ΕΝΩΣΕΙΣ (unions)

Οι ενώσεις έχουν παρόμοια σύνταξη με τις δομές με τη διαφορά ότι τα πεδία μιας ένωσης αντιστοιχούν στην ίδια θέση μνήμης αντί σε διαφορετικές θέσεις μνήμης μιας δομής. Η λέξη κλειδί union καθορίζει μια ένωση όπως ακριβώς η λέξη κλειδί struct καθορίζει μια δομή. Η πρόσβαση στα πεδία μιας ένωσης γίνεται όπως και με τις δομές. Παράδειγμα ένωσης:

```
union choice {      /* στην ένωση αυτή προβλέπεται η αποθήκευση: */
  short  digit;     /* ενός μικρού ακεραίου των 2 bytes */
  double bignum;    /* ή ενός πραγματικού αριθμού των 8 bytes */
  char   letter;    /* ή ενός χαρακτήρα του 1 byte */
};
```

Ορισμός μεταβλητών τύπου **choice**:

```
union choice c1;       /* variable of choice type */
union choice save[10]; /* array of 10 union elements */
union choice *pu;      /* pointer to a variable of choice type */
```

Με τη δήλωση της μεταβλητής c1 ο μεταγλωττιστής δεσμεύει αρκετό χώρο στη μνήμη ώστε να χωρέσει η μεγαλύτερη από τις διαφορετικές δυνατότητες. Στην προκειμένη περίπτωση είναι η double που απαιτεί 8 bytes. Για τον πίνακα save δεσμεύονται 10 x 8 = 80 bytes ενώ για τον δείκτη pu φυσικά δεσμεύονται 4 bytes.
Να θυμάστε ότι είναι ευθύνη του προγραμματιστή να παρακολουθεί τον τύπο δεδομένων που έχει αποθηκευθεί ανά πάσα στιγμή στη θέση μνήμης μιας ένωσης. Παράδειγμα:

```
c1.digit = 23;  /* αποθήκευση του 23 (χρησιμοποιούνται 2 bytes) */
c1.bignum = 141.0; /* εκκαθάριση του 23, αποθήκευση του 141.0 */
c1.letter = 'a'; /* εκκαθάριση του 141.0, αποθήκευση του 'a' */

c1.letter -= 32; /* τώρα έχει το 'A' */
fl_num = 3.0 * c1.bignum;  /* ERROR γιατί η c1 δεν έχει double */
```

οι pointers σε ενώσεις είναι όπως και στις δομές:

```
pu = &c1;
ch = pu->letter;  /* <==> ch = c1.letter; */
```

typedef

Με την typedef μπορούμε να δίνουμε μνημονικά ονόματα σε τύπους δεδομένων. Μοιάζει σε μεγάλο βαθμό με την #define ως προς αυτήν την άποψη αλλά με τις εξής τρεις διαφορές:

1. Σε αντίθεση με την #define, η typedef περιορίζεται στο να δίνει συμβολικά ονόματα σε τύπους δεδομένων μόνο.
2. Η typedef εκτελείται από τον μεταγλωττιστή, όχι από τον προεπεξεργαστή.
3. Η typedef είναι πιο ευέλικτη από την #define ως προς την ονοματοδοσία σε διάφορους (απλούς ή σύνθετους) τύπους δεδομένων.

Για να δώσουμε ένα μνημονικό όνομα σε έναν τύπο, ορίζουμε μια μεταβλητή με το όνομα αυτό και βάζουμε στην αρχή τη λέξη κλειδί typedef. Παραδείγματα:

```
typedef float real;
typedef char * String1;
typedef char String2[100];  /* Πίνακας 100 χαρακτήρων */

real x, y[10], *p1;
String1 name;        /* <==>   char *name;  */
String2 titlos;      /* <==>   char titlos[100];  */
```

```
typedef struct _iobuf FILE;  /* ή   #define FILE struct _iobuf  */
typedef int arr2D[100][100];
typedef struct cmplx {  /* το cmplx είναι ετικέτα (όνομα) δομής */
       float real;
       float imag;
}  COMPLEX;    /* COMPLEX είναι το συνώνυμο του struct cmplx */

FILE *ifp, *ofp;
COMPLEX cnum2, cnums[10][20];  /* Δεν χρειάζεται το struct */
arr2D map;  /* το map είναι διδιάστατος 100x100 πίνακας ακεραίων */

typedef union number {
    int num;
    char literal[10];
} NUM;
```

Παράδειγμα σύνθετου τύπου δεδομένων:

```
typedef char (*FRPTC())[5];  /* ο τύπος FRPTC είναι συνάρτηση που
       επιστρέφει δείκτη σε πίνακα 5 χαρακτήρων  */

FRPTC  func1, func2;
```

ΔΥΝΑΜΙΚΗ ΔΕΣΜΕΥΣΗ ΜΝΗΜΗΣ

Η συνάρτηση malloc:

```
void *malloc(unsigned int size);
```

όπου size είναι το μέγεθος της μνήμης που αιτούμαστε σε bytes. Η malloc επιστρέφει δείκτη στη δεσμευμένη μνήμη (σε void) τον οποίο μετατρέπουμε στον κατάλληλο τύπο δείκτη με type casting. Αν αποτύχει η δέσμευση μνήμης τότε η malloc επιστρέφει μηδενικό δείκτη (NULL). Η malloc δεν αρχικοποιεί τη μνήμη που δεσμεύει. Αν θέλουμε η δεσμευμένη μνήμη να έχει αρχικοποιηθεί στο μηδέν τότε μπορούμε να χρησιμοποιήσουμε τη συνάρτηση calloc:

```
void *calloc(unsigned int nr_of_items, unsigned int size);
```

όπου nr_of_items είναι το πλήθος των στοιχείων για τα οποία ζητάμε δέσμευση μνήμης και size είναι το μέγεθος των στοιχείων αυτών σε bytes. Παράδειγμα:

```
int *A, *B;
A = (int *) malloc(400);  /* η μνήμη περιέχει "σκουπίδια" */
B = (int *) calloc(100,4);  /* η μνήμη αρχικοποιείται στο 0 */
```

ή αν αφήσουμε να γίνει αυτόματα η μετατροπή του τύπου του δείκτη (δεν συνιστάται):

```
int *A = malloc(400), *B;
B = calloc(100,sizeof (int));
```

Η συνάρτηση realloc αλλάζει το μέγεθος του μπλοκ μνήμης στο οποίο δείχνει ένας pointer:

```
void *realloc(void *ptr, unsigned int size);
```

όπου ptr είναι ο δείκτης στο ήδη δεσμευμένο (με malloc, calloc ή realloc) μπλοκ μνήμης και size είναι το νέο συνολικό μέγεθος της μνήμης που αιτούμαστε σε bytes. Αν size = 0 και ο ptr δείχνει σε υπάρχον μπλοκ μνήμης, η μνήμη αποδεσμεύεται και η συνάρτηση επιστρέφει NULL. Όπως και με τις δύο προηγούμενες συναρτήσεις, η realloc επιστρέφει δείκτη στη νέα δεσμευμένη μνήμη.

Όταν δεν χρειαζόμαστε πλέον τη δεσμευμένη μνήμη, πρέπει να ενημερώσουμε το λειτουργικό σύστημα ότι την αποδεσμεύουμε ώστε να μπορέσει να τη χρησιμοποιήσει για τις ανάγκες άλλων προγραμμάτων. Αυτό ακριβώς κάνει η συνάρτηση free:

```
void *free(void *ptr);
```

Παράδειγμα:

```
int *A;
A = (int *)malloc(100*sizeof(int));
...
free(A);
```

Άλλες συναρτήσεις διαχείρισης μνήμης είναι οι εξής (δηλώνονται στο <string.h>):

```
void *memcpy(void *dest, const void *src, unsigned int size);
```

η οποία αντιγράφει size bytes από τη μνήμη στην οποία δείχνει ο src στη μνήμη στην οποία δείχνει ο δείκτης dest. Η memcpy διαφέρει από την strcpy στο ότι δεν σταματάει την αντιγραφή μόλις συναντήσει τερματικό χαρακτήρα '\0'. Είναι κατάλληλη για αντιγραφή μεγάλου όγκου δεδομένων καθώς πιο γρήγορη από χρήση επαναληπτικών βρόχων.

```
void *memmove(void *dest, const void *src, unsigned int size);
```

η οποία είναι παρόμοια με τη memcpy αλλά πιο εύρωστη (και πιο αργή) καθώς κάνει ελέγχους για τυχόν επικάλυψη μνήμης ώστε να εξασφαλίσει τη σωστή αντιγραφή των δεδομένων.

```
void *memcmp(const void *p1, const void *p2, unsigned int size);
```

η οποία συγκρίνει size bytes από τη μνήμη στην οποία δείχνει ο p1 με τα αντίστοιχα bytes της μνήμης στην οποία δείχνει ο p2. Η συμπεριφορά και η τιμή επιστροφής της memcmp είναι ανάλογη της strcmp χωρίς όμως να σταματάει μόλις συναντήσει τον τερματικό χαρακτήρα '\0'.

Ο ΠΡΟΕΠΕΞΕΡΓΑΣΤΗΣ ΤΗΣ C

Οι δύο οδηγίες του προεπεξεργαστή προς τον μεταγλωττιστή που έχουμε συναντήσει μέχρι στιγμής είναι η συμπερίληψη αρχείων (#include) και οι μακρο-αντικαταστάσεις (#define).Π.χ.

```
#include <stdio.h>      /* συμπεριλ. βιβλιοθήκη συστήματος */
#include "myheader.h"   /* συμπεριλ. δικές μας συναρτήσεις */
#define SIZE 100        /* μακρο-αντικατάσταση του SIZE με το 100 */
#define SQUARE(X) ((X)*(X)) /* μακρο-συνάρτηση υπολ. τετραγώνου */
π.χ. SQUARE(a-b) πρώτα θέτει X ≡ a-b και μετά αντικαθίσταται η
          SQUARE(a-b) από: ((a-b)*(a-b))   δηλαδή  από (a-b)²
 αν όμως δεν βάλουμε εσωτερικές παρενθέσεις: π.χ. SQUARE(X) (X*X)
   τότε   SQUARE(a-b) --> (a-b*a-b) = a – ab – b ≠ (a-b)²
 και αν δεν βάλουμε εξωτερικές παρενθέσεις: π.χ. SQUARE(X) (X)*(X)
   τότε   c / SQUARE(a-b) --> c/(a-b)*(a-b) = c ≠ c/(a-b)²
#define MAX(X,Y) (X > Y ? X : Y)
```