Προγραμματισμός Υπολογιστών

Ο προ-επεξεργαστής στη C και οι μακροεντολές

Νικόλαος Ζ. Ζάχαρης
Καθηγητής

Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Το στάδιο της προ-επεξεργασίας (preprocessing)

Σχεδόν σε όλους τους μεταγλωττιστές υπάρχει το στάδιο της προ-επεξεργασίας, οπου ο πηγαίος κώδικας μετατρέπεται σύμφωνα με κάποιες οδηγίες, και εν συνεχεία αρχίζει η μεταγλώττιση. Στη διάρκεια της προ-επεξεργασίας γίνονται διάφορες εργασίες όπως είναι για παράδειγμα, η διαγραφή των σχολίων του προγραμματιστή, γιατί δεν προσφέρουν κάτι στο επόμενο στάδιο της μεταγλώττισης.

Από το πρώτο πρόγραμμα στη γλώσσα C χρησιμοποιούμε την εντολή **#include** η οποία είναι μία οδηγία για το στάδιο της προ-επεξεργασίας, και συγκεκριμένα να τοποθετήσει όλες τις εντολές του αρχείου της επικεφαλίδας, στο συγκεκριμένο σημείο του προγράμματος. Για παράδειγμα η εντολή **#include <stdio.h>** θα τοποθετήσει τα περιεχόμενα της επικεφαλίδας <stdio.h> στο συγκεκριμένο σημείο του πηγαίου κώδικα.

Τα αρχεία των επικεφαλίδων που ανήκουν στο μεταγλωττιστή, επικεφαλίδες συστήματος, αναφέρονται μέσα στους χαρακτήρες < και >, για παράδειγμα

#include <stdlib.h>

Η αναζήτησή για την εισαγωγή τους, θα γίνει στους προκαθορισμένους φακέλους του συστήματος που δηλώνονται στη μεταβλητή περιβάλλοντος INCLUDE

ενώ τα αρχεία των επικεφαλίδων που έχουμε δημιουργήσει για την εφαρμογή μας εμπεριέχονται μέσα στους χαρακτήρες δίπλα εισαγωγικά " για παράδειγμα

#include "utils.h"

Η αναζήτησή θα γίνει στο φάκελο της εφαρμογής και σε αυτούς που δηλώνονται στη παράμετρο -iquote

Μακροεντολές (1-4)

Η γλώσσα προγραμματισμού C προσφέρει τη δυνατότητα της δημιουργίας μακροεντολών, δηλαδή παραμετροποιήσιμου κώδικα που θα τοποθετηθεί κατά το στάδιο της προ-επεξεργασίας (δηλαδή πριν από τη μεταγλώττιση) στα σημεία που εμφανίζεται η μακροεντολή μέσα στο πηγαίο κώδικα.

Αρκετοί προγραμματιστές παρομοιάζουν την διαδικασία με “εύρεση και αντικατάσταση κειμένου” μόνο που το κείμενο περιέχει κώδικα.

Πριν την σύνταξη της μακροεντολής προηγείται η λέξη #define εν συνέχεια το όνομα της, οι παράμετροι και τέλος ο κώδικας.

Όλοι οι μεταγλωττιστές της C ορίζουν αυτόματα κάποιες προκαθορισμένες μακροεντολές, όπως :

__LINE__	Η τρέχουσα γραμμή κώδικα
__FILE__	Το όνομα του πηγαίου αρχείου
__TIME__	Η τρέχουσα ώρα του συστήματος
__DATE__	Η τρέχουσα ημερομηνία του συστήματος
__STDC__	Υποστήριξη της ISO Standard C (Αν ναι επιστρέφει τη τιμή 1)

Μακροεντολές (2-4)

Για να δημιουργήσουμε μία μακροεντολή με το όνομα cube η οποία θα δέχεται ένα όρισμα και θα το επιστρέφει υψωμένο εις την δύναμη του 3 τότε θα πρέπει να γράψουμε :

```
#define cube(x) ((x) * (x) * (x))
```

Θα πρέπει να χρησιμοποιούμε πάντα παρενθέσεις στα ορίσματα της μακροεντολής γιατί διαφορετικά δημιουργούνται λάθος αποτελέσματα. Για παράδειγμα, αν η εντολή ήταν #define cube(x) (x * x * x)

```
printf("%d\n\n", cube(3));
```
Θα έχει σαν έξοδο το 27 που είναι σωστό αποτέλεσμα.

```
printf("%d\n\n", cube(2+1));
```
Θα έχει σαν έξοδο το 7 που είναι λάθος αποτέλεσμα γιατί θα εκτελέσει τη πράξη 2 + 1 * 2 + 1 * 2 + 1 = 7

Ο συνδυασμός (συνθήκη) ? (εντολέςΑ) : (εντολέςΒ) είναι η υλοποίηση της συνθήκης if else σαν μακροεντολή. Αν η συνθήκη είναι αληθής τότε θα εκτελεστούν οι εντολέςΑ διαφορετικά οι εντολέςΒ.

```
#define min(x ,y) ((x < y) ? x : y)
```

```
printf("Min number is: %d\n",min(2,3));
```

Θα έχει σαν αποτέλεσμα **Min number is: 2**

Μακροεντολές (3-4)

Η χρήση του χαρακτήρα # μέσα σε μακροεντολή τοποθετεί το όρισμα μέσα σε διπλά εισαγωγικά "

```
#define write(x)  printf("%s",#x);

int main(int argc, char *argv[]) {
    write(12+3);
     return 0;
}
```

Θα δημιουργήσει την εντολή printf("%s","12+3");

```
12+3
```

Η χρήση των χαρακτήρων ## μέσα σε μακροεντολή κάνει συνένωση των δύο ορισμάτων της

```
#define car_property(x, y)  (x ## y)

int main(int argc, char *argv[]) {
    int car_weight = 1255;
    printf("Car Weight : %d\n", car_property(car , _weight) );

            return 0;
}
```

Θα δημιουργήσει ένα ενιαίο όνομα το car_weight

```
Car Weight : 1255
```

Μακροεντολές (4-4)

Με την εντολή #define ορίζουμε μακρο-εντολές και μπορούμε σε οποιοδήποτε σημείο του προγράμματος να τις διαγράψουμε με την εντολή #undef, η σύνταξη της οποία είναι :

#undef ΟΝΟΜΑ_ΜΑΚΡΟΕΝΤΟΛΗΣ

Για τη διαγραφή της μακρο-εντολής cube θα πρέπει να γράψουμε

#undef cube

Δεν υπάρχει πρόβλημα στην περίπτωση που δεν υπάρχει το αναγνωριστικό που θέλουμε να διαγράψουμε.

```
#define PI 3.14
#define CircleArea(r) (PI * (r) * (r))

#undef PI

int main(int argc, char *argv[]) {
        return 0;
}
```

Δεν θα γίνει μεταγλώτιση στο διπλανό κώδικα επειδή δεν έχει οριστεί το PI. Αν όμως αλλάξουμε τη σειρά των εντολών τότε δεν υπάρχει πρόβλημα.

```
#undef PI
#define PI 3.14
#define CircleArea(r) (PI * (r) * (r))
```

Συνήθως ο ορισμός ή η διαγραφή μακροεντολών γίνεται υπό συνθήκες ελέγχου.

Μεταγλώττιση υπό συνθήκες

Σε μεγαλύτερα προγράμματα όπου αποτελούνται από πολλά αρχεία πηγαίου κώδικα, θα θέλαμε πάρα πολύ γρήγορα να συμπεριλαμβάνουμε συγκεκριμένα αρχεία επικεφαλίδων τα οποία περιέχουν είτε απαραίτητες μακροεντολές για την εφαρμογή είτε αναφέρονται σε συγκεκριμένο λειτουργικό σύστημα στο οποίο πρόκειται να γίνει η μεταγλώττιση.

Για να επιτευχθεί αυτή η λειτουργία, ο προ-επεξεργαστής υποστηρίζει μέσω των δηλώσεων #if, #ifdef, #ifndef, #else, #elif and #endif τη μεταγλώττιση υπό συνθήκες.

```
#ifdef __unix__
    # include <unistd.h>
#elif defined _WIN32
   # include <windows.h>
#endif
```

Η μακροεντολή __unix__ ορίζεται αυτόματα στους μεταγλωττιστές των συστημάτων τύπου unix ενώ στα συστήματα windows ορίζεται αυτόματα η μακροεντολή _WIN32

Το #ifdef είναι συντομογραφία του #if defined PI και το #ifndef είναι του #if !(defined PI)

Οι μακρο-εντολές μπορούν να κάνουν σύνθετες συνθήκες με τη χρήση των τελεστών && και || και ! όπως για παράδειγμα

```
#if !(defined CUBE64 || defined SQUARE64) && defined MATH64
    #include "mymath64.h"
#else
   #error MATH32 not supported
#endif
```

Η error σταματά τη μεταγλώττιση και εμφανίζει το μήνυμα που ακολουθεί "MATH32 not supported "

Η εφαρμογή

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

// #define DOLLAR 1

#include "test.h"
#define PI 3.14
#define CircleArea(r) (PI * (r) * (r))
#define MAX(x,y) ((x>y)?x:y)

int main(int argc, char *argv[]) {
    int a = 2,b = 7,max;
    max=MAX(a,b);
    printf("Maximum number is: %d\n",max);

    float radius = 2.45;
    float area = CircleArea(radius);
    printf("Circle area = %.4f\n", area);

    printf("Symbol         : %s\n",symbol);
    printf("Current Date   : %s\n",__DATE__);
    printf("Current File   : %s\n",__FILE__);
    printf("Current Line   : %d\n",__LINE__);
    printf("ISO Standard : %d\n",__STDC__);
    printf("Current time   : %s\n",__TIME__);
    return 0;
}
```

Τρέχουμε το πρόγραμμα και εμφανίζεται Symbol : Unknown και μετά βγάζουμε το σχόλιο και εμφανίζεται το Symbol : $.

test.h

```c
#if defined GREEK
   #define CAPITAL "Athens"
#else
   #define CAPITAL "Unknown"
#endif


#if defined EURO
   #define symbol "EURO"
#elif defined DOLLAR
   #define symbol "$"
#elif defined YENN
   #define symbol "Yn"
#else
   #define symbol "Unknown"
#endif
```

```
Maximum number is: 7
Circle area = 18.8479
Symbol         : Unknown
Current Date   : Jan 31 2019
Current File   : main.c
Current Line   : 23
ISO Standard : 1
Current time   : 10:57:59
```