Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Αντικειμενοστρεφής Προγραμματισμός (C++)

Ενότητα 2: Τα βασικά της C++

Καθηγήτρια Κλειώ Σγουροπούλου

Αναγνωριστικά

- **Αναγνωριστικά (identifiers) – Ονοματοδοσία**
  - Τα αναγνωριστικά ξεκινούν με αλφαβητικό χαρακτήρα και στη συνέχεια ακολουθεί οποιοσδήποτε συνδυασμός αλφαβητικών, αριθμητικών χαρακτήρων και του '_' (underscore).
  - Κάθε επόμενο από το πρώτο συνθετικό ενός αναγνωριστικού ξεκινά με **κεφαλαίο** γράμμα.
    - π.χ. `changeColor(), secondVar`
  - Δεν επιτρέπεται η χρήση λέξεων κλειδιών ως αναγνωριστικών.
  - Οι σταθερές πρέπει να είναι με κεφαλαία
    - π.χ. `X_AXIS, Math.PI`
  - Η C++ είναι case-sensitive
    - π.χ. `Result ǂ result ǂ RESULT`

Τύποι Δεδομένων 1/2

- Τρία είδη: απλοί, σύνθετοι και κλάσεις
- Απλοί τύποι:
  - **int**: τυπικός ακέραιος
  - **float**: πραγματικός μονής ακρίβειας
  - **double**: πραγματικός διπλής ακρίβειας
  - **short**/**long**: short (μισός)/long (διπλός) ακέραιος
  - **char**: 8-bit ακέραια τιμή
- Οι απλοί τύποι μπορεί να είναι **signed**/**unsigned**
- Το μέγεθος των απλών τύπων εξαρτάται από το σύστημα **sizeof (type)**
- Οι σύνθετοι τύποι ομαδοποιούν λογικά συσχετισμένα δεδομένα ώστε αυτά να αναφέρονται με ένα ενιαίο τρόπο π.χ. πίνακες

Τύποι Δεδομένων 2/2

Όνομα τύπου	Bytes*	Εύρος*
char	1	**signed:** -128 to 127 **unsigned:** 0 to 255
short	2	**signed:** -32768 to 32767 **unsigned:** 0 to 65535
long	4	**signed:**-2147483648 to 2147483647 **unsigned:** 0 to 4294967295
int	*	Βλέπε **short**, **long**
float	4	3.4e + / - 38 (7 digits)
double	8	1.7e + / - 308 (15 digits)
long double	10	1.2e + / - 4932 (19 digits)
bool	**1**	**true or false**
wchar_t	2	Wide characters

Δήλωση και αρχικοποίηση μεταβλητών

- **Δήλωση**: δέσμευση κατάλληλης ποσότητας bytes στη μνήμη και απόδοση (σε κάποιες περιπτώσεις) default τιμής.
  - **int** a;

    float mynumber;
  - **unsigned short** NumberOfSons;

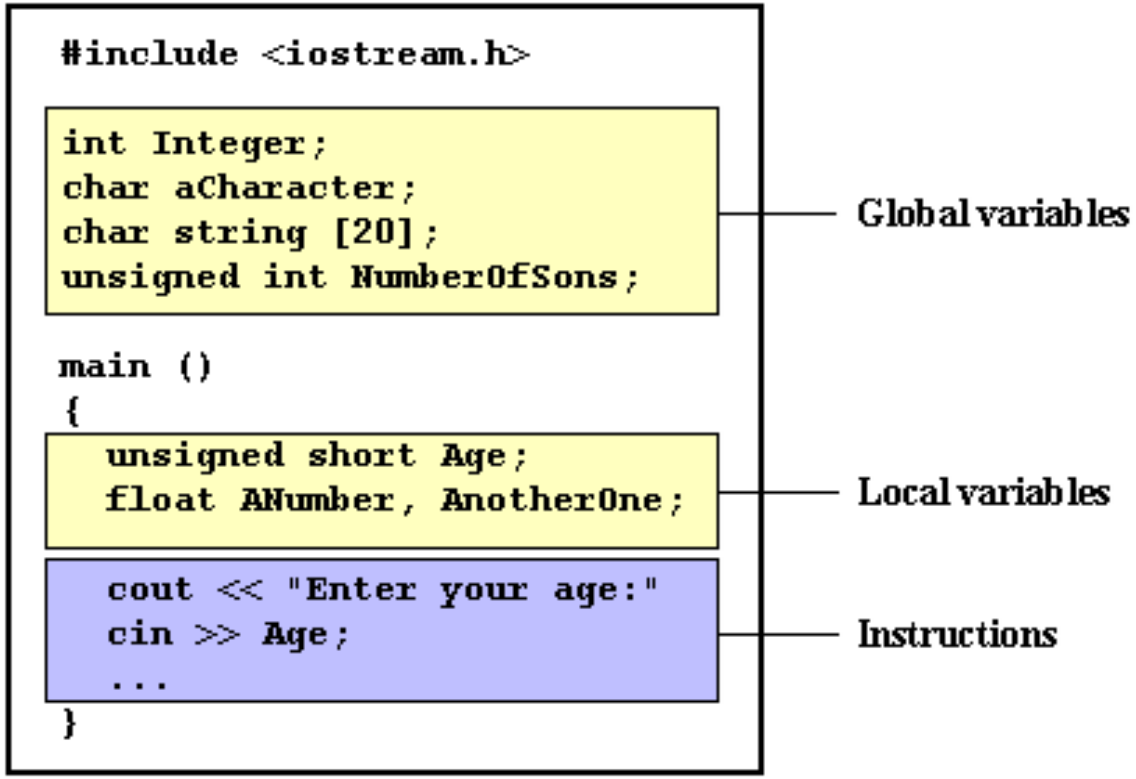
    signed int MyAccountBalance;

    (για τους τύπους char, short, long και int)
- **Αρχικοποίηση**: απόδοση επιθυμητής αρχικής τιμής
  - **int** a = 0;
  - **int** a (0);

Εμβέλεια 1/2

- Η **εμβέλεια** αναφέρεται στην **ορατότητα** μιας δομής
- Μια μεταβλητή γίνεται ορατή με τη δήλωσή της
- Οι μεταβλητές μπορούν να οριστούν ως **καθολικές** ή με εμβέλεια σε μια συγκεκριμένη συνάρτηση ή κλάση
- Τα όρια ενός τμήματος εμβέλειας ορίζονται από μια συνάρτηση, μια κλάση ή εντός ενός μπλοκ εντολών μεταξύ **{}** π.χ. **for** block
- Οι καθολικές μεταβλητές είναι ορατές σε όλο το πρόγραμμα
- Οι μεταβλητές που ορίζονται σε ένα συγκεκριμένο τμήμα εμβέλειας δεν είναι ορατές έξω από το τμήμα αυτό.

Εμβέλεια 2/2


#include <iostream.h>
int Integer;
char aCharacter;
char string [20];
unsigned int NumberOfSons;
main ()
{
unsigned short Age;
float ANumber, AnotherOne;
cout << "Enter your age:"
cin >> Age;
...
}
Global variables
Local variables
Instructions

Σταθερές

- Σταθερά είναι μια έκφραση με σταθερή τιμή.
- Διακρίνονται σε integer numbers, floating-point numbers, characters-strings
- Integer
  - `75 // decimal`
  - `0113 // octal`
  - `0x4b // hexadecimal`
- Floating-point
  - `3.14159 // 3.14159`
  - `6.02e23 // 6.02 x 1023`
  - `1.6e-19 // 1.6 x 10-19`
  - `3.0 // 3.0`
- Characters-Strings
  - `'z'`
  - `"Hello world"`
  - `'\n'`
  - `"Left \t Right"`
  - `"one\ntwo\nthree"`

Χαρακτήρες - Συμβολοσειρές

Escape codes

\n	newline
\r	carriage return
\t	tabulation
\v	vertical tabulation
\b	backspace
\f	page feed
\a	alert (beep)
\'	single quotes (')
\"	double quotes (")
\?	question (?)
\\	inverted slash (\)

ASCII Code Table

*	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Ορισμένες και δηλωμένες σταθερές

- Ορισμένες (defined) σταθερές
  - **#define** ***identifier value***
  - ```
    #define PI 3.14159265
    #define NEWLINE '\n'
    #define WIDTH 100
    ...
    circle = 2 * PI * r;
    cout << NEWLINE;
    ```
- Δηλωμένες (declared) σταθερές
  - **const** ***type identifier*** **=** ***value***;
  - ```
    const int width = 100;
    const char tab = '\t';
    const zip = 12440;*
    ```

*(όταν παραλείπεται ο τύπος υπονοείται int)

Τελεστές 1/3

- **Ανάθεσης** (=)
  - `a = 2 + (b = 5);  // C++`
  - `a = b = c = 5; // C++`
- **Αριθμητικοί** (+, -, *, /, %)
  - `a = 11 % 3  // υπόλοιπο διαίρεσης`
- **Σύνθετοι τελεστές ανάθεσης**
  (+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)
  - `value += increase; // value = value + increase;`
  - `price *= units + 1; // price = price * (units + 1);`
- **Τελεστές αύξησης και μείωσης** (++, --)
  - `B = 3; A = ++B; // A=?, B=?`
  - `B = 3; A = B++; // A=?, B=?`

Τελεστές 2/3

- **Σχεσιακοί τελεστές** (==, !=, >, <, >=, <=)
  - `(7 == 5);  // αποτέλεσμα?`
  - `(6 >= 6)  // αποτέλεσμα?`
  - `((b=2) == a) // αποτέλεσμα?, a=2, b=3`
- **Λογικοί τελεστές** (!, &&, ||)
  - `( (5 == 5) && (3 > 6) )// αποτέλεσμα?`
- **Τελεστής συνθήκης** (?)
  - *condition* **?** *result1* **:** *result2*
  - `7==5+2 ? 4 : 3  // αποτέλεσμα?`
- **Τελεστές bitwise** (&, |, ^, ~, <<, >>)
  - Μετατροπή μεταβλητών με βάση τη δυαδική τους αναπαράσταση

Τελεστές 3/3

- **Τελεστές μετατροπής τύπου**
  - (*type*)
  - ```
    int i;
    float f = 3.14;
    i = (int) f; ή i = int (f);
    ```
- **sizeof()**
  - ```
    a = sizeof (char);
    ```

Προτεραιότητα τελεστών 1/2

Προτ.	Τελεστής	Περιγραφή	Συσχετιστ.
1	::	scope	Left
2	() [ ] -> . sizeof		Left
3	++ --	increment/decrement	Right
	~	Complement to one (bitwise)	
	!	unary NOT	
	& *	Reference and Dereference (pointers)	
	(type)	Type casting	
	+ -	Unary less sign	
4	* / %	arithmetical operations	Left
5	+ -	arithmetical operations	Left
6	<< >>	bit shifting (bitwise)	Left

Προτεραιότητα τελεστών 2/2

Προτ.	Τελεστής	Περιγραφή	Συσχετιστ.
7	< <= > >=	Relational operators	Left
8	== !=	Relational operators	Left
9	& ^ |	Bitwise operators	Left
10	&& ||	Logic operators	Left
11	?:	Conditional	Right
12	= += -= *= /= %= >>= <<= &= ^= |=	Assignation	Right
13	,	Comma, Separator	Left

Εντολές ελέγχου 1/2

```
if (expression1)
    { statements
    }
else if
(expression2)
    { statements
    }
else
    { statements
    }
```

```
if (x > 0)
  cout << "x is
positive";
else if (x < 0)
  cout << "x is
negative";
else
  cout << "x is 0";
```

Εντολές ελέγχου 2/2

```
switch (expression) {
  case constant1:
    block of
    instructions 1
  break;
   case constant2:
    block of
    instructions 2
   break;
 ...
  default:
    default block of
    instructions
}
```

```
switch (x) {
  case 1:
    cout << "x is 1";
  break;
  case 2:
    cout << "x is 2";
  break;
  default:
    cout << "value of x
    unknown";
}
```

Εντολές επανάληψης 1/5

- **while (*expression*)**

```
{
        statements
}
```

```
#include <iostream>
int main ()
{
      int n;
      cout << "Enter the
      starting number > ";
      cin >> n;
      while (n>0) {
        cout << n << ", ";
        --n;
      }
      cout << "FIRE!";
      return 0;
}
```

Εντολές επανάληψης 2/5

```
do
{
  statements
}
while (condition)
```

```
#include <iostream>
int main ()
{
  unsigned long n;
  do {
    cout << "Enter number
    (0 to end): ";
    cin >> n;
    cout << "You entered:
"
    << n << "\n";
  } while (n != 0);
  return 0;
}
```

Εντολές επανάληψης 3/5

```
for (initialization; condition; increase)
{
  statements;
}
```

```
#include <iostream>
int main ()
{
  for (int n=10; n>0; n--) {
    cout << n << ", ";
}
cout << "FIRE!";
return 0;}
```

Εντολές επανάληψης 4/5

- `for (;;)`
  - τι κάνει;
  - ποιό είναι το ανάλογό του με χρήση `while;`
- `break`
  - **for (n=10; n>0; n--)**
  - **{ cout << n << ", ";**
  - **if (n==3)**
  - **{ cout << "countdown aborted!";**
  - **break;**
  - **}**
  - **}**

Εντολές επανάληψης 5/5

- **continue**
  - `for (int n=10; n>0; n--) {`
  - `    if (n==5) continue;`
  - `    cout << n << ", ";`
  - `  }`
- **exit()**
  - Συνάρτηση ορισμένη στην `stdlib.h`
  - Τερματίζει το τρέχων πρόγραμμα και επιστρέφει έναν κωδικό εξόδου (κατά σύμβαση 0 σημαίνει ότι το πρόγραμμα τερματίζει και οποιαδήποτε άλλη τιμή ότι έχει συμβεί σφάλμα)
  - **`void exit (int`** *`exit code`***`);`**

Συναρτήσεις

- *type name* (*argument1*, *argument2*, ...)

```
{
  statements;
}
```

```
#include <iostream.h>
int addition (int a, int b)
{
  int r;
  r = a + b;
  return (r);
}
```

Μοντέλο λειτουργίας συνάρτησης

```
int main ()
{
  int z;
  z = addition (5,3);
  cout << "The result
  is " << z;
  return 0;
}
```

Εκτέλεση κώδικα συνάρτησης

Πέρασμα παραμέτρων με τιμή & με αναφορά 1/2

- Πέρασμα **με τιμή**
  - Κατά την κλήση μιας συνάρτησης με παραμέτρους, αυτό που περνάμε στη συνάρτηση είναι οι τιμές των μεταβλητών και όχι αυτές καθαυτές τις μεταβλητές (η τιμή των οποίες παραμένει αμετάβλητη στην καλούσα συνάρτηση.
  - π.χ.

```
int addition (int a, int b)
int z
                          5     3
z = addition (  x   ,   y  );
```

Πέρασμα παραμέτρων με τιμή & με αναφορά 2/2

- Πέρασμα **με αναφορά**
  - Γίνεται στην περίπτωση που είναι επιθυμητή η μεταβολή της τιμής μιας εξωτερικής μεταβλητής από το εσωτερικό μιας συνάρτησης (τελεστής **&**).

```
#include <iostream.h>
void duplicate (int& a, int& b, int& c)
{
  a *= 2;
  b *= 2;
  c *= 2;
}
```

```
int main ()
{
  int x=1, y=3, z=7;
  duplicate (x, y, z);
  cout << "x=" << x << ",
  y=" << y << ", z=" << z;
  return 0;
}
```

```
void duplicate (int& a,int& b,int& c)
                    x     y     z
    duplicate (   x   ,   y   ,   z   );
```

Επιστροφή πολλαπλών τιμών

- Η τεχνική του περάσματος με αναφορά επιτρέπει την επιστροφή πολλαπλών τιμών από μια συνάρτηση

```
void prevnext (int x, int& prev, int& next)
{
  prev = x-1;
  next = x+1;
}
```

```
int main ()
{
  int x=100, y, z;
  prevnext (x, y, z);
  cout << "Previous=" << y
  << ", Next=" << z;
  return 0;
}
```

Χρήση προκαθορισμένων τιμών

- Όταν γράφουμε μια συνάρτηση μπορούμε να δηλώσουμε προκαθορισμένες (default) τιμές για κάθε μια από τις παραμέτρους της. Αυτές οι τιμές θα χρησιμοποιηθούν σε περίπτωση που κατά την κλήση, δεν περαστούν τιμές

```
#include <iostream.h>
int divide (int a, int b=2)
{
  int r;
  r=a/b;
  return (r);
}
```

```
int main ()
{
  cout << divide (12);
  cout << "\n";
  cout << divide (20,4);
  return 0;
}
```

Υπερφόρτωση συναρτήσεων

- Υπερφόρτωση (**overloading**) συνάρτησης έχουμε όταν ορίζουμε δύο ή περισσότερες συναρτήσεις με το ίδιο όνομα, αλλά διαφορετική λίστα παραμέτρων (είτε ως προς τον αριθμό ή ως προς τους τύπους)

```
#include <iostream.h>
int divide (int a, int b)
{
  return (a/b);
}
float divide (float a, float b)
{
  return (a/b);
}
```

```
int main ()
{
  int x = 5,y = 2;
  float n = 5.0,m = 2.0;
  cout << divide (x,y);
  cout << "\n";
  cout << divide (n,m);
  cout << "\n";
  return 0;
}
```

Συναρτήσεις inline

- Μια συνάρτηση ορίζεται ως inline με τη χρήση της ομώνυμης λέξης κλειδί πριν τον τύπο της συνάρτησης. Κατά τη μεταγλώττιση, η δήλωση αυτή απευθύνεται στον προεπεξεργαστή και έχει ως αποτέλεσμα την ενσωμάτωση του κώδικα στο σημείο κλήσης της συνάρτησης

```
inline type name (argument1, argument2, ...)
    {
      statements;
    }
```

Πρωτοτυποποίηση συναρτήσεων

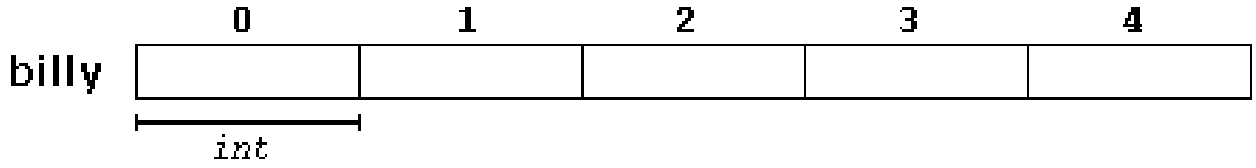
- Δεν είναι απαραίτητο ο κώδικας μιας συνάρτησης να γράφεται πριν τη χρήση της στη main. Μέσω της πρωτοτυποποίησης συναρτήσεων μπορούμε να δηλώνουμε απλά το πρωτότυπό της με τη μορφή:

```
type name ( argument_type1, argument_type2, ...);
```

- Η μορφή αυτή είναι ίδια με την επικεφαλίδα της συνάρτησης, εκτός του ότι:
  - Δεν αναφέρει τις εντολές που υλοποιούν τη λειτουργία της συνάρτησης
  - Τελειώνει με το σύμβολο τέλους των εντολών (;).
  - Στη λίστα των παραμέτρων αρκεί να δηλωθεί ο τύπος κάθε παραμέτρου (και όχι όνομα)

Πίνακες 1/2

- **Δήλωση πινάκων**
  - `type name [elements];`
  - `int billy [5];`



- **Αρχικοποίηση πινάκων**
  - Σε περίπτωση δήλωσης ενός πίνακα τοπικής εμβέλειας, ο πίνακας δεν αρχικοποιείται αυτόματα και οι τιμές του είναι απροσδιόριστες
  - Σε περίπτωση δήλωσης καθολικού πίνακα γίνεται αυτόματη αρχικοποίηση με όλες τις τιμές 0
  - `int billy [5] = { 16, 2, 77, 40, 12071 };`
  - `int billy [] = { 16, 2, 77, 40, 12071 };`

Πίνακες 2/2

- **Δήλωση διδιάστατων πινάκων**
  - *type name* [*rows*] [*columns*];
  - `int jimmy [3][5];` ⇔ `int jimmy [15];`

```
#include <iostream.h>

#define WIDTH 5
#define HEIGHT 3

int jimmy [HEIGHT][WIDTH];
int n, m;

int main ()
{
  for (n=0; n<HEIGHT; n++)
    for (m=0; m<WIDTH; m++)
    {
      jimmy[n][m]=(n+1)*(m+1);
    }
  return 0;
}
```

```
#include <iostream.h>

#define WIDTH 5
#define HEIGHT 3

int jimmy [HEIGHT * WIDTH];
int n, m;
int main ()
{
  for (n=0; n<HEIGHT; n++)
    for (m=0; m<WIDTH; m++)
    {
      jimmy [n * WIDTH +
      m]=(n+1)*(m+1);
    }
  return 0;
}
```

Πίνακες ως παράμετροι

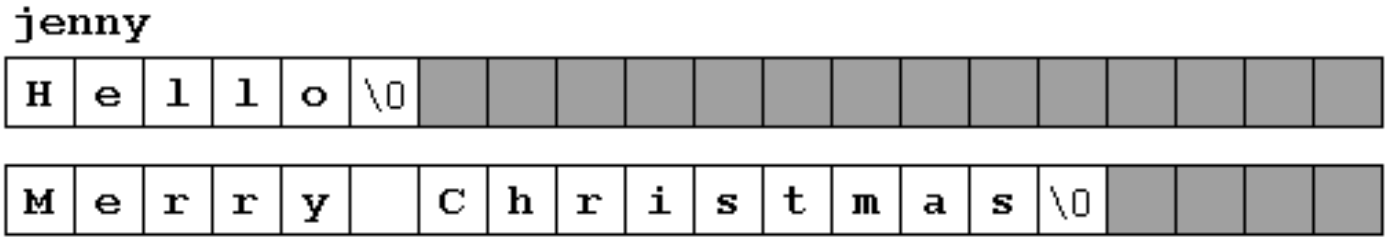
```
#include <iostream.h>

void printarray (int arg[], int length) {
  for (int n=0; n<length; n++)
    cout << arg[n] << " ";
  cout << "\n";
}

int main ()
{
  int firstarray[] = {5, 10, 15};
  int secondarray[] = {2, 4, 6, 8, 10};
  printarray (firstarray,3);
  printarray (secondarray,5);
  return 0;
}
```

Συμβολοσειρές 1/5

- **Δήλωση συμβολοσειρών**
  - Όπως στη C, έτσι και στη C++ δεν υπάρχει βασικός τύπος μεταβλητής για την αποθήκευση συμβολοσειρών. Αυτό επιτυγχάνεται με χρήση πινάκων χαρακτήρων
  - `char jenny [20];`



  - Οι συμβολοσειρές τερματίζουν με τον ειδικό χαρακτήρα '\0'

Συμβολοσειρές 2/5

- **Αρχικοποίηση συμβολοσειρών**
  - `char mystring [] = { 'H', 'e', 'l', 'l', 'o', '\0' };`
  - `char mystring [] = "Hello";`

  - **Προσοχή!** Ο παραπάνω τρόπος ανάθεσης είναι έγκυρος **μόνο** κατά την αρχικοποίηση
    - **`mystring = "Hello"; // λάθος`**
    - **`mystring[] = "Hello"; // λάθος`**
    - **`mystring = { 'H', 'e', 'l', 'l', 'o', '\0' }; //λάθος`**

Συμβολοσειρές 3/5

- **Ανάθεση τιμών σε μεταβλητές συμβολοσειράς**

```
mystring[0] = 'H';
mystring[1] = 'e';
mystring[2] = 'l';
mystring[3] = 'l';
mystring[4] = 'o';
mystring[5] = '\0';
```

ή

```
#include <string.h>

strcpy (string1, string2);
strcpy (mystring, "Hello");
```

```
#include <iostream.h>

void setstring (char szOut [], char
szIn [])
{
  int n=0;
  do {
    szOut[n] = szIn[n];
  } while (szIn[n++] != '\0');
}

int main ()
{
  char szMyName [20];
  setstring (szMyName,"C. Sgouro");
  cout << szMyName;
  return 0;
}
```

Συμβολοσειρές 4/5

- **Ανάθεση τιμών σε μεταβλητές συμβολοσειράς**
  - Εναλλακτικά της **strcpy** μπορεί να χρησιμοποιηθεί το ρεύμα εισόδου **cin** με δύο τρόπους:
    - χρήση της μεθόδου **getline**

```
#include <iostream.h>
int main ()
{
  char mybuffer [100];
  cout << "What's your name? ";
  cin.getline (mybuffer,100);
  cout << "Hello " << mybuffer << ".\n";
  cout << "Which is your favourite team? ";
  cin.getline (mybuffer,100);
  cout << "I like " << mybuffer << " too.\n";
  return 0;
}
```

```
cin.getline(char buffer[],int leng,char delim ='\n');
```

Συμβολοσειρές 5/5

- **Ανάθεση τιμών σε μεταβλητές συμβολοσειράς**
  - χρήση του τελεστή εξαγωγής >>

```
cin >> mybuffer;
```

    - Περιορισμοί σε σχέση με τη **getline**
    - Μπορεί να δεχτεί μόνο λέξεις (όχι προτάσεις), δεδομένου ότι χρησιμοποιεί σαν τερματιστή οποιοδήποτε μη απεικονίσιμο χαρακτήρα (συμπεριλαμβανομένων spaces, tabs, new lines και carriage returns)
    - Δεν επιτρέπεται ο καθορισμός μεγέθους για τον buffer. Αυτό καθιστά το πρόγραμμα ασταθές στην περίπτωση που η είσοδος του χρήστη είναι μεγαλύτερη από τον πίνακα υποδοχής.

Μετατροπή Συμβολοσειρών 1/2

- **Μετατροπή σε άλλους τύπους**
  - Η βιβλιοθήκη **`cstdlib (stdlib.h)`** παρέχει τρεις χρήσιμες συναρτήσεις για το σκοπό αυτό:
    - **`atoi`** μετατρέπει συμβολοσειρές σε **`int`**
    - **`atol`** μετατρέπει συμβολοσειρές σε **`long`**
    - **`atof`** μετατρέπει συμβολοσειρές σε **`float`**

Μετατροπή Συμβολοσειρών 2/2

- Μετατροπή σε άλλους τύπους

```
#include <iostream.h>
#include <stdlib.h>

int main ()
{
  char mybuffer [100]; float price; int quantity;
  cout << "Enter price: ";
  cin.getline (mybuffer,100);
  price = atof (mybuffer);
  cout << "Enter quantity: ";
  cin.getline (mybuffer,100);
  quantity = atoi (mybuffer);
  cout << "Total price: " << price*quantity;
  return 0;
}
```

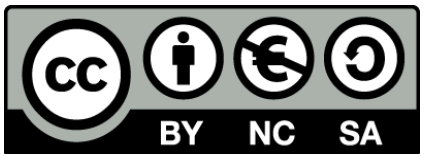
```
Enter price: 2.75
Enter quantity: 21
Total price: 57.75
```

Συναρτήσεις διαχείρισης συμβολοσειρών

- Η βιβλιοθήκη **cstring (string.h)** παρέχει συναρτήσεις διαχείρισης συμβολοσειρών:
  - `char* strcat (char* dest, const char* src);`
    Παραθέτει τη συμβολοσειρά `src` στο τέλος της `dest`. Επιστρέφει την `dest`.
  - `int strcmp (const char* string1, const char* string2);`
    Συγκρίνει τις συμβολοσειρές `string1` και `string2`. Επιστρέφει 0 όταν είναι ίσες.
  - `char* strcpy (char* dest, const char* src);`
    Αντιγράφει το περιεχόμενο της `src` στη `dest`. Επιστρέφει την `dest`.
  - `size_t strlen (const char* string);`
    Επιστρέφει το μήκος της συμβολοσειράς.

    ΣΗΜ: Η δήλωση **`char*`** είναι ισοδύναμη με **`char[]`**

ΤΕΛΟΣ ΕΝΟΤΗΤΑΣ



Σημειώματα

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Επεξήγηση όρων χρήσης έργων τρίτων

©	Δεν επιτρέπεται η επαναχρησιμοποίηση του έργου, παρά μόνο εάν ζητηθεί εκ νέου άδεια από το δημιουργό.
διαθέσιμο με άδεια CC-BY	Επιτρέπεται η επαναχρησιμοποίηση του έργου και η δημιουργία παραγώγων αυτού με απλή αναφορά του δημιουργού.
διαθέσιμο με άδεια CC-BY-SA	Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού, και διάθεση του έργου ή του παράγωγου αυτού με την ίδια άδεια.
διαθέσιμο με άδεια CC-BY-ND	Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού. Δεν επιτρέπεται η δημιουργία παραγώγων του έργου.
διαθέσιμο με άδεια CC-BY-NC	Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού. Δεν επιτρέπεται η εμπορική χρήση του έργου.
διαθέσιμο με άδεια CC-BY-NC-SA	Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού και διάθεση του έργου ή του παράγωγου αυτού με την ίδια άδεια. Δεν επιτρέπεται η εμπορική χρήση του έργου.
διαθέσιμο με άδεια CC-BY-NC-ND	Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού. Δεν επιτρέπεται η εμπορική χρήση του έργου και η δημιουργία παραγώγων του.
διαθέσιμο με άδεια CC0 Public Domain	Επιτρέπεται η επαναχρησιμοποίηση του έργου, η δημιουργία παραγώγων αυτού και η εμπορική του χρήση, χωρίς αναφορά του δημιουργού.
διαθέσιμο ως κοινό κτήμα	Επιτρέπεται η επαναχρησιμοποίηση του έργου, η δημιουργία παραγώγων αυτού και η εμπορική του χρήση, χωρίς αναφορά του δημιουργού.
χωρίς σήμανση	Συνήθως δεν επιτρέπεται η επαναχρησιμοποίηση του έργου.

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.