**A Παραδοτέο - Κώδικας**

#include <stdio.h>

#define SIZE 10

int main(void)

{

    char array\_char[SIZE];

    int array\_int[SIZE];

    double array\_double[SIZE];

    printf("Size of char array is %d bytes\n", sizeof(array\_char));

    printf("Size of int array is %d bytes\n", sizeof(array\_int));

    printf("Size of double array is %d bytes\n", sizeof(array\_double));

    return 0;

}

**A Παραδοτέο - Εκτέλεση**

Size of char array is 10 bytes

Size of int array is 40 bytes

Size of double array is 80 bytes

**A Παραδοτέο Περιγραφή**

* 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

**Β Παραδοτέο - 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 sumint=0,sumchar=0,sumdouble = 0;

    for (int i = 0; i <SIZE;i++)

    {

        sumchar+=sizeof(\*tbl\_char);

        sumint+=sizeof(\*tbl\_int);

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

    }

    printf("Size of char array is %d bytes\n",sumchar);

    printf("Size of int array is %d bytes\n",sumint);

    printf("Size of double array is %d bytes\n",sumdouble);

}

**Β Παραδοτέο - Εκτέλεση**

Size of char array is 10 bytes

Size of int array is 40 bytes

Size of double array is 80 bytes

**Β Παραδοτέο - Περιγραφή**

* 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

**Γ Παραδοτέο - 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\_int)++=0+index;

        \*(tbl\_double)++=0+2\*index;

    }

}

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]);

    }

}

**Γ Παραδοτέο - Εκτέλεση**

 0   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

 4   tbl\_int value 4

 4   tbl\_double value 8.000

 5   tbl\_char value F

 5   tbl\_int value 5

 5   tbl\_double value 10.000

 6   tbl\_char value G

 6   tbl\_int value 6

 6   tbl\_double value 12.000

 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

**Γ Παραδοτέο - Περιγραφή**

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[])

**Γ Παραδοτέο – Τροποποιημένος 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\_int)++=0+index;

        \*(tbl\_double)++=0+2\*index;

    }

}

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]);

    }

}

**Γ Παραδοτέο – Τροποποιημένος Κώδικας - Εκτέλεση**

Running.....

 0   tbl\_char value A and address: ffffcbd6

 0   tbl\_int value 0 and address: ffffcba0

 0   tbl\_double value 0.000 and address:  ffffcb50

 1   tbl\_char value B and address: ffffcbd7

 1   tbl\_int value 1 and address: ffffcba4

 1   tbl\_double value 2.000 and address:  ffffcb58

 2   tbl\_char value C and address: ffffcbd8

 2   tbl\_int value 2 and address: ffffcba8

 2   tbl\_double value 4.000 and address:  ffffcb60

 3   tbl\_char value D and address: ffffcbd9

 3   tbl\_int value 3 and address: ffffcbac

 3   tbl\_double value 6.000 and address:  ffffcb68

 4   tbl\_char value E and address: ffffcbda

 4   tbl\_int value 4 and address: ffffcbb0

 4   tbl\_double value 8.000 and address:  ffffcb70

 5   tbl\_char value F and address: ffffcbdb

 5   tbl\_int value 5 and address: ffffcbb4

 5   tbl\_double value 10.000 and address:  ffffcb78

 6   tbl\_char value G and address: ffffcbdc

 6   tbl\_int value 6 and address: ffffcbb8

 6   tbl\_double value 12.000 and address:  ffffcb80

 7   tbl\_char value H and address: ffffcbdd

 7   tbl\_int value 7 and address: ffffcbbc

 7   tbl\_double value 14.000 and address:  ffffcb88

 8   tbl\_char value I and address: ffffcbde

 8   tbl\_int value 8 and address: ffffcbc0

 8   tbl\_double value 16.000 and address:  ffffcb90

 9   tbl\_char value J and address: ffffcbdf

 9   tbl\_int value 9 and address: ffffcbc4

 9   tbl\_double value 18.000 and address:  ffffcb98

**Παρατήρηση**

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