Προγραμματισμός Υπολογιστών

Ορισμός τύπων και ενώσεις

Νικόλαος Ζ. Ζάχαρης
Καθηγητής

Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Ορισμός νέων τύπων δεδομένων

Η δήλωση **typedef** μας επιτρέπει να ορίσουμε νέους τύπους δεδομένων οι οποίοι βασίζονται σε βασικούς τύπους δεδομένων καθώς και σε δομές. Η σύνταξη της typedef έχει ως εξής :

typedef Τύπος_Δεδομένων Νέο_Ονομα

Πρόκειται για τη δημιουργία ένός νέου ονόματος αναφοράς σε ένα τύπο δεδομένων. Παράδειγμα μπορούμε να ορίσουμε το όνομα dekadikos αντί του float.

typedef float dekadikos;

Άλλο παράδειγμα είναι

typedef unsigned int WORD;

Τα πλεονεκτήματα του typedef είναι :

α) Λιγότερες εντολές κώδικα σε περίπτωση ενός πολυσύλλαβου τύπου δεδομένων

β) Γρήγορη αλλαγή σε περίπτωση που αλλάξουμε ένα τύπο με άλλο τύπο δεδομένων. Η αλλαγή θα γίνει σε μία γραμμή.

γ) Χρήση φιλικών ονομάτων

Την δήλωση typedef μπορούμε να την χρησιμοποιήσουμε και με τις δομές, όπως για παράδειγμα :

```
typedef struct {
    int x;
    int y;
} point;

int main(int argc, char *argv[])  {
  dekadikos x = 6.0f;
  WORD w = 12;

  printf("x = %f\n", x);
  printf("w = %u\n", w);

  point A;

  A.x = 22;
  A.y = 5;

  printf("A.x = %d and A.y= %d\n", A.x, A.y);

  return 0;
}
```

```
x = 6.000000
w = 12
A.x = 22 and A.y= 5
```

Η Απαρίθμηση (Enum) (1-2)

Η απαρίθμηση είναι ένας ακόμα τρόπος για να έχουμε διαφορετικού τύπου δεδομένα εκτός από τους βασικούς τύπους.

enum mycolors {black, blue, green, cyan, red, purple, yellow, white};

```
int main(int argc, char *argv[]){
 enum mycolors onecolor;

 onecolor = black;

 if (onecolor == black)
   onecolor = red;

 printf("onecolor = %d\n",onecolor);
 return 0;
}
```

Το πρόγραμμα εκτυπώνει 4 γιατί με την ανωτέρω δήλωση η C αποδίδει στα αναγνωριστικά (black, blue, …} ακέραιες τιμές ξεκινώντας από το 0 και αυξάνοντας κάθε φορά κατά 1.

Εναλλακτικά μπορούμε να κάνουμε την παρακάτω δήλωση :

enum mycolors {black=4, blue=7, green=18, cyan=20, red=22, white=25};

Αν τοποθετήσουμε την ανωτέρω πρόταση στο πρόγραμμα τότε θα εκτυπωθεί το 22 γιατί το αναγνωριστικό red έχει συγκεκριμένη τιμή.

Η Απαρίθμηση (Enum) (2-2)

```
enum mycolors {black, blue, green, cyan, red, purple, yellow, white};

typedef enum mycolors colors;

int main(int argc, char *argv[]){
 colors onecolor;

 onecolor = black;


 if (onecolor == black)
    onecolor = red;

 printf("onecolor = %d\n",onecolor);
 return 0;
}
```

Επιπρόσθετα θα μπορούσαμε να ορίσουμε ένα νέο τύπο για τα χρώματα με το όνομα **colors**

Προσοχή : απαγορεύεται να ξαναχρησιμοποιήσουμε τα ίδια ονόματα τιμών σε άλλη δομή. Η προσθήκη της νέας δομής setcolors στο ανωτέρω πρόγραμμα δημιουργεί σφάλμα.

Message
[Error] redeclaration of enumerator 'black'
[Note] previous definition of 'black' was here
[Error] redeclaration of enumerator 'blue'
[Note] previous definition of 'blue' was here
[Error] redeclaration of enumerator 'green'

```
enum mycolors {black, blue, green, cyan, red, purple, yellow, white};
enum setcolors {black, blue, green, cyan, red, purple, yellow, white};
```

Ένωση (union)

Η ένωση μας επιτρέπει να ενοποιήσουμε διαφορετικούς τύπους δεδομένων κάτω από ένα κοινό όνομα, δηλαδή όπως και η δομή (struct), αλλά μόνο ένας από αυτούς κάθε φορά έχει πρόσβαση στα δεδομένα. Πρακτικά όλοι οι τύποι δεδομένων που εμπεριέχονται μέσα σε μια ένωση καταλαμβάνουν τον ίδιο χώρο και συγκεκριμένα το μέγεθος όλης της ένωσης ισούται με το μέγεθος του μεγαλύτερου τύπου δεδομένων. Η σύνταξη της union έχει ως εξής :

```
Union Όνομα_Ένωσης {
        ΤύποςΔεδομμένων1  Αναγνωριστικό1;
        ΤύποςΔεδομμένων2  Αναγνωριστικό2;
        ΤύποςΔεδομμένων3  Αναγνωριστικό3;
        . . . . .
};
```

Αν αποδώσουμε μια τιμή σε ένα από τα μέλη της ένωσης τότε αυτό διατηρεί την τιμή του μέχρι του σημείου που θα αποδώσουμε σε κάποιο άλλο μέλος μια τιμή. Οπότε πρακτικά από όλα τα μέλη της ένωσης μόνο ένα μπορεί να χρησιμοποιηθεί κάθε στιγμή.

Αυτό το χαρακτηριστικό είναι πολύ χρήσιμο γιατί όπως θα δούμε στη συνέχεια κάνουμε χρήση λιγότερης μνήμης για την αναπαράσταση της πληροφορίας.

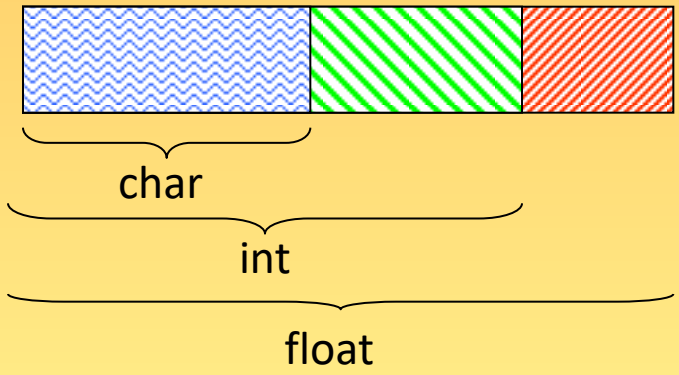
Παράδειγμα

Θα κατασκευάσουμε ένα union ικανό να αποθηκεύσει είτε ένα χαρακτήρα είτε ένα ακέραιο είτε ένα πραγματικό αριθμό.

Υλοποίηση

Η κατασκευή του union είναι μια αρκετά εύκολη υπόθεση

```
union OneUnion {
  char c;
  int i;
  float f;
};
```



Όταν δηλώσουμε μια μεταβλητή του τύπου **OneUnion** θα γίνει δέσμευση της μνήμης ικανή να αποθηκεύσει ένα πραγματικό αριθμό γιατί αυτός είναι ο τύπος με τις μεγαλύτερες απαιτήσεις σε μνήμη.

Παρατηρήσεις

Όπως φαίνεται και στο ανωτέρω σχήμα όταν αποθηκεύσουμε ένα char τότε θα χρησιμοποιηθεί η μπλε γραμμοσκίαση. Αν στη συνέχεια αποθηκεύουμε ένα int θα χρησιμοποιηθεί η μνήμη από την αρχή μέχρι και τη πράσινη γραμμοσκίαση αλλά καταστρέφουμε τα περιεχόμενα του char. Αν συνέχεια αποθηκεύσουμε ένα πραγματικό αριθμό θα χρησιμοποιηθεί όλη η μνήμη μέχρι και τη κόκκινη γραμμοσκίαση.

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

union OneUnion {
  char c;
  int i;
  float f;
};

union OneUnion test;

int main(int argc, char *argv[]) {
   printf("Assign a char value\n");
   test.c = 'c';
   printf("char : %c\n", test.c);
   printf("int : %d\n", test.i);
   printf("float : %f\n", test.f);

   printf("\nAssign an integer value\n");
   test.i = 65;
   printf("char : %c\n", test.c);
   printf("int : %d\n", test.i);
   printf("float : %f\n", test.f);

   printf("\nAssign a float value\n");
   test.f = 24.53;
   printf("char : %c\n", test.c);
   printf("int : %d\n", test.i);
   printf("float : %f\n", test.f);
   return 0;
}
```

```
Assign a char value
char : c
int : 99
float : 0.000000

Assign an integer value
char : A
int : 65
float : 0.000000

Assign a float value
char : q
int : 1103379825
float : 24.530001
```

```
struct InputEvent  {
    enum TypeOfEvent
    {
        KeyBoard, Mouse
     } Type;

    union
    {
        unsigned int KeyboardKey;
        struct {
              int ScreenX;
              int ScreenY;
              int MouseKey;
         } MouseEvent;
      } Data;
};
```

Στο διπλανό παράδειγμα δημιουργούμε μία δομή για την περιγραφή των διαφόρων γεγονότων που συμβαίνουν σε ένα υπολογιστικό σύστημα.

Ένα γεγονός μπορεί να είναι είτε από το πληκτρολόγιο είτε από το ποντίκι, οπότε χρησιμοποιούμε το enum TypeOfEvent για να δηλώσουμε το είδος του γεγονότος.

Αν σε περίπτωση το γεγονός είναι από το πληκτρολόγιο θα θέλαμε να αποθηκεύσουμε το πλήκτρο που πατήθηκε διαφορετικά αν το γεγονός προήλθε από το ποντίκι τότε να αποθηκεύσουμε τις συντεταγμένες στην οθόνη καθώς και το πλήκτρο του ποντικιού.

Εάν αντί για το union Data χρησιμοποιούσαμε μια δομή (struct) τότε σε αυτή την περίπτωση θα χρησιμοποιούσαμε επιπλέον χώρο στη μνήμη του υπολογιστή, ίσο με έναν ακέραιο αριθμό.

Union = 4 bytes (Type) + 3 (ScreenX, ScreenY, MouseKey) * 4 bytes = 16 bytes
Struct = 4 bytes (Type) + 4 bytes (KeyboardKey) + 3 (ScreenX, ScreenY, MouseKey) * 4 bytes = 20 bytes