Δ.Π.Θ Δομημένος Προγραμματισμός

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ

Γενικές Ασκήσεις

Dr. Αθανάσιος Μπαλαφούτης
Εργαστηριακό Διδακτικό Προσωπικό
Τομέας Συστημάτων Παραγωγής
Εργαστήριο Ρομποτικής και Αυτοματισμών
abalafou@pme.duth.gr
Γραφείο 304, τηλ.: 25410 – 79892

Από αρχείο σε Πίνακα 2D

Έστω το αρχείο `bit.txt`, που περιέχει σε κάθε γραμμή 16 bit (0 ή 1). Το αρχείο περιέχει 3 γραμμές

```
1 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0

0 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0

1 1 0 0 1 1 0 0 0 1 1 1 1 1 0 1
```

Θέλω να διαβάσω το αρχείο και να το τοποθετήσω σε έναν πίνακα 2D.
Στη συνέχεια θέλω να εκτυπώσω το περιεχόμενο του πίνακα.

Από αρχείο σε Πίνακα 2D (1/2)


```c
#include <stdio.h>
#include <stdlib.h>
int main() {
    FILE *file = fopen("files/bits.txt", "r");
    if (file == NULL) {
        printf("Σφάλμα: Δεν μπόρεσα να ανοίξω το αρχείο.\n");
        return 1;
    }
    int data[3][16];
    for(int row = 0; row < 3; row++){
        for (int col = 0; col < 16; col++) {
            int bit;
            fscanf(file, "%d", &bit);
            data[row][col] = bit;
        }
    }
```

Από αρχείο σε Πίνακα 2D (2/2)


```
    fclose(file);
    printf("Περιεχόμενα του πίνακα:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 16; j++) {
            printf("%d ", data[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Από αρχείο σε Πίνακα 2D

Αν το αρχείο `bit.txt`, διαχώριζε τους αριθμούς με κόμμα (συνηθισμένη μορφή αρχείων csv):

```
1,0,0,1,1,0,1,0,1,1,0,0,1,0,0,1
0,1,1,0,1,0,0,1,1,0,1,0,0,1,1,0
1,1,0,0,1,1,0,0,1,0,1,1,0,0,1,1
```

Η μοναδική αλλαγή στον προηγούμενο κώδικα θα ήταν στην εντολή fscanf:

```
fscanf(file, "%d,", &bit);
```

Από αρχείο σε δομή (struct)


```
1 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0
0 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0
1 1 0 0 1 1 0 0 0 1 1 1 1 1 0 1
```

Θέλουμε να φορτώσουμε τα δεδομένα του αρχείου με μια δομή με την ακόλουθη μορφή:

- lineID (αύξων αριθμός κάθε γραμμμής)
- firstByte[8] (σε ένα 1D πίνακα βάζω τους 8 πρώτους αριθμούς)
- secondByte[8] (σε ένα 1D πίνακα βάζω τους 8 τελευταίους αριθμούς)
- onesCount[2] (σε ένα 1D πίνακα βάζω το πλήθος των 1, για κάθε byte)

Από αρχείο σε δομή (struct)


```
1 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0
0 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0
1 1 0 0 1 1 0 0 0 1 1 1 1 1 0 1
```

Δομή (struct) `BitLine`:

```
typedef struct {
   int lineID;
   int firstByte[8];
   int secondByte[8];
   int onesCount[2];
} BitLine;
...
BitLine bitlines[3]; // main
```

Από αρχείο σε δομή (struct)


```
1 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0
0 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0
1 1 0 0 1 1 0 0 0 1 1 1 1 1 0 1
```

Δομή (struct) BitLine:

```
typedef struct {
   int lineID;
   int firstByte[8];
   int secondByte[8];
   int onesCount[2];
} BitLine;
...
BitLine bitlines[3]; // main
```

bitlines[3]

bitlines[0]
bitlines[1]
bitlines[2]

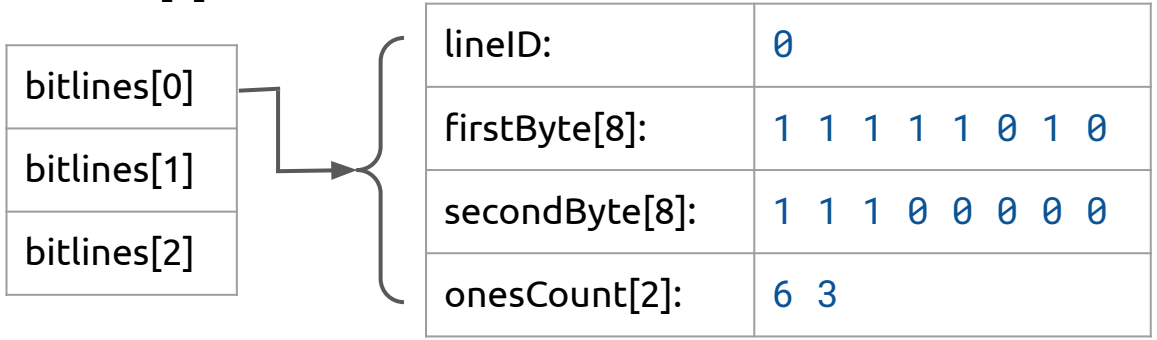
Από αρχείο σε δομή (struct)


```
1 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0
0 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0
1 1 0 0 1 1 0 0 0 1 1 1 1 1 0 1
```

Δομή (struct) `BitLine`:

```
typedef struct {
   int lineID;
   int firstByte[8];
   int secondByte[8];
   int onesCount[2];
} BitLine;
...
BitLine bitlines[3]; // main
```

bitlines[3]



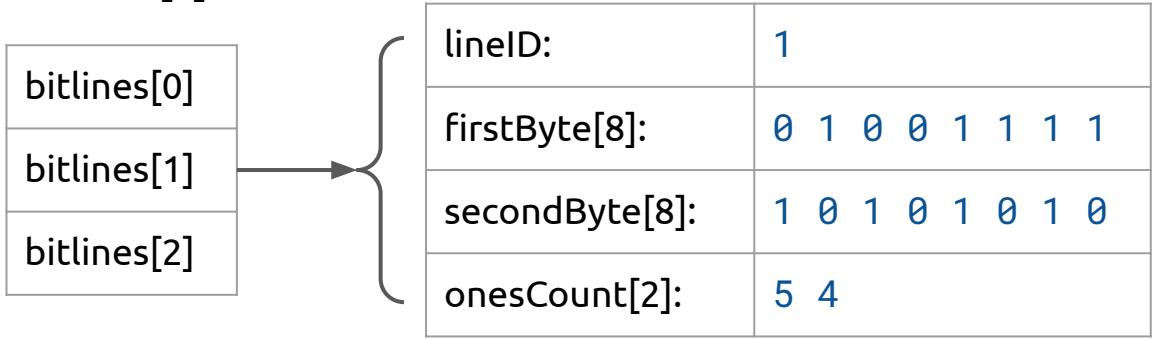
Από αρχείο σε δομή (struct)


```
1 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0
0 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0
1 1 0 0 1 1 0 0 0 1 1 1 1 1 0 1
```

Δομή (struct) `BitLine`:

```
typedef struct {
   int lineID;
   int firstByte[8];
   int secondByte[8];
   int onesCount[2];
} BitLine;
...
BitLine bitlines[3]; // main
```

bitlines[3]



Από αρχείο σε δομή (struct)


```
1 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0
0 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0
1 1 0 0 1 1 0 0 0 1 1 1 1 1 0 1
```

Δομή (struct) BitLine:

```
typedef struct {
   int lineID;
   int firstByte[8];
   int secondByte[8];
   int onesCount[2];
} BitLine;
...
BitLine bitlines[3]; // main
```

bitlines[3]

bitlines[0]
bitlines[1]
bitlines[2]

lineID:	2
firstByte[8]:	1 1 0 0 1 1 0 0
secondByte[8]:	0 1 1 1 1 1 0 1
onesCount[2]:	4 6

Από αρχείο σε δομή (1/3)


```c
#include <stdio.h>
#include <stdlib.h>
typedef struct {
   int lineID;
   int firstByte[8];
   int secondByte[8];
   int onesCount[2];
} BitLine;
int main() {
   FILE *file = fopen("files/bits.txt", "r");
   if (!file) {
       printf("Σφάλμα: Δεν μπόρεσα να ανοίξω το αρχείο.\n");
       return 1;
   }
   BitLine bitlines[3];
   int bit;
```

Από αρχείο σε δομή (2/3)


```
for(int row = 0; row < 3; row++){
        int ones1 = 0, ones2 = 0;
        int byte1[8];
        int byte2[8];

        for (int i = 0; i < 8; i++) {
            fscanf(file, "%d,", &bit);
            byte1[i] = bit;
            if (bit == 1) ones1++;
        }
        for (int i = 0; i < 8; i++) {
            fscanf(file, "%d,", &bit);
            byte2[i] = bit;
            if (bit == 1) ones2++;
        }
```

```
    // Αντιγραφή στη δομή
    bitlines[row].lineID = row + 1;
    for (int i = 0; i < 8; i++) {
        bitlines[row].firstByte[i] = byte1[i];
        bitlines[row].secondByte[i] = byte2[i];
     }
      bitlines[row].onesCount[0] = ones1;
      bitlines[row].onesCount[1] = ones2;
}

fclose(file);
```

Από αρχείο σε δομή (3/3)


```
    for (int i = 0; i < 3; i++) {
        printf("ID %d:\n", bitlines[i].lineID);
        printf("Πρώτο Byte: ");
        for (int j = 0; j < 8; j++)
            printf("%d ", bitlines[i].firstByte[j]);
        printf("\nΠλήθος 1: %d\n", bitlines[i].onesCount[0]);

        printf("Δεύτερο Byte: ");
        for (int j = 0; j < 8; j++)
            printf("%d ", bitlines[i].secondByte[j]);
        printf("\nΠλήθος 1: %d\n", bitlines[i].onesCount[1]);
    }

    return 0;
}
```

Από αρχείο σε δομή (struct)


```
1 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0
0 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0
1 1 0 0 1 1 0 0 0 1 1 1 1 1 0 1
```

Θέλουμε να φορτώσουμε τα δεδομένα του αρχείου με μια δομή με την ακόλουθη μορφή:

- lineID (αύξων αριθμός κάθε γραμμμής)
- firstByte[8] (σε ένα 1D πίνακα βάζω τους 8 πρώτους αριθμούς)
- secondByte[8] (σε ένα 1D πίνακα βάζω τους 8 τελευταίους αριθμούς)
- onesCount[2] (σε ένα 1D πίνακα βάζω το πλήθος των 1, για κάθε byte)

Πόσες και ποιες γραμμές έχουν άρτιο πλήθος από 1 και στα δύο Byte;

Άρτιο πλήθος από byte


```
printf("Γραμμές με άρτιο πλήθος 1 σε κάθε οκτάδα:\n");

for (int i = 0; i < 3; i++) {
    if (bitlines[i].onesCount[0] % 2 == 0 && bitlines[i].onesCount[1] % 2 == 0) {
        printf("  ID γραμμής: %d (%d, %d)\n",
            bitlines[i].lineID,
            bitlines[i].onesCount[0],
            bitlines[i].onesCount[1]);
    }
}
```

Άσκηση - Βαθμολογίες Φοιτητών

Το αρχείο `grades.txt` περιέχει τη βαθμολογία ενός φοιτητή (ή μίας φοιτήτριας) για όλη τη διάρκεια των σπουδών του (της). Το Πανεπιστημιακό Τμήμα στο οποίο ανήκει:

1. διενεργεί τρεις εξεταστικές περιόδους κάθε ημερολογιακό έτος (περίοδοι 1, 2, 3),
2. επιτρέπει σε κάθε εξεταστική τη συμμετοχή του φοιτητή (φοιτήτριας) σε 8 το πολύ εξετάσεις διαφορετικών μαθημάτων

Το αρχείο περιέχει στοιχεία για τα έτη 2011-2020. Κάθε γραμμή του αρχείου περιλαμβάνει κατά σειρά:

- Εξεταστική Περίοδος (1 ή 2 ή 3)
- Ημερολογιακό έτος (π.χ. 2019)
- 8 ζεύγη ακεραίων τιμών. Κάθε ζεύγος αποτελείται από τον κωδικό μαθήματος (τριψήφιος θετικός ακέραιος) και τη βαθμολογία (0-10). Όταν ο βαθμός είναι 0 τότε ο φοιτητής (φοιτήτρια) δεν συμμετείχε στην εξέταση του μαθήματος, παρόλο που το είχε δηλώσει.

Άσκηση - Βαθμολογίες Φοιτητών

Οι εγγραφές του αρχείου δεν είναι ταξινομημένες ημερολογιακά. Κάθε εξεταστική περίοδος για κάθε ημερολογιακό έτος υπάρχει μόνον μία φορά στο αρχείο.
Παράδειγμα ορισμένων γραμμών από το αρχείο `grades.txt`:

```
2 2014   218 3    125 6    219 2    188 4    279 9    382 0    555 0    715 3
1 2016   218 7    191 10   543 10   382 6    430 1    555 4    499 0    357 0
2 2017   109 6    263 2    372 2    272 4    118 3    363 4    432 6    531 9
.....................
```

Άσκηση - Βαθμολογίες Φοιτητών

Να γραφεί μια συνάρτηση με όνομα `create_student` η οποία θα έχει ως παραμέτρους εισόδου :

- Τα ζεύγη (κωδικός μαθήματος, βαθμολογία) από μία γραμμή του αρχείου

Η συνάρτηση θα πρέπει να επιστρέφει για κάθε γραμμή του αρχείου :

1. Τους κωδικούς μαθημάτων με βαθμολογία τουλάχιστον 5
2. Το πλήθος μαθημάτων με βαθμολογία <5
3. Το άθροισμα των βαθμών όλων των μαθημάτων με βαθμολογία τουλάχιστον 5

Θα μελετήσουμε 4 διαφορετικές προσεγγίσεις

1η ιδέα - create_student


```c
#include <stdio.h>
typedef struct {
   int passed_courses[8];          // Κωδικοί μαθημάτων με βαθμό >= 5
   int num_passed;                 // Πλήθος μαθημάτων με βαθμό >= 5
   int num_failed;                 // Πλήθος μαθημάτων με βαθμό < 5 και > 0
   int sum_passed_grades;          // Άθροισμα βαθμών για βαθμούς >= 5
} StudentResult;
```

```c
StudentResult create_student(int codes[], int grades[]) {

StudentResult result;
   ...

   return result;
}
```

2η ιδέα - create_student


```c
#include <stdio.h>
typedef struct {
   int passed_courses[8];          // Κωδικοί μαθημάτων με βαθμό >= 5
   int num_passed;                 // Πλήθος μαθημάτων με βαθμό >= 5
   int num_failed;                 // Πλήθος μαθημάτων με βαθμό < 5 και > 0
   int sum_passed_grades;          // Άθροισμα βαθμών για βαθμούς >= 5
} StudentResult;
```

```c
StudentResult create_student(int data[8][2]) {

StudentResult result;
   ...

   return result;
}
```

3η ιδέα - create_student


```
#include <stdio.h>


void create_student(int codes[], int grades[], int passed[], int *num_passed,
                    int *num_failed, int *sum_passed)
{

...

}
```

4η ιδέα - create_student


```c
#include <stdio.h>


// pairs = {218, 3, 125, 6, 219, 2, 188, 4, 279, 9, 382, 0, 555, 0, 715, 3}

void create_student(int pairs[], int passed[], int *num_passed, int *num_failed, int
                    *sum_passed)
{

   ...

}
```

1η ιδέα - create_student


```c
#include <stdio.h>
typedef struct {
   int passed_courses[8];          // Κωδικοί μαθημάτων με βαθμό >= 5
   int num_passed;                 // Πλήθος μαθημάτων με βαθμό >= 5
   int num_failed;                 // Πλήθος μαθημάτων με βαθμό < 5 και > 0
   int sum_passed_grades;          // Άθροισμα βαθμών για βαθμούς >= 5
} StudentResult;
```

```c
StudentResult create_student(int codes[], int grades[]) {

StudentResult result;
   ...

   return result;
}
```

1η ιδέα - create_student


```
StudentResult create_student(int codes[], int grades[]) {
    StudentResult result;
    result.num_passed = 0;
    result.num_failed = 0;
    result.sum_passed_grades = 0;
    for (int i = 0; i < 8; i++) {
        int code = codes[i];
        int grade = grades[i];
        if (grade >= 5) {
            result.passed_courses[result.num_passed++] = code;
            result.sum_passed_grades += grade;
        } else if (grade > 0)
            result.num_failed++;
    }
    return result;
}
```

1η ιδέα - main (για δοκιμή)


```c
int main() {
    // Παράδειγμα γραμμής από το αρχείο:
    // 2 2014    218 3    125 6    219 2    188 4    279 9    382 0    555 0    715 3
    int codes[8] = {218, 125, 219, 188, 279, 382, 555, 715};
    int grades[8] = {3, 6, 2, 4, 9, 0, 0, 3};

    StudentResult res = create_student(codes, grades);

    printf("Μαθήματα με βαθμό >= 5: ");
    for (int i = 0; i < res.num_passed; i++)
        printf("%d ", res.passed_courses[i]);
    printf("\nΠλήθος μαθημάτων με βαθμό < 5: %d\n", res.num_failed);
    printf("Άθροισμα βαθμών (>=5): %d\n", res.sum_passed_grades);

    return 0;
}
```

2η ιδέα - create_student


```c
#include <stdio.h>
typedef struct {
   int passed_courses[8];          // Κωδικοί μαθημάτων με βαθμό >= 5
   int num_passed;                 // Πλήθος μαθημάτων με βαθμό >= 5
   int num_failed;                 // Πλήθος μαθημάτων με βαθμό < 5 και > 0
   int sum_passed_grades;          // Άθροισμα βαθμών για βαθμούς >= 5
} StudentResult;
```

```c
StudentResult create_student(int data[8][2]) {

StudentResult result;
   ...

   return result;
}
```

2η ιδέα - create_student


```
StudentResult create_student(int data[8][2]) {
   StudentResult result;
   result.num_passed = 0;
   result.num_failed = 0;
   result.sum_passed_grades = 0;
   for (int i = 0; i < 8; i++) {
       int code = data[i][0];
       int grade = data[i][1];
       if (grade >= 5) {
           result.passed_courses[result.num_passed++] = code;
           result.sum_passed_grades += grade;
       } else if (grade > 0)
          result.num_failed++;
   }
   return result;
}
```

2η ιδέα - main (για δοκιμή)


```c
int main() {
    // Παράδειγμα γραμμής από το αρχείο:
    int data[8][2] = {
        {218, 3}, {125, 6}, {219, 2}, {188, 4},
        {279, 9}, {382, 0}, {555, 0}, {715, 3}
    };
    StudentResult res = create_student(data);
    printf("Μαθήματα με βαθμό >= 5: ");
    for (int i = 0; i < res.num_passed; i++) {
        printf("%d ", res.passed_courses[i]);
    }
    printf("\nΠλήθος μαθημάτων με βαθμό < 5: %d\n", res.num_failed);
    printf("Άθροισμα βαθμών (>=5): %d\n", res.sum_passed_grades);

    return 0;
}
```

3η ιδέα - create_student


```
#include <stdio.h>


void create_student(int codes[], int grades[], int passed[], int *num_passed,
                    int *num_failed, int *sum_passed)
{

...

}
```

3η ιδέα - create_student


```c
#include <stdio.h>

void create_student(int codes[], int grades[], int passed[], int *num_passed,
                    int *num_failed, int *sum_passed) {
   *num_passed = 0;
   *num_failed = 0;
   *sum_passed = 0;
   for (int i = 0; i < 8; i++) {
       if (grades[i] >= 5) {
           passed[*num_passed] = codes[i];
           (*num_passed)++;
           *sum_passed += grades[i];
       } else if (grade > 0)
           (*num_failed)++;
   }
}
```

3η ιδέα - main (για δοκιμή)


```c
int main() {
    int codes[8]  = {218, 125, 219, 188, 279, 382, 555, 715};
    int grades[8] = {3,   6,   2,   4,   9,   0,   0,   3};

    int passed[8];
    int num_passed, num_failed, sum_passed;
    create_student(codes, grades, passed, &num_passed, &num_failed, &sum_passed);
    printf("Μαθήματα με βαθμό >= 5: ");
    for (int i = 0; i < num_passed; i++) {
        printf("%d ", passed[i]);
    }
    printf("\nΠλήθος μαθημάτων με βαθμό < 5: %d\n", num_failed);
    printf("Άθροισμα βαθμών (>=5): %d\n", sum_passed);

    return 0;
}
```

4η ιδέα - create_student


```c
#include <stdio.h>


// pairs = {218, 3, 125, 6, 219, 2, 188, 4, 279, 9, 382, 0, 555, 0, 715, 3}

void create_student(int pairs[], int passed[], int *num_passed, int *num_failed, int
                    *sum_passed)
{

   ...

}
```

4η ιδέα - create_student


```
void create_student(int pairs[], int passed[], int *num_passed, int *num_failed, int
                    *sum_passed) {
    *num_passed = 0;
    *num_failed = 0;
    *sum_passed = 0;
    for (int i = 0; i < 8; i++) {
        int code = pairs[i * 2];
        int grade = pairs[i * 2 + 1];
        if (grade >= 5) {
            passed[*num_passed] = code;
            (*num_passed)++;
            *sum_passed += grade;
        } else if (grade > 0)
            (*num_failed)++;
    }
}
```

4η ιδέα - main (για δοκιμή)


```c
int main() {
    int pairs[16] = {218, 3, 125, 6, 219, 2, 188, 4, 279, 9, 382, 0, 555, 0, 715, 3};
    int passed[8];
    int num_passed, num_failed, sum_passed;

    create_student(pairs, passed, &num_passed, &num_failed, &sum_passed);

    printf("Μαθήματα με βαθμό >= 5: ");
    for (int i = 0; i < num_passed; i++)
        printf("%d ", passed[i]);

    printf("\nΠλήθος μαθημάτων με βαθμό < 5: %d\n", num_failed);
    printf("Άθροισμα βαθμών (>=5): %d\n", sum_passed);

    return 0;
}
```

Άσκηση - Βαθμολογίες Φοιτητών

Να οριστεί μια δομή με όνομα `student` που θα διαχειρίζεται και θα ομαδοποιεί τα δεδομένα κάθε γραμμής του αρχείου `grades.txt`. Τα μέλη της δομής είναι :

1. Ημερολογιακό έτος
2. Εξεταστική περίοδος
3. Κωδικοί των μαθημάτων με βαθμολογία τουλάχιστον 5
4. Πλήθος μαθημάτων με βαθμολογία <5
5. Το άθροισμα των βαθμών για όλα τα μαθήματα με βαθμολογία τουλάχιστον 5

Στη συνάρτηση main( ):
Να ορίσετε έναν πίνακα δομών του τύπου `student` με πλήθος στοιχείων όσες ακριβώς και οι εγγραφές του αρχείου `grades.txt`.

- Να εισάγετε τις τιμές των στοιχείων στον πίνακα δομών, μέσω προσπέλασης του αρχείου `grades.txt` και κλήσης της συνάρτησης `create_student` (απαιτείται έλεγχος για την ύπαρξη του αρχείου και τη δυνατότητα προσπέλασής του).
- Να εμφανίσετε όλα τα στοιχεία του πίνακα δομών, ένα σε κάθε γραμμή της οθόνης.

Δομή student

1. Ημερολογιακό έτος
2. Εξεταστική περίοδος
3. Κωδικοί των μαθημάτων με βαθμολογία τουλάχιστον 5
4. Πλήθος μαθημάτων με βαθμολογία <5
5. Το άθροισμα των βαθμών για όλα τα μαθήματα με βαθμολογία τουλάχιστον 5

```c
#include <stdio.h>

typedef struct {
   int year;
   int period;
   int passed_codes[8];
   int num_failed;
   int sum_passed;
} student;
```

create_student


```
student create_student(int period, int year, int *codes, int *grades) {
    student s;
    s.year = year;
    s.period = period;
    s.num_failed = 0;
    s.sum_passed = 0;

    int passed_index = 0;
    for (int i = 0; i < 8; i++) {
        if (grades[i] >= 5) {
            s.passed_codes[passed_index] = codes[i];
            passed_index++;
            s.sum_passed += grades[i];
        } else if (grades[i] > 0)
            s.num_failed++;
    }
```

```
 for (int i = passed_index; i < 8; i++)
        s.passed_codes[i] = 0;
 return s;
}
```

main (1/2)


```c
int main() {
    FILE *fp = fopen("files/grades.txt", "r");
    if (fp == NULL) {
        printf("Αδυναμία ανοίγματος του αρχείου.\n");
        return 1;
    }
    student students[100];
    int total_students = 0;
    int period, year;
    int codes[8], grades[8];
    while (fscanf(fp, "%d %d", &period, &year) != EOF) {
        for (int i = 0; i < 8; i++)
            fscanf(fp, "%d %d", &codes[i], &grades[i]);

        students[total_students] = create_student(period, year, codes, grades);
        total_students++;
    }
```

main (2/2)


```
    fclose(fp);
    for (int i = 0; i < total_students; i++) {
        printf("%d, %d, %d, %d, ",
            students[i].year,
            students[i].period,
            students[i].num_failed,
            students[i].sum_passed);

        for (int j = 0; j < 8 && students[i].passed_codes[j] != 0; j++)
            printf("%d ", students[i].passed_codes[j]);

        printf("\n");
    }
    return 0;
}
```

Άσκηση - Βαθμολογίες Φοιτητών

Χρησιμοποιώντας όλα τα στοιχεία του πίνακα δομών, να βρείτε και να εμφανίσετε:

1. πόσα μαθήματα πέρασε ο φοιτητής (φοιτήτρια) σε καθένα από τα 10 έτη (2011-2020)
2. τον μέσο όρο βαθμολογίας του (συμμετέχουν τα μαθήματα με βαθμό τουλάχιστον 5)

main (1/2)


```
// Υπολογισμός περασμένων μαθημάτων και μέσου όρου ανά έτος

int passed_per_year[10] = {0};   // index 0 -> 2011, 1 -> 2012, ..., 9 -> 2020
int sum_per_year[10] = {0};

for (int i = 0; i < total_students; i++) {
    int year_index = students[i].year - 2011;
        for (int j = 0; j < 8; j++) {
            if (students[i].passed_codes[j] != 0) {
                passed_per_year[year_index]++;
            }
        }
        sum_per_year[year_index] += students[i].sum_passed;
}
```

main (2/2)


```
for (int i = 0; i < 10; i++) {
    int year = 2011 + i;
    double average;
    if (passed_per_year[i] > 0) {
        average = (double)sum_per_year[i] / passed_per_year[i];
    } else {
        average = 0.0;
    }

    printf("%4d, %4d , %7.2f\n", year, passed_per_year[i], average);
}
```

Άσκηση - Βαθμολογίες Φοιτητών

Χρησιμοποιώντας όλα τα στοιχεία του πίνακα δομών, να βρείτε και να εμφανίσετε:

3. σε αύξουσα διάταξη, τους κωδικούς μαθημάτων που έχει περάσει ο φοιτητής – φοιτήτρια. Για την ταξινόμηση να χρησιμοποιήσετε κατάλληλη συνάρτηση ταξινόμησης `sort_array`

Συνάρτηση ταξινόμησης


```
void sort_array(int arr[], int n) {

   for (int i = 0; i < n - 1; i++) {
       for (int j = i + 1; j < n; j++) {
           if (arr[i] > arr[j]) {
               int temp = arr[i];
               arr[i] = arr[j];
               arr[j] = temp;
           }
       }
   }

}
```

main


```
for (int i = 0; i < total_students; i++) {
    sort_array(students[i].passed_codes, 8);
    printf("Φοιτητής %d: ", i + 1);

    for (int j = 0; j < 8; j++){
        if(students[i].passed_codes[j] > 0)
          printf("%d ", students[i].passed_codes[j]);
    }

    printf("\n");
}
```