Εισαγωγή στον Προγραμματισμό

Μέρος 5ο: Είσοδος - Έξοδος

Εξάμηνο Σπουδών: 3ο

Κωδικός Μαθήματος: 343

Τμήμα Μαθηματικών
Πανεπιστήμιο Ιωαννίνων

Μιχάλης Α. Μπέκος
bekos@uoi.gr

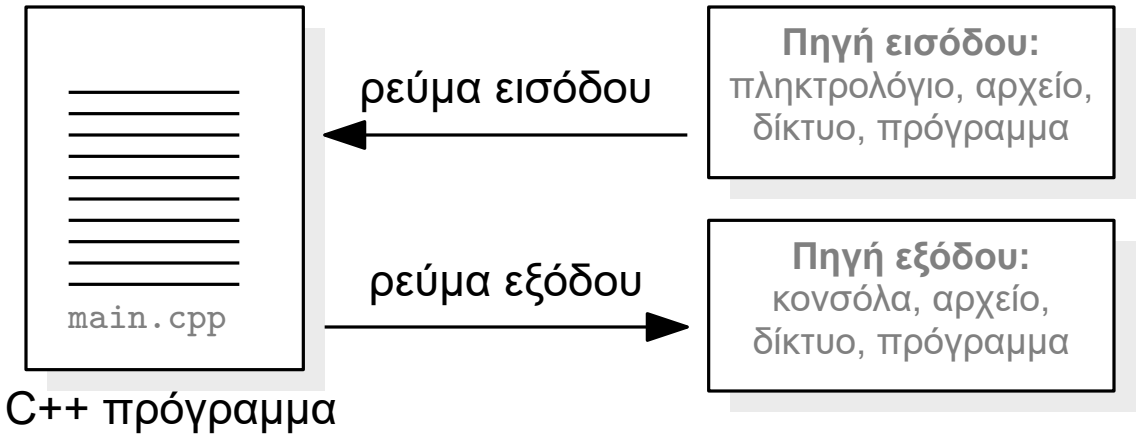
Εισαγωγή

- **Ρεύματα εισόδου/εξόδου** [I/O streams]: Πρόκειται για ακολουθίες χαρακτήρων
  - **Διαδραστικά** - `iostream`:

    `istream` - ρεύμα εισόδου, το οποίο συσχετίζεται με το πληκτρολόγιο (`cin`)
    `ostream` - ρεύμα εξόδου, το οποίο συσχετίζεται με την οθόνη (`cout`)

  - **Αρχεία** - `fstream`:

    `ifstream` - ρεύμα εισόδου, το οποίο συσχετίζεται με αρχείο
    `ofstream` - ρεύμα εξόδου, το οποίο συσχετίζεται με αρχείο



Μέρος 1°:

Απλά ρεύματα εισόδου - εξόδου

Απλά ρεύματα εισόδου - εξόδου

- Η βιβλιοθήκη `iostream` περιέχει έτοιμα αντικείμενα για τον χειρισμό διαδραστικών ρευμάτων εισόδου - εξόδου
  - `cin` Αντικείμενο τύπου `istream` που παρέχει σύνδεση με το πληκτρολόγιο
  - `cout` Αντικείμενο τύπου `ostream` που παρέχει σύνδεση με την οθόνη
  - `cerr` Αντικείμενο τύπου `ostream` που παρέχει σύνδεση με τη ροή σφαλμάτων

- Η πραγματοποίηση λειτουργιών εισόδου και εξόδου γίνεται στέλνοντας μηνύματα σε ένα από αυτά τα αντικείμενα.
  - Για είσοδο από το πληκτρολόγιο χρησιμοποιούμε τον τελεστή εξαγωγής (>>) στο αντικείμενο `cin`
  - Για έξοδο στην οθόνη χρησιμοποιούμε τον τελεστή εισαγωγής (<<) στο αντικείμενο `cout`
  - Για έξοδο σφάλματος χρησιμοποιούμε τον τελεστή εισαγωγής (<<) στο αντικείμενο `cerr`

Το αντικείμενο `cout`

- Το αντικείμενο `cout` έχει μεθόδους που καθορίζουν τον τρόπο εκτύπωσης των αντικειμένων.
- `void width(int w);`
  ορίζει το εύρος εκτύπωσης (για την επόμενη `cout` εντολή)
- `void setf(flags);`
  τροποποιεί μέσω προκαθορισμένων σημάνσεων την έξοδο
  - `ios::dec` εμφάνιση των ints ως δεκαδικών
  - `ios::oct` εμφάνιση των ints ως οκταδικών
  - `ios::hex` εμφάνιση των ints ως δεκαεξαδικών
  - `ios::showpos` εμφάνιση του συμβόλου +
  - `ios::left` στοίχιση αριστερά
  - `ios::right` στοίχιση δεξιά
- `void unsetf(flags);` αφαιρεί ισχύουσες σημάνσεις
  π.χ., `cout.unsetf(ios::dec | ios::oct | ios::hex);`

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

int main() {
  int A[2][2] = { {11, 5}, {2, 21}};
  // Print A in demical with + sign
  cout.setf(ios::showpos);
  for(int i=0; i<2; i++) {
    for (int j=0; j<2; j++) {
      cout.width(5); cout << A[i][j];
    }
    cout << endl;
  }
  // Print A in octal and aligned left
  cout.unsetf(ios::dec|ios::oct|ios::hex);
  cout.setf(ios::oct);
  cout.setf(ios::left);
  for(int i=0; i<2; i++) {
    for (int j=0; j<2; j++) {
      cout.width(5); cout << A[i][j
    }
    cout << endl;
  }
}
```

Part05/cout.cpp

Παράδειγμα

- Να γραφεί πρόγραμμα το οποίο να εκτυπώνει στο τερματικό ένα δεντράκι ως εξής:

```
      **
     ****
    ******
   ********
  **********
 ************
     ****
     ****
```

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

void repchar(char c, int n);
const int HEIGHT = 8;

int main() {
  for (int i=1; i<=HEIGHT; i++) {
    repchar(' ', HEIGHT-i);
    repchar('*', 2*i);
    cout << endl;
  }
  for (int i=1; i<=HEIGHT/4; i++) {
    repchar(' ', HEIGHT-2);
    repchar('*', 4);
    cout << endl;
  }
}

void repchar(char c, int n) {
  cout.width(n);
  cout.fill(c);
  cout << "";
}
```

Part05/printtree.cpp

Μέρος 2°:

Υπερφόρτωση τελεστών stream

Υπερφόρτωση αριθμητικών και συγκριτικών τελεστών

- Στην 4η εργαστηριακή άσκηση είδαμε πως γίνεται η υπερφόρτωση των αριθμητικών και συγκριτικών τελεστών:
  - `+, -, *, /`
  - `<, >, ==, !=, ≤, ≥`
- Όμοια γίνεται η υπερφόρτωση των τελεστών προσαύξησης:
  - `+=, -=, *=, /=`
- Σύνταξη:

```
return-type operatorop (arguments-list)
{
    //function body
}
```

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

class Fraction {
private:
  int numerator, denominator;
public:
  void setNumerator (int n);
  void setDenominator (int d);
  int getNumerator() const;
  int getDenominator() const;
  ...
};

Fraction operator+(Fraction f, Fraction g) {
  int n1 = f.getNumerator();
  int n2 = g.getNumerator();
  int d1 = f.getDenominator();
  int d2 = g.getDenominator();
  Fraction r;
  r.setNumerator(n1 * d2 + n2 * d1);
  r.setDenominator(d1 * d2);
  return r;
}
```

Part02/Fraction.cpp

Υπερφόρτωση αριθμητικών και συγκριτικών τελεστών

- Στην 4η εργαστηριακή άσκηση είδαμε πως γίνεται η υπερφόρτωση των αριθμητικών και συγκριτικών τελεστών:
  - `+, -, *, /`
  - `<, >, ==, !=, ≤, ≥`
- Όμοια γίνεται η υπερφόρτωση των τελεστών προσαύξησης:
  - `+=, -=, *=, /=`
- Σύνταξη:

```
return-type operatorop (arguments-list)
{
    //function body
}
```

Ο τύπος της τιμής που επιστρέφεται

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

class Fraction {
private:
  int numerator, denominator;
public:
  void setNumerator (int n);
  void setDenominator (int d);
  int getNumerator() const;
  int getDenominator() const;
  ...
};

Fraction operator+(Fraction f, Fraction g) {
  int n1 = f.getNumerator();
  int n2 = g.getNumerator();
  int d1 = f.getDenominator();
  int d2 = g.getDenominator();
  Fraction r;
  r.setNumerator(n1 * d2 + n2 * d1);
  r.setDenominator(d1 * d2);
  return r;
}
```

Part02/Fraction.cpp

Υπερφόρτωση αριθμητικών και συγκριτικών τελεστών

- Στην 4η εργαστηριακή άσκηση είδαμε πως γίνεται η υπερφόρτωση των αριθμητικών και συγκριτικών τελεστών:
  - `+, -, *, /`
  - `<, >, ==, !=, ≤, ≥`
- Όμοια γίνεται η υπερφόρτωση των τελεστών προσαύξησης:
  - `+=, -=, *=, /=`
- Σύνταξη:

```
return-type operatorop (arguments-list)
{
    //function body
}
```

Το όνομα της συνάρτησης, όπου `op` είναι ο τελεστής που υπερφορτώνεται

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

class Fraction {
private:
  int numerator, denominator;
public:
  void setNumerator (int n);
  void setDenominator (int d);
  int getNumerator() const;
  int getDenominator() const;
  ...
};

Fraction operator+(Fraction f, Fraction g) {
  int n1 = f.getNumerator();
  int n2 = g.getNumerator();
  int d1 = f.getDenominator();
  int d2 = g.getDenominator();
  Fraction r;
  r.setNumerator(n1 * d2 + n2 * d1);
  r.setDenominator(d1 * d2);
  return r;
}
```

Part02/Fraction.cpp

Υπερφόρτωση τελεστών stream

- Οι τελεστές εισαγωγής (<<) και εξαγωγής (>>) δεν λειτουργούν αυτόματα για αντικείμενα νέων κλάσεων.

- Η υπερφόρτωση τους γίνεται υλοποιώντας τις συναρτήσεις:

```
ostream& operator<<(ostream &os, type-name &o)
istream& operator>>(istream &is, type-name &o)
```

- Γιατί η πρώτη παράμετρος είναι `o/istream`;

  Το `cout` είναι τύπου `ostream`. Το `cin` είναι τύπου `istream`. Όταν γράφουμε:

  `cout << f;` ή `cin >> f;`

  η πρώτη παράμετρος στην υπερφόρτωση είναι η `cout` και η `cin` αντίστοιχα.

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

class Fraction {
private:
  int numerator, denominator;
public:
  void setNumerator (int n);
  void setDenominator (int d);
  int getNumerator() const;
  int getDenominator() const;
  ...
};
ostream& operator<<(ostream &os, Fraction &f)
{ os << f.getNumerator() << "/"
     << f.getDenominator();
  return os;
}
istream& operator>>(istream &is, Fraction &f)
{ int x, y;
  is >> x >> y;
  f.setNumerator(x); f.setDenominator(y);
  return is;
}
```

Part05/Fraction.cpp

Υπερφόρτωση τελεστών stream

- Οι τελεστές εισαγωγής (<<) και εξαγωγής (>>) δεν λειτουργούν αυτόματα για αντικείμενα νέων κλάσεων.
- Η υπερφόρτωση τους γίνεται υλοποιώντας τις συναρτήσεις:

```
ostream& operator<<(ostream &os, type-name &o)
istream& operator>>(istream &is, type-name &o)
```

- Γιατί η συνάρτηση υπερφόρτωσης δεν είναι `void`;

  Για την υποστήριξη διαδοχικών κλήσεων.

  Π.χ., `cout << f1 << " " << f2;`
  `cin >> f1 >> f2;`

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

class Fraction {
private:
  int numerator, denominator;
public:
  void setNumerator (int n);
  void setDenominator (int d);
  int getNumerator() const;
  int getDenominator() const;
  ...
};
ostream& operator<<(ostream &os, Fraction &f)
{ os << f.getNumerator() << "/"
     << f.getDenominator();
  return os;
}
istream& operator>>(istream &is, Fraction &f)
{ int x, y;
  is >> x >> y;
  f.setNumerator(x); f.setDenominator(y);
  return is;
}
```

Part05/Fraction.cpp

Μέρος 3°:
Αρχεία εισόδου - εξόδου

Ρεύματα εισόδου - εξόδου αρχείων

- Τα περισσότερα προγράμματα πρέπει να αποθηκεύουν δεδομένα σε αρχεία (στο δίσκο) και να τα διαβάσουν ξανά.

- Για το σκοπό αυτό, η C++ υποστηρίζει ένα σύνολο κλάσεων:
  - ifstream Υποστήριξη λειτουργιών ανάγνωσης
  - ofstream Υποστήριξη λειτουργιών εγγραφής
  - fstream Υποστήριξη λειτουργιών εγγραφής/ανάγνωσης

- Τα αντικείμενα αυτών των κλάσεων μπορούν να συσχετιστούν με αρχεία στο δίσκο.
  - Οι λειτουργίες ανάγνωσης και εγγραφής υποστηρίζονται μέσω αντίστοιχων μεθόδων.

Γενικά βήματα

1. Συμπεριλάβετε το αρχείο κεφαλίδας fstream στο πρόγραμμα
   - `#include <fstream>`
2. Δηλώστε μεταβλητές ροής αρχείων
   - `ofstream ofs;`
   - `ifstream ifs;`
3. Συσχετίστε κάθε μεταβλητή ροής με ένα αρχείο εισόδου/εξόδου ανοίγοντας το αρχείο
   - `ofs.open("path_to_file");`
   - `ifs.open("path_to_file");`
4. Χρησιμοποιήστε τις μεταβλητές ροής με τους τελεστές εισαγωγής (<<) ή εξαγωγής (>>) για έξοδο και είσοδο, αντίστοιχα.
   - `ofs << "output_goes_here";`
   - `ifs >> str;`
5. Κλείστε το αρχείο
   - `ofs.close();`
   - `ifs.close();`

Έξοδος

- Το άνοιγμα ενός αρχείου εξόδου συσχετίζει μια μεταβλητή ροής αρχείου του προγράμματος με ένα φυσικό αρχείο στο δίσκο.
- Αν το αρχείο εξόδου δεν υπάρχει, τότε δημιουργείται αυτόματα.
- Αν δεν καθορίσουμε τη διαδρομή προς το αρχείο, τότε το αρχείο θα πρέπει να είναι στον ίδιο φάκελο με το εκτελέσιμο.
- Μην ξεχάσετε στο τέλος να κλείσετε το αρχείο.
- https://cplusplus.com/reference/fstream/ofstream/

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

int main() {
  // Declare output file stream variable
  ofstream ofs;

  // Open the file fout.txt
  ofs.open("fout.txt");

  // Output text to the file
  ofs << "Follow your heart!" << endl;
  ofs << "But, of course..."  << endl;
  ofs << "take your brain with you.";

  // Close the file
  ofs.close();

  return 0;
}
```

Part05/oline.cpp

Έξοδος: Παράθεση περιεχομένου

- Το άνοιγμα ενός αρχείου εξόδου υποθέτει εξ'ορισμού ένα κενό (σ.σ., άδειο) αρχείο.
- Αν το αρχείο εξόδου υπάρχει στο δίσκο, τότε το περιεχόμενο του χάνεται.
- Για την παράθεση περιεχομένου, το άνοιγμα του αρχείου γίνεται ως εξής.
  - `ofstream ofs;`
    `ofs.open("file-to-append.txt", ios::app);`
- Αν το αρχείο δεν υπάρχει, τότε δημιουργείται. Αλλιώς, γίνεται παράθεση περιεχομένου.

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

int main() {
  // Declare output file stream variable
  ofstream ofs;

  // Open the file fout.txt
  ofs.open("fout.txt");

  // Output text to the file and close
  ofs << "Follow your heart!" << endl;
  ofs.close();

  // Append to file fout.txt and close
  ofs.open("fout.txt", ios::app);
  ofs << "Of course, but..." << endl;
  ofs << "take your brain with you.";
  ofs.close();

  return 0;
}
```

Part05/oappend.cpp

Τρόποι ανοίγματος αρχείων

- Διαφορετικοί τρόποι ανοίγματος αρχείου:
  - `ios::in` Ανοίγει ένα αρχείο εισόδου (για ανάγνωση)
  - `ios::out` Ανοίγει ένα αρχείο εξόδου (για εγγραφή)
  - `ios::app` Ανοίγει ένα αρχείο εξόδου (για παράθεση)
  - `ios::ate` Ανοίγει και μεταβαίνει στο τέλος του αρχείου
  - `ios::nocreate` Ανοίγει ένα αρχείο μόνο αν υπάρχει ήδη (διαφορετικά αποτυγχάνει)
  - `ios::noreplace` Ανοίγει ένα αρχείο μόνο αν δεν υπάρχει (διαφορετικά αποτυγχάνει)
  - `ios::trunc` Ανοίγει ένα αρχείο και διαγράφει το παλιό αρχείο, εάν υπάρχει
  - `ios::binary` Ανοίγει ένα αρχείο σε δυαδική μορφή (η προεπιλογή είναι μορφή κειμένου)
- Οι παραπάνω σημάνσεις μπορούν και να συνδυαστούν (με το λογικό ή)
  - Π.χ., άνοιξε ένα υπάρχον αρχείο και πρόσθεσε σε αυτό:

```
ofstream ofs("file.txt", ios::app | ios::nocreate);
```

Έξοδος → Ανά χαρακτήρα

- Η έξοδος μπορεί να γίνει και ανά χαρακτήρα.
- Στην περίπτωση αυτή, χρησιμοποιείται η μέθοδος `put(char c)` της μεταβλητής ροής αρχείου εξόδου του προγράμματος.
- **Παρατήρηση:** Το αρχείο εξόδου μπορεί να ορισθεί κατά τη δήλωση του ρεύματος εξόδου ως παράμετρος του κατασκευαστή.
  - `ofstream ofs("file-to-open.txt");`

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

int main() {
  string str = "Time is a great teacher";
  str += " but it kills all its pupils.";

  // An alternative way to open a file
  // through an output file stream as
  // supported by the constructor
  ofstream ofs("fout.txt");

  // Write each character to file
  for (int j = 0; j < str.size(); j++) {
    ofs.put(str[j]);
  }

  cout << "File written" << endl;
  return 0;
}
```

Part05/ochar.cpp

Είσοδος

- Το άνοιγμα ενός αρχείου εισόδου συσχετίζει μια μεταβλητή ροής αρχείου του προγράμματος με ένα φυσικό αρχείο στο δίσκο.
- Πριν αποκτήσετε πρόσβαση σε αυτό, επικυρώστε ότι το αρχείο υπάρχει και έχει ανοίξει.
- Μην ξεχάσετε στο τέλος να κλείσετε το αρχείο.
- https://cplusplus.com/reference/fstream/ifstream/

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

int main() {
  // Declare input file stream variable
  ifstream ifs;

  // Open the file fout.txt
  ifs.open("fout.txt");

  // Do until the end of file
  string line;
  while (ifs.is_open() && !ifs.eof()) {
    getline(ifs, line);
    cout << line << endl;
  }

  // Close the file
  ifs.close();

  return 0;
}
```

Part05/iline.cpp

Είσοδος → Ανά χαρακτήρα

- Η είσοδος μπορεί να γίνει και ανά χαρακτήρα.
- Στην περίπτωση αυτή, χρησιμοποιείται η μέθοδος `get(char c)` της μεταβλητής ροής αρχείου εισόδου του προγράμματος.

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

int main() {
  // Declare input file stream variable
  ifstream ifs;

  // Open the file fout.txt
  ifs.open("fout.txt");

  // Do until the end of file or error
  char ch;
  while (ifs.is_open() && !ifs.eof()) {
    ifs.get(ch);
    cout << ch;
  }

  // Close the file
  ifs.close();

  return 0;
}
```

Part05/ichar.cpp

Μέρος 4°:

Εφαρμογή: Αρχεία CSV

Αρχεία CSV

- Ένα αρχείο τιμών διαχωρισμένων με κόμματα [CSV; Comma Separated Values] είναι απλό αρχείο κειμένου που περιέχει μια λίστα δεδομένων διαχωρισμένα με κόμμα.

- Παράδειγμα:

```
Name,Email,Phone Number,Address
Bob Smith,bob@example.com,123-456-7890,123 SF
Alice Jones,alice@example.com,098-765-4321,321 LA
```

- Αυτά τα αρχεία χρησιμοποιούνται συχνά για την ανταλλαγή δεδομένων μεταξύ διαφορετικών εφαρμογών.
  - Π.χ., σχεδόν όλες οι σύγχρονες βάσεις δεδομένων υποστηρίζουν λειτουργίες εισαγωγής και εξαγωγής δεδομένων σε CSV αρχεία.

Πίνακας σε CSV αρχείο

- Μια συνήθης διεργασία είναι η αποθήκευση των τιμών ενός πίνακα σε αρχείο CSV.

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

int main() {
  string data[3][3] = {
   {"Name", "Email", "Phone"},
   {"Bob", "bob@msn.com", "12345678"},
   {"Alice", "alice@msn.com", "1987654"}
  };

  ofstream ofs;
  ofs.open("data.csv");

  for (int i=0; i<3; i++) {
    for (int j=0; j<3; j++) {
      ofs << data[i][j] << ",";
    }
    ofs << endl;
  }
  ofs.close();
  return 0;
}
```

Part05/ocsv.cpp

CSV αρχείο σε πίνακα

- Για την ανάγνωση των δεδομένων ενός αρχείου CSV ιδιαίτερα χρήσιμα είναι τα ρεύματα συμβολοσειρών `stringstream`.

- https://cplusplus.com/reference/sstream/stringstream/

```cpp
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
using namespace std;

const int ROWS = 4, COLS = 3;

int main() {
  int i=0, j=0;
  string data[ROWS][COLS];
  ifstream ifs("data.csv");

  string line, cell;       //do till eof
  while (ifs.is_open() && !ifs.eof()) {
    getline(ifs, line);

    stringstream ss(line); //stream it
    while (getline(ss, cell, ','))
      data[i][j++ % COLS] = cell;
    i++;
  }
  ifs.close();
}
```

Part05/icsv.cpp

Επιπλέον υλικό

- Κεφάλαιο 12:
  R. Lafore, Αντικειμενοστρεφής προγραμματισμός με τη C++,
  ISBN: 960-209-904-6, εκδόσεις Κλειδάριθμος, 2006.