ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2016-2017

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ

ΜΑΘΗΜΑ: ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

ΕΡΓΑΣΤΗΡΙΟ 2 - ΣΗΜΕΙΩΣΕΙΣ

Συναρτήσεις

Μια συνάρτηση είναι ένα 'μαύρο κουτί' το οποίο περιλαμβάνει εντολές που επιτελούν κάποιες ενέργειες. Η δήλωση μιας συνάρτησης στη C γίνεται ως εξής:

```
return_data_type function_name(data_type variable1, data_type variable2,..)
{
        .............
        statements
        .............
        return(variable);
}
```

Ενώ η κλήση της:

```
variable_name = function_name(variable1, variable2, ...);
```

Code 1

```
#include <stdio.h>
int evenodd(int); //FUNCTION DECLARATION
int main()
{
        int num, flag;
        printf("\n Enter the number : ");
        scanf("%d", &num);
        flag = evenodd(num); //FUNCTION CALL
        if (flag == 1)
                printf("\n %d is EVEN", num);
        else
                printf("\n %d is ODD", num);
        return 0;
}

int evenodd(int a) // FUNCTION HEADER
{
        // FUNCTION BODY
        if(a%2 == 0)
                return 1;
        else
                retun 0;
}
```

Για το πέρασμα παραμέτρων στις συναρτήσεις υπάρχουν δύο τρόποι:

A. Call by value. Η συνάρτηση που καλέι περνά μόνο τις τιμές των μεταβλητών. Οποιαδήποτε αλλαγή γίνεται σε αντίγραφα των μεταβλητών και όχι στις ίδιες τις μεταβλητές.

B. Call by reference. Η συνάρτηση που καλεί περνά τις διευθύνσεις των μεταβλητών. Οποιαδήποτε αλλαγή γίνεται στις ίδιες τις μεταβλητές.

Code 2

```
#include <stdio.h>
void add(int n);
int main()
{
	int num = 2;
	printf("\n The value of num before calling the function = %d", num);
	add(num);
	printf("\n The value of num after calling the function = %d", num);
	return 0;
}
void add(int n)
{
	n = n + 10;
	printf("\n The value of num in the called function = %d", n);
}
```

Code 3

```
#include <stdio.h>
void add(int *);
int main()
{
	int num = 2;
	printf("\n The value of num before calling the function = %d", num);
	add(&num);
	printf("\n The value of num after calling the function = %d", num);
	return 0;
}
void add(int *n)
{
	*n = *n + 10;
	printf("\n The value of num in the called function = %d", *n);
}
```

Δείκτες (pointers)

Κάθε μεταβλητή έχει ένα όνομα και μια τιμή. Μετά τη δήλωση της μεταβλητής, ένα τμήμα μνήμης δεσμεύεται για αυτή. Έτσι, κάθε μεταβλητή έχει τιμή και μια διεύθυνση μνήμης στην οποία έχει εκχωρηθεί η τιμή αυτή. Οι **δείκτες (pointers)** είναι μεταβλητές που περιέχουν τη διεύθυνση μνήμης μιας άλλης μεταβλητής. Η δήλωση των δεικτών γίνεται ως εξής:

```
data_type *ptr_name;
```

Παραδείγματα:

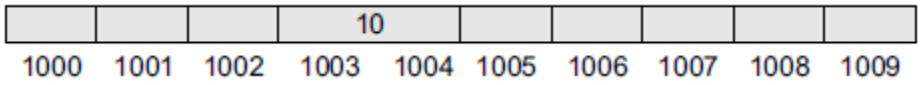
```
int *pnum;
char *pch;
float *pfnum;
```

Ακόμα και αν οι δείκτες ‘δείχνουν’ σε διαφορετικού τύπου δεδομένα, θα καταλάβουν τον ίδιο χώρο στη μνήμη αφού στην ουσία αποθηκεύουν διευθύνσεις μνήμης και όχι τιμές. Η χρήση του ***** ειδοποιεί το μεταγλωττιστή ότι πρόκειται για μια μεταβλητή τύπου δείκτη.

```
int x= 10;
int *ptr;
ptr = &x;
```

Στο παραπάνω παράδειγμα, το ptr δείχνει σε μια ακέραια μεταβλητή που καταλαμβάνει 2 bytes στη μνήμη. Το σύμβολο **&** ανακτά τη διεύθυνση μνήμης της μεταβλητής x.

Αν η μνήμη έχει την εξής μορφή:



Ο δείκτης θα πάρει την τιμή 1003.

Code 4

```
#include <stdio.h>
int main()
{
        int num, *pnum;
        pnum = &num;
        printf("\n Enter the number : ");
        scanf("%d", &num);
        printf("\n The number that was entered is : %d", *pnum);
        return 0;
}
```

Code 5

```
#include <stdio.h>
int main()
{
        int x=10;
        char ch = ‘A’;
        void *gp;
        gp = &x;
        printf("\n Generic pointer points to the integer value = %d", *(int*)gp);
        gp = &ch;
        printf("\n Generic pointer now points to the character= %c", *(char*)gp);
        return 0;
}
```

Code 6

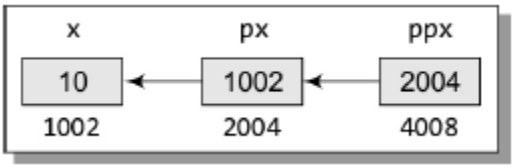
```
#include <stdio.h>
void sum (int*, int*, int*);
int main()
{
        int num1, num2, total;
        printf("\n Enter the first number : ");
```

```
        scanf("%d", &num1);
        printf("\n Enter the second number : ");
        scanf("%d", &num2);
        sum(&num1, &num2, &total);
        printf("\n Total = %d", total);
        return 0;
}
void sum (int *a, int *b, int *t)
{
        *t = *a + *b;
}
```

Τέλος, στη C μπορούμε να χρησιμοποιήσουμε δείκτες στο να δείχνουν σε άλλους δείκτες. Η δήλωση γίνεται μέσω της χρήσης του συμβόλου * σε κάθε επίπεδο των δεικτών.

Παράδειγμα:

```
int x=10;
int *px, **ppx;
px = &x;
ppx = &px;
```



Σε αυτό το παράδειγμα, η εντολή

```
printf("\n %d", **ppx);
```

Θα τυπώσει το 10.

Διάφορα κοινά λάθη που συμβαίνουν με τους δείκτες έχουν ως ακολούθως:

```
int x, *px;
x=10;
*px = 20;
```

Ο δείκτης px δεν έχει οριστεί και με την τρίτη εντολή θα γράψει στα περιεχόμενα της αντίστοιχης θέσης μνήμης την τιμή 20.

```
int x, *px;
x=10;
px = x;
```

Η σωστή εντολή είναι px = &x;

```
int x=10, y=20, *px, *py;
px = &x, py = &y;
if(px<py)
printf("\n x is less than y");
else
printf("\n y is less than x");
```

Η σωστή εντολή είναι *px < *py

Δομές Δεδομένων

Μια δομή δεδομένων (data structure) είναι μια ομάδα στοιχείων δεδομένων τα οποία τίθενται κάτω από ένα κοινό όνομα και καθορίζουν ένα συγκεκριμένο τρόπο αποθήκευσης και επεξεργασίας. Επίσης, για μια δομή δεδομένων ορίζεται ένα σύνολο επεξεργασιών για την αποδοτική διαχείριση της σε ένα υπολογιστικό σύστημα.

Κάποιες από τις πιο συχνά χρησιμοποιούμενες δομές δεδομένων είναι οι ακόλουθες:

- Πίνακας
- Λίστα
- Στοίβα
- Ουρά
- Δένδρα
- Γράφοι

Μέχρι το τέλος του εξαμήνου θα μελετήσουμε όλες τις παραπάνω δομές και θα υλοποιήσουμε προγραμματιστικά βασικές εφαρμογές με τη χρήση τους.

Γενικά, οι ενέργειες που μπορούν να γίνουν πάνω στις δομές δεδομένων έχουν ως ακολούθως:

- Διάσχιση
- Αναζήτηση
- Εισαγωγή
- Διαγραφή
- Ταξινόμηση
- Συγχώνευση
- Διαχωρισμός

Στο μάθημα των Αλγορίθμων του επόμενου εξαμήνου θα διδαχθείτε τον τρόπο υπολογισμού της πολυπλοκότητας διαφόρων αλγορίθμων που εφαρμόζονται πάνω στις δομές δεδομένων και θα τους μελετήσετε διεξοδικά.

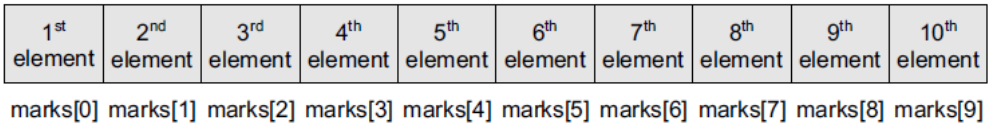
Πίνακες

Μια από τις πιο συχνά χρησιμοποιούμενες δομές δεδομένων είναι οι πίνακες. Ένας πίνακας (array) είναι μια συλλογή από παρόμοια στοιχεία δεδομένων. Αυτά τα στοιχεία έχουν τον ίδιο τύπο δεδομένων (data type). Τα στοιχεία αποθηκεύονται σε συνεχόμενες θέσεις στη μνήμη και προσπελαύνονται μέσω ενός δείκτη (index). Στη C η δήλωση ενός πίνακα γίνεται ως εξής:

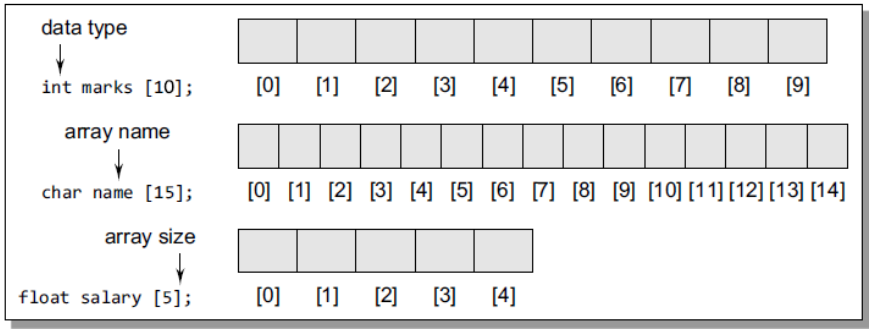
```
type name[size];
```

Παράδειγμα:

```
int marks[10];
```



Άλλα παραδείγματα δήλωσης πινάκων έχουν ως εξής:



Το ακόλουθο τμήμα κώδικα αναθέτει την τιμή -1 σε όλες τις θέσεις του πίνακα marks.

```
int i, marks[1 ];
for(i= ;i<1 ;i++)
        marks[i] = –1;
```

Συνεπώς, ο πίνακας θα έχει την ακόλουθη μορφή:

– 1	– 1	– 1	– 1	– 1	– 1	– 1	– 1	– 1	– 1
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

Αν θέλουμε να υπολογίσουμε τη θέση μνήμης στην οποία βρίσκεται το κάθε στοιχείο ενός πίνακα μπορούμε να εφαρμόσουμε τον εξής απλό μαθηματικό τύπο:

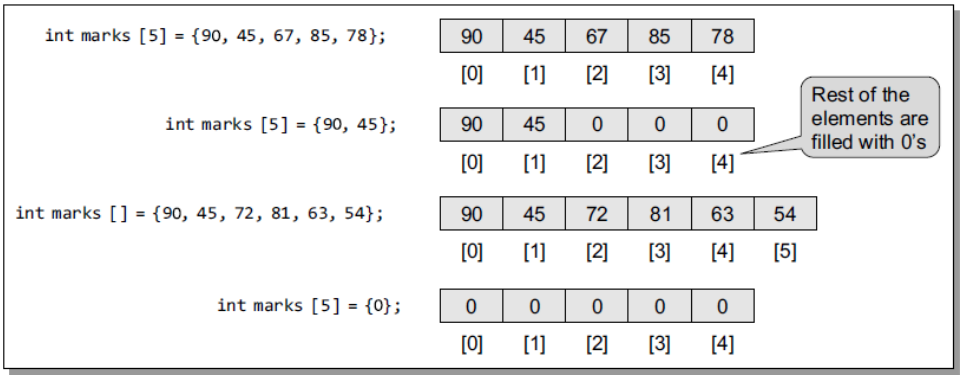
$$A[k] = BA(A) + w(k - lower\_bound)$$

Όπου το BA είναι η διεύθυνση βάσης και A είναι ο πίνακας. Το ακόλουθο παράδειγμα μας δείχνει τις θέσεις των στοιχείων του πίνακα int marks[] = {99,67,78,56,88,90,34,85} με διεύθυνση βάσης το 1000.

99	67	78	56	**88**	90	34	85
marks[0]	marks[1]	marks[2]	marks[3]	**marks[4]**	marks[5]	marks[6]	marks[7]
1000	1002	1004	1006	**1008**	1010	1012	1014

Στο συγκεκριμένο παράδειγμα έχουμε ότι w=2 που είναι το μέγεθος που καταλαμβάνει στη μνήμη ένας ακέραιος.

Ας δούμε όμως κάποια άλλα παραδείγματα δηλώσεων καθώς και τη μορφή που παίρνουν οι πίνακες στη μνήμη.



Τα ακόλουθα τμήματα κώδικα χρησιμοποιούνται για την αποθήκευση δεδομένων σε πίνακες:

<u>Παράδειγμα</u>

```
int i, marks[1 ];
for(i= ;i<1 ;i++)
        scanf("%d", &marks[i]);
```

Παράδειγμα

```
int i, arr1[1 ], arr2[1 ];
arr1[1 ] = { ,1,2,3,4,5,6,7,8,9};
for(i= ;i<1 ;i++)
        arr2[i] = arr1[i];
```

Παράδειγμα

```
int i,arr[1 ];
for(i=0;i<10;i++)
        arr[i] = i*2;
```

Σημείωση: Στη C μπορούμε να αυξήσουμε/τροποιήσουμε δυναμικά το μέγεθος ενός πίνακα με τον ακόλουθο τρόπο:

```
size_t myarray_size = 1000;
mystruct* myarray = malloc(myarray_size * sizeof(mystruct));

myarray_size += 1000;
mystruct* myrealloced_array = realloc(myarray, myarray_size * sizeof(mystruct));
if (myrealloced_array) {
   myarray = myrealloced_array;
} else {
   // deal with realloc failing because memory could not be allocated.
}
```

Διάσχιση Πινάκων

Αποτελεί βασική λειτουργία που επιτρέπει τη διαχείριση των στοιχείων του πίνακα.

Code 6

(εμφανίζει n αριθμούς με τη βοήθεια ενός πίνακα)

```
#include <stdio.h>
#include <conio.h>
int main()
{
        int i, n, arr[20];
        clrscr();
        printf("\n Enter the number of elements in the array : ");
        scanf("%d", &n);
        for(i=0;i<n;i++)
        {
                printf("\n arr[%d] = ", i);
                scanf("%d",&arr[i]);
        }
        printf("\n The array elements are ");
        for(i=0;i<n;i++)
                printf("\t %d", arr[i]);
        return 0;
}
```

Code 7

(εμφανίζει το μέσο όρο n αριθμών με τη βοήθεια ενός πίνακα)

```
#include <stdio.h>
#include <conio.h>
int main()
{
int i, n, arr[20], sum =0;
float mean = 0.0;
clrscr();
printf("\n Enter the number of elements in the array : ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
printf("\n arr[%d] = ", i);
scanf("%d",&arr[i]);
}
for(i=0;i<n;i++)
sum += arr[i];
mean = (float)sum/n;
printf("\n The sum of the array elements = %d", sum);
printf("\n The mean of the array elements = %.2f", mean);
return 0;
}
```

Code 8

(εμφανίζει τη θέση του μικρότερου στοιχείου ενός πίνακα)

```
#include <stdio.h>
#include <conio.h>
int main()
{
        int i, n, arr[20], small, pos;
        clrscr();
        printf("\n Enter the number of elements in the array : ");
        scanf("%d", &n);
        printf("\n Enter the elements : ");
        for(i=0;i<n;i++)
                scanf("%d",&arr[i]);
        small = arr[0]
        pos =0;
        for(i=1;i<n;i++)
        {
                if(arr[i]<small)
                {
                        small = arr[i];
                        pos = i;
                }
        }
        printf("\n The smallest element is : %d", small);
        printf("\n The position of the smallest element in the array is : %d", pos);
        return 0;
```

```
}
```

Code 9

(εμφανίζει τη θέση του δεύτερου μεγαλύτερου στοιχείου ενός πίνακα)

```
#include <stdio.h>
#include <conio.h>
int main()
{
	int i, n, arr[20], large, second_large;
	clrscr();
	printf("\n Enter the number of elements in the array : ");
	scanf("%d", &n);
printf("\n Enter the elements");
	for(i=0;i<n;i++)
		scanf("%d",&arr[i]);
	large = arr[0];
	for(i=1;i<n;i++)
	{
		if(arr[i]>large)
		large = arr[i];
	}
	second_large = arr[1];
	for(i=0;i<n;i++)
	{
		if(arr[i] != large)
		{
			if(arr[i]>second_large)
				second_large = arr[i];
		}
	}
	printf("\n The numbers you entered are : ");
	for(i=0;i<n;i++)
		printf("\t %d", arr[i]);
	printf("\n The largest of these numbers is : %d",large);
	printf("\n The second largest of these numbers is : %d",second_large);
	return 0;
}
```

Code 10

(εμφανίζει αν υπάρχει διπλή τιμή σε ένα πίνακα)

```
#include <stdio.h>
#include <conio.h>
int main()
{
	int array[10], i, n, j, flag =0;
	clrscr();
	printf("\n Enter the size of the array : ");
	scanf("%d", &n);
	for(i=0;i<n;i++)
	{
		printf("\n array[%d] = ", i);
```

```
            scanf("%d", &array[i]);
        }
        for(i=0;i<n;i++)
        {
            for(j=i+1;j<n;j++)
            {
                if(array[i] == array[j] && i!=j)
                {
                    flag =1;
                    printf("\n Duplicate numbers found at locations %d and %d", i,
                j);
                }
            }
        }
        if(flag==0)
        printf("\n No Duplicates Found");
        return 0;
}
```

Εισαγωγή Στοιχείων σε Πίνακα

Η εισαγωγή στοιχείου σε πίνακα μπορεί να γίνει σε οποιαδήποτε θέση του. Η πιο απλή περίπτωση είναι να γίνει η εισαγωγή στο τέλος του πίνακα. Αν ένας πίνακας έχει δηλωθεί ότι περιλαμβάνει 10 θέσεις και έχουμε εισάγει 8 στοιχεία τότε προφανώς υπάρχει χώρος για να προστεθεί ένα ακόμη στοιχείο. Αν όμως και οι 10 θέσεις έχουν καταληφθεί τότε ο χώρος δεν επαρκεί για την προσθήκη. Ο ακόλουθος κώδικας προσθέτει ένα στοιχείο σε μια συγκεκριμένη θέση στον πίνακα.

Code 11

(εισαγωγή ενός στοιχείου σε μια συγκεκριμένη θέση ενός πίνακα)

```
#include <stdio.h>
#include <conio.h>
int main()
{
        int i, n, num, pos, arr[10];
        clrscr();
        printf("\n Enter the number of elements in the array : ");
        scanf("%d", &n);
        for(i=0;i<n;i++)
        {
            printf("\n arr[%d] = ", i);
            scanf("%d", &arr[i]);
        }
        printf("\n Enter the number to be inserted : ");
        scanf("%d", &num);
        printf("\n Enter the position at which the number has to be added : ");
        scanf("%d", &pos);
        for(i=n–1;i>=pos;i—)
            arr[i+1] = arr[i];
        arr[pos] = num;
        n = n+1;
        printf("\n The array after insertion of %d is : ", num);
```

```
	for(i=0;i<n;i++)
		printf("\n arr[%d] = %d", i, arr[i]);
	getch();
	return 0;
}
```

Code 12

(εισαγωγή ενός στοιχείου σε ένα ταξινομημένο πίνακα)

```
#include <stdio.h>
#include <conio.h>
int main()
{
	int i, n, j, num, arr[10];
	clrscr();
	printf("\n Enter the number of elements in the array : ");
	scanf("%d", &n);
	for(i=0;i<n;i++)
	{
		printf("\n arr[%d] = ", i);
		scanf("%d", &arr[i]);
	}
	printf("\n Enter the number to be inserted : ");
	scanf("%d", &num);
	for(i=0;i<n;i++)
	{
		if(arr[i] > num)
		{
			for(j = n–1; j>=i; j—)
				arr[j+1] = arr[j];
			arr[i] = num;
			break;
		}
	}
	n = n+1;
	printf("\n The array after insertion of %d is : ", num);
	for(i=0;i<n;i++)
		printf("\n arr[%d] = %d", i, arr[i]);
	getch();
	return 0;
}
```

Διαγραφή Στοιχείων από Πίνακα

Code 13

(διαγραφή ενός στοιχείου από μια συγκεκριμένη θέση ενός πίνακα)

```
#include <stdio.h>
#include <conio.h>
int main()
{
	int i, n, pos, arr[10];
	clrscr();
```

```
	printf("\n Enter the number of elements in the array : ");
	scanf("%d", &n);
	for(i=0;i<n;i++)
	{
		printf("\n arr[%d] = ", i);
		scanf("%d", &arr[i]);
	}
	printf("\nEnter the position from which the number has to be deleted : ");
	scanf("%d", &pos);
	for(i=pos; i<n–1;i++)
		arr[i] = arr[i+1];
	n—;
	printf("\n The array after deletion is : ");
	for(i=0;i<n;i++)
		printf("\n arr[%d] = %d", i, arr[i]);
	getch();
	return 0;
}
```

Code 14

(διαγραφή ενός στοιχείου από μια συγκεκριμένη θέση ενός ταξινομημένου πίνακα)

```
#include <stdio.h>
#include <conio.h>
int main()
{
	int i, n, j, num, arr[10];
	clrscr();
	printf("\n Enter the number of elements in the array : ");
	scanf("%d", &n);
	for(i=0;i<n;i++)
	{
		printf("\n arr[%d] = ", i);
		scanf("%d", &arr[i]);
	}
	printf("\n Enter the number to be deleted : ");
	scanf("%d", &num);
	for(i=0;i<n;i++)
	{
		if(arr[i] == num)
		{
			for(j=i; j<n–1;j++)
				arr[j] = arr[j+1];
		}
	}
	n = n–1;
	printf("\n The array after deletion is : ");
	for(i=0;i<n;i++)
		printf("\n arr[%d] = %d", i, arr[i]);
	getch();
	return 0;
}
```