**Εργασία 2**

**Πάνου Χρήστος ΑΜ 1093457**

**Παραδοτέο Α**: Ανεβάστε στο eclass μια αναφορά σε μορφή pdf στην οποία θα ερμηνεύεται τα αποτελέσματα της εκτέλεσης του παραπάνω προγράμματος.

**1. Συγκεκριμένα ελέγξτε τις διευθύνσεις των τοπικών μεταβλητών (x,y) των συναρτήσεων foo1 και foo2. Τι παρατηρείτε και πως ερμηνεύετε τις παρατηρήσεις σας σύμφωνα με το μοντέλο διάταξης μνήμης;**

**2. Επίσης ελέγξτε τις διευθύνσεις των συναρτήσεων foo1, foo2 και τις καθολικής μεταβλητής g. Τι παρατηρείτε και πως ερμηνεύετε τις παρατηρήσεις σας σύμφωνα με το μοντέλο διάταξης μνήμης;**

**Αποτελέσματα Εκτέλεσης για Παραδοτέο Α**

PS C:\CLabs2023\output> & .\'ebdomada2.exe'

foo1 address 4299165908

x address 4294953932

y address 4294953928

g address 4299190272

foo2 address 4299165992

x address 4294953932

y address 4294953928

g address 4299190272

**Απάντηση στο Υποερώτημα 1**

* Όσον αφορά τη συνάρτηση foo1 στη Stack αποθηκεύονται οι τοπικές μεταβλητές x και y αποθηκεύονται στις διευθύνσεις 429495393216 (ή **285.960.649.01010**) και 429495392816 (ή **285.960.649.00010**).
* H διεύθυνση της foo1 αποθηκεύεται στη διεύθυνση μνήμης 429916590816 (ή **286.036.220.16810**) και η διεύθυνση της foo2 αποθηκεύεται στη διεύθυνση μνήμης 4299165992 (ή **286.036.220.30610**)
* Παρατηρούμε ότι οι διευθύνσεις των τοπικών μεταβλητών x και y βρίσκονται σε γειτονικές θέσεις μνήμης της stack που ξεκινά λογικά από τη διεύθυνση **285.960.649.00010** και οι διευθύνσεις των συναρτήσεων foo1 και foo2 βρίσκονται σε γειτονικές θέσεις μνήμης της Heap που ξεκινά λογικά από τη διεύθυνση **286.036.220.16810**
* Παρατηρούμε ότι οι διευθύνσεις των τοπικών μεταβλητών x και y είναι ίδιες και στις 2 κλήσεις των συναρτήσεων foo1 και foo2. Αυτό είναι αναμενόμενο διότι όταν τελειώσει η εκτέλεση της foo1() οι τοπικές μεταβλητές x, y αφαιρούνται από τη μνήμη οπότε όταν γίνει στη συνέχεια η κλήση της foo2 δεσμεύουν και πάλι τις ίδιες ακριβώς θέσεις μνήμης
* Σε σχέση με το μοντέλο διάταξης μνήμης παρατηρούμε ότι η Stack ξεκινά από τη διεύθυνση **285.960.649.00010** ενώ η Heap ξεκινά από τη διεύθυνση **286.036.220.16810**. Άρα η Heap είναι μετά τη Stack όπως υπαγορεύει και το μοντέλο διάταξης μνήμης. Συνεπώς επαληθεύεται το μοντέλο διάταξης μνήμης όσον αφορά το Stack και το Heap

**Απάντηση στο Υποερώτημα 2**

* Η καθολική μεταβλητή g αποθηκεύεται στη διεύθυνση **429919027216** (ή **286.036.394.61010**) και στην κλήση της foo1 και στην κλήση της foo2 κάτι αναμενόμενο καθώς η καθολική μεταβλητή g υπάρχει ανεξάρτητα από το ποια συνάρτηση εκτελείται
* Σε σχέση με το μοντέλο διάταξης μνήμης παρατηρούμε ότι το τμήμα Data ξεκινά από τη διεύθυνση **286.036.394.61010** ενώ η Heap ξεκινά από τη διεύθυνση **286.036.220.16810**. Άρα το τμήμα Data είναι μετά τη Heap όπως υπαγορεύει και το μοντέλο διάταξης μνήμης. Συνεπώς και πάλι επαληθεύεται το μοντέλο διάταξης μνήμης όσον αφορά το Data και το Heap

**Παραδοτέο Β**: Συμπληρώστε την αναφορά σας στην οποία θα ερμηνεύεται επιπλέον και τα αποτελέσματα της εκτέλεσης του παραπάνω προγράμματος.

1. Συγκεκριμένα ελέγξτε τις διευθύνσεις των μεταβλητών static int x και int y των συναρτήσεων foo1 και foo2. Τι παρατηρείτε και πως ερμηνεύετε τις παρατηρήσεις σας σύμφωνα με το μοντέλο διάταξης μνήμης;

2. Ελέγξτε τις διευθύνσεις των συναρτήσεων foo1, foo2 και τις καθολικής μεταβλητής g.

Τι παρατηρείτε και πως ερμηνεύετε τις παρατηρήσεις σας σύμφωνα με το μοντέλο διάταξης μνήμης;

3. Όπως μπορείτε να δείτε στην Β έκδοση του προγράμματος η συνάρτηση foo2 καλείτε μέσα στο σώμα της συνάρτησης foo1. Ποιες αλλαγές συνεπάγει το παραπάνω γεγονός στην διάταξη της μνήμης της εφαρμογής;

4.Τι πιστεύετε ότι θα γίνει αν στην συνάρτηση foo2 προσθέσουμε μια εντολή που θα καλεί την foo1(); πριν από το return;

**Αποτελέσματα Εκτέλεσης για Παραδοτέο Β**

foo1 address 4299165903

x address 4299190272

y address 4294953932

g address 4299169808

x address 4299190276

y address 4294953868

g address 4299169808

foo2 address 4299165995

**Απάντηση στο 1**

* Όσον αφορά τις διευθύνσεις των **μεταβλητών static int x και int y** των συναρτήσεων foo1 και foo2 αυτές αποθηκεύονται στις διευθύνσεις 429919027216 (**ή 286.036.394.61010**) και 429495393216 (**ή 285.960.649.01010**)
* Σε σχέση με το μοντέλο διάταξης μνήμης παρατηρούμε ότι η Stack ξεκινά από τη διεύθυνση **285.960.649.00010** ενώ το τμήμα Data ξεκινά από τη διεύθυνση **286.036.394.61010**. Άρα το τμήμα Data είναι μετά τη Stack όπως υπαγορεύει και το μοντέλο διάταξης μνήμης. Συνεπώς επαληθεύεται το μοντέλο διάταξης μνήμης όσον αφορά το Stack και το Data

**Απάντηση στο 2**

* H διεύθυνση της foo1 αποθηκεύεται στη διεύθυνση μνήμης 429916590316 (ή **286.036.220.16310**) και η διεύθυνση της foo2 αποθηκεύεται στη διεύθυνση μνήμης 4299165992 (ή **286.036.220.30910**). Άρα η Heap ξεκινά από τη διεύθυνση **286.036.220.16310** που είναι πριν το τμήμα Data το οποίο ξεκινά από τη διεύθυνση **286.036.394.61010** και η Heap είναι μετά το Stack που ξεκινά από τη διεύθυνση **285.960.649.00010** οπότε επιβεβαιώνεται το μοντέλο διάταξης μνήμης αναφορικά με τις θέσεις των Stack, Heap και Data
* Επειδή η συνάρτηση foo2 καλείτε μέσα στο σώμα της συνάρτησης foo1 η διεύθυνση που αποθηκεύεται η foo2 είναι η **286.036.220.30910** ενώ η διεύθυνση που αποθηκεύεται η foo1 είναι η **286.036.220.16310**. Παρατηρούμε ότι και οι 2 διευθύνσεις είναι στο Heap και η διεύθυνση της foo2 είναι μεταγενέστερη της διεύθυνσης της foo1 κάτι απόλυτα λογικό αφού η foo2 καλείτε μέσω της foo1
* Η καθολική μεταβλητή g αποθηκεύεται στη διεύθυνση **429916980816** (ή **286.036.236.29610**) και στις 2 περιπτώσεις

**Απάντηση στο 3**

* H μεταβλητή static int x της foo2 αποθηκεύεται στη διεύθυνση **285.960.649.010**10 και η τοπική μεταβλητή y της foo2 αποθηκεύεται στη διεύθυνση **285.960.649.000**10. Παρατηρούμε ότι η μεταβλητή static int x της foo2 δεν αποθηκεύεται στο τμήμα Data που αρχίζει στη διεύθυνση **286.036.394.61010** αλλά στο τμήμα Stack που ξεκινά στη διεύθυνση **285.960.649.00010** και αυτό αποτελεί την αλλαγή που συνεπάγει το παραπάνω γεγονός στην διάταξη της μνήμης της εφαρμογής

**Απάντηση στο 4**

* Αν στην συνάρτηση foo2 προσθέσουμε μια εντολή που θα καλεί την foo1(); πριν από το return τότε θα προκληθεί ένας άπειρος βρόγχος διότι η μια συνάρτηση θα καλεί συνεχώς την άλλη δηλ. η foo1() θα καλεί τη foo2() και η foo2() θα καλεί τη foo1()