Αντικειμενοστραφείς Γλώσσες Προγραμματισμού C++ / ROOT

Ιωάννης Παπαδόπουλος

Τμήμα Φυσικής, Πανεπιστήμιο Ιωαννίνων

Οκτώβριος 2018

Περιεχόμενα

Λέξεις κλειδιά (keywords) της C++
(τυποποίηση c++11)

alignas	char32_t	enum	namespace	return	try
alignof	class	explicit	new	short	typedef
and	compl	export	noexcept	signed	typeid
and_eq	const	extern	not	sizeof	typename
asm	constexpr	false	not_eq	static	union
auto	const_cast	float	nullptr	static_assert	unsigned
bitand	continue	for	operator	static_cast	using
bitor	decltype	friend	or	struct	virtual
bool	default	goto	or_eq	switch	void
break	delete	if	private	template	volatile
case	do	inline	protected	this	wchar_t
catch	double	int	public	thread_local	while
char	dynamic_cast	long	register	throw	xor
char16_t	else	mutable	reinterpret_cast	true	xor_eq

Μαθηματικές συναρτήσεις, μέσω της επικεφαλίδας <cmath> *(τυποποίηση c++11)*

Βασικές πράξεις

abs, fabs	|x|
fmod	υπόλοιπο διαίρεσης αριθμών κινητής υποδιαστολής
remainder	εμπρόσημο υπόλοιπο διαίρεσης
remquo	εμπρόσημο υπόλοιπο διαίρεσης και ακέραιο πηλίκο
fma	(x*y)+z
fmax	μέγιστος δύο αριθμών κινητής υποδιαστολής
fmin	ελάχιστος δύο αριθμών κινητής υποδιαστολής
fdim	θετική διαφορά δύο αριθμών κινητής υποδιαστολής, max(0, x-y)
nan, nanf, nanl	not-a-number (NaN)

Μαθηματικές συναρτήσεις, μέσω της επικεφαλίδας <cmath> *(τυποποίηση c++11)*

Εκθετικές, λογαριθμικές και συναρτήσεις δυνάμεων

exp	e^x
exp2	2^x
expm1	$e^x - 1$
log	$\ln(x)$
log10	$\log_{10} x$
log2	$\log_2 x$
log1p	$\ln(1 + x)$

pow	x^y
sqrt	$\sqrt{x}$
cbrt	$\sqrt[3]{x}$
hypot	$\sqrt{x^2 + y^2}$

Μαθηματικές συναρτήσεις, μέσω της επικεφαλίδας <cmath> *(τυποποίηση c++11)*

Τριγωνομετρικές και υπερβολικές συναρτήσεις

sin	$\sin x$
cos	$\cos x$
tan	$\tan x$
asin	$\sin^{-1} x$
acos	$\cos^{-1} x$
atan	$\tan^{-1} x$
atan2	$\tan^{-1}$ του σημείου (x, y)

sinh	$\sinh x$
cosh	$\cosh x$
tanh	$\tanh x$
asinh	$\sinh^{-1} x$
acosh	$\cosh^{-1} x$
atanh	$\tanh^{-1} x$

Μαθηματικές συναρτήσεις, μέσω της επικεφαλίδας <cmath> *(τυποποίηση c++11)*

Συναρτήσεις Γ και σφάλματος

erf	συνάρτηση σφάλματος
erfc	συμπληρωματική συνάρτηση σφάλματος
tgamma	συνάρτηση Γ
lgamma	φυσικός λογάριθμος της συνάρτησης Γ

Μαθηματικές συναρτήσεις, μέσω της επικεφαλίδας <cmath> *(τυποποίηση c++11)*

Συναρτήσεις στρογγυλοποίησης

ceil	πλησιέστερος ακέραιος όχι μικρότερος της δοθείσας τιμής
floor	πλησιέστερος ακέραιος όχι μεγαλύτερος της δοθείσας τιμής
trunc	πλησιέστερος ακέραιος όχι μεγαλύτερος κατ' απόλυτη τιμή της δοθείσας τιμής
round	
lround	
llround	
nearbyint	
rint	
lrint	
llrint	

άλλες συναρτήσεις στρογγυλοποίησης

Μαθηματικές συναρτήσεις, μέσω της επικεφαλίδας <cmath> *(τυποποίηση c++11)*

Συναρτήσεις χειρισμού αριθμών κινητής υποδιαστολής

frexp	decomposes a number into significand and a power of 2
ldexp	multiplies a number by 2 raised to a power
modf	decomposes a number into integer and fractional parts
scalbn	multiplies a number by FLT_RADIX raised to a power
scalbln	
ilogb	extracts exponent of the number
logb	extracts exponent of the number
nextafter	next representable floating point value towards the given value
nexttoward	
copysign	copies the sign of a floating point value

Μαθηματικές συναρτήσεις, μέσω της επικεφαλίδας <cmath> *(τυποποίηση c++11)*

Συναρτήσεις κατηγοριοποίησης και σύγκρισης

fpclassify	categorizes the given floating point value
isfinite	the given number has finite value?
isinf	the given number is infinite?
isnan	the given number is NaN?
isnormal	the given number is normal?
signbit	the given number is negative?
isgreater	the first floating-point argument is greater than the second?
isgreaterequal	the first floating-point argument is greater or equal than the second?
isless	the first floating-point argument is less than the second?
islessequal	the first floating-point argument is less or equal than the second?
islessgreater	the first floating-point argument is less or greater than the second?
isunordered	checks if two floating-point values are unordered

Εντολή if

```
if ( συνθήκη ) εντολήΑνΣυνθήκηΑληθής

if ( συνθήκη ) εντολήΑνΣυνθήκηΑληθής else εντολήΑνΣυνθήκηΨευδής
```

- συνθήκη: οποιαδήποτε έκφραση της C++ που μπορεί να αποτιμηθεί σε τιμή boolean (true ή false). Υπενθυμίζεται ότι true=1 και false=0.
- εντολήΑνΣυνθήκηΑληθής: εντολή ή μπλοκ εντολών της C++, που εκτελείται εφόσον η συνθήκη είναι αληθής.
- εντολήΑνΣυνθήκηΨευδής: εντολή ή μπλοκ εντολών της C++, που εκτελείται εφόσον η συνθήκη είναι ψευδής.

Εντολή if

```
#include <iostream>

int main() {
int i, j;
  std::cin >> i >> j ;
  if (i>2)
    std::cout << "Το i είναι μεγαλύτερο του 2." << std::endl;
  else
    std::cout << "i≤2" << std::endl;

  if ( i!=0 ) {
    if ( j==0 ) {
      std::cout << "Δεν μπορώ να υπολογίσω το k=84/(i*j) επειδή j=0" << std::endl;
    }
    else {
      std::cout << "k=84/(i*j)="<< 84/(i*j) << std::endl;
    }
  }
  else {
    if (j==0) {
      std::cout << "Δεν μπορώ να υπολογίσω το k=84/(i*j) επειδή i=0 και j=0" << std::endl;
    }
    else {
      std::cout << "Δεν μπορώ να υπολογίσω το k=84/(i*j) επειδή i=0" << std::endl;
    }
  }
  return 0;
}
```

Εντολή if


```cpp
#include <iostream>

double d(int i, int j) {  // Kronecker delta
  if (i==j)
    return 1;
  else
    return 0;
}

int main() {
  int i, j;
  std::cin >> i >> j ;
  std::cout << "Kronecker δ(" << i << "," << j << ")=" << d(i,j) << std::endl;
  return 0;
}
```

Τελεστής συνθήκης (conditional operator): ? :


```
συνθήκη ? έκφρασηΑνΣυνθήκηΑληθής : έκφρασηΑνΣυνθήκηΨευδής
```

- συνθήκη: οποιαδήποτε έκφραση της C++ που μπορεί να αποτιμηθεί σε τιμή boolean (true ή false). Υπενθυμίζεται ότι true=1 και false=0.
- έκφρασηΑνΣυνθήκηΑληθής: εφόσον η συνθήκη είναι αληθής, υπολογίζεται και επιστρέφεται η τιμή της
- έκφρασηΑνΣυνθήκηΨευδής: εφόσον η συνθήκη είναι ψευδής, υπολογίζεται και επιστρέφεται η τιμή της

Τελεστής συνθήκης (conditional operator): ? :


```cpp
#include <iostream>

double d(int i, int j) {  // Kronecker delta
  return i==j ? 1 : 0 ;
}

int main() {
  int i, j;
  std::cin >> i >> j ;
  std::cout << "Kronecker δ(" << i << "," << j << ")=" << d(i,j) << std::endl;
  return 0;
}
```

Εντολή switch


```
switch ( έκφρασηΜεΑκέραιοΑποτέλεσμα ) εντολή
```

- έκφρασηΜεΑκέραιοΑποτέλεσμα: Μπορεί να χρησιμοποιηθεί για να ελεγχθεί η τιμή της από ετικέτες `case:` για τον έλεγχο της ροής του προγράμματος.
- εντολή: Συνήθως πρόκειται για μπλοκ εντολών, όπου μπορούν να χρησιμοποιηθούν οι ετικέτες `case:` και `default:`, και μπορεί να χρησιμοποιηθεί η εντολή `break;`.

Εντολή switch

```
switch(1) {
    case 1 : cout << '1';   // τυπώνει "1",
    case 2 : cout << '2';   // έπειτα τυπωνει "2"
}

switch(1) {
    case 1 : cout << '1';   // τυπώνει "1"
             break;         // έξοδος από το switch
    case 2 : cout << '2';
             break;
}

switch(6) {                 // θα εκτελεστεί το μέρος του default:
    case 1 : cout << '1';
             break;
    case 2 : cout << '2';
             break;
    default: cout << "Δεν καλύπτεται από τις επιμέρους περιπτώσεις" << endl;
}
```

Εντολή for

```
for ( εντολήΑρχικοποίησης  συνθήκη  ;  έκφρασηΕπανάληψης )
    εντολή
```

- εντολήΑρχικοποίησης: Χρησιμοποιείται προαιρετικά για αρχικοποίηση (αλλιώς μπορεί να είναι η κενή εντολή `;`).
- συνθήκη: Χρησιμοποιείται προαιρετικά. Τότε η τιμή της υπολογίζεται (true ή false), και σε περίπτωση που είναι ψευδής, το πρόγραμμα εξέρχεται από το βρόχο.
- έκφρασηΕπανάληψης: Χρησιμοποιείται προαιρετικά. Εκτελείται στο τέλος κάθε ανακύκλωσης, και πριν υπολογιστεί εκ νέου η συνθήκη.
- εντολή: Συνήθως πρόκειται για μπλοκ εντολών. Εκτελείται σε κάθε ανακύκλωση, όσο η συνθήκη παραμένει αληθής.

Εντολή for


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

double d(int i, int j) {  // Kronecker delta
  return i==j ? 1 : 0 ;
}

int main() {
  for ( int i=0 ; i<5 ; i++ )
    for ( int j=0 ; j<5 ; j++ )
      cout << "Kronecker δ(" << i << "," << j << ")=" << d(i,j) << endl;
  return 0;
}
```

Εντολή for

Ατέρμονας βρόχος:

```
for ( ; ; ) {
   cout << "θα τυπώνω αυτή γραμμή για πάντα..." << endl;
}
```

- εντολήΑρχικοποίησης: κενή εντολή `;`.
- συνθήκη: Δεν χρησιμοποιείται (θεωρείται πάντοτε αληθής).
- έκφρασηΕπανάληψης: Δεν χρησιμοποιείται.
- εντολή: Εκτελείται σε κάθε ανακύκλωση, για πάντα...
  - Έξοδος από ατέρμονα βρόχο μπορεί να επιτευχθεί μέσω μίας εντολής `break` ή `return` μετά από κάποιον έλεγχο.

Εντολή for

- Έξοδος από ατέρμονα βρόχο μπορεί να επιτευχθεί μέσω μίας εντολής `break` ή `return` μετά από κάποιον έλεγχο. Π.χ.

```
for ( ; ; ) {
   if ( CurrentTemperature()>35 )
     cout << "Έχει πολλή ζέστη! Ας περιμένουμε..." << endl;
   else {
     cout << "Ώρα να συνεχίσουμε..." << endl;
     break;   // το πρόγραμμα θα βγει από την εντολή for
   }
}
```

Εντολή for για περιοχές (range-for)

```
for ( δήλωσηΠεριοχής : έκφρασηΠεριοχής ) εντολή
```

- δήλωσηΠεριοχής: Δήλωση ονόματος μεταβλητής, με τύπο αυτόν των στοιχείων που εισάγονται στην έκφρασηΠεριοχής, ή μία αναφορά στον τύπο αυτό (&). Συχνά, χρησιμοποιείται αυτόματος προσδιορισμός του τύπου, μέσω του `auto`.
- έκφρασηΠεριοχής: Οποιαδήποτε έκφραση που παριστά μία κατάλληλη ακολουθία (π.χ. ένας πίνακας) ή μία λίστα αρχικοποίησης εντός `{ }`.
- εντολή: Συνήθως πρόκειται για μπλοκ εντολών. Εκτελείται σε κάθε ανακύκλωση.

Εντολή for για περιοχές (range-for)

```
#include <iostream>
#include <vector>
int main() {
  for (int n : {0, 1, 2, 3, 4, 5})   // περιοχή = αρχικοποίηση εντός { }
    std::cout << n << ' ';
  std::cout << '\n';

  int a[] = {0, 1, 2, 3, 4, 5};
  for (int n : a)                    // η περιοχή μπορεί να είναι ένας πίνακας
    std::cout << n << ' ';
  std::cout << '\n';

  for (int n : a)
    std::cout << 1 << ' ';           // δεν είναι απαραίτητο να χρησιμοποιείται η
  std::cout << '\n';                 // μεταβλητή ανακύκλωσης
}
```

Εντολή for για περιοχές (range-for)

Πιο προχωρημένο παράδειγμα:

```
#include <iostream>
#include <vector>
int main() {
  std::vector<int> v = {0, 1, 2, 3, 4, 5};

  for (const int& i : v)     // access by const reference
    std::cout << i << ' ';   // cannot modify i (const int&)
  std::cout << '\n';

  for (int& i : v)                  // access by reference
    {std::cout << i << ' '; i++;}   // v is modified
  std::cout << '\n';

  for (auto i : v) // access by value, i is int
    {std::cout << i << ' '; i++;}   // v is not modified  (only its local copy is)
  std::cout << '\n';

  for (auto&& i : v)                // access by forwarding reference, i is int&
    {std::cout << i << ' '; i++;}   // v is modified
  std::cout << '\n';

  const auto& cv = v;
  for (auto&& i : cv)        // access by f-d reference, i is const int&
    std::cout << i << ' ';   // cannot modify i (const int&)
  std::cout << '\n';
}
```

Εντολή while

```
while ( συνθήκη ) εντολή
```

- συνθήκη: Η τιμή της υπολογίζεται (true ή false), και εφόσον είναι αληθής, η εντολή εκτελείται και επανυπολογίζεται η τιμή της συνθήκης.
- εντολή: Συνήθως πρόκειται για μπλοκ εντολών. Εκτελείται σε κάθε ανακύκλωση, όσο η συνθήκη παραμένει αληθής.
- Υπάρχει περίπτωση η εντολή να μην εκτελεστεί ποτέ, αν την πρώτη φορά υπολογισμού η συνθήκη είναι ψευδής.

Εντολή do-while

```
do  εντολή  while  συνθήκη
```

- εντολή: Συνήθως πρόκειται για μπλοκ εντολών. Εκτελείται σε κάθε ανακύκλωση, όσο η συνθήκη παραμένει αληθής.
- συνθήκη: Η τιμή της υπολογίζεται (true ή false), και εφόσον είναι αληθής, η εντολή εκτελείται και πάλι. Έπειτα, επανυπολογίζεται η τιμή της συνθήκης κ.ο.κ.
- Η εντολή θα εκτελεστεί τουλάχιστον μία φορά, αφού ο υπολογισμός της συνθήκης έπεται αυτής.

Εντολές μετάβασης: break, continue, return και goto

Οι εντολές μετάβασης καθοδηγούν τη ροή της εκτέλεσης του προγράμματος. Πιο συγκεκριμένα:

break;

Οδηγεί τη ροή του προγράμματος έξω από την περιοχή στην οποία καλείται. Η περιοχή αυτή μπορεί να είναι μία σύνθετη εντολής ανακύκλωσης (βρόχος) ή μία εντολή επιλογής switch.

continue;

Χρησιμοποιείται εντός μίας σύνθετης εντολής ανακύκλωσης ενός βρόχου for, range-for, while ή do-while, και οδηγεί τη ροή του προγράμματος στο τέλος της σύνθετης εντολής ανακύκλωσης.

Εντολές μετάβασης: break, continue, return και goto

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

int main() {
  vector<int> v;   // ορίζω ένα διάνυσμα ακεραίων, το v
  srand(123);      // αρχικοποιώ τη γεννήτρια τυχαίων αριθμών
  cout << "Γεμίζω το v με τυχαίους αριθμούς από 0 έως και 100." << endl;
  for  (int i=0 ; i<10 ; i++)  v.push_back(rand()%101);

  auto pos=v.begin();
  for (auto elem : v) {
    pos++;
    if (elem>80) {
      cout << "Αφαιρώ το πρώτο στοιχείο του v με τιμή >80, το ";
      cout << elem;
      cout << " που βρίσκεται στην " << distance(v.begin(),pos);
      cout << "η θέση." << endl;
      v.erase(pos-1);
      break;       // <========= έξοδος από το for
    }
  }
  cout << "  ---> Το διάνυσμα έχει "<< v.size() << " στοιχεία:";
  for  (int i=0 ; i<v.size() ; i++)  cout << " " << v.at(i);
  cout << endl;
}
```

Εντολές μετάβασης: break, continue, return και goto

```cpp
#include <iostream>
#include <boost/format.hpp>
using namespace std;
using namespace boost;

int main() {
  cout << '\n';

  for ( int j=0 ; j<10 ; j++ ) {
    for ( int k=0 ; k<20 ; k++ ) {
      if ( (k==3) || (k%4==0) ) continue;     // <===== συνέχισε με το επόμενο k
      cout << format("%4d") % j << k << " ";  // δεν εκτελείται όποτε k==3 ή k%4==0
    }
    cout << endl;
  }
}
```

Εντολές μετάβασης: break, continue, return και goto

return *έκφραση* ;

Οδηγεί τη ροή του προγράμματος έξω από τη συνάρτηση στην οποία καλείται. Επιστρέφει την τιμή της *έκφρασης* αφού πρώτα την υπολογίσει μετατρέποντάς την έμμεσα στον τύπο της συνάρτησης (implicit conversion).

goto *ετικέτα* ;

Η χρήση του θεωρείται κακός προγραμματισμός! Πολύ εύκολα μπορεί να κάνει τον πηγαίο κώδικα ακατανόητο και να εισάγει λογικά λάθη...

(Βλέπε «*Goto* Statement Considered Harmful»)

Οδηγεί τη ροή του προγράμματος σε συγκεκριμένο σημείο εντός της συνάρτησης στην οποία καλείται, το οποίο δηλώνεται με μία ετικέτα.

Εντολές μετάβασης: break, continue, return και goto


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

int gcd(int i, int j) {  // μέγιστος κοινός διαιρέτης
  for ( int k=(i<j?i:j) ; k>1 ; k-- ) {
    if ( (i%k==0) && (j%k==0) ) return k;   //   <===== επιστροφή τιμής
  }
  return 1;                                 //   <===== επιστροφή τιμής
}

int main() {  // Πρόγραμμα εκτύπωσης των πρώτων αριθμών από το i1 έως το i2
  int i1=3, i2=500;
  cout << "Πρώτοι αριθμοί στο διάστημα [" << i1 << "," << i2 << "]:\n";
  for ( int i=i1 ; i<=i2 ; i++ ) {
    for ( int j=2 ; j<i ; j++ ) {
      if ( gcd(i,j)>1 )
        goto next_i;      // ο i δεν είναι πρώτος -> τέλος βρόχου -> επόμενο i
    }
    cout << " " << i;     // ο i είναι πρώτος -> τύπωσέ τον
next_i:
    ;
  }
  cout << endl;
}                 // Η main σιωπηλά επιστρέφει μηδέν, εφόσον δεν βάλαμε return.
                  // Η απουσία return αποτελεί προνόμιο MONON της main,
                  // οπότε ο μεταγλωττιστής το επιτρέπει.
```

Άσκηση 1: Το παραγοντικό n!

$$n! = \prod_{i=1}^{n} i\,, \quad 0! = 1$$

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

int main() {
  int n;
  cout << "Υπολογισμός του n! Δώσε το n: ";
  cin >> n;
  assert(n>=0);  // Αν δεν ισχύει, το πρόγραμμα τερματίζει με μήνυμα λάθους
  double p=1;
  for ( int i=2 ; i<=n ; i++ ) {
    p*=i;
  }
  cout << n << "!=" << p << endl;
}
```

Άσκηση 2: Το παραγοντικό n! (με αναδρομή)

$$n! = \prod_{i=1}^{n} i = n \cdot (n-1)!\,, \quad 0! = 1$$

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

double factorial(int i) {
  if ( (i==0) || (i==1) )  return 1;
  return i*factorial(i-1);
}

int main() {
  int n;
  cout << "Υπολογισμός του n! Δώσε το n: ";
  cin >> n;
  assert(n>=0);  // Αν δεν ισχύει, το πρόγραμμα τερματίζει με μήνυμα λάθους
  cout << n << "!=" << factorial(n) << endl;
}
```

Άσκηση 3: Υπολογισμός αναδρομικής σχέσης

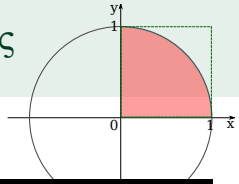
$$a(n) = a_{n-1} - 3a_{n-2}\,, \quad a_0 = 3\,, \quad a_1 = 2$$

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

double a(int i) {
  if ( i==0 )  return 3;
  if ( i==1 )  return 2;
  return a(i-1)-3*a(i-2);
}

int main() {
  int n;
  cout << "Δώσε το n: ";
  cin >> n;
  assert(n>=0);  // Αν δεν ισχύει, το πρόγραμμα τερματίζει με μήνυμα λάθους
  cout << "a(" << n << ")=" << a(n) << endl;
}
```

Άσκηση 4: Υπολογισμός του π με τυχαίους αριθμούς



$$\lim_{N\to\infty}\frac{N_{inside}}{N}=\frac{S_{in}}{S_{out}}=\frac{\frac{\pi 1^2}{4}}{1^2}=\frac{\pi}{4}\quad\Rightarrow\quad\pi=\lim_{N\to\infty}\frac{4N_{inside}}{N}$$

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main() {
  srand(time(NULL)); // αρχικοποίηση της γεννήτριας τυχαίων με την ώρα του Η/Υ
  int Ninside=0;     // μετρητής σημείων εντός του τεταρτοκυκλίου
  int N=100000000;   // ολικός αριθμός σημείων

  for ( int i=0 ; i<N ; i++) {
    double x=(double)rand()/RAND_MAX; // x,y : τυχαίοι στην περιοχή [0,1]
    double y=(double)rand()/RAND_MAX; // (x,y) : σημείο εντός του τετραγώνου
    if (x*x+y*y<1) Ninside++;
  }
  //  Ninside/N = Sin/Sout = (π1²/4) / 1² = π/4   =>   π = 4 Ninside / N
  cout << "π=" << 4.0*Ninside/N << endl;
}
```