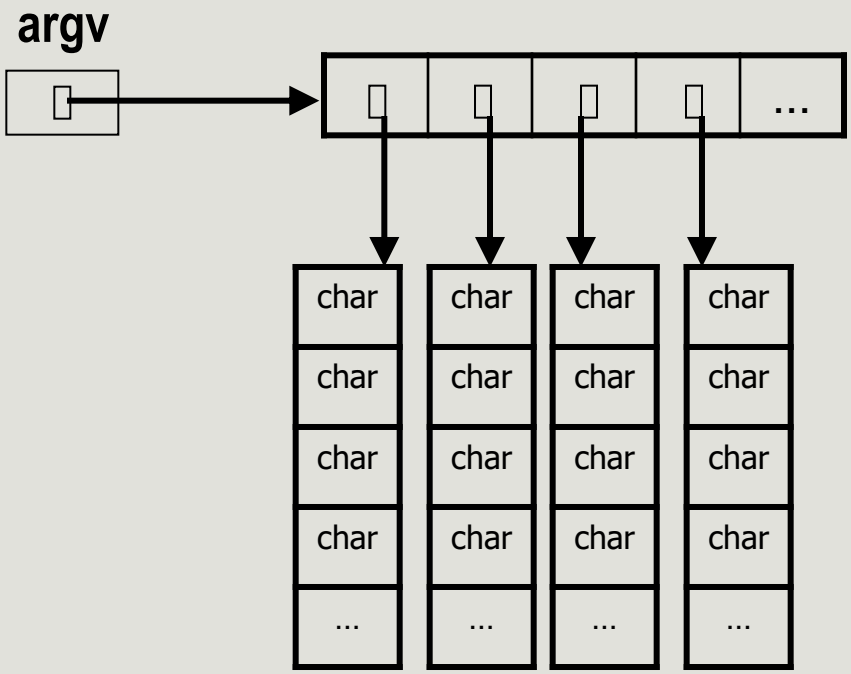
```
    for (j=0; j<up; j++)
    {
        if (comp(ar[j], ar[j+1])>0)
        {
            int temp;
            temp=ar[j];
            ar[j]=ar[j+1];
            ar[j+1]=temp;
        }
    }
}
```

```
void bubblesort(int ar[], int size, int (*comp)(int, int))
{
    int i;
     for (i=N-1; i>=1; i--) helper(ar, i, comp);
}

int unsorted(int ar[], int size)
{
    int i;

    for (i=0; i<=size-2; i++)
       if (ar[i]>ar[i+1])  return 1;
        return 0;
}
```

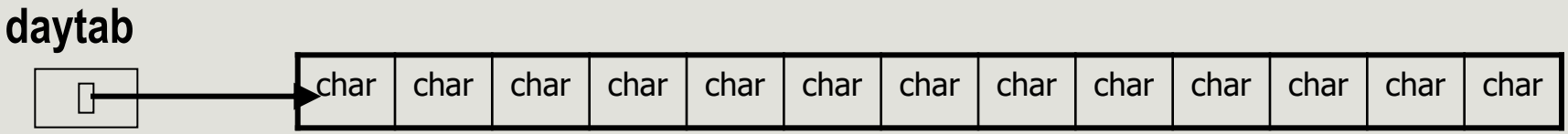
Παραδείγματα Δηλώσεων

char **argv	δείκτης σε δείκτη σε char
int (*daytab)[13]	δείκτης σε πίνακα[13] int
int *daytab[13]	πίνακας[13] με δείκτες σε int
void *comp()	συνάρτηση που επιστρέφει δείκτη σε void
void (*comp)()	δείκτης σε συνάρτηση που επιστρέφει void
char (*(*x())[])()	συνάρτηση που επιστρέφει δείκτη σε πίνακα δεικτών σε συνάρτηση που επιστρέφει char
char (*(*x[3])())[5]	πίνακας [3] με δείκτες σε συνάρτηση που επιστρέφει δείκτη σε πίνακα [] char

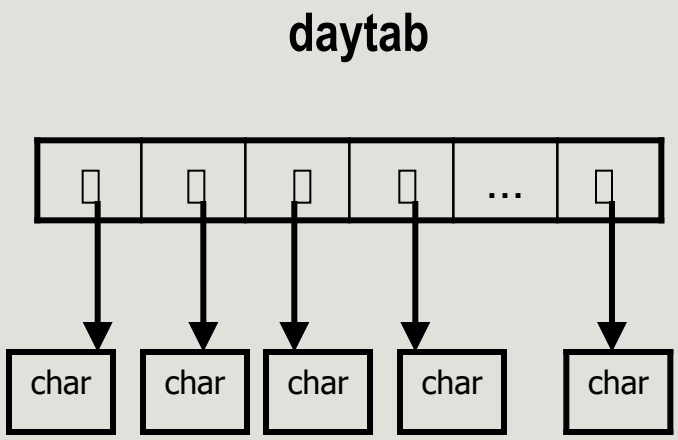
char **argv


argv
...
char
char
char
char
...
char
char
char
char
...
char
char
char
char
...
char
char
char
char
...

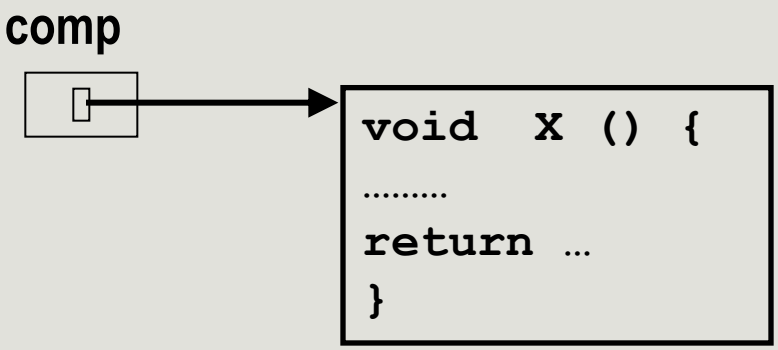
char (*daytab)[13]



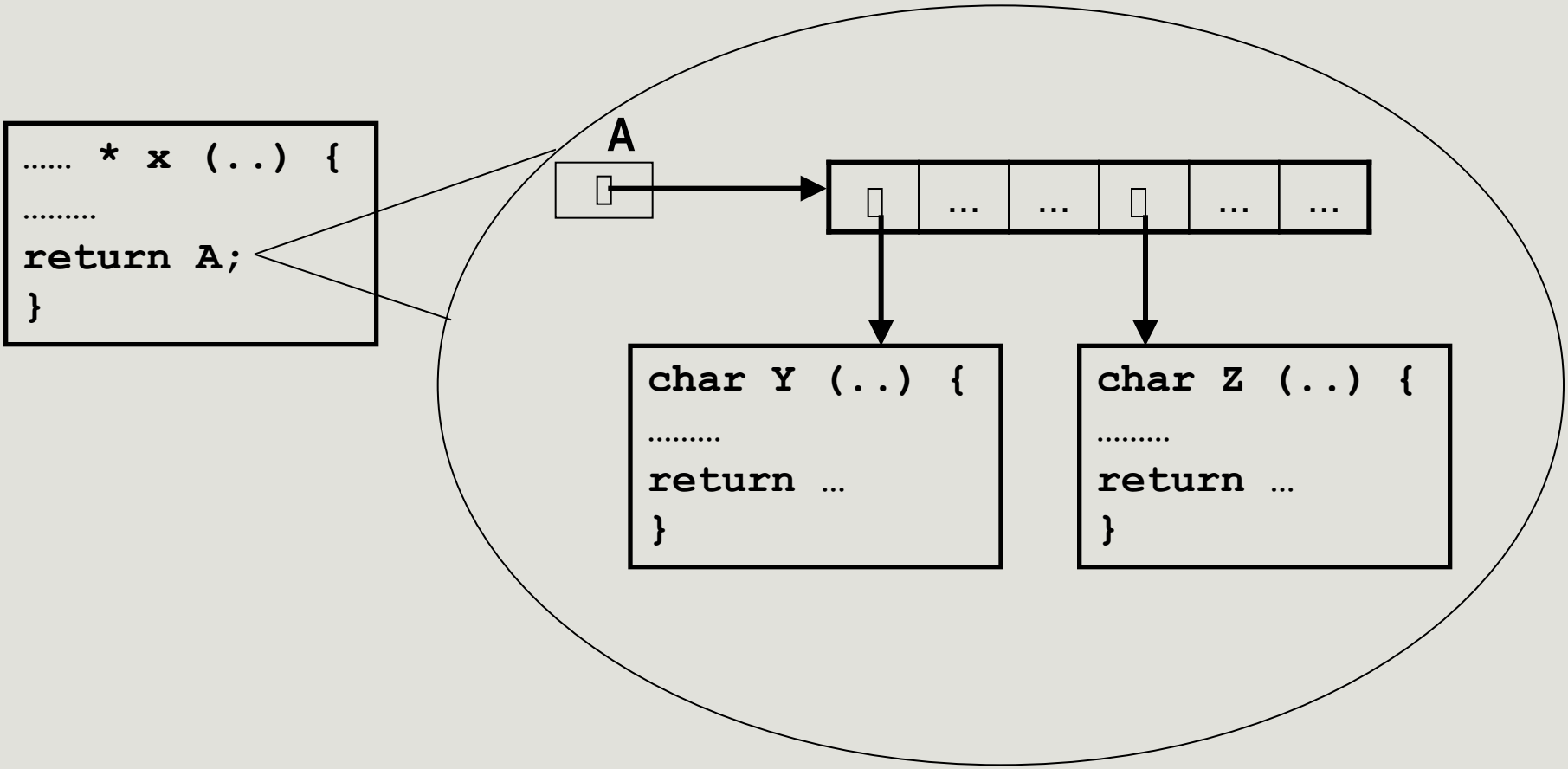
char *daytab[13]



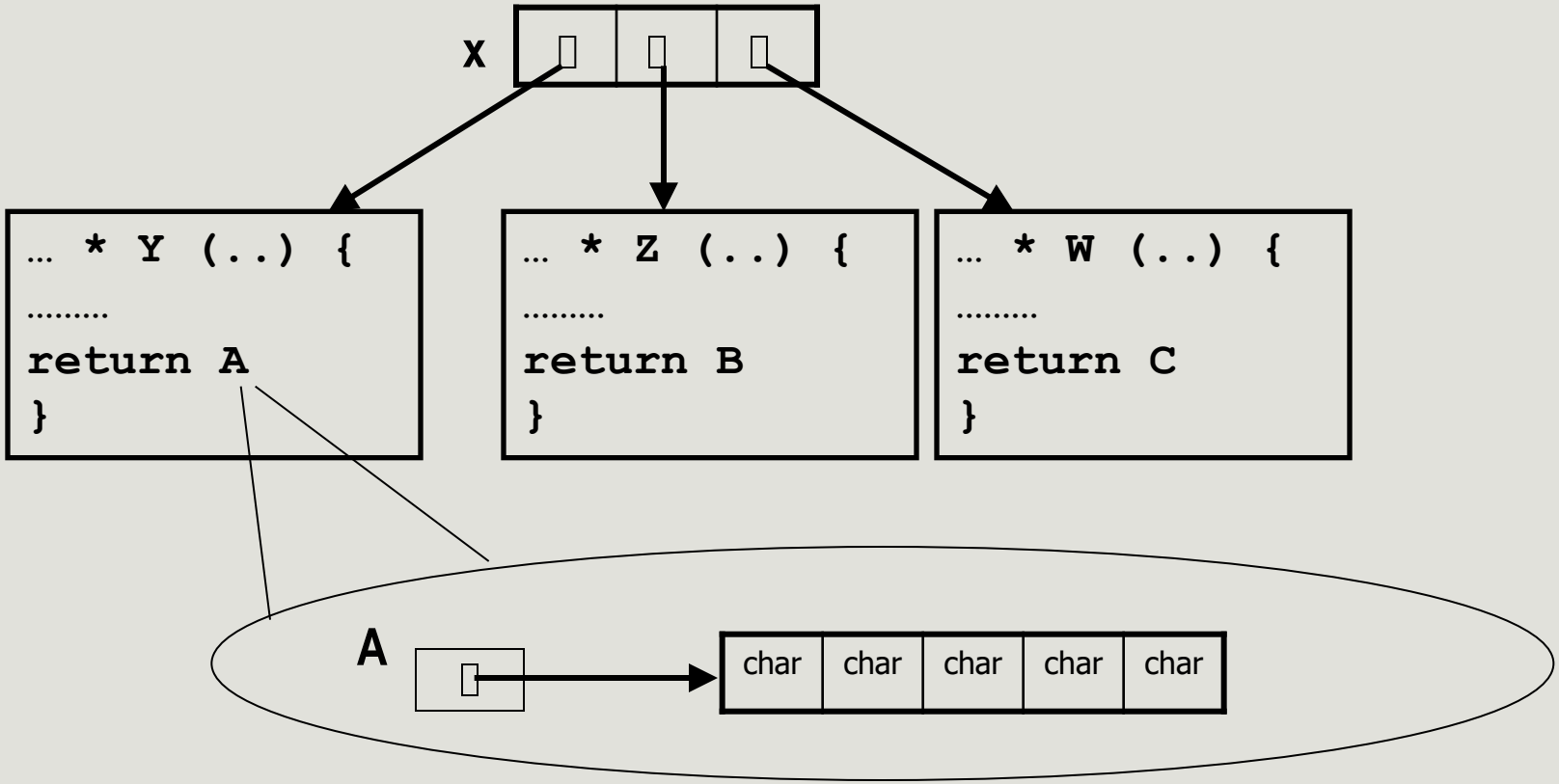
void (*comp)()



char (*(*x())[])()



char (*(*x[3])())[5]



Επεξεργασία Αρχείων στη C

- Με τη χρήση `FILE *` αναπαριστάνουμε δείκτη σε ένα αρχείο.
- `fopen` χρησιμοποιείται για να ανοίγει ένα αρχείο. Επιστρέφει την ειδική τιμή `NULL` για να δείξει ότι δεν μπορεί να ανοίξει ένα αρχείο.

```
FILE *fptr;
char filename[]= "file2.dat";
fptr= fopen (filename,"w");
if (fptr == NULL) {
    fprintf (stderr, “ERROR”);
     /* DO SOMETHING */
}
```

Τρόποι Ανοίγματος Αρχείου

- FILE * fopen(const char * filename, const char * mode)
- Το δεύτερο όρισμα της `fopen` καθορίζει τον τρόπο με τον οποίο ανοίγουμε ένα αρχείο.

  Υπάρχουν οι εξής επιλογές:
- "r" ανοίγει ένα αρχείο για διάβασμα
- "w" ανοίγει ένα αρχείο για εγγραφή
- "a" ανοίγει ένα αρχείο για προσθήκη / εγγραφή
- “b” δυαδικά δεδομένα (binary)
- rb, wb, ab, r+, w+, a+, rb+,wb+,ab+

Διαχείριση Αρχείων

- Κλείσιμο αρχείων

  int fclose (FILE * fp);
- Είσοδος – έξοδος χαρακτήρων

  int putc(int c, FILE * fp);

  (ισοδύναμα fputc())

  int getc(FILE * fp);

  (ισοδύναμα fgetc())
- Είσοδος – έξοδος συμβολοσειρών

  int fputs (const char * s, FILE * fp);

  char * fgets (char * s, int n, FILE * fp);

- int *feof(FILE* *fp)

  Not 0: end of file

- void *rewind* (FILE *fp);

- int *fflush(FILE* *fp)

  0: success, EOF: error

- int *fseek*(FILE *fp, long int apostasi, int thesi)

  thesi=SEEK_SET (0), SEEK_CUR (1), SEEK_END (2)

  int *ftell*(FILE *fp)

- int fread(void *buffer, int arbytes, int fores, FILE *fp)

  Διάβασε fores στοιχεία, μεγέθους arbytes το καθένα, στο buffer array

- int fwrite(void *buffer, int arbytes, int fores, FILE *fp)

  Γράψε fores στοιχεία, μεγέθους arbytes το καθένα, από το buffer array

Παράδειγμα

```
 #include <stdio.h>
void filecopy(FILE *, FILE *);

int main (int argc, char *argv[])
{ FILE *fp;
 if (argc==1) filecopy(stdin, stdout);
 else
 while (--argc>0)   if ((fp=fopen(*++argv, “r”))==NULL)
                          { printf(“cat: δεν μπορώ να ανοίξω το %s\n”, argv);
                            return 1; }
  else {filecopy(fp, stdout); fclose(fp);}
 return 0;  }

void filecopy(FILE *ifp, FILE *ofp)
{            int c;
 while ((c=getc(ifp))!=EOF)  putc(c,ofp);
}
```

Γράφοντας σε Αρχείο με *fprintf*

- Η fprintf δουλεύει όπως η printf και η sprintf με την διαφορά ότι το πρώτο όρισμα είναι ένας δείκτης σε αρχείο.

```
FILE *fptr;
fptr= fopen ("file.dat","w");
/* Check it's open */
fprintf (fptr,"Hello World!\n");
```

- Θα μπορούσαμε επίσης να διαβάσουμε αριθμούς από ένα αρχείο χρησιμοποιώντας fscanf – αλλά υπάρχει και καλύτερος τρόπος.

Διαβάζοντας από Αρχείο με *fgets*

- *fgets* είναι ένας καλύτερος τρόπος ανάγνωσης από αρχείο
- Μπορούμε να διαβάσουμε από αρχείο με *fgets*

```
FILE *fptr;
char line [1000];
/* Open file and check it is open */
while (fgets(line,1000,fptr) != NULL) {
   printf ("Read line %s",line);
```

fgets έχει 3 ορίσματα, μία συμβολοσειρά, ένα μέγιστο αριθμό χαρακτήρων προς ανάγνωση και ένα δείκτη αρχείου. It returns NULL if there is an error (such as EOF)

fgets(char*s, int n, FILE *fstream); (διαβάζει το πολύ n-1 χαρακτήρες σταματώντας αν συναντήσει χαρακτήρα νέας γραμμής)

Κλείνοντας ένα Αρχείο

- Μπορούμε να κλείσουμε ένα αρχείο απλώς χρησιμοποιώντας fclose και το δείκτη αρχείου. Ακολουθεί παράδειγμα "hello files".

```
FILE *fptr;
char filename[]= "myfile.dat";
fptr= fopen (filename,"w");
if (fptr == NULL) {
    printf ("Cannot open file to write!\n");
    exit(-1);
}
fprintf (fptr,"Hello World of filing!\n");
fclose (fptr);
```

Παράδειγμα (3)

Να υλοποιηθούν και να χρησιμοποιηθούν οι ακόλουθες συναρτήσεις:

1. int **createRandFile**(char *fileName,int N): Δημιουργεί ένα αρχείο με όνομα fileName στο οποίο γράφει N τυχαίους αριθμούς. Επιστρέφει 1 σε επιτυχία και -1 σε αποτυχία
2. int ***loadData**(char *fileName,int N): Διαβάζει από το αρχείο fileName N τυχαίους αριθμούς και τους επιστρέφει σε ένα δείκτη ακεραίων.
3. int **writeArray2File**(char *fileName,int *array,int N): Γράφει τους αριθμούς του διανύσματος array στο αρχείο fileName. Επιστρέφει 1 σε επιτυχία και -1 σε αποτυχία.

```
int createRandFile(char *fileName,int N)
 {
    FILE *fp;
    int i;

    fp = fopen(fileName,"w");
    if (fp)
    {
       srand((unsigned)time(NULL));
       for (i=0;i<N;i++)
       {
          fprintf(fp,"%d\n",(MAX*rand())/RAND_MAX);
       }
       fclose(fp);
    return 1;
    }
    else
       return -1;
 }
```

stdlib.h → srand, rand

time.h -> time()

```
int *loadData(char *fileName,int N)
 {
    int *array = (int*)malloc(sizeof(int)*N);
    FILE *fp;
    int i;

    fp = fopen(fileName,"r");
    if (fp)
    {
       for (i=0;i<N;i++)
       {
          fscanf(fp,"%d",&array[i]);
       }
       fclose(fp);
       return array;
    }
    else
       return NULL;
 }
```

```
int writeArray2File(char *fileName,int *array,int N)
 {
    FILE *fp;
    int i;

    fp = fopen(fileName,"w");
    if (fp)
    {
       for (i=0;i<N;i++)
       {
          fprintf(fp,"%d\n",array[i]);
       }
       fclose(fp);
       return 1;
    }
    else
    return -1;
 }
```