ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Τεχνολογίες Υλοποίησης Αλγορίθμων

Χρήστος Ζαρολιάγκης

Καθηγητής
Τμήμα Μηχ/κων Η/Υ & Πληροφορικής
Πανεπιστήμιο Πατρών
email: zaro@ceid.upatras.gr

Εισαγωγή στην C++ - 1

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο **«Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών»** έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Περιεχόμενα

Απλό Παράδειγμα

```
// we want to do I/O
#include <iostream>

int main()
{
    int v1, v2;
    // cout: standard output stream object
    // << : stream insertion operator
    std::cout << "Please give me two integers: ";
    // cin: standard input stream object
    // >> : stream extraction operator
    std::cin >> v1 >> v2;
    std::cout << "The numbers are " << v1;
    std::cout << " and " << v2 << std::endl;
    return 0;
}
```

Παράδειγμα I/O με Μεταβλητές

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string s;
    cout << "Type in your name: " ;
    cin >> s ;
    cout << "The name you typed is: " << s << endl;
    return 0 ;
}
```

Παράδειγμα I/O με Αρχεία

```
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

int main()
{
    ifstream infile( "input_file" );
    if ( ! infile ) {
        cerr << "error: unable to open input file!"
                << endl;
        return -1;
    }
    ofstream outfile( "out_file" );
    if ( ! outfile ) {
        cerr << "error: unable to open output file!"
                << endl;
        return -2;
    }
    string word;
    while ( infile >> word )
        outfile << word << ' ';
    return 0;
}
```

(ένα αρχείο)

```
g++ [-o prog] prog.c
```

▷ **Μεταγλώττιση (Compilation) και Σύνδεση (Linking)**

(πολλά αρχεία)

- *Μεταγλώττιση*

```
g++ -c prog1.c ⟹ prog1.o
g++ -c prog2.c ⟹ prog2.o
...
g++ -c progn.c ⟹ progn.o
g++ -c main.c ⟹ main.o
```

- *Σύνδεση*

```
g++ [-o main] main.o prog1.o prog2.o ... progn.o
```

Βασική Δομή ενός Προγράμματος C++

```
#include<iostream>
// other preprocessor directives

function-definition 1;
function-definition 2;
. . .

int main()
{
    declaration 1;
    declaration 2;
    . . .

    execution-statement 1;
    execution-statement 2;
    . . .

    return 0;
}
```

Σύνταξη ενός Προγράμματος C++

΄Ονομα (Identifier)

- Ακολουθία από ≥ 1 χαρακτήρες
- Χαρακτήρες $\in \{\text{letters, digits, \_}\}$
- Πεζά γράμματα $\neq$ κεφαλαία γράμματα
- **Πρέπει** να αρχίζει με γράμμα ή _

Σχόλια (Comments)

- Σχόλιο σε μια γραμμή:

```
// ...
```

- Σχόλιο πολλών γραμμών:

```
/* ... */
```

  - ▷ Προσοχή: `/*`, `*/` δεν φωλιάζουν!
    Π.χ., το ακόλουθο δεν δουλεύει:

```
/* outer-comment
     OUT
/* inner-comment
     IN
*/
     OUT (cntd)
*/
```

Τύποι Δεδομένων (Data Types)

Ακέραιοι τύποι δεδομένων (Integral data types)

Data Type	Bytes	Values
`[signed] char c;`	1	$[-128, 127]$
`unsigned char c;`	1	$[0, 255]$
`short int x;`	2	$[-32768, 32767]$
`unsigned short int x;`	2	$[0, 65535]$
`int x;`	4	$[-2^{31}, 2^{31} - 1]$
`unsigned int x;`	4	$[0, 4294967295]$
`long int x;`	4	$[-2^{31}, 2^{31} - 1]$
`unsigned long int x;`	4	$[0, 4294967295]$
`long long int x;`	8	$[-2^{63}, 2^{63} - 1]$
`unsigned long long int x;`	8	$[0, 2^{64} - 1]$

$2^{31} = 2,147,483,648$
$2^{63} = 9,223,372,036,854,775,808$
$2^{64} = 18,446,744,073,709,551,616$

Αυτές οι τιμές εξαρτώνται από το συγκεκριμένο υπολογιστικό σύστημα και περιγράφονται στο αρχείο limits.

Τύποι πραγματικών αριθμών (Real numeric types)

Data Type	Bytes	Values
`float x;`	4	$1.17549435 \times 10^{-38}$
		$3.40282347 \times 10^{38}$
`double x;`	8	$2.2250738585072014 \times 10^{-308}$
		$1.7976931348623157 \times 10^{308}$
`long double x;`	16	$\approx 10^{-4931}$
		$\approx 10^{4932}$

Αυτές οι τιμές εξαρτώνται από το συγκεκριμένο υπολογιστικό σύστημα και περιγράφονται στο αρχείο limits.

Λογικοί και Απαριθμητικοί Τύποι

Λογικός τύπος (Boolean Type)

```
bool flag; // takes value true or false
bool BC_deposition = false;
```

Τύπος Απαρίθμησης (Enumeration Type)

- τύπος δεδομένων οριζόμενος από τον χρήστη
- (υπο)σύνολο ακέραιων σταθερών που αναπαρίστανται από αντιπροσωπευτικά ονόματα

Παράδειγμα:

```
#include <iostream>

int main()
{
    enum Day {Monday, Tuesday, Wednesday, Thursday,
                    Friday, Saturday, Sunday};
    Day today;
    today = Monday;
    std::cout << today << std::endl; // prints 0
    return 0;
}
```

Παραστάσεις και Τελεστές

Απλή Παράσταση: σύνθεση ενός ή περισσοτέρων *τελεσταίων (operands)* και ενός *τελεστή (operator)* που εφαρμόζεται σε αυτούς.
MISTHOS - FOROS

Σύνθετη Παράσταση: συνδιασμός δύο ή περισσοτέρων παραστάσεων.

KATHARA_KERDH = ESODA - FOROS - EKSODA

Αριθμητικοί Τελεστές

Τελεστής	Λειτουργία	Χρήση	Προσεταιριστικότητα
( )	παρενθέσεις	(expr_list)	
*	πολλαπλασιασμός	expr * expr	
/	διαίρεση	expr / expr	
%	υπόλοιπο διαίρεσης	expr % expr	$A \rightarrow \Delta$
+	πρόσθεση	expr + expr	
-	αφαίρεση	expr - expr	

```
int i = 4, j = 7, k = 5;
val = i * j / 2 + k % 2 - 3;      // val has value 12
val = (i * j / (2 + k)) % 2 - 3; // val has value -3

int i = 21/6;          // value of i is 3
rem = 3.14159 % 3; // error: floating point operand
```

Λογικοί και συσχετιστικοί τελεστές

Τελεστής	Λειτουργία	Χρήση	Προσεταιρ.
!	λογικό NOT	!expr	$\Delta \rightarrow A$
<	μικρότερο	expr < expr	
<=	μικρότερο ή ίσο	expr <= expr	
>	μεγαλύτερο	expr > expr	
>=	μεγαλύτερο ή ίσο	expr >= expr	
==	ίσο	expr == expr	$A \rightarrow \Delta$
!=	διάφορο	expr != expr	
&&	λογικό AND	expr && expr	
||	λογικό OR	expr || expr	

```
if (iter <= max && found == true)
    // do something
```

Λογικοί τελεστές πράξεων bits (Bitwise Operators)

Χρησιμοποιούνται για πράξεις επί των bits ακέραιων τελεσταίων

Τελεστής	**Λειτουργία**	**Χρήση**	**Προσεταιρ.**
~	λογικό NOT στα bits	~ expr	$\Delta \rightarrow A$
<<	ολίσθηση αριστερά	expr << expr	
>>	ολίσθηση δεξιά	expr >> expr	
&	λογικό AND στα bits	expr & expr	$A \rightarrow \Delta$
^	λογικό XOR στα bits	expr ^ expr	
|	λογικό OR στα bits	expr | expr	

```
// Assume int are 4 bits
unsigned int a = 11;      // 1011
unsigned int b = !a;      // 0100
unsigned int c = a << 1;  // 0110
unsigned int d = a >> 2;  // 0010
unsigned int e = a & c;   // 0010
unsigned int f = a ^ c;   // 1101
unsigned int g = a | c;   // 1111
```

Τελεστές ανάθεσης (Assignment operators)

Τελεστής	Λειτουργία	Χρήση	Προσετ.
=	απλή ανάθεση	lvalue = expr	
*=	πολλ/σμός με ανάθεση	lvalue *= expr	
/=	διαίρεση με ανάθεση	lvalue /= expr	
%=	υπόλοιπο διαίρ. με ανάθεση	lvalue %= expr	
+=	πρόσθεση με ανάθεση	lvalue += expr	
-=	αφαίρεση με ανάθεση	lvalue -= expr	$\Delta \rightarrow A$
<<=	ολίσθηση αριστ. με ανάθεση	lvalue <<= expr	
>>=	ολίσθηση δεξιά με ανάθεση	lvalue >>= expr	
&=	λογικό AND με ανάθεση	lvalue &= expr	
|=	λογικό OR με ανάθεση	lvalue |= expr	
^=	λογικό XOR με ανάθεση	lvalue ^= expr	

a op= b είναι ισοδύναμο με a = a op b

```
sum += a[i];   // sum = sum + a[i]
div /= 2;      // div = div/2;
```

Μονομελείς τελεστές (Unary operators)

Τελεστής	**Λειτουργία**	**Χρήση**	**Προσετ.**
++	επιθεματική βηματική αύξηση	lvalue++	$A \rightarrow \Delta$
--	επιθεματική βηματική μείωση	lvalue--	$A \rightarrow \Delta$
sizeof	μέγεθος αντικειμένου	sizeof expr	
sizeof	μέγεθος τύπου	sizeof(type)	
++	προθεματική βηματική αύξηση	++lvalue	
--	προθεματική βηματική μείωση	--lvalue	$\Delta \rightarrow A$
-	μονομελές μείον	-expr	
+	μονομελές πλήν	+expr	

Παράδειγμα με τελεστές

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

int main()
{
    int x=5;
    cout << x << endl;    // prints 5
    cout << x++ << endl; // prints 5
    cout << x << endl;    // prints 6

    x=5;
    cout << x << endl;    // prints 5
    cout << ++x << endl; // prints 6
    cout << x << endl;    // prints 6

    cout << sizeof(int) << endl; // size of an int in bytes
    cout << sizeof(x) << endl;   // size of x in bytes
    // alternatively: sizeof x << endl;
    return 0;
}
```

Τελεστές (σε σειρά φθίνουσας προτεραιότητας)

Τελεστής	**Λειτουργία**	**Χρήση**	**Προσεταιρ.**
()	value construction	type(expr_list)	
++	post increment	lvalue++	L-to-R
--	post decrement	lvalue--	
sizeof	size of object	sizeof expr	
sizeof	size of type	sizeof(type)	
++	pre increment	++lvalue	
--	pre decrement	--lvalue	
~	bitwise NOT	~expr	
!	logical NOT	!expr	R-to-L
-	unary minus	-expr	
+	unary plus	+expr	
&	address of	&expr	
*	dereference	*expr	
&	bitwise AND	expr & expr	
^	bitwise XOR	expr ^ expr	
|	bitwise OR	expr | expr	RL-to-R
&&	logical AND	expr && expr	
||	logical OR	expr || expr	
?:	conditional expression	expr?expr:expr	
<<	stream insertion	expr << expr	L-to-R
>>	stream extraction	expr >> expr	

Τελεστές (σε σειρά φθίνουσας προτεραιότητας - συνέχεια)

Τελεστής	Λειτουργία	Χρήση	Προσεταιρ.
*	multiply	expr * expr	
/	divide	expr / expr	
%	modulo or remainder	expr % expr	
+	plus	expr + expr	
-	minus	expr - expr	
<<	shift left	expr << expr	
>>	shift right	expr >> expr	L-to-R
<	less than	expr < expr	
<=	less than or equal	expr <= expr	
>	greater than	expr > expr	
>=	greater than or equal	expr >= expr	
==	equal	expr == expr	
!=	not equal	expr != expr	
=	conventional assignment	lvalue = expr	
*=	multiply and assign	lvalue *= expr	
/=	divide and assign	lvalue /= expr	
%=	modulo and assign	lvalue %= expr	
+=	add and assign	lvalue += expr	
-=	subtract and assign	lvalue -= expr	R-to-L
<<=	shift left and assign	lvalue <<= expr	
>>=	shift right and assign	lvalue >>= expr	
& =	AND and assign	lvalue &= expr	
| =	OR and assign	lvalue | = expr	
^=	XOR and assign	lvalue ^= expr	

Σταθερές (Constants) – const

- δηλώνει ένα αντικείμενο ως μη τροποποιήσιμο
- ένα αντικείμενο τύπου const πρέπει πάντοτε να αρχικοποιείται

Παράδειγμα:

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

int main()
{
    const int x; // Error: x must be initialised
    x=7;         // Error: x is not modifiable

    const int x=7;
    cout << "Value of x is " << x << endl;
    return 0;
}
```

Σειρές χαρακτήρων (strings)

- Μπορούν να αναπαρασταθούν με δύο τρόπους:

  (1) πίνακας τύπου char που τελειώνει με τον χαρακτήρα null (\0)(μη ασφαλής – να αποφεύγεται)

  (2) αντικείμενα του τύπου δεδομένων string

Παράδειγμα:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    char course[40];
    strcpy(course, "Algorithm "); // takes care of \0
    strcat(course, "Engineering ");
    strcat(course, "and Experimentation");
    cout << course << endl;
    return 0;
}
```

΄Αλλες συναρτήσεις σειρών χαρακτήρων

- strcmp(s1,s2); // comparison
- strlen(s1); // returns the length of s1

Παράδειγμα με strings

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string course;
    course = "Algorithm ";
    course += "Engineering ";
    course += "and Experimentation";
    cout << course << endl;
    return 0;
}
```

΄Αλλες λειτουργίες που αφορούν τον τύπο string

Συναρτήσεις Κατασκευής : string s;
string s1(s2);

Σύγκριση : γίνεται χρησιμοποιώντας τους συνήθεις συσχετιστικούς τελεστές

Συνένωση : s1+s2;

Μήκος : s.length();

Κενότητα : s.empty();

Ρητή μετατροπή τύπου (casting)

- (type) expression – C στυλ, να αποφεύγεται
- type (expression)
- static_cast<type> (expression)

Παράδειγμα

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

int main ()
{
    char c, ch[] = "a";
    float a, b;
    int i=1, j;
    a = static_cast<float>(i);
    b = float(i);
    c = ch[0];
    j = int(c);

    cout << i << endl;
    cout << a << endl;
    cout << b << endl;
    cout << ch << endl; // prints a
    cout << j << endl;  // prints 97
    return 0;
}
```

Προαγωγή (έμμεση μετατροπή τύπου)

```
int counter;
float total, average;
...

average = total/counter;
```

Ιεραρχία προαγωγής των βασικών ενσωματωμένων τύπων

```
long double
double
float
unsigned long int
long int
unsigned int
int
unsigned short int
short int
unsigned char
char
```

Μετατροπές Λογικών Τύπων

- bool $\rightarrow$ int (προαγωγή): true $\Rightarrow$ 1, false $\Rightarrow$ 0.
- αριθμητική τιμή ή τιμή δείκτη $\rightarrow$ bool:
  zero value $\Rightarrow$ false, non-zero value $\Rightarrow$ true.

Παράδειγμα:

```
bool found = false;
int occurrence_count = 0;

. . .

if (found)
    occurrence_count += found; // promotion
. . .

if (occurrence_count)
{
    cout << "No. of occurrences ";
    cout << occurrence_count << endl;
}
else
    cout << "Nothing has been found" << endl;
```

Τέλος Ενότητας

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση **1.0**.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Σημείωμα Ιστορικού Εκδόσεων 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