*Aναστάσιος Βήττας 197*

*24.11.2024*

*Προηγμενεσ δικτυακεσ τεχνολογιεσ*

Μέρος 1 – Υλοποίηση

Table of Contents

[Υλοποίηση 2](#_Toc183367671)

[Αποτελέσματα 2](#_Toc183367672)

[Άσκηση Hello.c 2](#_Toc183367673)

[Άσκηση String.c 3](#_Toc183367674)

[Άσκηση Pointers.c 3](#_Toc183367675)

[Άσκηση Args.c 4](#_Toc183367676)

[Άσκηση Structure.c 4](#_Toc183367677)

[Άσκηση Κατανόησης 5](#_Toc183367678)

[Βιβλιογραφία 6](#_Toc183367679)

# Υλοποίηση

Για την υλοποίηση των σεναρίων ο υπολογιστής μας θα πρέπει να έχει την C/C++ εγκατεστημένη και περασμένη στο Path του.

A screenshot of a computer

Description automatically generated

Το λειτουργικό του υπολογιστή που χρησιμοποίησα για την εργασία είναι Unix based, οπότε δεν χρειάστηκε να εγκαταστήσω κάποιο VM και να βάλω Linux, όπως ζητήθηκε.

Για την συγγραφή του κώδικα χρειάστηκα το Visual Studio Code και το Terminal του υπολογιστή μου. Ο κώδικας γραφόταν στο VSC, αποθηκευόταν και ύστερα γινόταν compile στο Terminal.

Για ένα πρόγραμμα αποθηκευμένο ως hello.c οι εντολές για την εκτέλεση του είναι οι εξής:

1. Για την μεταγλώττιση του.



**gcc:** Καλεί τον μεταγλωττιστή GCC (GNU Compiler Collection) για τη μεταγλώττιση του κώδικα.

**-o hello:** Ορίζει το όνομα του εκτελέσιμου αρχείου ως hello. Χωρίς αυτό παράγει ένα εκτελέσιμο αρχείο με όνομα a.out.

**-Wall:** Ενεργοποιεί όλες τις βασικές προειδοποιήσεις για πιθανά λάθη στον κώδικα. Είναι πολύ χρήσιμο για τον εντοπισμό προβλημάτων κατά την ανάπτυξη.

**hello.c:** Ορίζει το αρχείο πηγαίου κώδικα που πρέπει να μεταγλωττιστεί.

1. Για την εκτέλεση του.



# 

# Αποτελέσματα

## Άσκηση Hello.c

Η πρώτη άσκηση ήταν δοκιμαστική με σκοπό να δούμε αν η εγκατάσταση της C έχει γίνει με επιτυχία.

Το αποτέλεσμα μεταγλώττισης και εκτέλεσης της είναι το εξής:



## 

## Άσκηση String.c

Στην δεύτερη άσκηση καλούμαστε να φτιάξουμε έναν πίνακα χαρακτήρων που το τέλος του θα ορίζεται με τον χαρακτήρα τερματισμού '\0'.

char greeting[6 ] = {'H','e','l','l','o','\0'};

Η έξοδος του προγράμματος είναι η εξής:



**Αν αφαιρεθεί ο χαρακτήρας '\0' από τον κώδικα θα εκτελεστεί το πρόγραμμα; Γιατί;**

Πρώτον το πρόγραμμα θα εκτελεστεί κανονικά αλλά έχει τις βασικές διαφοροποιήσεις:

1. Δεν θεωρείται συμβολοσειρά. Είναι απλώς ένας πίνακας χαρακτήρων χωρίς το \0.
2. Δεν μπορούν να χρησιμοποιηθούν βασικές συναρτήσεις της βιβλιοθήκης string.h όπως strlen, strcpy κλπ., διότι αυτές βασίζονται στο '\0' για να καθορίσουν πού τελειώνει η συμβολοσειρά.
3. Αν χρησιμοποιηθεί η printf με %s όπως στο δικό μας παράδειγμα, ενδέχεται να διαβαστούν τυχαία δεδομένα πέρα από τα όρια του πίνακα. Αυτό μπορεί να οδηγήσει σε εκτύπωση garbage ή ακόμα και σε σφάλμα μνήμης.

## Άσκηση Pointers.c

Η τρίτη άσκηση έχει σκοπό την εκμάθηση των δεικτών στην C. Ένας δείκτης (pointer) είναι μια μεταβλητή που περιέχει διευθύνσεις μνήμης.

**Τι θα σας εμφανίζει η εκτέλεση του κώδικα pointers.c;**

Κατά την εκτέλεση του κώδικα που μας δόθηκε έχουμε την παρακάτω έξοδο.

A screen shot of a computer code

Description automatically generated

1. **Address of var variable:** Η διεύθυνση μνήμης όπου αποθηκεύεται η μεταβλητή var. Αυτή είναι μια συγκεκριμένη τιμή που εξαρτάται από την μνήμη που αποδίδεται από το σύστημα.
2. **Address stored on ip variable:** Η ίδια τιμή με την παραπάνω, αφού ο δείκτης ip παίρνει την διεύθυνση της var.

ip = &var;

1. **Value of \*ip variable:** Η τιμή που δείχνει ο δείκτης ip. Δεδομένου ότι δείχνει στη var, η τιμή είναι 20.

## Άσκηση Args.c

Για την ανάγνωση ορισμάτων από την γραμμή εντολών χρησιμοποιούμε την συνάρτηση

int main(int argc, char \*argv[])

argc: Ο αριθμός των ορισμάτων που θέλουμε να πάρει από το Terminal.

argv[]: Δείκτης των ορισμάτων.

**Πως θα εκτελέσετε τον παραπάνω κώδικα;**

Ο κώδικας args.c θα εκτελεστεί κατά τον παρακάτω τρόπο.

A screen shot of a computer code

Description automatically generated

Το πρώτο όρισμα που παίρνει το σύστημα είναι το όνομα του εκτελέσιμου δηλαδή argv[0], από εκεί και πέρα:

1. Αν είναι λιγότερο από 2, εκτυπώνει ότι χρειάζεται ένα ακόμα όρισμα.
2. Αν τα ορίσματα είναι 2, εκτυπώνει το όρισμα.
3. Αν είναι μεγαλύτερο από 2, εκτυπώνει ότι δόθηκαν πολλά ορίσματα.

## Άσκηση Structure.c

Η δομή (structure) περιέχει δεδομένα διαφορετικών τύπων. Η εκτέλεση του κώδικα structure.c που μας δόθηκε έχει ως αποτέλεσμα:

A black background with white text

Description automatically generated

**Τροποποιήστε τον κώδικα structure.c ώστε να διαβάζει το μήνυμα ως όρισμα από την γραμμή εντολών.**

Η τροποποίηση που επιδέχεται ο κώδικας που μας δόθηκε αφορά κυρίως την χρήση της συνάρτησης που είδαμε και στην άσκηση args.c

int main(int argc, char \*argv[])

Έτσι θα μπορέσουμε να δεχθούμε το όρισμα από την γραμμή εντολών και όχι χειροκίνητα κατά την συγγραφή του κώδικα.

Επίσης άλλη μια αλλαγή θα είναι να «πειράξουμε» και την συνάρτηση strncpy της βιβλιοθήκης string.h έτσι ώστε να παίρνει το όρισμα της θέσης argv[1] και να αφαιρεί την κενή θέση που προσπίπτει από το τερματικό χαρακτήρα \0 που είδαμε και στην άσκηση string.c.

Η έξοδος είναι η εξής:

A screen shot of a computer code

Description automatically generated

## Άσκηση Κατανόησης

Στην τελευταία άσκηση καλούμαστε να εξηγήσουμε την λειτουργία του παρακάτω κώδικα.

typedef struct { int a; short s[2]; } MSG;

MSG \*mp, m = {4, 1, 0};

char \*fp, \*tp;

mp = (MSG \*)malloc(sizeof(MSG));

printf("%d", (mp + 1));

for (fp = (char \*)m.s, tp = (char \*)mp->s; tp < (char \*)(mp + 1);) {

\*tp++ = \*fp++;

}

Αυτός ο κώδικας δεσμεύει δυναμική μνήμη για μια δομή MSG και αντιγράφει byte προς byte δεδομένα.

Αν συμπληρώνουμε τον κώδικα έτσι ώστε να είναι εκτελέσιμος θα πάρουμε την συγκεκριμένη έξοδο.

A black screen with white text

Description automatically generated

Αρχικά η malloc χρησιμοποιείται για να δεσμεύσει μνήμη για τον δείκτη mp. Αν δεν το κάνουμε, ο mp δείχνει σε μη έγκυρη μνήμη, οδηγώντας σε σφάλματα.

Φαίνεται λοιπόν ότι εντός της for loop μεταφέρουμε δεδομένα από το m.s στο mp->s. Ο δείκτης fp δείχνει στο m.s και ο tp στο mp->s.

## Βιβλιογραφία

1. <https://www.tutorialspoint.com/cprogramming/index.htm>
2. <https://stackoverflow.com/questions/52323643/what-happens-in-c-if-you-dont-use-0-at-the-end-of-a-string>
3. <https://www.programiz.com/c-programming/c-structures-pointers>
4. <https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/>