Εμβέλεια μεταβλητών

Εμβέλεια μιας μεταβλητής είναι ο χώρος του προγράμματος μέσα στον οποίο είναι γνωστή η μεταβλητή

- Τοπικές
- Καθολικές
- Στατικές

Τοπικές μεταβλητές (local variables)

- Οι μεταβλητές μιας συνάρτησης καθώς και οι τυπικές παράμετροι της καλούνται τοπικές μεταβλητές. Έχουν ισχύ μόνο μέσα στη συνάρτηση που δηλώθηκαν και είναι άγνωστες στο υπόλοιπο πρόγραμμα
- Όταν η συνάρτηση επιστρέψει, τα περιεχόμενα των τοπικών μεταβλητών χάνονται

Καθολικές μεταβλητές (global variables)

- Έχουν εμβέλεια σε όλο το πρόγραμμα και μπορούν να προσπελαστούν και να χρησιμοποιηθούν από οποιαδήποτε συνάρτηση του προγράμματος
- Γενικά η μέθοδος μεταβίβασης τιμών σε μια συνάρτηση μέσω καθολικών μεταβλητών πρέπει να αποφεύγεται
- Ο χρόνος ζωής μιας καθολικής μεταβλητής είναι όλη η διάρκεια εκτέλεσης του προγράμματος

Στατικές τοπικές μεταβλητές (static)

- Όταν θέλουμε μια τοπική μεταβλητή να διατηρεί την τιμή της πρέπει να τη δηλώνουμε ως static.
- Όταν μια τοπική μεταβλητή δηλώνεται ως static τότε η πρόταση δήλωσης (ή/και η ανάθεση τιμής) της εκτελείται μόνο μια φορά
- Ο χρόνος ζωής τους είναι όλη η διάρκεια εκτέλεσης του προγράμματος

Δείκτες

- Η C++ διαθέτει τις ίδιες δυνατότητες στη χρήση δεικτών όπως και η C
- Η χρήση των δεικτών στη C++ είναι πιο περιορισμένη λόγω των νέων χαρακτηριστικών που διαθέτει η γλώσσα
- Μια μεταβλητή δείκτη περιέχει τη διεύθυνση μνήμης μιας άλλης μεταβλητής
- Μπορούμε να αναθέσουμε σε μια μεταβλητή δείκτη την τιμή μιας συμβολοσειράς και να αναφερόμαστε στη συμβολοσειρά μέσω του δείκτη

Οι τελεστές & και *

- & τελεστής απόδοσης μιας διεύθυνσης μεταβλητής σε δείκτη
- * τελεστής αναφοράς ώστε να προσπελαστεί μια μεταβλητή μέσω ενός δείκτη

Αριθμητική δεικτών

- Αύξηση κατά 1 για char και bool
- Αύξηση κατά 4 για int και float
- Αύξηση κατά 8 για double

Δείκτες τύπου void

void *ptr

Ένας δείκτης τύπου void δεν δείχνει σε δεδομένα συγκεκριμένου τύπου άρα δεν μπορούμε να εφαρμόσουμε αριθμητικούς τελεστές σε αυτόν

Δείκτης NULL

```
int *p;
p=NULL;
```

- Όταν θέλουμε να δηλώσουμε ότι ένας δείκτης δεν δείχνει πουθενά του αναθέτουμε την τιμή NULL
- Αμέσως μετά τη δήλωση ο δείκτης έχει απροσδιόριστη τιμή και όχι τιμή 0

Δείκτες και συμβολοσειρές

char *p;

p=“nikos”;

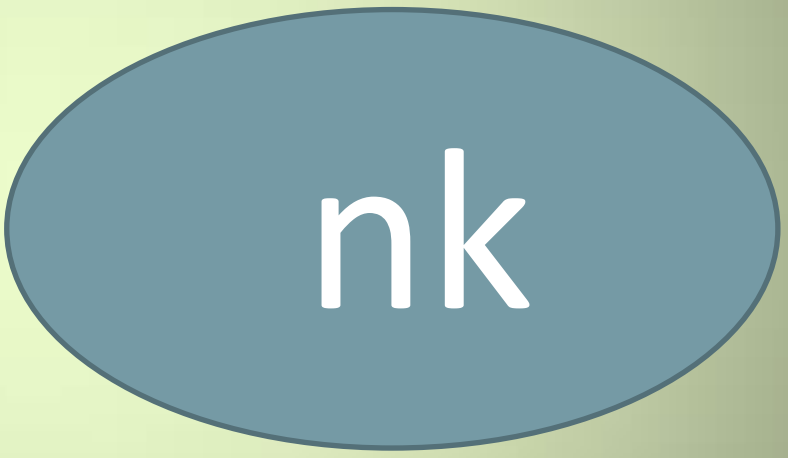
Στη μεταβλητή δείκτη p καταχωρίζεται η διεύθυνση της συμβολοσειράς nikos

cout << “nikos”;

Στο αντικείμενο cout μεταβιβάζεται η διεύθυνση της συμβολοσειράς και όχι οι ίδιοι οι χαρακτήρες της

Δείκτες και συμβολοσειρές

```
char *p;
 p="nikos";
 cout << *p;
 cout << *(p+2);
```



Δείκτες και συμβολοσειρές

- Σε όλες τις συναρτήσεις και τα αντικείμενα της C++ τα οποία διαχειρίζονται συμβολοσειρές μεταβιβάζονται οι διευθύνσεις των συμβολοσειρών. Οι συναρτήσεις και το τα αντικείμενα διασχίζουν τις συμβολοσειρές μέχρι να εντοπίσουν τον χαρακτήρα τερματισμού \0
- Η κλάση string που διαθέτει η standard library της C++ μας δίνει την δυνατότητα να χειριζόμαστε τις συμβολοσειρές με ποιο απλό τρόπο από αυτόν με δείκτες

Δείκτης σε δείκτη

```
int a, *p1, **p2;
p1= &a;
p2=&p1;
```

Προχωρημένα θέματα συναρτήσεων

- Κατά την κλήση μιας συνάρτησης κατ αξία οι τιμές των ορισμάτων αντιγράφονται στις αντίστοιχες τυπικές παραμέτρους της συνάρτησης και η συνάρτηση δεν μπορεί να τροποποιήσει τις τιμές που χρησιμοποιήθηκαν ως ορίσματα κατά την κλήση της

Κλήση συνάρτησης με δείκτη

- Ένας έμμεσος τρόπος για να δώσουμε σε μια συνάρτηση τη δυνατότητα να προσπελάζει τις μεταβλητές των ορισμάτων με τα οποία κλήθηκε
- Έτσι στη συνάρτηση μεταβιβάζονται ως ορίσματα διευθύνσεις μνήμης
- Το παρακάτω πρόγραμμα χρησιμοποιεί τη συνάρτηση swap για να αντιμεταθέσει τα περιεχόμενα δυο ακέραιων μεταβλητών

```
#include <iostream>
using namespace std;
void swap(int *a, int *b);
int main()
{
        int x,y;
        cout << "Δώσε δύο αριθμούς :";
        cin >> x >> y;
        cout << "x=" << x << " y=" << y <<endl;
        swap(&x,&y);
        cout << "x=" << x << " y=" << y <<endl;
        return 0;
}

void swap(int *a, int *b)
{
        int c;
        c=*a;
        *a=*b;
        *b=c;
}
```

Κλήση συνάρτησης με αναφορά

- Μια αναφορική μεταβλητή αναφέρεται σε άλλη μεταβλητή η οποία της ανατίθεται κατά τη δήλωση της

```
int a=23, &b=a;
b=44;
```

Η μεταβλητή b δηλώνεται ως αναφορική (ψευδώνυμο της a) και αναφέρονται και οι δυο στην ίδια θέση μνήμης. Ο αριθμός 44 θα καταχωριστεί στη μεταβλητή a. Στη C++ οι αναφορικές μεταβλητές χρησιμοποιούνται για τη μεταβίβαση παραμέτρων με αναφορά στην κλήση μιας συνάρτησης

Κατά τη κλήση μια συνάρτησης με αναφορά η συνάρτηση μπορεί να μεταβάλλει τις τιμές των ορισμάτων με τα οποία κλήθηκε

```
#include <iostream>
using namespace std;
void swap(int &x, int &y)
{
        int temp;
        temp=x;
        x=y;
        y=temp;
}
int main()
{
        int i=10,j=12;
        cout << i << ", " << j << endl;
        swap(i,j);
        cout << i << ", " << j << endl;
        return 0;
}
```

Οι παράμετροι x, y αναφέρονται στις μεταβλητές i και j αντίστοιχα
Στη κλήση συναρτήσεων με αναφορά τα ορίσματα πρέπει να είναι μεταβλητές και όχι σταθερές

Συναρτήσεις που επιστρέφουν ως τιμή έναν δείκτη

- Μια τέτοια συνάρτηση επιστρέφει μια διεύθυνση μνήμης

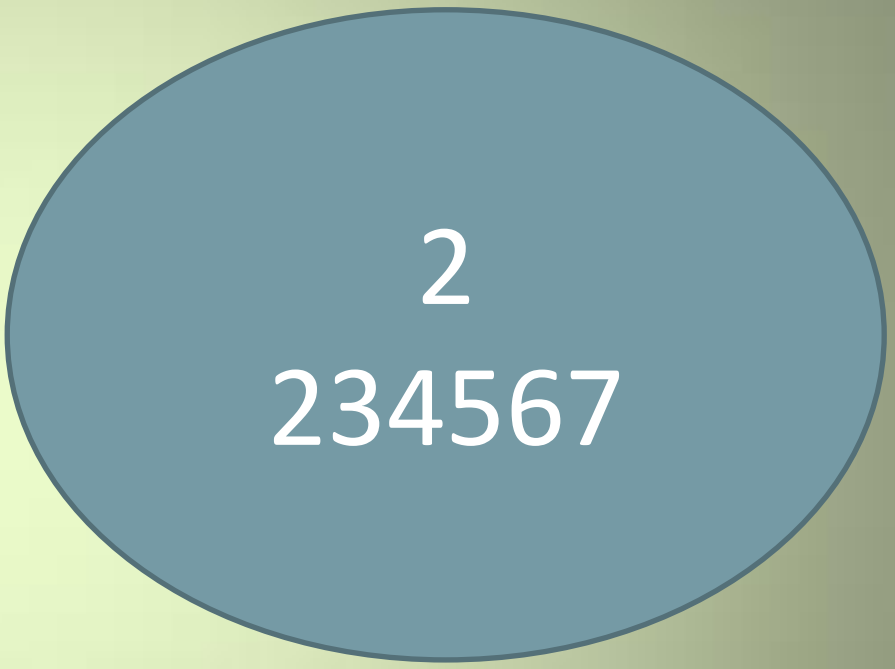
```
char *func1()
int *func2()
```

Το επόμενο πρόγραμμα καλεί τη συνάρτηση second() και της μεταβιβάζει τη συμβολοσειρά 1234567

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

char *second(char *str);
int main()
{
        char *p;
        p=second("1234567");
        cout << *p << endl;
        cout << p << endl;
        return 0;
}

char *second(char *str)
{
        return str+1;
}
```



Συναρτήσεις που επιστρέφουν ως τιμή έναν δείκτη

- Στο πρόγραμμα που ακολουθεί η συνάρτηση max() επιστρέφει ως τιμή έναν δείκτη προς μια από τις δυο μεταβλητές k ή l ανάλογα με το ποια έχει μεγαλύτερη τιμή

```
#include <iostream>
using namespace std;
int k=12,l=4;
int *max()
{
        if (k > l)
                return &k;
        else if (l > k)
                return &l;
        else
                return NULL;
}
int main()
{
        int *ptr;
   ptr=max();
        cout    << *ptr << "," << *max() << "," << k ;
        *ptr = 100;
        cout << "," << k << endl;
   return 0;
}
```

Συναρτήσεις που επιστρέφουν ως τιμή μια αναφορά

- Μια αναφορική συνάρτηση επιστρέφει μια αναφορά σε μια θέση μνήμης
- Στο παρακάτω πρόγραμμα η συνάρτηση max() επιστρέφει μια αναφορά προς μια από τις δυο μεταβλητές k ή l ανάλογα με το ποια έχει μεγαλύτερη τιμή

```
#include <iostream>
using namespace std;
int k=12,l=4;
int &max()
{
        if (k >= l)
                return k;
        else
                return l;
}
int main()
{
        cout    << max() << "," << k;
        max() = 100;
        cout << "," << k << endl;
        return 0;
}
```

Συναρτήσεις που επιστρέφουν ως τιμή μια αναφορά

- Μια αναφορική συνάρτηση πρέπει να επιστρέφει μια αναφορά σε μια θέση μνήμης η οποία να έχει εμβέλεια στον χώρο από τον οποίο καλείται η συνάρτηση

Παραστάσεις αριστερής και δεξιάς τιμής

- Οι παραστάσεις που μπορούν να βρίσκονται αριστερά του τελεστή = καλούνται παραστάσεις αριστερής τιμής (lvalue) και είναι όλες παραστάσεις που αποδίδουν αναφορά σε μια θέση μνήμης
- Οι παραστάσεις που τοποθετούνται δεξιά από τον τελεστή = καλούνται παραστάσεις δεξιάς τιμής (rvalue)

Δείκτες σε συναρτήσεις

- Στη C++ ένας δείκτης μπορεί να δείχνει σε μια συνάρτηση
- Αν αποθηκεύσουμε τη διεύθυνση μιας συνάρτησης σε μια μεταβλητή δείκτη, μπορούμε να καλούμε τη συνάρτηση, όχι πια με το όνομα της αλλά χρησιμοποιώντας τον συγκεκριμένο δείκτη

Δήλωση δείκτη σε συνάρτηση

int (*ptr) (**int** x, **int** y);

Η πρόταση αυτή δηλώνει μια μεταβλητή δείκτη με όνομα ptr ο οποίος δείχνει σε συναρτήσεις τύπου int με δυο παραμέτρους επίσης τύπου int

Ανάθεση τιμής σε μεταβλητή δείκτη σε συνάρτηση

- Η ανάθεση τιμής σε δείκτη σε συνάρτηση γίνεται απλά με την ανάθεση της διεύθυνσης της συνάρτησης στη μεταβλητή δείκτη ή με την ανάθεση του ονόματος της συνάρτησης. Η διεύθυνση αποδίδεται με τον τελεστή &

```
ptr = &func1;
ptr = func1;
```

Κλήση συνάρτησης με δείκτη σε συνάρτηση

- Η κλίση της συνάρτησης με τη χρήση του δείκτη στον οποίον έχει ανατεθεί η διεύθυνση της γίνεται απλά με τη χρήση του ονόματος της ακολουθούμενο από τα ορίσματα της κλήσης:

ptr (όρισμα1, όρισμα2…..) ή

(*ptr)(όρισμα1, όρισμα2….)

Στο παρακάτω πρόγραμμα ορίζονται δυο συναρτήσεις. Στο κυρίως πρόγραμμα δηλώνονται 2 μεταβλητές δείκτη που δείχνουν τις 2 συναρτήσεις και μέσω αυτών καλούνται

```
#include <iostream>
using namespace std;
int func1(int x, int y)
{
        return x*y;
}
int func2(int x, int y)
{
        return x+y;
}
int main()
{
        int (*ptr1)(int x, int y), (*ptr2)(int x, int y);
        ptr1=&func1; //ο δείκτης ptr1 δείχνει στη συνάρτηση func1()
        ptr2=&func2; //ο δείκτης ptr2 δείχνει στη συνάρτηση func2()
        cout    << "Κλήση της func1 : "
                        << ptr1(10,12) << endl;
        cout    << "Κλήση της func2 : "
                        << ptr2(10,12) << endl;
   ptr1=ptr2;
        cout    << "Κλήση της func2 : "
                        << ptr1(10,12) << endl;
        return 0;
}
```

Έλεγχος των περιεχομένων ενός δείκτη σε συνάρτηση

- Είναι δυνατόν να ελέγξουμε σε ποια συνάρτηση δείχνει ένας δείκτης χρησιμοποιώντας τον τελεστή αναφορά

```
if (ptr1==&func1)
```

Δείκτες σε συναρτήσεις ως παράμετροι

- Ένας δείκτης σε συνάρτηση μπορεί να μεταβιβαστεί ως παράμετρος σε μια άλλη συνάρτηση. Για παράδειγμα, η ακόλουθη συνάρτηση διαθέτει μια παράμετρο p η οποία είναι ένας δείκτης σε συναρτήσεις τύπου int με δυο παραμέτρους επίσης τύπου int

```
void call_ptr (int (*p) int x, int y))
{
    cout << “κλήση της συνάρτησης μέσω του δείκτη
                    p: “ << p(10, 100) <<endl
}
```

Δείκτες σε συναρτήσεις και αντικείμενα της C++

- Η χρήση των δεικτών σε συναρτήσεις βρίσκει ευρεία εφαρμογή στα αντικειμενοστρεφή χαρακτηριστικά της γλώσσας C++: τις κλάσεις και τα αντικέιμενα

Εμβόλιμες συναρτήσεις

- Σε περιπτώσεις μικρών συναρτήσεων, όπου ο κώδικας που απαιτείται για την κλήση και την επιστροφή της είναι περισσότερος από τον κώδικα της ίδιας της συνάρτησης, για την επίτευξη της βέλτιστης απόδοσης του προγράμματος η C++ επιτρέπει τη χρήση των εμβόλιμων συναρτήσεων
- Ο κώδικας μιας εμβόλιμης συνάρτησης παρεμβάλλεται στο κυρίως πρόγραμμα τόσες φορές όσες καλείται η συνάρτηση
- Όταν ο μεταγλωττιστής συναντήσει τη κλήση μια εμβόλιμης συνάρτησης τότε παρεμβάλει τον κώδικα της συνάρτησης στο σημείο της κλήσης
- Μια εμβόλιμη συνάρτηση δηλώνεται όπως και μια οποιαδήποτε άλλη συνάρτηση με τη διαφορά ότι πριν από τον τύπο της βάζουμε το πρόθεμα **inline**
- Για να αντιμετωπιστεί μια συνάρτηση ως εμβόλιμη θα πρέπει ο ορισμός της να έχει γίνει πριν από την πρώτη κλήση της

Πίνακες 1Δ

- Το παρακάτω πρόγραμμα γεμίζει έναν πίνακα με 10 τυχαίους αριθμούς και στη συνέχεια εμφανίζει μόνον αυτούς που είναι μεγαλύτεροι από το 100

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
	int i,a[10];
	for(i=0;i<10;i++)
		a[i]=rand();
	for(i=0;i<10;i++)
		if (a[i]>1000) cout << a[i] << endl;
	return 0;
}
```

- Το επόμενο πρόγραμμα γεμίζει τον πίνακα με 10 αριθμούς που δίνει ο χρήστης και στη συνέχεια υπολογίζει και εμφανίζει το άθροισμα των αριθμών

```
#include <iostream>
using namespace std;
int main()
{
	int i;
	float a[10],sum;
	for (i=0;i<10;i++)
		cin >> a[i];
	sum=0;
	for (i=0;i<10;i++)
		sum=sum+a[i];
	cout << "Το άθροισμα των αριθμών είναι:"  << sum << endl;
	return 0;
}
```

- Το επόμενο πρόγραμμα γεμίζει έναν πίνακα με 10 τυχαίους αριθμούς και τους εμφανίζει αντίστροφα. Στη συνέχεια υπολογίζει και εμφανίζει το πλήθος των ζυγών αριθμών του πίνακα

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
	int i,a[10],pl=0;
	for (i=0;i<10;i++)
		a[i]=rand();
	for (i=9;i>=0;i--)
		cout << a[i] << endl;
	for (i=0;i<10;i++)
		if (a[i]%2==0) pl++;
	cout << "Πλήθος ζυγών = " << pl << endl;
	return 0;
}
```

Χειρισμός πινάκων 1Δ

- Το επόμενο πρόγραμμα δημιουργεί έναν πίνακα 100 θέσεων και ον γεμίζει με τυχαίους αριθμούς. Στη συνέχεια υπολογίζει το άθροισμα των στοιχείων του, και ψάχνει αν ένας αριθμός που έδωσε ο χρήστης υπάρχει ή δεν υπάρχει μέσα στον πίνακα

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
	int i,a[100],sum=0,ar;
	bool found=false;
	for (i=0;i<100;i++)
		a[i]=rand();
	for (i=0;i<100;i++)
		cout << a[i] << endl;
	for (i=0;i<100;i++)
		sum=sum+a[i];
	cout << "Το άθροισμα των αριθμών είναι: " << sum << endl;
	cout << "Δώσε αριθμό για εύρεση:";
	cin >> ar;
	for (i=0;i<100;i++)
		if (a[i]==ar) found=true;
	if (found)
		cout << "Υπάρχει στον πίνακα" << endl;
	else
		cout << "Δεν υπάρχει στον πίνακα" << endl;
	return 0;
}
```

- Το επόμενο πρόγραμμα γεμίζει έναν πίνακα πραγματικών αριθμών 10 θέσεων με αριθμούς που δίνει ο χρήστης και στη συνέχει βρίσκει και εμφανίζει τον μικρότερο και τον μεγαλύτερο αριθμό του πίνακα καθώς και τις θέσεις του πίνακα στις οποίες τους εντόπισε

```
#include <iostream>
using namespace std;
int main()
{
		int i,th_min,th_max;
		float a[10],min,max;
		for (i=0;i<10;i++)		cin >> a[i];
		min=max=a[0];
		th_min=th_max=0;
		for (i=0;i<10;i++)
		{
			if (a[i]<min)
			{
				min=a[i];
				th_min=i;
			}
			if (a[i]>max)
			{
				max=a[i];
				th_max=i;
			}
		}
		cout << "Ο μικρότερος είναι: "<<min<<" στη θέση "<<th_min<<endl;
		cout << "Ο μεγαλύτερος είναι: "<<max<<" στη θέση "<<th_max<<endl;
		return 0;
}
```

Πίνακες 1Δ και συμβολοσειρές

- Το πρόγραμμα που ακολουθεί καταχωρίζει σε έναν πίνακα χαρακτήρων τυχαίους χαρακτήρες με κωδικούς ASCII από το 65 μέχρι το 74 (A-J). Έπειτα εμφανίζει τους χαρακτήρες αυτούς στην οθόνη

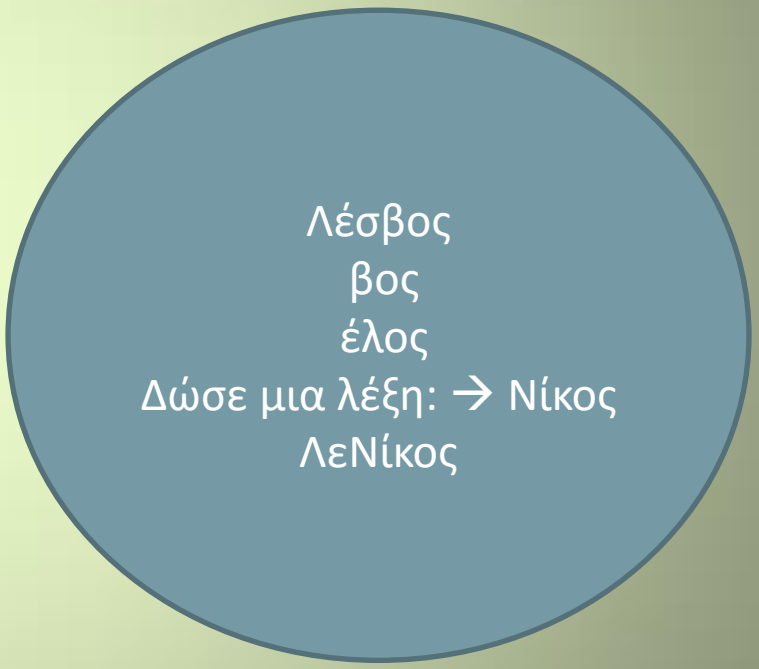
```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
	char ch[10];
	int i;
	for (i=0;i<10;i++)
		ch[i]=65+rand()%26; // Η παράσταση
65+rand()%26 παίρνει τυχαίες τιμές από 65 μέχρι 90.
	for (i=0;i<10;i++)
		cout << ch[i];
	return 0;
}
```

- Στη C++ τις συμβολοσειρές τις αντιμετωπίζουμε ως αντικείμενα (κλάση string)

Πίνακες και αντικείμενα cin, cout

```
#include <iostream>
using namespace std;
int main()
{
        char lex[20];
        cout << "Δώσε μια φράση->";
        cin >> lex;
        cout << "Έδωσες το:"<<lex<<endl;
        cout << "Δώσε και άλλη φράση->";
        cin >> lex;
        cout << "Έδωσες το:"<<lex<<endl;
        return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
        char lex[10]="Λέσβος";
        cout << lex <<endl;
        cout << lex+3 <<endl;
        cout << "Τέλος"+1 <<endl;
        cout << "Δώσε μια λέξη :";
        cin >> lex+2;
        cout << lex << endl;
        return 0;
}
```



Συναρτήσεις και συμβολοσειρές

- Στο επόμενο πρόγραμμα η συνάρτηση find_star() εντοπίζει τον πρώτο χαρακτήρα * του πίνακα lexi και επιστρέφει ως τιμή έναν δείκτη ο οποίος δείχνει στη θέση μνήμης που τον εντόπισε. Έτσι η cout<<prt εμφανίζει στην οθόνη το τμήμα της λέξης από το πρώτο * μέχρι το τέλος της. Αν δεν βρει τον χαρακτήρα επιστρέφει δείκτη Null

```
char *find_star(char *str)
{
        int i=0;
        bool found=false;
        while (str[i] != '\0')
        {
                if (str[i] == '*')
                {
                        found=true;
                        break;
                }
                i++;
        }
        if (found) return(str+i); else return(NULL);
}
```

```
#include <iostream>
using namespace std;
char *find_star(char *str);

int main()
{
        char lexi[40], *ptr;
        cout << " Δώσε μια λέξη :";
        cin >> lexi;
        ptr=find_star(lexi);
        if (ptr != NULL)
        {
                cout << «Η λέξη από το * και μετά είναι:";
                cout << ptr;
        }
        else
                cout << " Δεν υπάρχει * στη λέξη που έδωσες \n";
}
```

Συναρτήσεις βιβλιοθήκης που εφαρμόζονται σε συμβολοσειρές

#include <cstring>

strlen()

strcmp()

strcpy()

strcat()

strstr()

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
	char lexi[40];
	int len;
	len=strlen("Σκουλικομυρμηγκότρυπα");
	cout << "Η λέξη έχει" << len <<" χαρακτήρες" << endl;
	cout << "Δώσε μια λέξη:";
	cin >> lexi;
	len=strlen(lexi);
	cout << "η λέξη που έδωσες έχει" << len <<" χαρακτήρες" <<
endl;
	cout << strlen("Τέλος") << endl;
	return 0;
}
```

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
	char lexi1[20],lexi2[20];
	cout << "Δώσε μια λέξη:";
	cin >> lexi1;
	cout << "Δώσε μια άλλη λέξη:";
	cin >> lexi2;
	if (strstr(lexi1,lexi2)!=NULL)
		cout << "Η λέξη "<<lexi2<<" βρίσκεται μέσα στη
"<<lexi1<<endl;
	else
		cout << "Η λέξη "<<lexi2<<" δεν βρίσκεται μέσα στη
"<<lexi1<<endl;
	return 0;
}
```

Αντικείμενα string και πίνακες

- Ένα αντικείμενο string είναι ένας αποθηκευτικός χώρος ο οποίος μπορεί να μεγαλώνει και να μικραίνει ώστε να προσαρμόζεται ακριβώς στο μέγεθος της ακολουθίας χαρακτήρων που περιέχει. Ο τελευταίος χαρακτήρας του αντικειμένου είναι ο χαρακτήρας τερματισμού \0
- Η δυνατότητα των αντικειμενοστρεφών γλωσσών να δημιουργούμε αντικείμενα τα οποία μπορούμε να χρησιμοποιούμε χωρίς να μας ενδιαφέρουν οι τεχνικές λεπτομέρειες υλοποίησης τους ονομάζεται αφαιρετικότητα

getline()

getline(cin, string str)

Η πρώτη παράμετρος καθορίζει ως ρεύμα εισόδου το πληκτρολόγιο, ενώ η δεύτερη καθορίζει το αντικείμενο string στο οποίο θα αποθηκευτεί η συμβολοσειρά

- Στο πρόγραμμα που ακολουθεί δημιουργείται ένας πίνακας string 5 θέσεων και σε κάθε μια καταχωρίζεται μια συμβολοσειρά

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
        string lex[5];
   int i;
   lex[0] = "Αθήνα";
   lex[1] = "Τεχνολογία";
   lex[2] = "C++";
   lex[3] = "Πανεπιστήμιο";
   lex[4] = "Πάτρας";
   for (i=0;i<5;i++)
      cout << lex[i]<<endl;
   cout << "====================="<<endl;
   for (i=0;i<5;i++)
      cout << lex[i][0];
        return 0;
}
```

Αθήνα
Τεχνολογία
C++
Πάτρας
==========
ΑΤΣΠΠ

Άθροισμα στοιχείων πίνακα 2Δ

```
#include <iostream>
using namespace std;
int main()
{
	int i,j,a[10][5],ss=0;
	for (i=0;i<10;i++)
		for (j=0;j<5;j++)
			cin >> a[i][j];
	for (i=0;i<10;i++)
		for (j=0;j<5;j++)
			ss=ss+a[i][j];
	cout << "Αθροισμα=" << ss << endl;
	return 0;
}
```

Εύρεση max, min σε πίνακα 2Δ

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
		int i,j,a[10][5],max,min;
		for (i=0;i<10;i++)
				for (j=0;j<5;j++)
						a[i][j]=rand();
		max=min=a[0][0];
		for (i=0;i<10;i++)
				for (j=0;j<5;j++)
				{
						if (a[i][j]>max)
								max=a[i][j];
						else if (a[i][j]<min)
								min=a[i][j];
				}
		for (i=0;i<10;i++)
		{
				for (j=0;j<5;j++)
						cout << a[i][j] << " ";
				cout << endl;
		}
		cout << "μεγαλύτερη τιμή= " << max << endl;
		cout << "μικρότερη τιμή= " << min << endl;
		return 0;
}
```

Αρχικές τιμές πινάκων 2Δ

```
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;
int main()
{
		int i,j;
		string theseis[4][3]={"Νίκος","Λευτέρης","Βαγγέλης","Θοδωρής","Κώστας","Ουρανία",
"Μαρία","Πάνος","Μιχαήλ",
"Μάρκος","Σοφία","Ιωάννα"};
		for (j=0;j<3;j++)
  {
				for (i=0;i<4;i++)
      cout << theseis[i][j]<<" ";
     cout << endl;
  }
		return 0;
}
```

Μεταβίβαση πινάκων πολλών διαστάσεων σε συναρτήσεις

- Το πρόγραμμα που ακολουθεί καταχωρίζει τις θερμοκρασίες 100 πόλεων μιας χώρας, για κάθε ώρα της ημέρας για ένα μήνα και τελικά υπολογίζει και εμφανίζει τη μέση θερμοκρασία της χώρας

```
#include <iostream>
using namespace std;
int main()
{
	float therm[100][31][24],ath=0,mo;
	int i,j,k;
	for (i=0;i<100;i++)
		for (j=0;j<31;j++)
		{
			cout <<"Πόλη:"<<i+1<<" Ημέρα:"<<j+1<<endl;
			for (k=0;k<24;k++)
			{
				cout << "Θερμοκρασία ώρας "<<k+1<<":";
				cin >> therm[i][j][k];
			}
		}
	for (i=0;i<100;i++)
		for (j=0;j<31;j++)
			for (k=0;k<24;k++)
				ath=ath+therm[i][j][k];
	mo=ath/(100*31*24);
	cout << "Μέση θερμοκρασία πόλεων:"<<mo<<endl;
	return 0;
}
```