Αναφορά Εργαστηρίου 1

Προεργασία

Για την προεργασία του 1ου εργαστηρίου χρειάστηκε να ετοιμάσουμε τους κώδικες των 5 ερωτημάτων της άσκησης σε γλώσσα C. Επίσης χρειάστηκε να τρέξουμε τα επιμέρους προγράμματα και να παρατηρήσουμε τι συμβαίνει στη μνήμη.

Περιγραφή Ζητουμένων

Σκοπός της άσκησης ήταν να κατανοήσουμε μέσω κάποιων εντολών της C τι συμβαίνει στη μνήμη. Οι τοπικές μεταβλητές αποθηκεύονται στην περιοχή Stack, οι global στην Heap και κάποιες συναρτήσεις όπως η malloc στην Static. Επιπλέον σκοπός ήταν να εξασκηθούμε στην αναπαράσταση της μνήμης σε δεκαεξαδικό σύστημα αρίθμησης.

Περιγραφή Εκτέλεσης

Στο 1ο ερώτημα παρατηρήσαμε ότι οι θέσεις στη μνήμη των παραστάσεων A+1 και (int)A+1 απέχουν μεταξύ τους 3 θέσεις μνήμης.Αυτό γίνεται διότι το A είναι ένας ακεραίος και όταν προστίθεται το 1 μετακινείται στην επόμενη θέση μνήμης και γίνεται μια νέα μεταβλητη.Όμως με τον τελεστή int η τιμή του A αυξάνεται κατά ένα.

Στο 2ο ερώτημα παρατηρήσαμε πως η C ταξινομεί τις θέσεις των μεταβλητών στη μνήμη κατά φθίνουσα σειρά.

Στο 3ο ερώτημα παρατηρήσαμε ότι καμία δομή δεν είχε το πραγματικό μέγεθος των 6 bytes. Αυτό συμβαίνει επειδή η C κατανέμει τις μεταβλητές στη μνήμη με τέτοιο τρόπο έτσι ώστε η διεύθυνση της μνήμης να είναι πολλαπλάσιο του 4.

Στο 4ο ερώτημα παρατηρήσαμε ότι σε κάθε περίπτωση η απόσταση των διαδοχικών malloc στην μνήμη είναι 24 bytes ανεξάρτητα από το μέγεθος το οποίο δεσμέυσαμε.

Στο 5ο ερώτημα παρατηρήσαμε πως οι διευθύνσεις των main,global variable και malloc είναι πολύ κοντά.Αντίθετα η διεύθυνση της τοπικής μεταβλητής είναι μακρυά από τις υπόλοιπες.

Συμπεράσματα

Ένα από τα συμπεράσματα που προέκυψαν είναι ότι η malloc ως συνάρτηση της C χρειάζεται για την λειτουργία της κάποια επιπλέον bytes.Γι αυτό προέκυψαν περισσότερα bytes από αυτά που απαιτήσαμε για δέσμευση.Επίσης συμπεράναμε πως οι περιοχές Heap και Static είναι πολύ κοντά σε σύγκριση με την Stack.

Παράρτημα - Κώδικας

1ο ερώτημα

```
#include <stdio.h>

#include <stdlib.h>

int main() {

int A[10],i;

for (i = 0; i < 10; i++){

   A[i] = i;

 }

printf("\nA: %d",A);

printf("\nA: %x",A);

printf("\n\nA+1: %d",A+1);

printf("\nA+1: %x",A+1);
```

```
printf("\n\n(((int)A)+1): %d",((int)A)+1);
printf("\n(((int)A)+1): %x",((int)A+1));
printf("\n\n&A[1]: %d",&A[1]);
printf("\n&A[1]: %x",&A[1]);
printf("\n\nsizeof(A): %d",sizeof(A));
printf("\nsizeof(A[1]): %d",sizeof(A[0]));
return 0;
}
```

2ο ερώτημα

```
#include <stdio.h>
#include <stdlib.h>
int main() {
int a,b,c;
printf("\n%d",&a);
printf("\n%d",&b);
printf("\n%d",&c);
return 0;
}
```

3ο ερώτημα

```
#include <stdio.h>
#include <stdlib.h>
struct A {
        char X;
        int C;
```

```
        char Y;
};
struct B{
        char X;
        char Y;
        int C;
};
int main() {
printf("\n%d",sizeof(struct A));
printf("\n%d",sizeof(struct B));
return 0;
}
```

4ο ερώτημα

```
#include <stdio.h>
#include <stdlib.h>
int main() {
malloc(10);
malloc(10);
malloc(16);
malloc(32);
printf("\n%d",malloc(10));
printf("\n%d",malloc(10));
printf("\n%d",malloc(16));
printf("\n%d",malloc(32));
```

```
return 0;
}
```

5ο ερώτημα

```
#include <stdio.h>
#include <stdlib.h>
int i;
int main() {
int a;
printf("\n%d",main);
printf("\n%d",&i);
printf("\n%d",&a);
printf("\n%d",&malloc);
return 0;
}
```