ΕΚΤΗ ΔΙΑΛΕΞΗ

Τύπος `string` (1/2)

Για την αποθήκευση και τον χειρισμό σειράς χαρακτήρων παρέχεται ο τύπος `std::string` από το `<string>`.

Δήλωση (με αρχική τιμή)

```
std::string s; // Κενό string
std::string q{}; // Κενό string
std::string p{'a','b','c','d'}; // p <- "abcd"
std::string g{u8"Δώσε αριθμό:"}; // g <- "Δώσε αριθμό:"
std::string r{g}; // r <- "Δώσε αριθμό:"
```

Τύπος `string` (2/2)

Άλλοι τρόποι προσδιορισμού αρχικής τιμής:

- Ένα τμήμα άλλου `string`, από μια θέση (η αρχική θέση είναι η μηδενική) και πέρα:

```
std::string s1{"Give number:"};
// copy all chars from position 3
std::string s2{s1,3}; // s2 <- "e number:"
```

- Ένα τμήμα άλλου `string`, από μια συγκεκριμένη θέση και με συγκεκριμένο πλήθος χαρακτήρων:

```
std::string s1{"Give number:"};
// copy 5 bytes from position 3
std::string s2{s1,3,5}; // s2 <- "e num"
```

- Επανάληψη ενός χαρακτήρα

```
std::string s(6, '*'); // s <- "******"
```

Χειρισμός `string` (1/2)

- Αντιγραφή με τον τελεστή =:

```
std::string s{"in"}; // s <- "in"
s = "out";  // s <- "out"
```

- Ένωση με τον τελεστή +:

```
std::string s1{"Give"};
std::string s2{"number:"};
std::string s3 = s1 + ' ' + s2; // s3 <- "Give number:"
```

- Συμπλήρωση με τον τελεστή +=:

```
std::string s{"Give"};
s+= ' ';   // s <- "Give "
s+="number:"; // s <- "Give number:"
```

Χειρισμός `string` (2/2)

- Επιλογή χαρακτήρα σε συγκεκριμένη θέση με τον τελεστή `[]`:

```
std::string s{"Give number:"};
std::cout << s[0] << '\n';  // 'G'
std::cout << s[10] << '\n';  // 'r'
```

- Ανάγνωση ή εκτύπωση με τους τελεστές `>>` και `<<`:

```
std::string s;
std::cin  >> s;
std::cout << s;
```

  Η ανάγνωση σταματά στον πρώτο κενό χαρακτήρα (ή στο τέλος της ροής).

- Σύγκριση με τους τελεστές `==`, `!=`, `<=`, `>=`, `>`, `<`. Το πρώτο ζεύγος άνισων χαρακτήρων καθορίζει το αποτέλεσμα.

Μετατροπή `string` σε αριθμό και αντίστροφα

- Στο `<string>` παρέχονται συναρτήσεις (`std::stoi()`, `std::stol()`, `std::stod()`, `std::stof()`, κλπ.) για τη μετατροπή ενός string που αρχίζει με αριθμητικούς χαρακτήρες σε ακέραιο ή πραγματικό αριθμό.

```
std::string s{"12 chars"};
auto k = std::stoi(s); // k = 12, int

std::string p{"12.32 + 5.0"};
auto d = std::stod(p); // d = 12.32, double
```

- Μετατροπή αριθμού σε `string` γίνεται με τη συνάρτηση `std::to_string()`:

```
auto s = std::to_string(6.2); // s <- "6.200000", std::string
```

Γεννήτρια τυχαίων αριθμών (1/3)

Η C++ παρέχει στο `<random>` συναρτήσεις και κλάσεις για την παραγωγή τυχαίων αριθμών. Υπάρχουν

- *Μηχανισμοί* παραγωγής σειράς τυχαίων bits.
  Κάθε bit έχει ίδια πιθανότητα να είναι 0 ή 1.
- *Κατανομές* που χρησιμοποιούν κάποιον από τους μηχανισμούς για την παραγωγή τυχαίων αριθμών με καθορισμένη πιθανότητα εμφάνισης.
  Ενδεικτικές κατανομές:

  ομοιόμορφη: ισοπίθανοι τυχαίοι σε κάποιο διάστημα,

  κανονική: τυχαίοι με πιθανότητα που μειώνεται εκθετικά με το τετράγωνο της απόστασής τους από μια μέση τιμή.

Γεννήτρια τυχαίων αριθμών (2/3)

Α' Παράδειγμα

Τυχαίοι ακέραιοι, ομοιόμορφα κατανεμημένοι στο διάστημα $[5, 20]$.

```
#include <random>

std::default_random_engine e{};
std::uniform_int_distribution<int> d{5, 20};

auto r = d(e);
// Καθε φορά που καλούμε το d(e), έχει τυχαία ακέραια τιμή.
 ... // Χρήση του r
```

Παρατήρηση

Κάθε φορά που δημιουργείται ο μηχανισμός e με τη δήλωση όπως γράφηκε παραπάνω, παράγει την ίδια σειρά τυχαίων bits. Αν θέλουμε διαφορετική σε κάθε εκτέλεση, γράφουμε:

```
std::random_device rd{};
std::default_random_engine e{rd()};
```

Γεννήτρια τυχαίων αριθμών (3/3)

Β' Παράδειγμα

Τυχαίοι πραγματικοί, ομοιόμορφα κατανεμημένοι στο διάστημα $[a, b)$.

```
#include <random>

std::default_random_engine e{};
std::uniform_real_distribution<double> d{a,b};

auto r = d(e);
// Καθε φορά που καλούμε το d(e) έχει τυχαία πραγματική τιμή.
 ... // Χρήση του r
```

Παρατήρηση

Το άνω όριο, b (με $b > a$), δεν περιλαμβάνεται στο πεδίο των τυχαίων αριθμών.

Ροές σε Αρχεία

Η C++ υποστηρίζει ανάγνωση από και εγγραφή σε αρχεία.
Στο `<fstream>` παρέχονται τα `std::ifstream` και `std::ofstream` για τη δημιουργία *ροής (stream) σε αρχείο*.

Δήλωση ροής για ανάγνωση

```
#include <fstream>
std::ifstream όνομα{"όνομα αρχείου"};
```

Χρήση ροής για ανάγνωση

```
όνομα >> μεταβλητή;
```

Δήλωση ροής για εγγραφή

```
#include <fstream>
std::ofstream όνομα{"όνομα αρχείου"};
```

Χρήση ροής για εγγραφή

```
όνομα << μεταβλητή/σταθερή/έκφραση/...;
```

Επιτυχία εισόδου–εξόδου δεδομένων

Η επιτυχία εκτύπωσης ή ανάγνωσης από κάποιο stream ελέγχεται μέσω της «τιμής» του stream: αν είναι **`false`** *τότε η εγγραφή ή η ανάγνωση έχει αποτύχει· αν είναι* **`true`** *έχει επιτύχει.*

Παράδειγμα

Ανάγνωση άγνωστου πλήθους ακεραίων από το stream in γίνεται με χρήση της εντολής επανάληψης **`while`** ως εξής:

```
int i;
while (in >> i) {
  ...
}
```

Εκτελείται η εντολή `in >> i` και κατόπιν ελέγχεται η «τιμή» του in. Αν η εκχώρηση τιμής στη μεταβλητή i έγινε κανονικά, η «τιμή» του ισοδυναμεί με **`true`** και εκτελείται το σώμα εντολών του **`while`**. Αλλιώς, η «τιμή» του `in` ισοδυναμεί με **`false`** και διακόπτεται η επανάληψη.

Διαμορφώσεις εκτύπωσης (1/5)

Η C++ παρέχει αντικείμενα («διαμορφωτές») που όταν στέλνονται για εκτύπωση, αλλάξουν τον τρόπο εκτύπωσης των επόμενων ποσοτήτων.

Στο header `<iomanip>` υπάρχουν

- ο `std::setprecision()` που ως όρισμα δέχεται ένα ακέραιο αριθμό. Το όρισμα προσδιορίζει το μέγιστο πλήθος των *σημαντικών* ή των δεκαδικών ψηφίων στην εκτύπωση πραγματικών αριθμών. Η προκαθορισμένη τιμή είναι 6.
- ο `std::setw()` που ως όρισμα δέχεται το ελάχιστο πλήθος των θέσεων στο οποίο θα τυπωθεί η επόμενη ποσότητα.
- ο `std::setfill()` που ως όρισμα δέχεται το χαρακτήρα με τον οποίο θα γεμίσουν οι κενές θέσεις αν ο `std::setw()` όρισε περισσότερες από την ακρίβεια. Ο προεπιλεγμένος χαρακτήρας είναι ο κενός, '␣'.

Διαμορφώσεις εκτύπωσης (2/5)

Παράδειγμα

```
#include <iomanip>
#include <iostream>

int main()
{
   std::cout << std::setprecision(4)
             << 1.2345678 << '\n'; // 1.235

   std::cout << std::setw(3)
             << std::setfill('0') << 3 <<'\n'; // 003
}
```

Διαμορφώσεις εκτύπωσης (3/5)

Στο `<ios>` παρέχονται οι

- `std::noboolalpha` (και `std::boolalpha`). Προσδιορίζουν ότι κατά την εκτύπωση λογικών ποσοτήτων θα εμφανίζονται οι αριθμοί 1 και 0 (ή οι λέξεις **`true`** και **`false`**).
- `std::scientific`: οι πραγματικοί τυπώνονται με τη μορφή $\pm d.ddddde \pm dd$.
- `std::fixed`: οι πραγματικοί τυπώνονται με τη μορφή $\pm dddd.dd$.
- `std::defaultfloat`: οι πραγματικοί τυπώνονται με τη μορφή που επιλέγει ο μεταγλωττιστής. Είναι η προκαθορισμένη επιλογή μορφής εκτύπωσης.
- `std::right` (και `std::left`). προκαλούν δεξιά (ή αριστερή) στοίχιση στην εκτύπωση.

Διαμορφώσεις εκτύπωσης (4/5)

Παράδειγμα

```
#include <iostream>
#include <ios>

int main()
{
  double a{256.123456789987};
  std::cout << "default\t" << a << '\n'; // 256.123
  std::cout << "scientific\t"  << std::scientific
            << a << '\n'; // 2.561235e+02
  std::cout << "fixed\t" << std::fixed
            << a << '\n'; // 256.123457
}
```

Διαμορφώσεις εκτύπωσης (5/5)

Αντί για τους διαμορφωτές `setprecision()`, `setw()` και `setfill()`, μπορούμε να χρησιμοποιήσουμε τις αντίστοιχες *συναρτήσεις-μέλη* κάθε ροής, `precision()`, `width()` και `fill()`.

Παράδειγμα

```
#include <iostream>

int main()
{
  std::cout.width(5);
  std::cout.fill('0');
  std::cout << 3; // -> 00003
  std::cout.precision(9);
  std::cout << 256.123456789987 << '\n'; // 256.123457
}
```

Εσωτερικά αρχεία (1/2)

Υπάρχει η δυνατότητα χρήσης εσωτερικού αρχείου (αποθηκεύεται προσωρινά στη μνήμη και όχι μόνιμα στο σκληρό δίσκο). Ο header `<sstream>` παρέχει τις κλάσεις `std::istringstream` και `std::ostringstream`.

Η εκτύπωση σε αντικείμενο τύπου `std::ostringstream` δημιουργεί ένα `std::string` με συνένωση (και πιθανή διαμόρφωση) των εκτυπούμενων ποσοτήτων. Η εξαγωγή του string γίνεται με τη συνάρτηση μέλος `str()`.

Παράδειγμα

```
#include <sstream>

std::ostringstream os;
os << "filename_" << std::setw(3)
   << std::setfill('0') << 3 << ".dat";
std::string s = os.str();  // s is "filename_003.dat"
```

Εσωτερικά αρχεία (2/2)

Αντικείμενο τύπου `std::istringstream` χρησιμοποιείται για ανάγνωση τιμών από το string που έχει εσωτερικά, όπως θα γινόταν από ροή αρχείου.

Παράδειγμα

```
#include <sstream>

int main()
{
  std::istringstream is{"5 6 7 a"};
  int i,j,k;
  char c;
  is >> i >> j >> k >> c; // i = 5, j = 6, k = 7, c = 'a'
}
```

Δομή **struct**: Ορισμός τύπου

Το διάνυσμα ή ο πίνακας ομαδοποιεί ποσότητες ίδιου τύπου. Πώς ομαδοποιώ σχετιζόμενες ποσότητες διαφορετικού τύπου;

Η C++ παρέχει τη σύνθετη δομή **struct**.

Σύνταξη

struct *όνομα_δομής {*
τύποςΑ μέλοςΑ;
τύποςΒ μέλοςΒ;
... // άλλα μέλη
};

Παρατηρήσεις

- Το όνομα_δομής αποτελεί νέο τύπο.
- Ο ορισμός του τύπου καλό είναι να γίνει έξω από κάθε συνάρτηση (ιδανικά σε header).

Δομή `struct`: Δήλωση αντικειμένου

Δήλωση χωρίς αρχική τιμή

όνομα_δομής όνομα_μεταβλητής;

Όσα μέλη είναι θεμελιώδους τύπου (αριθμητικού, χαρακτήρα κλπ.) ή `std::array<>` αποκτούν απροσδιόριστες τιμές.
Τα υπόλοιπα (π.χ. `std::string`, `std::vector<>`) αποκτούν τις αντίστοιχες προκαθορισμένες τιμές.

Δήλωση με αρχική τιμή

Απόδοση αρχικής τιμής σε ποσότητα του τύπου κατά τη δήλωση μπορεί να γίνει

- με λίστα τιμών:

  όνομα_δομής όνομα_μεταβλητής{τιμήA, τιμήB, ...};

  Οι τιμές αντιστοιχούν στα μέλη του αντικειμένου με τη σειρά που δηλώθηκαν στον ορισμό της δομής.

- με αντιγραφή κατά μέλη άλλου αντικειμένου ίδιου τύπου:

  όνομα_δομής όνομα1{τιμήA, τιμήB, ...};
  όνομα_δομής όνομα2{όνομα1};

Δομή **struct**: Πρόσβαση στα μέλη

Ατομική πρόσβαση στα μέλη αντικειμένου μιας δομής γίνεται με τον τελεστή '.'. Το όνομα του αντικειμένου ακολουθείται από '.' και το όνομα του μέλους. Π.χ.

όνομα_μεταβλητής.μέλοςΑ = ...

Δομή `struct`

Παράδειγμα: χημικό στοιχείο

Ορισμός νέου τύπου

```
struct ChemicalElement {
   double mass;  // atomic mass
   int Z; // atomic number
   std::string name;
   std::string symbol;
};
```

Δήλωση με αρχική τιμή

```
ChemicalElement hydrogen{1.008, 1, u8"Υδρογόνο", "H"};
```

Πρόσβαση στα μέλη

```
std::cout << u8"Το χημικό στοιχείο " << hydrogen.name
          << u8" έχει σύμβολο το "<< hydrogen.symbol
          << u8" και ατομική μάζα " << hydrogen.mass <<'\n';
```