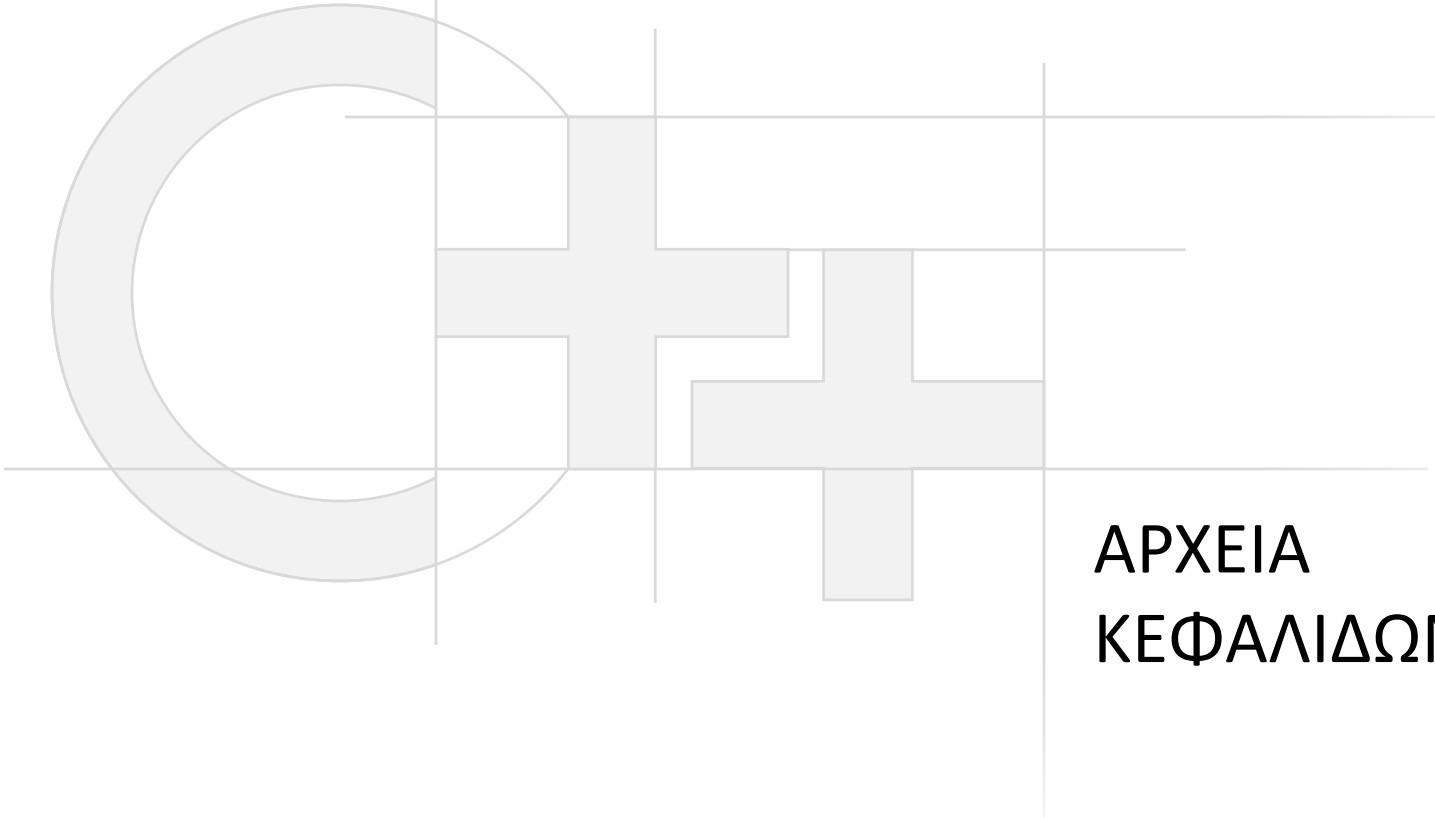
Προγραμματισμός Υπολογιστών με C++



ΑΡΧΕΙΑ ΚΕΦΑΛΙΔΩΝ

Γεώργιος Παπαϊωάννου (2013-20)
gepap@aueb.gr

Περιεχόμενο Παρουσίασης

- Περιγραφή:
  - Πολλαπλά αρχεία κώδικα
  - Αρχεία κεφαλίδων

- Τελευταία ενημέρωση: Νοέμβριος 2020

Πολλά Αρχεία Πηγαίου Κώδικα

- Στη C/C++ τον κώδικά μας τον οργανώνουμε συνήθως σε αρχεία, ανάλογα με εννοιολογικό διαχωρισμό, ή με βάση τις κλάσεις
  - Π.χ. τις ελεύθερες συναρτήσεις τις οργανώνουμε σύμφωνα με τη λειτουργία τους (I/O, μαθηματικά, επεξ. Πινάκων, κλπ.)
  - Τις κλάσεις τις βάζουμε συνήθως σε ένα αρχείο υλοποίησης για κάθε αυτόνομη κλάση
- Γιατί;
  - Στη C++ δεν έχουμε την υποχρέωση να χωρίσουμε τον κώδικά μας όπως στη Java, αλλά:
  - Γίνεται ο κώδικας πιο αρθρωτός και επαναχρησιμοποιήσιμος
  - Γίνεται ο κώδικας πιο ευανάγνωστος
  - Δε χρειάζεται να κάνουμε τα πάντα compile για μια μικρή αλλαγή

Παράδειγμα

Πώς θα γνωστοποιήσουμε τις πράξεις που φτιάξαμε στο mymath.cpp στη main();

mymath.cpp

```
float dotProduct(float* a, float* b){
  if (!a || !b)
    return 0.f;
  return a[0]*b[0]+a[1]*b[1]+a[2]*b[2];
}

bool isZero(float* a){
  return a==nullptr ? true :
    a[0]==.0f && a[1]==.0f && a[2]==.0f;
}

…
```

application.cpp

```
#include <iostream>
...

  int main(int argc, char *argv[]){
  float v1[] = {1.f, 0.f, 0.f};
  float v2[] = {2.f, 1.f, 0.f};
  std::cout << dotProduct(v1,v2);
  return 0;
}
```

Αρχεία Κεφαλίδων (Header Files)

- Τα αρχεία κεφαλίδων είναι αρχεία στα οποία ξεχωρίζουμε τις δηλώσεις συναρτήσεων, τύπων, μακροεντολών κλπ ώστε:
  - Τα αρχεία κώδικα .c, .cpp .c++ να περιέχουν μόνο υλοποιήσεις συναρτήσεων και μεθόδων
  - Να μπορούμε να κάνουμε Include τα αρχεία κεφαλίδων σε πολλαπλά αρχεία υλοποίησης κώδικα

Παράδειγμα Αρχείων Κεφαλίδων

mymath.h

```
float dotProduct(float* a, float* b);
bool isZero(float* a);
…
```

mymath.cpp

```
#include "mymath.h"
float dotProduct(float* a, float* b){
  if (!a || !b)
    return 0.f;
  return a[0]*b[0]+a[1]*b[1]+a[2]*b[2];
}

bool isZero(float* a){
  return a==nullptr ? true :
    a[0]==.0f && a[1]==.0f && a[2]==.0f;
}
…
```

application.cpp

```
#include <iostream>
#include "mymath.h"
  int main(int argc, char *argv[]){
  float v1[] = {1.f, 0.f, 0.f};
  float v2[] = {2.f, 1.f, 0.f};
  std::cout << dotProduct(v1,v2);
  return 0;
}
```

gcc –o application.exe mymath.cpp application.cpp

Αρχεία Κεφαλίδων και inline Συναρτήσεις

mymath.h

```
float dotProduct(float* a, float* b);
inline bool isZero(float* a){
  return a==nullptr ? true :
    a[0]==.0f && a[1]==.0f && a[2]==.0f;
}
…
```

mymath.cpp

```
#include "mymath.h"
float dotProduct(float* a, float* b){
  if (!a || !b)
    return 0.f;
  return a[0]*b[0]+a[1]*b[1]+a[2]*b[2];
}
…
```

application.cpp

```
#include <iostream>
#include "mymath.h"
  int main(int argc, char *argv[]){
  float v1[] = {1.f, 0.f, 0.f};
  float v2[] = {2.f, 1.f, 0.f};
  std::cout << dotProduct(v1,v2);
  return 0;
}
```

Τις συναρτήσεις inline τις ορίζουμε στο αρχείο κεφαλίδας, γιατί απαιτείται να ορίζονται ξανά σε κάθε αρχείο .cpp!

Τι Μπαίνει στα Αρχεία Κεφαλίδων;

- **Δηλώσεις συναρτήσεων**.
- **Ορισμοί συναρτήσεων inline**.
- **Ορισμοί κλάσεων** (επόμενες διαλέξεις).
- **Σχεδιότυπα** (templates - επόμενες διαλέξεις).
- Ορισμοί δομών, δηλώσεις καθολικών μεταβλητών και καθολικών σταθερών

Φωλιασμένη Ενσωμάτωση Κεφαλίδων

- Πολύ συχνά κεφαλίδες καλούνται μέσα από άλλες κεφαλίδες, συνδυάζοντας λειτουργίες
- Αν σε κάποιο σημείο μια κεφαλίδα καλείται πάνω από μια φορά, θα έχουμε διπλές δηλώσεις!

geometry.h :

```
#include "mymath.h"
float projection(float * base, float * vec);
…
```

signals.h :

```
#include "mymath.h"
void findCoefs(float* coefs, float* params);
```

MyProg.cpp :

```
#include "geometry.h"
#include "signals.h"
```

Πρόβλημα: Διπλές δηλώσεις των περιεχομένων του mymath.h

Λύση: Ενσωμάτωση Κώδικα υπό Συνθήκη

- Δηλώνουμε όλο το περιεχόμενο του αρχείου κεφαλίδας μέσα σε:

```
#ifndef _Kapoio_monadiko_onoma_
#define _Kapoio_monadiko_onoma_
… [Κώδικας κεφαλίδας]
#endif
```

- Όσες φορές και να γίνει #include το παραπάνω αρχείο, ο κώδικας μέσα στο #ifndef … #endif θα διαβαστεί από τον προεπεξεργαστή μόνο μια φορά
- Ισοδύναμη δήλωση προεπεξεργαστή: #pragma once

Παράδειγμα

mymath.h

```
#ifndef _MYMATH_
#define _MYMATH_
float dotProduct(float* a, float* b);
bool isZero(float* a);
…
#endif
```

Ισοδύναμα:

```
#pragma once
float dotProduct(float* a, float* b);
bool isZero(float* a);
…
```

geometry.h :

```
#include "mymath.h"
float projection(float * base, float * vec);
…
```

signals.h :

```
#include "mymath.h"
void findCoefs(float* coefs, float* params);
```

MyProg.cpp :

```
#include "geometry.h"
#include "signals.h"
```

Εδώ το _MYMATH_ έχει ήδη δηλωθεί, οπότε δε θα μπει στο #define…#endif της mymath.h