Δομές Δεδομένων και Αλγόριθμοι - Εργαστήριο 1
Βασικές έννοιες στη C και στη C++

Τ.Ε.Ι. Ηπείρου, Τμήμα Μηχανικών Πληροφορικής Τ.Ε.
Χρήστος Γκόγκος - Αναπληρωτής Καθηγητής

1 Εισαγωγή

Στο πρώτο αυτό εργαστήριο θα επιχειρηθεί μια παρουσίαση των βασικών γνώσεων που απαιτούνται έτσι ώστε να είναι δυνατή η κατανόηση των εργαστηρίων που ακολουθούν. Ειδικότερα, θα γίνει αναφορά σε δείκτες, στη δυναμική δέσμευση και αποδέσμευση μνήμης, στο πέρασμα παραμέτρων σε συναρτήσεις, στους πίνακες, στις δομές, στα αντικείμενα και τέλος στην ανάγνωση και εγγραφή σε αρχεία. Θα παρουσιαστούν λυμένα παραδείγματα καθώς και εκφωνήσεις ασκήσεων προς επίλυση. Ο κώδικας όλων των παραδειγμάτων βρίσκεται στο https://github.com/chgogos/ceteiep_dsa. Αναλυτικότερη παρουσίαση των ανωτέρω θεμάτων γίνεται στα ελεύθερα διαθέσιμα βιβλία [1], [2], [3], [4] που παρατίθενται ως αναφορές στο τέλος του κειμένου του εργαστηρίου.

2 Δείκτες

Κάθε θέση μνήμης στην οποία μπορούν να αποθηκευτούν δεδομένα βρίσκεται σε μια διεύθυνση μνήμης. Η δε μνήμη του υπολογιστή αποτελείται από ένα συνεχόμενο χώρο διευθύνσεων. Αν μια μεταβλητή δηλωθεί ως τύπου int * τότε η τιμή που θα λάβει ερμηνεύεται ως μια διεύθυνση που δείχνει σε μια θέση μνήμης η οποία περιέχει έναν ακέραιο. Από την άλλη μεριά το σύμβολο & επιτρέπει τη λήψη της διεύθυνσης μιας μεταβλητής. Στον ακόλουθο κώδικα δηλώνονται 2 ακέραιες μεταβλητές (a και b) και ένας δείκτης (p) σε ακέραια τιμή. Ο δείκτης p λαμβάνει ως τιμή τη διεύθυνση της μεταβλητής a. Στη συνέχεια, οι μεταβλητές a και b λαμβάνουν τιμές μέσω του δείκτη p. Για να συμβεί αυτό γίνεται έμμεση αναφορά ή αλλιώς αποαναφορά (dereference) του δείκτη με το *p. Συνεπώς, το *p αντιστοιχεί στο περιεχόμενο της διεύθυνσης μνήμης που έχει ο δείκτης p.

```
1  #include <cstdio>
2
3  int main(int argc, char **argv) {
4    int a, b;
5    int *p;
6    p = &a;
7    *p = 5;
8    b = *p + 1;
9    printf("variable a=%d address=%p\n", a, &a);
10   printf("variable b=%d address=%p\n", b, &b);
11   printf("pointer p=%p *p=%d\n", p, *p);
12   return 0;
13 }
```

Κώδικας 1: Παράδειγμα με δείκτες (lab01_01.cpp)

```
1  variable a=5 address=000000000022fe44
2  variable b=6 address=000000000022fe40
3  pointer p=000000000022fe44 *p=5
```

Ένα συνηθισμένο λάθος με δείκτες παρουσιάζεται όταν γίνεται dereference ενός δείκτη (δηλαδή, δεδομένου ενός δείκτη p όταν χρησιμοποιείται το *p) χωρίς ο δείκτης να έχει αρχικοποιηθεί πρώτα δείχνοντας σε μια έγκυρη θέση μνήμης. Σε αυτή την περίπτωση το πρόγραμμα καταρρέει.

```cpp
#include <iostream>

using namespace std;

int main(int argc, char **argv) {
  int *p;
  *p = 2;
  cout << *p << endl;
  return 0;
}
```

Κώδικας 2: Λανθασμένη χρήση δείκτη (lab01_02.cpp)

```
segmentation fault
```

Αν η μεταγλώττιση του κώδικα γίνει με το flag -Wall τότε θα εμφανιστεί μήνυμα που θα προειδοποιεί για τη λάθος χρήση του δείκτη.

```
g++ -Wall lab01_02.cpp -o lab01_02
lab01_02.cpp: In function 'int main(int, char**)':
lab01_02.cpp:8:11: warning: 'p' is used uninitialized in this function [-Wuninitialized]
     *p = 2;
          ^
```

3 Κλήση με τιμή και κλήση με αναφορά

Οι δείκτες μπορούν να χρησιμοποιηθούν έτσι ώστε να επιτευχθεί, εφόσον απαιτείται, κλήση με αναφορά (call by reference) στις παραμέτρους μιας συνάρτησης και όχι κλήση με τιμή (call by value) που είναι ο προκαθορισμένος τρόπος κλήσης συναρτήσεων. Στο παράδειγμα που ακολουθεί η συνάρτηση swap (σε αντίθεση με τη συνάρτηση swap_impotent) επιτυγχάνει την αντιμετάθεση των δύο μεταβλητών που δέχεται ως ορίσματα καθώς χρησιμοποιεί δείκτες που αναφέρονται στις ίδιες τις μεταβλητές του κυρίου προγράμματος και όχι σε αντίγραφά τους.

```cpp
#include <cstdio>

void swap_not_ok(int x, int y) {
  int temp = x;
  x = y;
  y = temp;
}

void swap(int *x, int *y) {
  int temp = *x;
  *x = *y;
  *y = temp;
}

int main(int argc, char **argv) {
  int a = 5, b = 7;
  printf("BEFORE a=%d b=%d\n", a, b);
  swap_not_ok(a, b);
  printf("AFTER a=%d b=%d\n", a, b);
  printf("BEFORE a=%d b=%d\n", a, b);
  swap(&a, &b);
  printf("AFTER a=%d b=%d\n", a, b);
  return 0;
}
```

Κώδικας 3: Αντιμετάθεση μεταβλητών με δείκτες (lab01_03.cpp)

```
BEFORE a=5 b=7
AFTER a=5 b=7
BEFORE a=5 b=7
AFTER a=7 b=5
```

Η γλώσσα C++ προκειμένου να απλοποιήσει την κλήση με αναφορά εισήγαγε την έννοια των ψευδωνύμων (aliases). Τοποθετώντας στη δήλωση μιας παραμέτρου συνάρτησης το σύμβολο & η παράμετρος λειτουργεί ως ψευδώνυμο για τη μεταβλητή που περνά στην αντίστοιχη θέση. Η συγκεκριμένη συμπεριφορά παρουσιάζεται στον ακόλουθο κώδικα.

```
#include <cstdio>

void swap_cpp(int &x, int &y) {
   int temp = x;
   x = y;
   y = temp;
}

int main(int argc, char **argv) {
   int a = 5, b = 7;
   printf("BEFORE a=%d b=%d\n", a, b);
   swap_cpp(a, b);
   printf("AFTER a=%d b=%d\n", a, b);
   return 0;
}
```

Κώδικας 4: Αντιμετάθεση μεταβλητών με αναφορές (lab01_04.cpp)

```
BEFORE a=5 b=7
AFTER a=7 b=5
```

4 Πίνακες

Ένας πίνακας είναι μια συλλογή από στοιχεία του ίδιου τύπου καθένα από τα οποία μπορεί να αναγνωριστεί από την τιμή ενός ακεραίου δείκτη (index). Το γεγονός αυτό επιτρέπει την τυχαία προσπέλαση (random access) στα στοιχεία του πίνακα. Οι δείκτες των πινάκων ξεκινούν από το μηδέν.

4.1 Μονοδιάστατοι πίνακες

Οι μονοδιάστατοι πίνακες είναι η πλέον απλή δομή δεδομένων. Η αναφορά στα στοιχεία του πίνακα γίνεται συνήθως με μια δομή επανάληψης (π.χ. for). Στο ακόλουθο παράδειγμα δύο μονοδιάστατοι πίνακες αρχικοποιούνται κατά τη δήλωσή τους και εν συνεχεία υπολογίζεται το εσωτερικό γινόμενό τους δηλαδή το άθροισμα των γινομένων των στοιχείων των πινάκων που βρίσκονται στην ίδια θέση.

```
#include <cstdio>

int main(int argc, char **argv) {
   double a[] = {3.2, 4.1, 7.3};
   double b[] = {6.0, .1, −5.3};
   double sum = .0;
   for (int i = 0; i < 3; i++)
      sum += a[i] * b[i];
   printf("inner product %.2f\n", sum);
   return 0;
}
```

Κώδικας 5: Υπολογισμός εσωτερικού γινομένου δύο πινάκων (lab01_05.cpp)

```
inner product −19.08
```

4.2 Δυναμικοί πίνακες

Δυναμικοί πίνακες χρησιμοποιούνται όταν το μέγεθος του πίνακα πρέπει να αλλάζει κατά τη διάρκεια εκτέλεσης του προγράμματος και συνεπώς δεν μπορεί να ορισθεί κατά τη μεταγλώττιση. Πριν χρησιμοποιηθεί ένας δυναμικός πίνακας θα πρέπει δεσμευτούν οι απαιτούμενες θέσεις μνήμης. Επίσης, θα πρέπει να απελευθερωθεί ο χώρος που καταλαμβάνει όταν πλέον δεν χρησιμοποιείται. Στο ακόλουθο παράδειγμα ο χρήστης εισάγει το μέγεθος ενός μονοδιάστατου πίνακα και ο απαιτούμενος χώρος δεσμεύεται κατά την εκτέλεση του κώδικα. Στη συνέχεια ο πίνακας γεμίζει με τυχαίες ακέραιες τιμές στο διάστημα [1,100]. Παρουσιάζονται δύο εκδόσεις του κώδικα, μια που χρησιμοποιεί τις συναρτήσεις malloc και free της γλώσσας C και μια που χρησιμοποιεί τις εντολές new και delete της C++ για τη δέσμευση και την αποδέσμευση μνήμης. Επιπλέον, χρησιμοποιείται διαφορετικός τρόπος για τη δημιουργία των τυχαίων τιμών στα δύο προγράμματα.

```c
#include <cstdio>
#include <cstdlib>
#include <ctime>

int main(int argc, char **argv) {
   int n;
   printf("Enter the size of the vector: ");
   fflush(stdout);
   scanf("%d", &n);
   int *a = (int *)malloc(sizeof(int) * n);
   // srand(time(NULL));
   srand(1821);

   for (int i = 0; i < n; i++)
      a[i] = (rand() % 100) + 1;

   for (int i = 0; i < n; i++)
      printf("%d ", a[i]);

   free(a);
   return 0;
}
```

Κώδικας 6: Δημιουργία δυναμικού πίνακα με συναρτήσεις της C (lab01_06.cpp)

```
Enter the size of the vector: 10
86 72 71 16 32 49 75 35 1 70
```

```cpp
#include <iostream>
#include <random>
using namespace std;
int main(int argc, char **argv) {
   mt19937 mt(1821);
   uniform_int_distribution<int> dist(1, 100);
   int n;
   cout << "Enter the size of the vector: ";
   cin >> n;
   int *a = new int[n];
   for (int i = 0; i < n; i++)
      a[i] = dist(mt);
   for (int i = 0; i < n; i++)
      cout << a[i] << " ";
   delete[] a;
```

```
16   return 0;
17 }
```

Κώδικας 7: Δημιουργία δυναμικού πίνακα με συναρτήσεις της C++ (lab01_07.cpp)

```
1 Enter the size of the vector: 10
2 73 44 67 66 1 98 52 85 19 24
```

Για να γίνει η μεταγλώττιση του κώδικα 7 θα πρέπει να χρησιμοποιηθεί το flag -std=c++11 όπως φαίνεται στην ακόλουθη εντολή.

```
1 g++ -std=c++11 lab01_07.cpp -o lab01_07
```

4.3 Πίνακας ως παράμετρος συνάρτησης

Ένας πίνακας μπορεί να περάσει ως παράμετρος σε μια συνάρτηση. Συχνά χρειάζεται να περάσουν ως παράμετροι και οι διαστάσεις του πίνακα. Στον ακόλουθο κώδικα η συνάρτηση simple_stats δέχεται ως παράμετρο έναν μονοδιάστατο πίνακα ακεραίων και το πλήθος των στοιχείων του και επιστρέφει μέσω κλήσεων με αναφορά το μέσο όρο, το ελάχιστο και το μέγιστο από όλα τα στοιχεία του πίνακα.

```
1  #include <cstdio>
2
3  void simple_stats(int a[], int N, double *avg, int *min, int *max) {
4    int sum;
5    sum = *min = *max = a[0];
6    for (int i = 1; i < N; i++) {
7      sum += a[i];
8      if (*max > a[i])
9        *max = a[i];
10     if (*min < a[i])
11       *min = a[i];
12   }
13   *avg = (double)sum / (double)N;
14 }
15
16 int main(int argc, char **argv) {
17   int a[] = {3, 2, 9, 5, 1};
18   double avg;
19   int min, max;
20   simple_stats(a, 5, &avg, &min, &max);
21   printf("Average= %.2f min=%d max=%d\n", avg, min, max);
22   return 0;
23 }
```

Κώδικας 8: Δυναμικός πίνακας ως παράμετρος συνάρτησης (lab01_08.cpp)

```
1 Average= 4.00 min=9 max=1
```

4.4 Δισδιάστατοι πίνακες

Ένας δισδιάστατος πίνακας αποτελείται από γραμμές και στήλες και η αναφορά στα στοιχεία του γίνεται με δύο δείκτες από τους οποίους ο πρώτος δείκτης υποδηλώνει τη γραμμή και ο δεύτερος υποδηλώνει τη στήλη του πίνακα. Οι πίνακες είναι ιδιαίτερα σημαντικοί για την εκτέλεση μαθηματικών υπολογισμών (π.χ. πολλαπλασιασμό πινάκων, επίλυση συστημάτων γραμμικών εξισώσεων κ.α.). Στον ακόλουθο κώδικα δίνεται ένα παράδειγμα δήλωσης ενός δισδιάστατου πίνακα 5 x 4 ο οποίος περνά ως παράμετρος στη συνάρτηση sums_row_wise. Η δε συνάρτηση επιστρέφει το άθροισμα κάθε γραμμής του πίνακα.

```cpp
#include <cstdio>

void sums_row_wise(int a[][4], int M, int N, int *row) {
  for (int i = 0; i < M; i++) {
    row[i] = 0;
    for (int j = 0; j < N; j++)
      row[i] += a[i][j];
  }
}

int main(int argc, char **argv) {
  int a[5][4] = {{5, 4, 0, −1},
                 {1, 5, 42, 2},
                 {−3, 7, 8, 2},
                 {7, 312, −56, 6},
                 {19, 45, 6, 5}};
  int row[5];
  sums_row_wise(a, 5, 4, row);
  for (int i = 0; i < 5; i++)
    printf("sum of row %d is %d\n", i, row[i]);
  return 0;
}
```

Κώδικας 9: Δισδιάστατος πίνακας ως παράμετρος συνάρτησης (lab01_09.cpp)

```
sum of row 0 is 8
sum of row 1 is 50
sum of row 2 is 14
sum of row 3 is 269
sum of row 4 is 75
```

4.5 Πολυδιάστατοι πίνακες

Αν και οι μονοδιάστατοι και οι δισδιάστατοι πίνακες χρησιμοποιούνται συχνότερα, υποστηρίζονται και πίνακες μεγαλύτερων διαστάσεων. Στη συνέχεια δίνεται ένα παράδειγμα δήλωσης και αρχικοποίησης ενός τρισδιάστατου πίνακα 3x3x2 και ενός τετραδιάστατου πίνακα 3x3x3x2.

```cpp
int main() {
  int d[3][3][2];
  int e[3][3][2][2];
  for (int i = 0; i < 3; i++)
    for (int j = 0; j < 3; j++)
      for (int k = 0; k < 2; k++) {
        d[i][j][k] = 1;
        for (int l = 0; l < 2; l++)
          e[i][j][k][l] = 1;
      }
}
```

Κώδικας 10: Δήλωση και αρχικοποίηση τρισδιάστατου και τετραδιάστατου πίνακα (lab01_10.cpp)

4.6 Πριονωτοί πίνακες

Εφόσον ένας πολυδιάστατος πίνακας δημιουργείται δυναμικά μπορεί να οριστεί με τέτοιο τρόπο έτσι ώστε η κάθε γραμμή του να μην έχει τον ίδιο αριθμό στοιχείων. Στον ακόλουθο κώδικα δημιουργείται ένας δισδιάστατος πίνακας 5 γραμμών με την πρώτη γραμμή να έχει 1 στοιχείο και κάθε επόμενη γραμμή ένα περισσότερο στοιχείο από την προηγούμενη της.

```cpp
#include <iostream>
using namespace std;
int main(int argc, char **argv) {
   int *a[5];
   for (int i = 0; i < 5; i++) {
      a[i] = new int[i + 1];
      for (int j = 0; j < i + 1; j++)
         a[i][j] = i + j;
   }
   for (int i = 0; i < 5; i++) {
      for (int j = 0; j < i + 1; j++)
         cout << a[i][j] << " ";
      cout << endl;
   }
   for (int i = 0; i < 5; i++)
      delete[] a[i];
   return 0;
}
```

Κώδικας 11: Παράδειγμα πριονωτού πίνακα με 5 γραμμές (lab01_11.cpp)

```
0
1 2
2 3 4
3 4 5 6
4 5 6 7 8
```

5 Δομές

Οι δομές χρησιμοποιούνται όταν απαιτούνται σύνθετοι τύποι δεδομένων οι οποίοι αποτελούνται από επιμέρους στοιχεία. Στο παράδειγμα που ακολουθεί ορίζεται η δομή Book με 3 πεδία. Στη συνέχεια δημιουργούνται 3 μεταβλητές που πρόκειται να αποθηκεύσουν πληροφορίες για ένα βιβλίο η κάθε μια. Η τρίτη μεταβλητή είναι δείκτης προς τη δομή Book και προκειμένου να χρησιμοποιηθεί θα πρέπει πρώτα να δεσμευθεί μνήμη (new) ενώ με τον τερματισμό του προγράμματος θα πρέπει η μνήμη αυτή να επιστραφεί στο σύστημα (delete).

```cpp
#include <iostream>

using namespace std;

typedef struct {
   string title;
   int price;
   bool isHardpack;
} Book;

// struct Book
// {
// string title;
// int price;
// bool isHardpack;
// };

void print_book(Book b) {
   cout << "Title: " << b.title << " Price: " << b.price / 100.0
        << " Hardcover: " << (b.isHardpack ? "YES" : "NO") << endl;
}

int main(int argc, char **argv) {
```

```
24    Book b1, b2, *pb;
25    b1.title = "The SIMPSONS and their mathematical secrets";
26    b1.price = 1899;
27    b1.isHardpack = false;
28    b2 = b1;
29    b2.isHardpack = true;
30    b2.price = 2199;
31    pb = new Book;
32    pb->title = "Bad Science";
33    pb->price = 999;
34    pb->isHardpack = false;
35    print_book(b1);
36    print_book(b2);
37    print_book(*pb);
38    delete pb;
39    return 0;
40 }
```

Κώδικας 12: Μεταβλητές τύπου δομής Book (lab01_12.cpp)

```
1 Title: The SIMPSONS and their mathematical secrets Price: 18.99 Hardcover: NO
2 Title: The SIMPSONS and their mathematical secrets Price: 21.99 Hardcover: YES
3 Title: Bad Science Price: 9.99 Hardcover: NO
```

6 Κλάσεις - Αντικείμενα

Ο αντικειμενοστρεφής προγραμματισμός εντοπίζει τα αντικείμενα που απαρτίζουν την εφαρμογή και τα συνδυάζει προκειμένου να επιτευχθεί η απαιτούμενη λειτουργικότητα. Για κάθε αντικείμενο γράφεται μια κλάση η οποία είναι υπεύθυνη για τη δημιουργία των επιμέρους στιγμιοτύπων (object instances). Κάθε αντικείμενο έχει μεταβλητές και συναρτήσεις οι οποίες μπορεί να είναι είτε ιδιωτικές (private) είτε δημόσιες (public) (είτε προστατευμένες-protected). Τα ιδιωτικά μέλη χρησιμοποιούνται εντός της κλάσης που ορίζει το αντικείμενο ενώ τα δημόσια μπορούν να χρησιμοποιηθούν και από κώδικα εκτός της κλάσης. Στο ακόλουθο παράδειγμα ορίζεται η κλάση Box η οποία έχει 3 ιδιωτικά μέλη (length, width, height) και 1 δημόσιο μέλος, τη συνάρτηση volume. Στη main δημιουργούνται με τη βοήθεια του κατασκευαστή (constructor) δύο αντικείμενα (στιγμιότυπα) της κλάσης Box και καλείται για καθένα από αυτά η δημόσια συνάρτηση μέλος της Box, volume.

```
1  #include <iostream>
2
3  using namespace std;
4
5  class Box {
6  public:
7    // constructor declaration with default values
8    Box(double l = 1, double w = 1, double h = 1);
9    // member function
10   double volume();
11
12 private:
13   // member data
14   double length;
15   double width;
16   double height;
17 };
18
19 // constructor using initializer list
20 Box::Box(double l, double w, double h) : length(l), width(w), height(h) {}
21
22 double Box::volume() { return length * width * height; }
```

```
23
24 int main(int argc, char **argv) {
25   Box b1(10, 5, 10);
26   cout << "The volume is " << b1.volume() << endl;
27   Box *pb = new Box();
28   cout << "The volume is " << pb->volume() << endl;
29   delete pb;
30   return 0;
31 }
```

Κώδικας 13: Παράδειγμα κλάσης Box (lab01_13.cpp)

```
1 The volume is 500
2 The volume is 1
```

7 Αρχεία

Συχνά χρειάζεται να αποθηκεύσουμε δεδομένα σε αρχεία ή να επεξεργαστούμε δεδομένα τα οποία βρίσκονται σε αρχεία. Ο ακόλουθος κώδικας πρώτα δημιουργεί έναν αρχείο με 100 τυχαίους ακεραίους στον τρέχοντα κατάλογο και στη συνέχεια ανοίγει το αρχείο και εμφανίζει τα στοιχεία του.

7.1 Εγγραφή και ανάγνωση δεδομένων από αρχείο με συναρτήσεις της C

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <ctime>
4
5  int main(int argc, char **argv) {
6    FILE *fp = fopen("data_int_100.txt", "w");
7    srand(time(NULL));
8    for (int i = 0; i < 100; i++)
9      fprintf(fp, "%d ", rand() % 1000 + 1);
10   fclose(fp);
11
12   fp = fopen("data_int_100.txt", "r");
13   if (fp == NULL) {
14     printf("error opening file");
15     exit(-1);
16   }
17   int x;
18   while (!feof(fp)) {
19     fscanf(fp, "%d", &x);
20     printf("%d ", x);
21   }
22   fclose(fp);
23   return 0;
24 }
```

Κώδικας 14: Εγγραφή 100 ακέραιων αριθμητικών δεδομένων σε αρχείο και ανάγνωση τους από το ίδιο αρχείο (lab01_14.cpp)

```
1 811 718 632 412 529 957 359 735 498 302 855 265 749 756 336 625 489 870 120 177 ...
```

7.2 Εγγραφή και ανάγνωση δεδομένων από αρχείο με συναρτήσεις της C++

Η C++ έχει προσθέσει νέους τρόπους με τους οποίους μπορεί να γίνει η αλληλεπίδραση με τα αρχεία. Ακολουθεί ένα παράδειγμα εγγραφής και ανάγνωσης δεδομένων από αρχείο με τη χρήση των fstream και sstream.

```cpp
#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>


using namespace std;

int main(int argc, char **argv) {
   int constexpr N = 10;
   fstream filestr;
   string buffer;
   int i = 0;
   string names[N] = {"nikos", "maria", "petros", "sofia", "kostas",
                      "dimitra", "giorgos", "christos", "anna", "apostolis"};
   int grades[N] = {55, 30, 70, 80, 10, 25, 75, 90, 100, 30};
   filestr.open("data_student_struct10.txt", ios::out);
   if (!filestr.is_open()) {
      cerr << "file not found" << std::endl;
      exit(-1);
   }
   for (i = 0; i < N; i++)
      filestr << names[i] << "\t" << grades[i] << endl;
   filestr.close();

   filestr.open("data_student_struct10.txt");
   if (!filestr.is_open()) {
      cerr << "file not found" << std::endl;
      exit(-1);
   }
   string name;
   int grade;
   while (getline(filestr, buffer)) {
      stringstream ss(buffer);
      ss >> name;
      ss >> grade;
      cout << name << " " << setprecision(1) << fixed
           << static_cast<double>(grade) / 10.0 << endl;
   }
   filestr.close();
   return 0;
}
```

Κώδικας 15: Εγγραφή και ανάγνωση αλφαριθμητικών και ακεραίων από αρχείο (lab01_15.cpp) (lab01_15.cpp)

```
nikos 5.5
maria 3.0
petros 7.0
sofia 8.0
kostas 1.0
dimitra 2.5
giorgos 7.5
christos 9.0
anna 10.0
apostolis 3.0
```

8 Παραδείγματα

8.1 Παράδειγμα 1

Γράψτε κώδικα που να δημιουργεί μια δομή με όνομα Point και να έχει ως πεδία 2 double αριθμούς (x και y) που υποδηλώνουν τις συντεταγμένες του σημείου στο καρτεσιανό επίπεδο. Δημιουργήστε έναν πίνακα με όνομα points με 5 σημεία με απευθείας εισαγωγή τιμών για τα ακόλουθα σημεία: (4, 17), (10, 21), (5, 32), (-1, 16), (-4, 7). Γράψτε τον κώδικα που εμφανίζει τα 2 πλησιέστερα σημεία. Ποια είναι τα πλησιέστερα σημεία και ποια η απόσταση μεταξύ τους;

```
#include <climits>
#include <cmath>
#include <cstdio>

struct Point {
   double x;
   double y;
};

int main(int argc, char **argv) {
   struct Point points[5] = {{4, 17}, {10, 21}, {5, 32}, {−1, 16}, {−4, 7}};

   double min = INT_MAX;
   Point a, b;
   for (int i = 0; i < 5; i++)
      for (int j = i + 1; j < 5; j++) {
         Point p1 = points[i];
         Point p2 = points[j];
         double distance = sqrt(pow(p2.x − p1.x, 2.0) + pow(p2.y − p1.y, 2.0));
         if (distance < min) {
            min = distance;
            a = p1;
            b = p2;
         }
      }
   printf("Min %.4f\n", min);
   printf("Point A: (%.4f, %.4f)\n", a.x, a.y);
   printf("Point B: (%.4f, %.4f)\n", b.x, b.y);
   return 0;
}
```

Κώδικας 16: Λύση παραδείγματος 1 (lab01_16.cpp)

```
Min 5.0990
Point A: (4.0000, 17.0000)
Point B: (−1.0000, 16.0000)
```

8.2 Παράδειγμα 2

Με τη γεννήτρια τυχαίων αριθμών mt19937 δημιουργήστε 10000 τυχαίες ακέραιες τιμές στο διάστημα 0 έως 10000 με seed την τιμή 1729. Τοποθετήστε τις τιμές σε ένα δισδιάστατο πίνακα 100 x 100 έτσι ώστε να συμπληρώνονται οι τιμές στον πίνακα κατά σειρές από πάνω προς τα κάτω και από αριστερά προς τα δεξιά. Να υπολογιστεί το άθροισμα της κάθε γραμμής του πίνακα. Ποιος είναι ο αριθμός της γραμμής με το μεγαλύτερο άθροισμα και ποιο είναι αυτό;

```
#include <iostream>
#include <random>

using namespace std;
```

```
5
6  int main(int argc, char **argv) {
7    int seed = 1729;
8    int a[100][100];
9    mt19937 mt(seed);
10   uniform_int_distribution<int> dist(0, 10000);
11   for (int i = 0; i < 100; i++)
12     for (int j = 0; j < 100; j++)
13       a[i][j] = dist(mt);
14   int b[100];
15   int max = 0;
16   for (int i = 0; i < 100; i++) {
17     int sum = 0;
18     for (int j = 0; j < 100; j++)
19       sum += a[i][j];
20     b[i] = sum;
21     if (sum > max)
22       max = sum;
23   }
24   cout << "Maximum row sum: " << max << endl;
25   for (int i = 0; i < 100; i++)
26     if (b[i] == max)
27       cout << "Occurs at row: " << i << endl;
28   return 0;
29 }
```

Κώδικας 17: Λύση παραδείγματος 2 (lab01_17.cpp)

```
1 Maximum row sum: 586609
2 Occurs at row: 40
```

8.3 Παράδειγμα 3

Γράψτε 10000 τυχαίες ακέραιες τιμές στο διάστημα [1,10000] στο αρχείο data_int_10000.txt χρησιμοποιώντας τις συναρτήσεις rand και srand και seed την τιμή 1729. Διαβάστε τις τιμές από το αρχείο. Εντοπίστε τη μεγαλύτερη τιμή στα δεδομένα. Ποιες είναι οι τιμές που εμφανίζονται τις περισσότερες φορές στα δεδομένα;

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <iostream>
4
5
6  using namespace std;
7
8  int main(int argc, char **argv) {
9    const char *fn = "data_int_10000.txt";
10   int N = 10000;
11   srand(1729);
12
13   FILE *fp = fopen(fn, "w");
14   if (fp == NULL) {
15     printf("error opening file");
16     exit(-1);
17   }
18   for (int i = 0; i < N; i++)
19     fprintf(fp, "%d ", rand() % 10000 + 1);
20   fclose(fp);
21
22   fp = fopen(fn, "r");
23   if (fp == NULL) {
```

```
24      printf("error opening file");
25      exit(-1);
26    }
27    int *data = new int[N];
28    int i = 0;
29    while (!feof(fp)) {
30      fscanf(fp, "%d", &data[i]);
31      i++;
32    }
33    fclose(fp);
34    int max = data[0];
35    for (i = 1; i < N; i++)
36      if (data[i] > max)
37        max = data[i];
38    printf("Maximum value %d\n", max);
39    int *freq = new int[max + 1];
40    for (i = 0; i < max + 1; i++)
41      freq[i] = 0;
42    for (i = 0; i < N; i++)
43      freq[data[i]]++;
44    int max2 = 0;
45    for (i = 0; i < max + 1; i++)
46      if (freq[i] > max2)
47        max2 = freq[i];
48    for (i = 0; i < max + 1; i++)
49      if (freq[i] == max2)
50        printf("Value %d exists %d times\n", i, max2);
51    delete[] freq;
52    return 0;
53  }
```

Κώδικας 18: Λύση παραδείγματος 3 (lab01_18.cpp)

```
1  Maximum value 9999
2  Value 885 exists 6 times
3  Value 1038 exists 6 times
4  Value 3393 exists 6 times
5  Value 4771 exists 6 times
6  Value 5482 exists 6 times
7  Value 8722 exists 6 times
8  Value 9501 exists 6 times
```

8.4 Παράδειγμα 4

Γράψτε κώδικα που να δημιουργεί μια δομή με όνομα student (σπουδαστής) και να έχει ως πεδία το name (όνομα) τύπου string και το grade (βαθμός) τύπου int. Διαβάστε τα περιεχόμενα του αρχείου που έχει δημιουργηθεί με τον κώδικα 15 (data_student_struct10.txt) και τοποθετήστε τα σε κατάλληλο πίνακα. Βρείτε τα ονόματα και το μέσο όρο βαθμολογίας των σπουδαστών με βαθμό άνω του μέσου όρου όλων των σπουδαστών. Θεωρείστε ότι οι βαθμοί έχουν αποθηκευτεί στο αρχείο data_student_struct10.txt ως ακέραιοι αριθμοί από το 0 μέχρι και το 100, αλλά η εμφάνισή τους θα πρέπει να γίνεται εφόσον πρώτα διαιρεθούν με το 10. Δηλαδή, ο βαθμός 55 αντιστοιχεί στο βαθμό 5.5.

```
1  #include <fstream>
2  #include <iostream>
3  #include <sstream>
4
5
6  using namespace std;
7
8  struct student {
```

```
9     string name;
10    int grade;
11  };
12
13  int main(int argc, char **argv) {
14    int N = 10, i = 0;
15    student students[10];
16    const char *fn = "data_student_struct10.txt";
17    fstream filestr;
18    string buffer;
19    filestr.open(fn);
20    if (!filestr.is_open()) {
21      cerr << "File not found" << std::endl;
22      exit(-1);
23    }
24    while (getline(filestr, buffer)) {
25      stringstream ss(buffer);
26      ss >> students[i].name;
27      ss >> students[i].grade;
28      i++;
29    }
30    filestr.close();
31    double sum = 0.0;
32    for (i = 0; i < N; i++)
33      sum += students[i].grade;
34    double avg = sum / N;
35    cout << "Average grade =" << avg / 10.0 << endl;
36
37    double sum2 = 0.0;
38    int c = 0;
39    for (i = 0; i < N; i++)
40      if (students[i].grade > avg) {
41        cout << students[i].name << " grade = " << students[i].grade / 10.0
42             << endl;
43        sum2 += students[i].grade;
44        c++;
45      }
46    double avg2 = sum2 / c;
47    cout << "Average grade for students having grade above the average grade = "
48         << avg2 / 10.0 << endl;
49    return 0;
50  }
```

Κώδικας 19: Λύση παραδείγματος 4 (lab01_19.cpp)

```
1  Average grade =5.65
2  petros grade = 7
3  sofia grade = 8
4  giorgos grade = 7.5
5  christos grade = 9
6  anna grade = 10
7  Average grade for students having grade above the average grade = 8.3
```

9 Ασκήσεις

1. Γράψτε μια συνάρτηση που να δέχεται έναν πίνακα ακεραίων και το μέγεθός του και να επιστρέφει το μέσο όρο των τιμών καθώς και το πλήθος των τιμών που απέχουν το πολύ 10% από το μέσο όρο. Δοκιμάστε την κλήση της συνάρτησης για έναν πίνακα 100 θέσεων με τυχαίες ακέραιες τιμές στο διάστημα

[1,100] οι οποίες θα δημιουργηθούν με τη χρήση των συναρτήσεων srand() και rand() της C. Χρησιμοποιήστε ως seed για την αρχικοποίηση των τυχαίων τιμών την τιμή 12345.

2. Γράψτε πρόγραμμα που να διαβάζει τα στοιχεία υπαλλήλων (όνομα, μισθό και έτη προϋπηρεσίας) από το αρχείο data_ypallhlos_struct20.txt και να εμφανίζει τα στοιχεία του κάθε υπαλλήλου μέσω μιας συνάρτησης που θα δέχεται ως παράμετρο μια μεταβλητή τύπου δομής υπαλλήλου. Στη συνέχεια να υπολογίζει και να εμφανίζει το ποσό που θα συγκεντρωθεί αν για κάθε υπάλληλο με περισσότερα από 5 έτη προϋπηρεσίας παρακρατηθεί το 5% του μισθού του ενώ για τους υπόλοιπους υπαλλήλους παρακρατηθεί το 7% του μισθού τους.

3. Γράψτε το προηγούμενο πρόγραμμα ξανά χρησιμοποιώντας κλάση στη θέση της δομής. Επιπλέον ορίστε constructor και getters/setters για τα μέλη δεδομένων του αντικειμένου υπάλληλος.

4. Γράψτε ένα πρόγραμμα που να γεμίζει έναν πίνακα a, 5 γραμμών και 5 στηλών, με τυχαίες ακέραιες τιμές στο διάστημα 1 έως και 1000 (χρησιμοποιήστε ως seed την τιμή 12345). Γράψτε μια συνάρτηση που να δέχεται ως παράμετρο τον πίνακα a και να επιστρέφει σε μονοδιάστατο πίνακα col το άθροισμα των τιμών κάθε στήλης του πίνακα. Οι τιμές που επιστρέφονται να εμφανίζονται στο κύριο πρόγραμμα το οποίο να εμφανίζει επιπλέον και τον αριθμό στήλης με το μεγαλύτερο άθροισμα.

Αναφορές

[1] Σταμάτης Σταματιάδης. Εισαγωγή στη γλώσσα προγραμματισμού C++11. Τμήμα Επιστήμης και Τεχνολογίας Υλικών, Πανεπιστήμιο Κρήτης, 2017, https://www.materials.uoc.gr/el/undergrad/courses/ETY215/notes.pdf.

[2] Allen B. Downey. How to think like a computer scientist, C++ version, 2012, http://www.greenteapress.com/thinkcpp/.

[3] Juan Soulié. C++ Language Tutorial. cplusplus.com, 2007, http://www.cplusplus.com/files/tutorial.pdf.

[4] Brian Hall. Beej's Guide to C Programming, 2007, http://beej.us/guide/bgc/.