Προγραμματισμός Υπολογιστών με C++

Ο ΠΡΟΕΠΕΞΕΡΓΑΣΤΗΣ

Γεώργιος Παπαϊωάννου (2013-14)
gepap@aueb.gr

Περιεχόμενο Παρουσίασης

- Περιγραφή:
  - Ο Προεπεξεργαστής
  - Χρήση του προεπεξεργαστή
- Τελευταία ενημέρωση: Ιούνιος 2013

Ο Ρόλος του Προεπεξεργαστή

- Ο προεπεξεργαστής «φιλτράρει» τα αρχεία κώδικα και τα τροποποιεί με βάση:
  - Ρυθμίσεις του μεταγλωττιστή
  - Δηλωμένες από το χρήστη εξωτερικές τιμές και παραμέτρους
  - Εντολές προεπεξεργασίας μέσα στον κώδικα
- Βασικές χρήσεις:
  - Ενσωμάτωση εξωτερικών τμημάτων/αρχείων κώδικα
  - Επιλεκτική απομάκρυνση/ενσωμάτωση κώδικα
  - Υλοποίηση «μακροεντολών»

Ενσωμάτωση Εξωτερικών Αρχείων

- Οι εντολές προεπεξεργαστή μέσα στον κώδικα ξεκινούν με το σύμβολο #
- Η εντολή προεπεξεργαστή #include ανοίγει και ενσωματώνει ένα αρχείο επί τόπου στο σημείο της δήλωσης
- Συνήθως, κατά σύμβαση, ενσωματώνουμε αρχεία δηλώσεων ή «κεφαλίδων» (header files) ώστε να μπορεί ο μεταγλωττιστής να εντοπίσει τις δηλώσεις τύπων και συναρτήσεων
- Τα header files έχουν συνήθως κατάληξη (αν έχουν):

  .h ή .hpp

Εύρεση Εξωτερικών Αρχείων για Ενσωμάτωση (1)

- Ο προεπεξεργαστής ψάχνει:
  - Στα μονοπάτια που είναι γνωστά στον μεταγλωττιστή από την εγκατάσταση της γλώσσας
  - Σε μονοπάτια που έχουν προσδιοριστεί από το χρήστη είτε από τη γραμμή εντολών (-I [path] στον compiler) είτε από το IDE (πάλι μέσω ορισμάτων στη γραμμή εντολών ορίζονται, αλλά έμμεσα)
  - Στον τοπικό φάκελο που γίνεται η μεταγλώττιση
  - Σε ρητά δοσμένα μονοπάτια στο filesystem

Εύρεση Εξωτερικών Αρχείων για Ενσωμάτωση (2)

- Παραδείγματα:

```
#include “myfunctions.h”
#include <math.h>
#include <vector>
#include <gl/gl.h>
#include “code/gepap/mathlib”
```

Ψάχνει τοπικά ή στα ορισμένα από τον προγραμματιστή μονοπάτια (`-I`)

Ψάχνει στα «default» include paths

Ψάχνει στα «default» include paths, σε σχετικό κατάλογο (gl/). Να προτιμάτε τις «unix-style» δηλώσεις μονοπατιών στο #include (cross-platform)

Ψάχνει σε σχετικό κατάλογο τοπικά ή στα ορισμένα από τον προγραμματιστή μονοπάτια (`-I`)

Μακροεντολές

- Η εντολή #define του προεπεξεργαστή χρησιμεύει για να ορίζουμε πράγματα που στο στάδιο της προεπεξεργασίας αντικαθίστανται μέσα στον κώδικα
- Ο, τι έχουμε ορίσει με #define ισχύει από το σημείο αυτό και μέχρι να καταργηθεί ο ορισμός με την εντολή προεπεξεργαστή #undef
- Ο μηχανισμός αυτός μας χρησιμεύει για να αντικαθιστούμε μαζικά κομμάτια κώδικα και σύμβολα

Η Εντολή #define

```
#define M_PI 3.1415936f

#define MAX(x,y) ( (x) > (y)? (x) : (y) )

#define PRINT_DEBUG_INFO(a) {cout << “Info: ” << a << endl;}
#include <iostream>
…
int main()
{
    float val1 = M_PI;
    float max_val = MAX( MAX( val1, 1 ), 0 );
    PRINT_DEBUG_INFO (max_val);
    return 0;
}
```

Ορισμός συμβόλου

Ορισμός μακροεντολής με ορίσματα

Η χρήση της iostream μετά την εμφάνιση της cout στη #define δεν προκαλεί σφάλμα. Γιατί;

Ορισμός μπλοκ κώδικα ως μακροεντολή

Χρήση των #defines

Η Εντολή #define: Μετά την Προεπεξεργασία

```
…  // Δηλώσεις από το iostream.h
…

int main()
{
    float val1 = 3.1415936f;
    float max_val = ( ((val1)>(1)?(val1):(1))>(0)?
       ((val1)>(1)?(val1):(1)) : (0) );
    {cout << “Info: ” << max_val << endl;}
    return 0;
}
```

Ενσωμάτωση Κώδικα υπό Συνθήκη (1)

- Με τη χρήση των εντολών ροής ελέγχου του προεπεξεργαστή, μπορούμε να συμπεριλάβουμε ή όχι τμήματα κώδικα και δηλώσεις υπό συνθήκη:

```
#define _DEBUG

…
double findRoot( double x) {
    if (x<=0) {
#ifdef _DEBUG
        cout << “Invalid value in findRoot(): ” << x << endl;
#endif
        return 0.0;
    }
    else return sqrt(x);
}
```

Ενσωμάτωση Κώδικα υπό Συνθήκη (2)

- Με την υπο συνθήκη ενσωμάτωση κώδικα από τον προεπεξεργαστή μπορούμε να αποφύγουμε και διπλές δηλώσεις:

```
#ifndef MYFUNCTIONS
#define MYFUNCTIONS

double findRoot( double x) {
    if (x<=0) {
#ifdef _DEBUG
        cout << “Invalid value in findRoot(): ” << x << endl;
#endif
        return sqrt(x);
    }
    else return 0.0;
}
#endif
```

Το αντίθετο του #ifdef : (n)ot defined

Ο προεπεξεργαστής θα ενσωματώσει αυτόν τον κώδικα μόνο την πρώτη φορά που θα συναντήσει αυτό το μπλοκ. Γιατί;