C++: Από τη Θεωρία στην Εφαρμογή

Κεφάλαιο 1ο

Εισαγωγή στη Γλώσσα C++

Σύντομη Αναδρομή

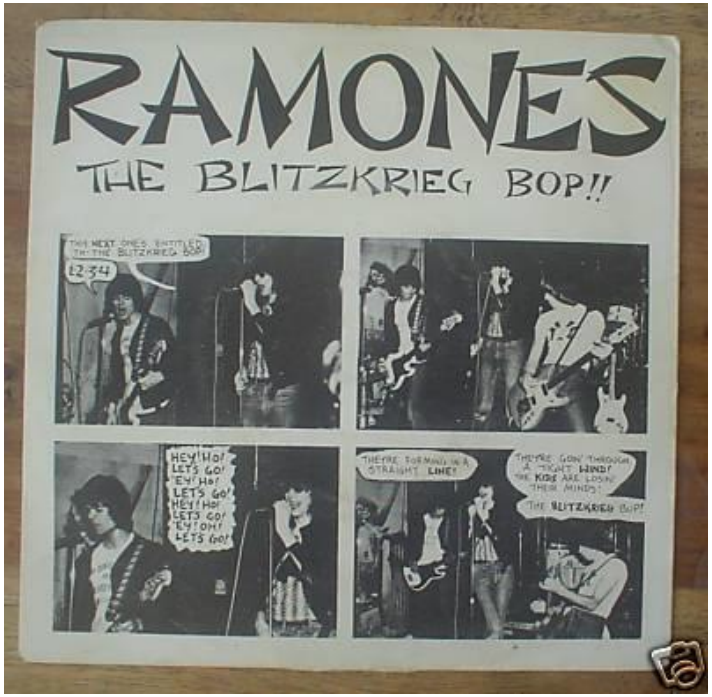
- Η C++ προέρχεται από τη γλώσσα C, η οποία αναπτύχθηκε στις αρχές της δεκαετίας του 1970
- Η C++ άρχισε να αναπτύσσεται από τον Bjarne Stroustrup και συναδέλφους του, στις αρχές της δεκαετίας του 1980
- Η C++ υποστηρίζει *διαδικασιακό* προγραμματισμό με τη χρήση δομών ελέγχου, εντολών επανάληψης και συναρτήσεων, *αντικειμενοστρεφή* προγραμματισμό με την υλοποίηση και χρήση ιεραρχιών κλάσεων και αντικειμένων και *γενικευμένο* προγραμματισμό με την υλοποίηση και την χρήση προτύπων και γενικών αλγορίθμων
- Η C++ τυποποιήθηκε αρχικά το 1998 και από τότε συνεχίζει να εξελίσσεται με την παραγωγή νέων προτύπων
- Ένας μεγάλος αριθμός δημοφιλών εφαρμογών που χρησιμοποιούμε στην καθημερινή μας ζωή είναι γραμμένες σε C++

Κύκλος Δημιουργίας Προγράμματος

- Συγγραφή προγράμματος. Για τη συγγραφή του κώδικα μπορεί να χρησιμοποιηθεί οποιοσδήποτε διαθέσιμος κειμενογράφος
- Μεταγλώττιση προγράμματος. Ο μεταγλωττιστής (compiler) δημιουργεί ένα αρχείο, το οποίο περιέχει τον κώδικα του προγράμματος μεταφρασμένο στη γλώσσα που κατανοεί ο υπολογιστής, η οποία ονομάζεται γλώσσα μηχανής (machine language). Αυτό το αρχείο ονομάζεται αρχείο αντικειμένου (object file)
- Σύνδεση προγράμματος. Ο συνδέτης (linker) αναλαμβάνει να συνδέσει τον κώδικα αντικειμένου με τον κώδικα βιβλιοθήκης (library code) που χρησιμοποιεί το πρόγραμμα και άλλη απαραίτητη πληροφορία. Ο κώδικας βιβλιοθήκης περιέχει τον κώδικα μηχανής κλάσεων και συναρτήσεων βιβλιοθήκης που χρησιμοποιεί το πρόγραμμα
- Εκτέλεση προγράμματος. Ο συνδέτης παράγει το εκτελέσιμο αρχείο (π.χ. .exe στα Windows), το οποίο μπορούμε να εκτελέσουμε

Το Πρώτο Πρόγραμμα

```
#include <iostream>
int main()
{
        std::cout << "Hey Ho, Let's Go\n";
        return 0;
}
```



Η οδηγία #include

- #include <iostream>

  - Με την οδηγία #include <όνομα_αρχείου> ο μεταγλωττιστής υποχρεώνεται να συμπεριλάβει (include) το περιεχόμενο του αρχείου στον κώδικα του προγράμματος

  - Το αρχείο iostream περιέχει πληροφορία για κλάσεις και συναρτήσεις που είναι απαραίτητες για το διάβασμα δεδομένων (π.χ. από το πληκτρολόγιο) και την εμφάνιση δεδομένων (π.χ. στην οθόνη)

Παρατηρήσεις

- Το αρχείο `iostream` συμπεριλαμβάνεται σχεδόν πάντα σε ένα C++ πρόγραμμα, αφού περιέχει την απαραίτητη πληροφορία για την εμφάνιση και διάβασμα δεδομένων

- Οι οδηγίες (π.χ. `#include`) ξεκινούν πάντα με τον χαρακτήρα **#** και στο τέλος της οδηγίας δεν προστίθεται το ερωτηματικό **;** ή κάποιος άλλος χαρακτήρας

- Ο προγραμματιστής μπορεί να δημιουργήσει και δικά του αρχεία και να τα συμπεριλάβει στο πρόγραμμά του με την οδηγία `#include`

- Κατά τη μεταγλώττιση του προγράμματος, ο μεταγλωττιστής ψάχνει να βρει σε ποιο σημείο του δίσκου είναι αποθηκευμένο το αρχείο για να το συμπεριλάβει στο πρόγραμμα

Η Συνάρτηση `main()`

- `int main()`
  - Ένα πρόγραμμα μπορεί να περιέχει πολλές συναρτήσεις
    - πρέπει όμως οπωσδήποτε να περιέχει τη συνάρτηση `main()`, η οποία αποτελεί την «κύρια συνάρτηση του προγράμματος»
  - Η `main()` καλείται αυτόματα από το λειτουργικό σύστημα όταν εκτελεστεί το πρόγραμμα
  - Η εκτέλεση του προγράμματος αρχίζει με την πρώτη εντολή της `main()` και τελειώνει με την τελευταία εντολή της, εκτός αν κληθεί νωρίτερα μία εντολή εξόδου, όπως η `return`
  - Οι εντολές της συνάρτησης `main()` ή αλλιώς το «σώμα της συνάρτησης» πρέπει να περιέχονται μέσα σε άγκιστρα `{}`

Η Συνάρτηση `main()`

- `int main()`
  - Η ειδική λέξη `int` σημαίνει ότι η `main()` πρέπει να επιστρέψει έναν ακέραιο αριθμό στο λειτουργικό σύστημα όταν τερματίσει
  - Αυτός ο ακέραιος επιστρέφεται με την εντολή `return`. Η τιμή 0 δηλώνει τον ομαλό τερματισμό του προγράμματος. Συνήθως, μία μη-μηδενική τιμή υποδεικνύει το λάθος που συνέβη

Το cout Αντικείμενο

- `std::cout << "Hey Ho, Let's Go\n";`
  - Οι εντολές γράφονται συνήθως μία σε κάθε γραμμή και σχεδόν πάντα τελειώνουν με το ελληνικό ερωτηματικό (;)
  - Το `cout` είναι ένα αντικείμενο το οποίο δηλώνεται στον χώρο ονομάτων `std` και συνδέεται με την προκαθορισμένη έξοδο (π.χ. οθόνη)
  - Το `cout` ξέρει πώς να εμφανίζει διαφορετικούς τύπους δεδομένων, όπως αλφαριθμητικά, αριθμούς και χαρακτήρες
  - Η σημειογραφία `<<` δηλώνει ότι το αλφαριθμητικό αποστέλλεται στο `cout`
  - Ο χαρακτήρας `'\n'` δημιουργεί μία νέα γραμμή μετά την εμφάνιση του μηνύματος στην οθόνη
    - Ισοδυναμεί δηλ. με το πάτημα του πλήκτρου Enter

Χώροι Ονομάτων (1)

- Ο χώρος ονομάτων είναι ένα τμήμα του προγράμματος, στο οποίο δηλώνονται ορισμένα ονόματα (π.χ. ονόματα μεταβλητών). Αυτά τα ονόματα δεν είναι γνωστά (ορατά) έξω από αυτόν τον χώρο
- Για παράδειγμα, όλα τα ονόματα της καθιερωμένης βιβλιοθήκης ορίζονται σε ένα χώρο ονομάτων που ονομάζεται `std`
- Για να έχουμε πρόσβαση σε ένα στοιχείο του χώρου πρέπει να γράψουμε το όνομα του χώρου ακολουθούμενο από τον τελεστή επίλυσης εμβέλειας `::`. Έτσι, γράφοντας `std::cout` έχουμε πρόσβαση στο `cout` αντικείμενο που δηλώνεται στον `std` χώρο

Χώροι Ονομάτων (2)

- Αν θέλουμε να είναι διαθέσιμα όλα τα ονόματα του `std` χώρου γράφουμε: `using namespace std;` Τώρα, δεν χρειάζεται να προσθέτουμε το πρόθεμα `std::` πριν από τα ονόματα που χρησιμοποιούμε. Για παράδειγμα, μπορούμε να γράψουμε: `cout << "Hey Ho, Let's Go\n";`
- Εναλλακτικά, μπορούμε να κάνουμε διαθέσιμα μόνο τα ονόματα που χρησιμοποιούνται στο πρόγραμμά μας. Αυτό επιτυγχάνεται με αντίστοιχες `using` δηλώσεις. Για παράδειγμα: `using std::cout;` Τώρα, δεν χρειάζεται να προσθέτουμε το πρόθεμα `std::` όταν χρησιμοποιούμε το `cout`, ενώ πρέπει με άλλα ονόματα του `std`

Εισαγωγή Σχολίων σε Πρόγραμμα (1)

- **Τι είναι τα «Σχόλια»;**
  - Είναι κείμενο το οποίο εισάγεται από τον προγραμματιστή στον κώδικα προκειμένου να καταστήσει τον ίδιο τον κώδικα περισσότερο ευανάγνωστο και σαφή
  - Τα σχόλια εισάγονται είτε για επεξήγηση προς τρίτους (που πιθανόν διαβάσουν τον κώδικά μας) είτε και για μας τους ίδιους (ιδίως όταν πρόκειται να διαβάσουμε τον κώδικά μας μετά από αρκετό χρονικό διάστημα)
  - Περιέχεται μεταξύ `/*` και `*/` (και αγνοείται από τον υπολογιστή)
  - Επίσης σχόλιο είναι και το κείμενο στην ίδια γραμμή μετά από `//`

Εισαγωγή Σχολίων σε Πρόγραμμα (2)

- π.χ.

```
/* Ο σκοπός του πρώτου προγράμματος είναι να εμφανίσει ένα μήνυμα στην
οθόνη. */
#include <iostream>
using std::cout; // Θα χρησιμοποιήσουμε το αντικείμενο cout.

int main()
{
        cout << "Hey Ho, Let's Go\n";
        return 0;
}
```

Λάθη Μεταγλώττισης (1)

- Τα πιο συνηθισμένα λάθη μεταγλώττισης, ιδίως από αρχάριους προγραμματιστές, είναι τα συντακτικά
- Η C++ είναι μία γλώσσα προγραμματισμού που κάνει διάκριση μεταξύ πεζών και κεφαλαίων γραμμάτων (case sensitive)
  - Δηλαδή, αν γράψουμε `Cout` αντί για `cout` ο μεταγλωττιστής θα εμφανίσει μήνυμα λάθους, γιατί ο χαρακτήρας `'C'` είναι διαφορετικός από τον χαρακτήρα `'c'`

- Μερικές φορές το λάθος που υποδεικνύει ένας μεταγλωττιστής μπορεί να μη βρίσκεται στη γραμμή που υποδεικνύει, αλλά σε κάποια προηγούμενη, με πιθανότερη την αμέσως προηγούμενη

- Αν ο μεταγλωττιστής εμφανίζει πολλά μηνύματα λάθους, τότε διορθώστε μόνο το πρώτο λάθος ή μαζί και κάποια άλλα προφανή λάθη και μεταγλωττίστε πάλι το πρόγραμμά σας. Είναι πολύ πιθανό, η νέα μεταγλώττιση να εμφανίσει πολύ λιγότερα ή και καθόλου λάθη

Λάθη Μεταγλώττισης (2)

- Ο μεταγλωττιστής μπορεί να μην εμφανίσει μηνύματα λάθους, αλλά να εμφανίσει μηνύματα προειδοποίησης (warnings). Ένα μήνυμα προειδοποίησης μπορεί να ενημερώνει τον προγραμματιστή για ενδεχόμενη δυσλειτουργία του προγράμματός του, οπότε καλό είναι να μην τα αγνοείτε

- Ο μεταγλωττιστής ανιχνεύει λανθασμένη εφαρμογή των κανόνων της γλώσσας που τον εμποδίζει να μεταγλωττίσει το πρόγραμμα σωστά (π.χ. ορθογραφικά λάθη, συντακτικά λάθη, αδήλωτες μεταβλητές, ...) και όχι λάθη λογικής που ενδέχεται να περιέχονται στο πρόγραμμά σας

- Δηλαδή, ο μεταγλωττιστής δεν είναι «μέσα στο κεφάλι σας» για να ξέρει ποιος είναι ο σκοπός του προγράμματός σας

- Άρα, αν το πρόγραμμά σας μεταγλωττίζεται σωστά, δεν σημαίνει απαραίτητα ότι θα παράγει και σωστά αποτελέσματα

Λάθη Μεταγλώττισης (3)

- Π.χ. αν θέλετε το πρόγραμμά σας να εμφανίζει τη λέξη Less αν η τιμή του a είναι μικρότερη από 5 και γράψετε:

```
if(a > 5) // Λογικό λάθος.
        cout << "Less\n";
```

τότε, αν και αυτός ο κώδικας μεταγλωττίζεται επιτυχημένα, το πρόγραμμα δεν θα λειτουργεί όπως θα θέλατε

- Λάθη σαν και αυτό ονομάζονται **λογικά λάθη** (bugs) και δεν εντοπίζονται από τον μεταγλωττιστή. Αποτελεί υποχρέωση του προγραμματιστή να τα εντοπίσει και να τα διορθώσει
- Η διαδικασία ανίχνευσης λαθών ονομάζεται αποσφαλμάτωση *(debugging)*. Οι μεταγλωττιστές συνήθως συνοδεύονται από ένα περιβάλλον ανίχνευσης λαθών, το οποίο βοηθάει τον προγραμματιστή να εντοπίσει τα λάθη του προγράμματός του