**Παραδοτέο Α**

**Κώδικας**

int main(void)

{

char array\_char[SIZE];

int array\_int[SIZE];

double array\_double[SIZE];

//ΚΩΔΙΚΑΣ ΠΟΥ ΘΑ ΥΠΟΛΟΓΙΖΕΙ ΤΗΝ ΜΝΗΜΗ ΣΕ BYTES ΠΟΥ ΚΑΤΑΛΑΜΒΑΝΕΙ ΚΑΘΕ ΠΙΝΑΚΑΣ

size\_t array\_char\_size=sizeof(array\_char);

printf("Size of array array\_char in bytes: %zu\n", array\_char\_size);

size\_t array\_int\_size=sizeof(array\_int);

printf("Size of array array\_int in bytes: %zu\n", array\_int\_size);

size\_t array\_double\_size=sizeof(array\_double);

printf("Size of array array\_double in bytes: %zu\n", array\_double\_size);

return 0;

}

**Αποτελέσμστα Εκτέλεσης**

Size of array array\_char in bytes: 10

Size of array array\_int in bytes: 40

Size of array array\_double in bytes: 80

**Αιτιολόγηση Αποτελεσμάτων**

* To μέγεθος του πίνακα array\_char είναι συνολικά 10 bytes. Αυτό δικαιολογείται απόλυτα από τη θεωρία διότι η συνάρτηση sizeof(array\_char) επιστρέφει το μέγεθος του πίνακα χαρακτήρων που είναι 10 bytes δηλ. 10 θέσεις μνήμης × 1 byte η κάθε θέση άρα σύνολο 10 bytes.
* To μέγεθος του πίνακα array\_int είναι συνολικά επίσης10 bytes. Αυτό δικαιολογείται απόλυτα από τη θεωρία διότι η συνάρτηση sizeof(array\_int) επιστρέφει το μέγεθος του πίνακα ακεραίων που είναι 40 bytes δηλ. 10 θέσεις μνήμης × 4 byte η κάθε θέση άρα σύνολο 40 bytes.
* To μέγεθος του πίνακα array\_double είναι συνολικά επίσης10 bytes. Αυτό δικαιολογείται απόλυτα από τη θεωρία διότι η συνάρτηση sizeof(array\_double) επιστρέφει το μέγεθος του πίνακα double που είναι 80 bytes δηλ. 10 θέσεις μνήμης × 8 byte η κάθε θέση άρα σύνολο 80 byte

**Παραδοτέο B**

**Κώδικας**

#include <stdio.h>

#define SIZE 10

void print\_memory\_size(char\* tbl\_char, int \*tbl\_int, double \*tbl\_double);

int main(void)

{

char array\_char[SIZE];

int array\_int[SIZE];

double array\_double[SIZE];

print\_memory\_size(array\_char, array\_int, array\_double);

return 0;

}

void print\_memory\_size(char\* tbl\_char, int \* tbl\_int, double \*tbl\_double)

{

int sum = 0;

for (int i = 0; i <SIZE;i++)

sum+=sizeof(\*tbl\_char);

printf("Size of array array\_char in bytes: %d\n", sum);

sum=0;

for (int i = 0;i <SIZE;i++)

sum+=sizeof(\*tbl\_int);

printf("Size of array array\_int in bytes: %d\n", sum);

sum=0;

for (int i = 0; i <SIZE;i++)

sum+=sizeof(\*tbl\_double);

printf("Size of array array\_double in bytes: %d\n", sum);

}

**Αποτελέσμστα Εκτέλεσης**

Size of array array\_char in bytes: 10

Size of array array\_int in bytes: 40

Size of array array\_double in bytes: 80

**Αιτιολόγηση Αποτελεσμάτων**

* Με το 1ο for προσθέτουμε στον αθροιστή sum το μέγεθος σε bytes του στοιχείου στο οποίο δείχνει ο δείκτης tbl\_char. Το στοιχείο αυτό είναι το 1ο στοιχείο του πίνακα array\_char του main διότι όταν μεταβιβάζουμε ένα πίνακα σε συνάρτηση μεταβιβάζεται η διεύθυνση του αρχικού του στοιχείου στον αντίστοιχο δείκτη άρα εδώ ο δείκτης tbl\_char δείχνει στο 1ο στοιχείο του πίνακα array\_char. Άρα σε κάθε επανάληψη αθροίζεται το μέγεθος κάθε στοιχείου του πίνακα array\_char που είναι ένας χαρακτήρας αφού ο πίνακας έχει στοιχεία του ίδιου τύπου και στο τέλος τυπώνουμε το μέγεθος όλου του πίνακα array\_char που είναι 10 bytes όπως και πριν
* Με το 2ο for προσθέτουμε στον αθροιστή sum το μέγεθος σε bytes του στοιχείου στο οποίο δείχνει ο δείκτης tbl\_int. Το στοιχείο αυτό είναι το 1ο στοιχείο του πίνακα array\_int του main διότι όταν μεταβιβάζουμε ένα πίνακα σε συνάρτηση μεταβιβάζεται η διεύθυνση του αρχικού του στοιχείου στον αντίστοιχο δείκτη άρα εδώ ο δείκτης tbl\_int δείχνει στο 1ο στοιχείο του πίνακα array\_int. Άρα σε κάθε επανάληψη αθροίζεται το μέγεθος κάθε στοιχείου του πίνακα array\_int που είναι ένας ακέραιος αφού ο πίνακας έχει στοιχεία του ίδιου τύπου και στο τέλος τυπώνουμε το μέγεθος όλου του πίνακα array\_int που είναι 40 bytes όπως και πριν
* Με το 3ο for προσθέτουμε στον αθροιστή sum το μέγεθος σε bytes του στοιχείου στο οποίο δείχνει ο δείκτης tbl\_double. Το στοιχείο αυτό είναι το 1ο στοιχείο του πίνακα array\_ double του main διότι όταν μεταβιβάζουμε ένα πίνακα σε συνάρτηση μεταβιβάζεται η διεύθυνση του αρχικού του στοιχείου στον αντίστοιχο δείκτη άρα εδώ ο δείκτης tbl\_double δείχνει στο 1ο στοιχείο του πίνακα array\_double. Άρα σε κάθε επανάληψη αθροίζεται το μέγεθος κάθε στοιχείου του πίνακα array\_double που είναι ένας double αφού ο πίνακας έχει στοιχεία του ίδιου τύπου και στο τέλος τυπώνουμε το μέγεθος όλου του πίνακα array\_ double που είναι 80 bytes όπως και πριν

**Παραδοτέο Γ**

**Κώδικας**

#include <stdio.h>

#define SIZE 10

void initialize(char\* tbl\_char, int \* tbl\_int, double []);

void print(char [], int [], double []);

int main(void)

{

char array\_char[SIZE];

int array\_int[SIZE];

double array\_double[SIZE];

initialize(array\_char, &array\_int[0],&array\_double[0]);

print(array\_char, array\_int, array\_double);

return 0;

}

void initialize(char\* tbl\_char, int \* tbl\_int, double tbl\_double[])

{

//ΣΥΜΠΛΗΡΩΣΤΕ ΤΟΝ ΚΩΔΙΚΑ ΓΙΑ ΤΗΝ ΑΡΧΙΚΟΠΟΙΗΣΗ

int index;

for (index=0;index<SIZE; index++)

{

(tbl\_char)='A'+ index;

tbl\_char++;

\*(tbl\_int)=0+index;

tbl\_int++;

\*(tbl\_double)=0+2\*index;

tbl\_double++;

}

}

void print(char tbl\_char[], int tbl\_int[], double tbl\_double[])

{

int index;

for (index = 0;index<SIZE; index++)

{

printf(" %d \t tbl\_char value %c \n", index, tbl\_char[index]);

printf(" %d \t tbl\_int value %d \n", index, tbl\_int[index]);

printf(" %d \t tbl\_double value %.3f \n", index, tbl\_double[index]);

}

}

**Αποτελέσμστα Εκτέλεσης**

tbl\_char value A

0 tbl\_int value 0

0 tbl\_double value 0.000

1 tbl\_char value B

1 tbl\_int value 1

1 tbl\_double value 2.000

2 tbl\_char value C

2 tbl\_int value 2

2 tbl\_double value 4.000

3 tbl\_char value D

3 tbl\_int value 3

3 tbl\_double value 6.000

4 tbl\_char value E

7 tbl\_char value H

7 tbl\_int value 7

7 tbl\_double value 14.000

8 tbl\_char value I

8 tbl\_int value 8

8 tbl\_double value 16.000

9 tbl\_char value J

9 tbl\_int value 9

9 tbl\_double value 18.000

**Αιτιολόγηση Αποτελεσμάτων**

Η κλήση της συνάρτησης initialize(array\_char, &array\_int[0],&array\_double[0]); γίνεται με αναφορά διότι μεταβιβάζουμε τη διεύθυνση του αρχικού στοιχείου του πίνακα array\_char στο δείκτη \* tbl\_char, μεταβιβάζουμε στη συνέχεια τη διεύθυνση του αρχικού στοιχείου του πίνακα array\_int στο δείκτη \* tbl\_int και τέλος μεταβιβάζουμε τη διεύθυνση του αρχικού στοιχείου του πίνακα array\_double στο δείκτη \* tbl\_double

H κλήση της συνάρτησης print(array\_char, array\_int, array\_double) γίνεται και πάλι με αναφορά διότι όταν μεταβιβάζουμε ένα πίνακα σε συνάρτηση, ουσιαστικά αυτό που μεταβιβάζεται πάντα είναι μόνο η διεύθυνση του αρχικού στοιχείου του εκάστοτε πίνακα έστω και αν στην υλοποίηση της συνάρτησης print χρησιμοποιούμε τους τοπικούς πίνακες-παραμέτρους char tbl\_char[], int tbl\_int[], double tbl\_double[])

**Κώδικας με τροποποιημένη print ώστε να τυπώνει και τη διεύθυνση κάθε στοιχείου**

#include <stdio.h>

#define SIZE 10

void initialize(char\* tbl\_char, int \* tbl\_int, double []);

void print(char [], int [], double []);

int main(void)

{

char array\_char[SIZE];

int array\_int[SIZE];

double array\_double[SIZE];

initialize(array\_char ,&array\_int[0],&array\_double[0]);

print(array\_char,array\_int,array\_double);

return 0;

}

void initialize(char\* tbl\_char, int \* tbl\_int, double tbl\_double[])

{

//ΣΥΜΠΛΗΡΩΣΤΕ ΤΟΝ ΚΩΔΙΚΑ ΓΙΑ ΤΗΝ ΑΡΧΙΚΟΠΟΙΗΣΗ

int index;

for (index=0;index<SIZE; index++)

{

(tbl\_char)='A'+ index;

tbl\_char++;

\*(tbl\_int)=0+index;

tbl\_int++;

\*(tbl\_double)=0+2\*index;

tbl\_double++;

}

}

void print(char tbl\_char[], int tbl\_int[], double tbl\_double[])

{

int index;

for (index = 0;index<SIZE; index++)

{

printf(" %d \t tbl\_char value %c and address %x \n", index, tbl\_char[index], &tbl\_char[index]);

printf(" %d \t tbl\_int value %d and address %x\n", index, tbl\_int[index], &tbl\_int[index]);

printf(" %d \t tbl\_double value %.3f and address %x\n", index, tbl\_double[index], &tbl\_double[index]);

}

}

Αν στην συνάρτηση print στον βρόχο επανάληψης αντί για το SIZE πληκτρολογήσουμε κατά λάθος SIZE+10 θα τυπώνονται και απροσδιόριστες θέσεις μνήμης