Η ΓΛΩΣΣΑ C

Μάθημα 25:

Μακροεντολές

Δημήτρης Ψούνης

www.psounis.gr

Περιεχόμενα Μαθήματος

Α. Πίνακες

1. Προεπεξέργαστης
    1. Μεταγλώττιση Προγράμματος
    2. Λειτουργίες του Προεπεξέργαστή
    3. Η οδηγία include
    4. Η οδηγία define
2. Μακροεντολές
    1. Δημιουργία συναρτήσεων ως μακροεντολές
    2. Συναρτήσεις με περισσότερα ορίσματα
    3. Μετατροπή σε συμβολοσειρά
    4. Τελεστής συνένωσης συμβολοσειρών
    5. Σύνοψη για τις μακροεντολές
    6. Δήλωση σταθερών και αποφυγή πολλαπλών #include

Β. Ασκήσεις

Α. Θεώρια

1. Προεπεξεργαστής

1. Μεταγλώττιση προγράμματος

- Η μεταγλώττιση του προγράμματος είναι η διαδικασία της μετατροπής των αρχείων κώδικα σε εκτελέσιμο αρχείο.
- Η διαδικασία αποτελείται από 3 βήματα τα οποία είναι:
  - Προεπεξεργασία (preprocessing):
    - Οι εντολές #include αντικαθίστανται από τον κώδικα των αρχείων στα οποία αναφέρονται (και αντίστοιχα για άλλες οδηγίες του προεπεξεργαστή)
  - Μεταγλώττιση (compiling):
    - Κάθε αρχείο .c μεταγλωττίζεται μόνο του και παράγει ένα νέο αρχείο (συνήθως με προέκταση .o) το οποίο έχει τη μετάφραση του αρχείου σε γλώσσα μηχανής και λέγεται αντικειμενικό αρχείο
  - Σύνδεση (linking):
    - Τα αντικειμενικά αρχεία (.o) συνδυάζονται σε ένα τελικό εκτελέσιμο αρχείο (.exe)

Α. Θεώρια

1. Προεπεξεργαστής

2. Λειτουργίες του Προεπεξεργαστή

- Η βασική λειτουργία του προεπεξεργαστή είναι να κάνει αντικατάσταση των οδηγιών που έχουν δοθεί στο πρόγραμμα με κανονικό κώδικα C
  - Οι οδηγίες είναι εντολές που ξεκινούν με τη #
    - π.χ. #define, #include κ.α.
  - Οι οδηγίες αντικαθίστανται από κώδικα
    - π.χ. μία #include ενσωματώνει τον κώδικα του αρχείου που την ακολουθεί.

- Υπάρχουν διάφορες οδηγίες για τον προεπεξεργαστή και θα ασχοληθούμε με αυτές στο σημερινό μάθημα.

Α. Θεώρια

1. Προεπεξεργαστής

3. Η οδηγία include

- Έχουμε ήδη χρησιμοποιήσει την οδηγία του προεπεξεργαστή #include για να ενσωματώσουμε βιβλιοθήκες στο πρόγραμμά μας.
  - Π.χ. με την οδηγία:

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

    - Η γραμμή αυτή θα αντικατασταθεί με τα περιεχόμενα του αρχείου stdio.h
    - Τα σύμβολα <,> χρησιμοποιούνται για να υποδείξουν ότι το αρχείο αυτό βρίσκεται στη συνηθισμένη θέση που κρατάει τις βιβλιοθήκες του ο μεταγλωττιστής που χρησιμοποιούμε.
      - Και ο φάκελος που τηρούνται είναι μία ρύθμιση του μεταγλωττιστή.
  - Ενώ π.χ. με την οδηγία:

```
#include “mylib.h”
```

    - Ενσωματώνουμε μια βιβλιοθήκη χρήστη.
    - Ο μεταγλωττιστής θα αναζητήσει τη βιβλιοθήκη αυτή, στον ίδιο φάκελο που βρίσκεται και το αρχείο κώδικα.
    - Ο διαχωρισμός γίνεται με βάση τα διπλά εισαγωγικά.

Α. Θεώρια

1. Προεπεξεργαστής

4. Η οδηγία define

- Η οδηγία #define ορίζει μια συμβολική σταθερά στο πρόγραμμα μας.
  - και ο προεπεξεργαστής αντικαθιστά κάθε εμφάνιση της στο πρόγραμμα με τη συμβολική σταθερά.
- Π.χ. όταν γράφαμε στο πρόγραμμά μας την οδηγία:

```
#define SIZE 100
```

  - Ο προεπεξεργαστής αντικαθιστά κάθε εμφάνιση του SIZE με το 100.
  - Ο μεταγλωττιστής δεν γνωρίζει καν για την ύπαρξη του SIZE μιας και θα δει μόνο το 100.
  - Γενικά το συντακτικό είναι:

```
#define name replacement_text
```

    - το name αντικαθίσταται από το replacement_text.
      - και πρέπει να είναι σε μία γραμμή (χρησιμοποιώντας το \ για «αλλαγή» γραμμής).
- Εκτός από τη λειτουργία που ήδη ξέρουμε,
  - μπορούμε να ορίσουμε και περισσότερο περίπλοκες αντικαταστάσεις.
  - Οι αντικαταστάσεις αυτές λέγονται μακροεντολές, και θα τις μελετήσουμε αναλυτικά στην επόμενη ενότητα.

- Οι αντικαταστάσεις που γίνονται με τη #define λέγονται και **μακροεντολές**

Α. Θεώρια

2. Μακροεντολές

1. Δημιουργία συναρτήσεων ως μακροεντολές

- Μακροεντολή είναι ένα τμήμα κώδικα που έχει ονομαστεί.
  - π.χ. η #define SIZE 100 δίνει σε έναν κώδικα (Το 100) ένα όνομα (το SIZE)
  - Έτσι οι #define που χρησιμοποιήσαμε ως τώρα, είναι μακροεντολές.
- Μπορούμε ωστόσο να έχουμε και μακροεντολές που δέχονται ορίσματα.
- Παράδειγμα:
  - Η μακροεντολή:

```
#define half(x) ((x)/2)
```

  - Θα αντικαταστήσει κάθε εμφάνιση της x με το (x)/2
    - όπου x μπορεί να είναι μια περίπλοκη παράσταση.

- Βλέπουμε και ολοκληρωμένο το παράδειγμα στο επόμενο πρόγραμμα:

Α. Θεώρια

2. Μακροεντολές

1. Δημιουργία συναρτήσεων ως μακροεντολές

```
/* macro_function.c */

#include <stdio.h>

#define half(x) ((x)/2)

main()
{
   int i=5;
   float f=2.2;

   printf("half(%d)=%d\n", i, half(i));
   printf("half(%.1f)=%.1f\n", f, half(f));
}
```

Α. Θεώρια

2. Μακροεντολές

1. Δημιουργία συναρτήσεων ως μακροεντολές

Παρατηρήσεις για τις μακροεντολές:

- Είναι χωρίς τύπους!
  - Άρα η ίδια μακροεντολή μπορεί να χρησιμοποιηθεί για διαφορετικούς τύπους (int, float)
- Προσοχή στις παρενθέσεις
  - Οι «μεταβλητές» πρέπει να έχουν παρενθέσεις και επίσης η συνολική παράσταση πρέπει να έχει παρένθεση.
  - Αν δεν βάλουμε παρενθέσεις, τότε μπορεί να προκύψουν λανθασμένες αντικαταστάσεις
  - Π.χ. αν είχαμε ορίσει τη μακροεντολή ως: #define half(x) (x/2) τότε
    - ο κώδικας half(1+2) θα γινόταν αντικατάσταση με τον (1+2/2) αντί για το σωστό ((1+2)/2)

Α. Θεώρια

2. Μακροεντολές

2. Συναρτήσεις με περισσότερα ορίσματα

- Μπορούμε να έχουμε και μακροεντολές που έχουν όσα ορίσματα θέλουμε:

```
#define sum(x,y) ((x)+(y))
```

  - Παρατηρούμε πάλι την παρενθετοποίηση των μεταβλητών στο σώμα της αντικατάστασης.
- Και ένα ολοκληρωμένο παράδειγμα:

```
/* macro_function2.c */

#include <stdio.h>

#define sum(x,y) ((x)+(y))

main()
{
   int v = sum(5,5*3);
   printf("sum=%d\n", v);
}
```

- Σημαντικό: Όλες οι παράμετροι πρέπει να χρησιμοποιούνται
  - π.χ. η μακροεντολή **#define sum(x,y,z) ((x)+(y))** θα οδηγήσει σε σφάλμα.

Α. Θεώρια

2. Μακροεντολές

3. Μετατροπή σε συμβολοσειρά

- Με τον τελεστή # μπορούμε να μετατρέψουμε ένα όρισμα σε συμβολοσειρά:

```
#define out(x) printf(#x)
```

  - Στο όρισμα (x) θα μπουν διπλά εισαγωγικά και θα αντικατασταθεί το #x.
- Και ένα ολοκληρωμένο παράδειγμα:

```
/* macro_function3.c */

#include <stdio.h>

#define out(x) printf(#x)

main()
{
   out(This is a string);
}
```

- Το παραπάνω πρόγραμμα:
  - Θα αντικαταστήσει την out() ως printf(“This is a string”)
  - και τελικά θα τυπώσει: This is a string

Α. Θεώρια

2. Μακροεντολές

4. Τελεστής συνένωσης συμβολοσειρών

- Με τον τελεστή ## μπορούμε να συνενώσουμε δύο συμβολοσειρές. Π.χ.

```
#define name(x) v ## x
```

  - Το name(1) θα επιστρέψει v1, το name(2) θα επιστρέψει v2 κ.ο.κ.
- Και ένα ολοκληρωμένο παράδειγμα:

```
/* macro_function4.c */

#include <stdio.h>

#define v(x) v ## x

main()
{
   int i, v1=1, v2=2;

   printf("%d", v(1));
   printf("%d", v(2));
}
```

Α. Θεώρια

2. Μακροεντολές

5. Σύνοψη για τις μακροεντολές

- Αν και οι μακροεντολές φαίνονται πολύ αποδοτικές, οι απόψεις διίστανται για τη χρήση τους:
  - Φαίνεται ότι μπορούμε να κάνουμε δήλωση συναρτησοειδών με μακροεντολές που θα είναι πιο γρήγορες από τις συναρτήσεις (αφού δεν θα γίνεται κλήση, αλλά απλή συντακτική αντικατάσταση),
    - αλλά από την άλλη, χάνουμε τον έλεγχο τύπων που είναι ένα από τα ατού της C
  - Απλοποιούν σε κάποιες περιπτώσεις τον κώδικα,
    - αλλά από την άλλη η μεταγλώττιση και η αποσφαλμάτωση είναι πολύ δύσκολη.
    - Τα λάθη μεταγλώττισης, αφορούν τον κώδικα αφού έχει γίνει η προεπεξεργασία, με αποτέλεσμα να μην βλέπουμε τον πραγματικό κώδικα του οποίου μας αναφέρονται τα σφάλματα από το μεταγλωττιστή.
  - Είναι δύσκολες στη σύνταξη (όλη η σύνταξη πρέπει να γίνει σε μια γραμμή)

- Γενικά η χρήση μακροεντολών, έχει υποκειμενική επιθυμία χρήσης και αφήνεται η επιλογή τους στην ευχέρεια του αναγνώστη.
  - Ωστόσο έχουν γραφεί πολλές χρήσιμες μακροεντολές, π.χ. η ακόλουθη απλοποιεί τη σύνταξη της malloc:

```
#define MALLOC(t,n) (t *) malloc((n)*sizeof(t))
```

Α. Θεώρια

2. Μακροεντολές

6. Δήλωση σταθερών και αποφυγή πολλαπλών #include βιβλιοθηκών χρήστη

- Μπορούμε να δηλώσουμε ότι απλά κάτι έχει οριστεί. Π.χ.

```
#define FLAG
```

  - Η εντολή αυτή απλά δηλώνει ότι το FLAG έχει οριστεί (χωρίς τιμή)
- Και έπειτα μπορούμε να κάνουμε ελέγχους για το αν έχει οριστεί το FLAG:

```
#ifdef FLAG
  …
#endif
```

- ή αν δεν έχει οριστεί το FLAG:

```
#ifndef FLAG
  …
#endif
```

- Υπάρχουν και οι #elif και #else με προφανή σημασία.

Α. Θεώρια

2. Μακροεντολές

6. Δήλωση σταθερών και αποφυγή πολλαπλών #include βιβλιοθηκών χρήστη

- Το πρόβλημα της διπλής ενσωμάτωσης (double inclusion) προκύπτει όταν:
  - όταν γίνεται δύο φορές ενσωμάτωση της ίδιας βιβλιοθήκης και σε αυτήν υπάρχει π.χ. ένα struct.
- Παράδειγμα:
  - Έχουμε μία ιεραρχία βιβλιοθηκών:
  - Στη main χρησιμοποιούμε τις βιβλιοθήκες h1 και h2
  - Οι h1 και h2 χρησιμοποιούν μία κοινή βιβλιοθήκη, την h
  - Και η h έχει ορίσει (μεταξύ άλλων) ένα struct.
  - (βλέπε επόμενη διαφάνεια)
  - Η μεταγλώττιση του προγράμματος θα αποτύχει…

Α. Θεώρια

2. Μακροεντολές

6. Δήλωση σταθερών και αποφυγή πολλαπλών #include

```
/* h.h */
void h();
struct temp {
    int x;
};
```

```
/* h1.h */
#include "h.h"

void h1();
```

```
/* h.c */
#include <stdio.h>
#include "h.h"

void h()
{
   struct temp t;
   printf("in h\n");
}
```

```
/* h2.h */
#include "h.h"

void h2();
```

```
/* h1.c */
#include <stdio.h>
#include "h1.h"

void h1()
{
    printf("in h1\n");
    h();
}
```

```
/* h2.c */
#include <stdio.h>
#include "h2.h"

void h2()
{
    printf("in h2\n");
    h();
}
```

```
#include <stdio.h>
#include "h1.h"
#include "h2.h"

main()
{
   h1();
   h2();
}
```

Α. Θεώρια

2. Μακροεντολές

6. Δήλωση σταθερών και αποφυγή πολλαπλών #include βιβλιοθηκών χρήστη

- Η μεταγλώττιση του προγράμματος οδηγεί στο λάθος διπλής ενσωμάτωσης:
  - Μέσω του include της h1.h έχει ενσωματωθεί το struct
  - Και μέσω του include της h2.h έχει ενσωματωθεί 2η φορά το struct.
  - Έτσι ο μεταγλωττιστής διαμαρτύρεται ότι δηλώνεται το ίδιο struct δύο φορές.

- Η λύση έρχεται μέσω των «φρουρών ενσωμάτωσης» (inclusion guards)

- Σε κάθε αρχείο κεφαλίδας .h περικλείουμε τον κώδικα ανάμεσα σε φρουρούς.
- Π.χ. για το h1.h κάνουμε την ενσωμάτωση των φρουρών:

```
/* h1.h */
#include "h.h"

void h1();
```

```
/* h1.h */
#ifndef H_H
#define H_H
#include "h.h"

void h1();

#endif
```

Α. Θεώρια

2. Μακροεντολές

6. Δήλωση σταθερών και αποφυγή πολλαπλών #include βιβλιοθηκών χρήστη

- Συνηθίζεται ο φρουρός να είναι μια σταθερά που απλά ορίζεται:
  - Με όνομα, το όνομα της βιβλιοθήκης με κεφαλαία γράμματα
  - ακολουθούμενο με _H

```
/* library.h */
#ifndef LIBRARY_H
#define LIBRARY_H
…
#endif
```

- Και θεωρείται καλή πολιτική κάθε βιβλιοθήκη να έχει τους φρουρούς της!
- Τώρα το πρόγραμμα μας, πλέον θα μεταγλωττίζεται
  - (ολοκληρωμένο είναι το project2.dev)

Β. Ασκήσεις

1. Μακροεντολή για δέσμευση μνήμης

Μελετήστε την ακόλουθη μακροεντολή:

```
#define MALLOC(p,t,n) p=(t *)malloc((n)*sizeof(t)); \
  if (!p) { \
    fprintf(stderr, "Error Allocating Memory"); \
    exit(errno); \
  }
```

(Σημείωση: Το \ επιτρέπει στη μακροεντολή να εκτείνεται σε περισσότερες από μία γραμμές)
Κατασκευάστε μία main που να χρησιμοποιεί την παραπάνω μακροεντολή.

Β. Ασκήσεις

2. Μακροεντολή για άνοιγμα αρχείου

Σε αντιστοιχία με την προηγούμενη άσκηση, κατασκευάστε μία μακροεντολή που να ανοίγει ένα αρχείο κειμένου:

- Οι παράμετροι της μακροεντολής να είναι ο δείκτης αρχείου, το όνομα του αρχείου (χωρίς τα εισαγωγικά) και ο τρόπος ανοίγματος του αρχείου.
- Να ενσωματώνεται στην μακροεντολή και ο τυπικός έλεγχος που κάνουμε για το αν το αρχείο άνοιξε σωστά.