Εργαστήριο 6ο

Pointers – Δυναμικές μεταβλητές

Δημιουργία και χρήση πινάκων με χρήση Pointers

Άσκηση 1 (χρήση πινάκων)

Να δημιουργήσουμε ένα πρόγραμμα με το όνομα Lab5E1 το οποίο δημιουργεί και αρχικοποιεί έναν πίνακα ακεραίων με 4 στοιχεία. Στην συνέχεια τυπώνει τα στοιχεία αυτά και την θέση στην οποία βρίσκονται. Το πρόγραμμα είναι το παρακάτω

```
#include"pch.h"
#include<iostream>
using namespace std;

int main() {
	int a[4] = { 1,3,5,7 };

	a[3] = 11;

	for (int i = 0; i < 4; ++i) {
		cout << "element with index " << i << " is " << a[i] << endl;
	}

	for (auto &x : a)
		x = 88;

	return 0;
}
```

Να απαντήσετε στις παρακάτω ερωτήσεις

1. Ποιες πληροφορίες περιέχει η δήλωση ενός πίνακα;
2. Μπορεί ένας πίνακας να έχει διαφορετικού τύπου στοιχεία;
3. Τι πρέπει να ισχύει για το μέγεθος ενός πίνακα στην δήλωση του;
4. Ποια είναι η αναπαράσταση ενός πίνακα στην μνήμη του υπολογιστή;
5. Ποιες είναι οι indexed variables (ή στοιχεία) ενός πίνακα;
6. Ποια είναι η εμβέλεια (scope) της μεταβλητής i;
7. Θα μπορούσαμε να μεταβάλουμε την συνθήκη της for σε x<4;

Άσκηση 2 (Πίνακες και συναρτήσεις)

Να δημιουργήσετε ένα πρόγραμμα που να περιέχει δύο συναρτήσεις. Η μία θα παίρνει σαν παράμετρο έναν ακέραιο και θα επιστρέφει το μισό του (δεκαδικό). Η δεύτερη θα παίρνει 2 παραμέτρους η πρώτη θα είναι ένας πίνακας ακεραίων και η δεύτερη ένας ακέραιος αριθμός που θα δηλώνει το μέγεθος του πίνακα της πρώτης παραμέτρου. Η συνάρτηση θα επιστρέφει τον πίνακα όπου κάθε στοιχείο θα έχει την διπλάσια τιμή του. Το κυρίως πρόγραμμα θα ορίζει έναν πίνακα ακεραίων και θα καλεί τις συναρτήσεις. Ο κώδικας είναι ο παρακάτω:

```
#include<iostream>
using namespace std;

double make_half(int x);
void make_double(int x[], int size);
```

```
int main() {
	int a[]={1,3,5,7};


	cout <<"the half of element 3 is "<< make_half(a[2]) <<endl;

	make_double(a, 4);
	for (int i = 0; i < 4; ++i) {
		cout << "Element " << i << " is " << a[i] << endl;
	}
	return 0;
}

double make_half(int x) {
	return x / 2.0;
}

void make_double(int x[], int size) {
	for (int i = 0; i < size; ++i) {
		x[i] *= 2;
	}
	return;
}
```

Να απαντήσετε στις παρακάτω ερωτήσεις:

1. Μπορούμε να καλούμε συναρτήσεις δίνοντας σαν ορίσματα στοιχεία ενός πίνακα; Έχει διαφορά στην χρήση της το στοιχείο ενός πίνακα με μια κανονική μεταβλητή του ίδιου τύπου;
2. Μπορούμε να δίνουμε σαν όρισμα σε μια συνάρτηση έναν ολόκληρο πίνακα; Η κλήση είναι με τιμή (call by value) ή με αναφορά (call by reference);
3. Γιατί στην συνάρτηση make_double χρειάζεται να περαστεί και η δεύτερη παράμετρος;

Να πραγματοποιήσετε τις παρακάτω αλλαγές (η κάθε μία ξεχωριστά από τις άλλες):

1. Στην make_double κάντε const την παράμετρο τύπου πίνακα σε ακέραιους. Γίνεται compile το πρόγραμμα; Γιατί;
2. Τροποποιήστε την make_half έτσι ώστε να εφαρμόζεται σε όλον τον πίνακα

Άσκηση 3 (πολυδιάστατοι πίνακες)

Να δημιουργήσετε ένα πρόγραμμα με το όνομα Lab5E3 που δημιουργεί έναν πίνακα ακεραίων 3X3. Στην συνέχεια καλεί δύο συναρτήσεις. Μία που διαβάζει τον πίνακα από το πληκτρολόγιο και μία που στην συνέχεια τον τυπώνει. Ο κώδικας είναι ο παρακάτω

```
#include<iostream>
using namespace std;

void read_array(int a[][3], int rsize, int csize);
void print_array(const int (&a)[3][3]);

int main() {
	int m[3][3];

	read_array(m, 3, 3);
	print_array(m);

}

void read_array(int a[][3], int rsize, int csize)
```

```
{
	for (int i = 0; i < rsize; ++i) {
		for (int j = 0; j < csize; ++j)
		{
			cout << "Give the element in row " << i << " and column " << j <<
": ";
			cin >> a[i][j];
		}
	}
}

void print_array(const int (&a)[3][3]) {
	for (auto &row : a) {
		for (auto elem : row) {
			cout << elem <<" ";
		}
		cout << endl;
	}
}
```

Να απαντήσετε στις παρακάτω ερωτήσεις:

1. Πως δηλώνουμε έναν πίνακα πολλών διαστάσεων στην C++;
2. Πως αναφερόμαστε σε ένα στοιχείο ενός πίνακα πολλών διαστάσεων στην C++;
3. Πως γίνεται η αποθήκευση στην μνήμη ενός πίνακα πολλών διαστάσεων;
4. Τι είναι ο τύπος auto;
5. Πως δίνουμε σαν όρισμα έναν πίνακα πολλών διαστάσεων (όταν ΔΕΝ θέλουμε στην συνάρτηση να χρησιμοποιήσουμε range for);
6. Πως δίνουμε σαν όρισμα έναν πίνακα αν θέλουμε στην συνάρτηση να χρησιμοποιήσουμε την range for;
7. Πως χρησιμοποιούμε τις εμφωλευμένες range for;

Να κάνετε τις παρακάτω τροποποιήσεις

1. Να τυπώνει το αθροίσματα για κάθε γραμμή και κάθε στήλη
2. Να τυπώνει το άθροισμα των διαγώνιων

Άσκηση 4 (Δημιουργία/χρήση pointer variables)

Να δημιουργήσουμε το παρακάτω πρόγραμμα με το όνομα Lab7E1 το εξηγεί την δημιουργία και τις βασικές χρήσεις των pointer variables.

```
#include <iostream>
using namespace std;

int main()
{
	int x = 8, *p1, *p2;
	cout << "Variable x has value " << x << " and is in memory " << &x << endl;
	p1 = &x;
	cout << "Variable p1 has value " << p1 << " and points to " << *p1 << endl;
	p2 = p1;
	cout << "Variable p2 has value " << p1 << " and points to " << *p1 << endl;
	*p2 = 44;
	cout << "Pointer p1 points to " << *p1 << " and p2 points to " << *p2 << endl;
	cout << "Variable x has value " << x << endl;

	return 0;
}
```

Να απαντήσετε στις παρακάτω ερωτήσεις

8. Πως δηλώνουμε μια pointer variable και ποια είναι η τιμή που περιέχει;
9. Γιατί χρειάζεται η δήλωσε σε τι τύπο δείχνει ένας pointer;
10. Τι σημαίνει η έκφραση &x όταν η x είναι μια μεταβλητή;
11. Πως κάνω μια pointer variable να «δείχνει» σε μια υπάρχουσα μεταβλητή;
12. Πως μπορώ μέσα από μια pointer variable να μεταβάλω την τιμή στην μνήμη που αυτή «δείχνει»;
13. Έχοντας δηλώσει μια μεταβλητή pointer (π.χ. int *p;) τι σημαίνει το p και τι σημαίνει το *p;

Να πραγματοποιήσετε τις παρακάτω αλλαγές (κάθε μία ξεχωριστά από τις άλλες)

1. Δηλώστε μια μεταβλητή y τύπου double. Μετά εκτελέσετε την ανάθεση “p1 = &y”. Την δέχεται ο compiler; Αν όχι γιατί;
2. Προσπαθήστε να αναθέσετε σε μια μεταβλητή τύπου pointer μια ακέραια τιμή (π.χ. p=4002;) Την δέχεται ο compiler; Τι συμπέρασμα βγάζετε από αυτό; Είναι οι μεταβλητές τύπου pointer ακέραιοι αριθμοί;
3. Αλλάξτε την τιμή της μεταβλητής x (π.χ. x=22) θα αλλάξουν τιμές οι μεταβλητές pointer στο παράδειγμα μας;

Άσκηση 5 (δυναμικές μεταβλητές)

Να δημιουργήσετε το παρακάτω πρόγραμμα Lab7E2 που παρουσιάζει την δημιουργία και χρήση των δυναμικών μεταβλητών στην C++.

```
#include "stdafx.h"
#include <iostream>
using namespace std;

void create_var(int *&p, int value);
int *create_var(int value);

int main()
{
	int *p1,*p2;

	create_var(p1, 22);
	cout <<"Var p1 has value "<<p1<<" and points to "<< *p1 << endl;
	p2 = create_var(33);
	cout << "Var p2 has value " << p2 << " and points to " << *p2 << endl;
	delete p1,p2;
	return 0;
}

void create_var(int *&p, int value) {
	p = new int;

	*p = value;
}

int *create_var(int value) {
	int *p;
	p = new int;
	*p = value;
	return p;
```

```
}
```

Να απαντήσετε στις παρακάτω ερωτήσεις:

1. Ποιος είναι ο τελεστής που δημιουργεί μια δυναμική μεταβλητή;
2. Έχουν όνομα οι δυναμικές μεταβλητές; Πως μπορώ να έχω πρόσβαση σε μια δυναμική μεταβλητή;
3. Έχει διαφορά στην χρήση της μια δυναμική από μια κανονική μεταβλητή;
4. Τι πρέπει να κάνω όταν δεν χρειάζομαι πλέον μια δυναμική μεταβλητή;
5. Με ποιους δύο τρόπους μπορώ να επιστρέφω μία τιμή τύπου pointer από μία συνάρτηση.

Άσκηση 6 (δυναμικοί πίνακες)

Να δημιουργήσετε το παρακάτω πρόγραμμα Lab6E3 που παρουσιάζει την δημιουργία και τις βασικές χρήσεις των δυναμικών πινάκων.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
void print_arr(double *p);

int main()
{
	double a[] = { 1.1,2.2,3.3,4.4 };
	double *p;

	p = a;
	print_arr(p);

	p = new double[4];
	cout << "Give me 4 doubles: ";
	for (int i = 0; i < 4; i++) {
		cin >> *(p + i);
	}

	print_arr(p);

	delete [] p;
}

void print_arr(double * p)
{
	for (int i = 0; i < 4; i++) {
		cout << "Element " << i << " is " << *(p + i)<<endl;
	}
}
```

Απαντήστε στις παρακάτω ερωτήσεις

1. Αν ορίσουμε έναν πίνακα με την εντολή int a[10], τι συμβολίζει το a;
2. Πως δημιουργούμε έναν δυναμικό πίνακα;
3. Ποιοι είναι οι δύο τρόποι με τους οποίους μπορούμε να έχουμε πρόσβαση στα στοιχεία ενός δυναμικού πίνακα;

Να εκτελέσετε τις παρακάτω αλλαγές (μία κάθε φορά)

1. Να τροποποιήσετε την print_arr έτσι ώστε να τυπώνει τα στοιχεία του πίνακα ανάποδα (από το τελευταίο προς το πρώτο)

Άσκηση 7

Να γράψουμε ένα πρόγραμμα που τυπώνει το «εναλλακτικό άθροισμα» ενός μονοδιάστατου πίνακα μεγέθους 10. Το εναλλακτικό άθροισμα υπολογίζεται ως εξής

$$alt_{sum} = a_0 - a_1 + a_2 - a_3 + a_4 - \cdots$$

Άσκηση 8

Να δημιουργήσετε ένα πρόγραμμα που δημιουργεί μαγικά τετράγωνα (έχουν το ίδιο άθροισμα για κάθε γραμμή / στήλη / διαγώνιο). Ένας αλγόριθμος που δημιουργεί n X n μαγικά τετράγωνα (όπου n περιττός) είναι ο ακόλουθος

Τοποθετήστε το 1 στην μέση της τελευταίας γραμμής. Το επόμενο αριθμό θα τοποθετείτε στην επόμενη γραμμή και σε μία στήλη δεξιά

- Σε περίπτωση που φτάσετε στο άκρο του πίνακα ξεκινήστε από την αρχική γραμμή / στήλη.
- Σε περίπτωση που το κελί είναι ήδη συμπληρωμένο τότε συμπληρώστε το κελί που είναι ακριβώς από πάνω.

Ο πίνακας που θα δημιουργήσετε θα παίρνει τιμή από μία σταθερά (π.χ. 5)

Άσκηση 9

Να δημιουργήσετε μία συνάρτηση insert που θα έχει σαν παραμέτρους έναν pointer σε double και έναν double και έναν int. Η συνάρτηση θα επιστρέφει pointer σε double. Π.χ.

double *insert(double *order, double number, int n)

Η συνάρτηση θα καλείται δίνοντας της σαν όρισμα α) έναν δυναμικό πίνακα order με double αριθμούς ταξινομημένο με αύξουσα σειρά, β) έναν double αριθμό number και γ) έναν ακέραιο n που θα περιέχει το μέγεθος του δυναμικού πίνακα.

Στην συνέχεια η συνάρτηση θα δημιουργεί έναν καινούργιο δυναμικό πίνακα με n+1 στοιχεία όπου θα έχει τα στοιχεία του πίνακα που τις δόθηκε μαζί με τον αριθμό που τις δόθηκε στην κατάλληλη θέση (ο πίνακας θα πρέπει να είναι ταξινομημένος πάλι). Η συνάρτηση θα επιστρέφει τον δείκτη στην αρχή του καινούργιου πίνακα και να ελευθερώνει τον χώρο από τον παλιό πίνακα.

Δημιουργήστε και μία main που θα καλεί και θα δοκιμάζει την ορθή λειτουργία της συνάρτησης που υλοποιήσατε.

Άσκηση 10

Να δημιουργήσετε μία συνάρτηση extract που θα παίρνει σαν παράμετρο έναν pointer σε double order και έναν ακέραιο αριθμό pos καθώς και το μέγεθος του δυναμικού πίνακα στον οποίο «δείχνει» ο order και θα επιστρέφει έναν pointer σε double. Π.χ.

double *extract(double *order, int pos, int n)

Η συνάρτηση θα καλείται δίνοντας της σαν όρισμα έναν δυναμικό πίνακα με n double αριθμούς και έναν int αριθμό. Η συνάρτηση θα δημιουργεί έναν καινούργιο δυναμικό πίνακα με n-1 αριθμούς που θα είναι όλοι οι αριθμοί του αρχικού πίνακα εκτός από τον αριθμό στην θέση pos. Στην συνέχεια η συνάρτηση θα επιστρέφει τον pointer για τον καινούργιο πίνακα και θα ελευθερώνει τον χώρο από τον παλιό πίνακα

Δημιουργήστε μια main που θα καλεί τις insert και extract και θα ελέγχει την ορθή λειτουργία τους