ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Προγραμματισμός σε C++, Python και Matlab

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

της

ΑΝΤΩΝΑΡΙΔΟΥ ΜΑΡΙΑ

(ΑΕΜ: 1645)

Επιβλέπων : Στυλιανός Μαυρατζάς

Καστοριά Μάιος - 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Προγραμματισμός σε C++, Python και Matlab

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

της

ΑΝΤΩΝΑΡΙΔΟΥ ΜΑΡΙΑ

(ΑΕΜ: 1645)

Επιβλέπων : Στυλιανός Μαυρατζάς

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την Δευτέρα 29 Μαΐου 2023

Καστοριά Μάιος - 2022

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή κύριο Στυλιανό Μαυρατζά για την πολύτιμη βοήθειά του και την καθοδήγηση. Επίσης, θα ήθελα να ευχαριστήσω την οικογένεια μου για την στήριξή τους.

Περίληψη

Αντικείμενο της παρούσας πτυχιακής εργασίας είναι η σύγκριση των γλωσσών προγραμματισμού C++, Python και Matlab και συγκεκριμένα η μελέτη των συντακτικών διαφορών που παρουσιάζουν οι συναρτήσεις στις προαναφερθείσες γλώσσες προγραμματισμού. Για τον σκοπό αυτό, έγινε βιβλιογραφική έρευνα αναφορικά με τις συναρτήσεις στην κάθε γλώσσα προγραμματισμού και αναπτύχθηκαν προγράμματα στις τρεις γλώσσες προγραμματισμού που επιλύουν τα ίδια προβλήματα. Χρησιμοποιήθηκαν έξι προβλήματα, τα οποία ανήκουν στα πεδία των Μαθηματικών, των Αλγορίθμων και της Φυσικής και test cases από την γνωστή πλατφόρμα codingame.com. Οι λύσεις των προγραμμάτων στην C++ και στην Python επαληθεύτηκαν με χρήση της πλατφόρμας, ενώ για τα προγράμματα στην Matlab χρησιμοποιήθηκε το λογισμικό πακέτο Matlab 2019a. Σε γενικότερο πλαίσιο η καταλληλότητα της γλώσσας προγραμματισμού εξαρτάται από το είδος της εφαρμογής, τα προγράμματα που αναπτύχθηκαν στην παρούσα πτυχιακή εργασία αναδεικνύουν ότι η Python τόσο σε επίπεδο συναρτήσεων όσο και σε επίπεδο σύνταξης του προγράμματος από την αρχή, είναι πιο ευέλικτη, περιεκτική και χρηστική σε σχέση με τις C++ και Matlab.

Λέξεις Κλειδιά: *C++, Python, Matlab, διαφορές, σύγκριση, κώδικας.*

Abstract

The subject of this thesis is the comparison of the programming languages C++, Python and Matlab and, in particular, the study of the syntax differences among the functions of the aforementioned programming languages. For this purpose, a literature review on the functions of each programming language was conducted and programs in the three programming languages solving the same problems were developed. Six problems in the fields of Mathematics, Algorithms and Physics and test cases from the popular platform codingame.com were used. The program solutions in C++ and Python were verified with the aid of the platform, whereas Matlab programs were verified by executing them in Matlab 2019a. Although, in general, the suitability of the programming language depends on the kind of the application, the developed programs of this thesis show that Python is more flexible, concise and useful than C++ and Matlab regarding functions and developing a program from scratch.

Key Words: *C++, Python, Matlab, differences, comparison, code.*

Πίνακας Περιεχομένων

Λίστα Εικόνων

Λίστα Πινάκων

Εισαγωγή

Η Python και η C++ είναι γλώσσες προγραμματισμού γενικού σκοπού, αλλά διαφέρουν σε αρκετά σημεία. Η C++ προέρχεται από την γλώσσα προγραμματισμού C και χρησιμοποιεί μεταγλωττιστή για την εκτέλεση των προγραμμάτων. Η Python ως γλώσσα προγραμματισμού υψηλού επιπέδου χρησιμοποιεί διερμηνευτή για την εκτέλεση των προγραμμάτων. Επιπλέον, στην Python, σε αντίθεση με την C++, οι μεταβλητές μπορούν να χρησιμοποιηθούν χωρίς να δηλωθούν προηγουμένως στον κώδικα. Στην Python υπάρχει η δυνατότητα να γραφεί το πρόγραμμα μία φορά και να εκτελεστεί σε οποιοδήποτε λειτουργικό σύστημα είναι εγκατεστημένη η Python.

Αναφορικά με την διαχείριση μνήμης, στην C++ είναι πιθανή η διαρροή μνήμης (memory leak), δεν παρέχεται η δυνατότητα συλλογής απορριμμάτων (garbage collection) και χρησιμοποιούνται pointers σε μεγάλο βαθμό. Η Python διαθέτει την δυνατότητα συλλογής απορριμμάτων και δυναμικής κατανομής στην μνήμη με αποτέλεσμα την αποτελεσματικότερη διαχείριση μνήμης.

Η C++ επίσης χαρακτηρίζεται ως γλώσσα μεσαίου επιπέδου καθώς εμπεριέχει τόσο στοιχεία γλώσσας προγραμματισμού υψηλού επιπέδου όσο και στοιχεία γλώσσας προγραμματισμού χαμηλού επιπέδου. Υποστηρίζει λειτουργικότητα αντικειμενοστραφούς προγραμματισμού, την υπερφόρτωση τελεστών, την πολλαπλή κληρονομικότητα, εικονικές συναρτήσεις και τον χειρισμό εξαιρέσεων. Η Python είναι γνωστή για την απλότητά της, ο κώδικάς της διαβάζεται εύκολα και κατατάσσεται στις γλώσσες προγραμματισμού υψηλού επιπέδου αλλά και στις γλώσσες αντικειμενοστραφούς προγραμματισμού.

Οι βασικές διαφορές μεταξύ C++ και Python σχετικά με την σύνταξη του κώδικα, εντοπίζονται στη χρήση κενών διαστημάτων, στις εκφράσεις τύπου Boolean, στην χρήση μεταβλητών, pointers και συνοπτικών λιστών.

Από την άλλη πλευρά, η γλώσσα προγραμματισμού Matlab είναι γνωστή για το υψηλής ποιότητας ολοκληρωμένο περιβάλλον ανάπτυξης κώδικα που παρέχει για προγράμματα που περιλαμβάνουν πίνακες ή εφαρμογές γραμμικής άλγεβρας. Σε αντίθεση με την Python και την C++, το περιβάλλον Matlab δεν είναι λογισμικό ανοικτού κώδικα. Η ειδοποιός διαφορά μεταξύ Matlab και των δύο γλωσσών, είναι ότι στην Matlab κάθε οντότητα αντιμετωπίζεται ως πίνακας, ενώ, παραδείγματος χάρη, στην Python κάθε οντότητα αντιμετωπίζεται ως αντικείμενο. Οι συντακτικές διαφορές μεταξύ της Matlab και των δύο γλωσσών εντοπίζονται στον ορισμό των σχολίων, στη χρήση κενών, στον ορισμό βρόγχων επανάληψης, στην κλήση και χρήση των συναρτήσεων, στις εντολές ελέγχου διακλάδωσης, σε κάποιους

μαθηματικούς τελεστές όπως ο τελεστής ύψωσης σε δύναμη και στις συναρτήσεις βιβλιοθήκης.

Στην παρούσα πτυχιακή εργασία μελετάται η σύνταξη των συναρτήσεων στις γλώσσες προγραμματισμού C++, Python και Matlab. Για τον σκοπό αυτό αναπτύχθηκαν τα ίδια προβλήματα, έξι τον αριθμό, με κλίμακα δυσκολίας σε κάθε μία από αυτές.

Στα τέσσερα πρώτα κεφάλαια αναλύεται η θεωρία συναρτήσεων στις υπό μελέτη γλώσσες προγραμματισμού και αναδεικνύονται οι διαφορές τους.

Στο πέμπτο κεφάλαιο αναπτύσσονται αλγοριθμικά προγράμματα και προγράμματα από τα επιστημονικά πεδία των Μαθηματικών και της Φυσικής με στόχο την περαιτέρω εμβάθυνση στις υπό μελέτη γλώσσες προγραμματισμού και την ανάλυση του θέματος της πτυχιακής εργασίας.

Στο έκτο κεφάλαιο αναφέρονται τα συμπεράσματα της μελέτης.

Ακολουθούν η σχετική βιβλιογραφία και τα τεχνικά παραρτήματα. Στο Παράρτημα Α παρατίθενται βασικές και δημοφιλείς συναρτήσεις βιβλιοθήκης των γλωσσών προγραμματισμού C++, Python και Matlab, όπως και οι τιμές εισόδου και εξόδου των test cases μέρους των προγραμμάτων που αναπτύχθηκαν στο πλαίσιο της πτυχιακής εργασίας. Στο Παράρτημα Κώδικα παρατίθενται τα προγράμματα σε C++, Python και Matlab των προβλημάτων του πέμπτου κεφαλαίου.

1. Συναρτήσεις στην C++

Οι συναρτήσεις είναι τα θεμελιώδη blocks στην C++ μέσα στα οποία λαμβάνουν χώρα όλες οι ενέργειες του προγράμματος [1].

1.1 Γενική μορφή συνάρτησης

Η γενική μορφή μιας συνάρτησης στην C++ είναι η εξής [1]:

```
ret-type function-name(parameter list) {

    body of the function
}
```

Εικόνα 1 Γενική μορφή συνάρτησης στην C++ [1]

Το ret-type αναφέρεται στον τύπο των δεδομένων που επιστρέφει η συνάρτηση. Μια συνάρτηση μπορεί να επιστρέφει οποιονδήποτε τύπο δεδομένων εκτός από πίνακα. Το parameter list είναι μία λίστα με ονόματα μεταβλητών και τον τύπο κάθε μεταβλητής, που χωρίζονται μεταξύ τους με κόμμα. Οι μεταβλητές αυτές λαμβάνουν τις τιμές των ορισμάτων κατά την κλήση της συνάρτησης. Η λίστα παραμέτρων έχει την μορφή *(type varname1, type varname2, … ,type varnameN)*. Στην περίπτωση που η συνάρτηση δεν περιέχει παραμέτρους, το parameter list είναι κενό, αλλά η παρένθεση χρησιμοποιείται στην σύνταξη της συνάρτησης.

1.2 Κανόνες συναρτήσεων

Κάθε συνάρτηση είναι ένα διακριτό block κώδικα. Ο κώδικας της συνάρτησης ανήκει μόνο στην συνάρτηση και δεν μπορεί να προσπελαστεί από κώδικα άλλης συνάρτησης, εκτός αν κληθεί η συγκεκριμένη συνάρτηση. Ο κώδικας που αποτελεί το σώμα της συνάρτησης είναι κρυμμένος από το υπόλοιπο πρόγραμμα και δεν μπορεί να επηρεαστεί ή να επηρεάσει άλλο τμήμα του προγράμματος, παρά μόνο στην περίπτωση που χρησιμοποιεί καθολικές μεταβλητές (global variables) [1].

Οι μεταβλητές που δηλώνονται εντός μιας συνάρτησης ονομάζονται τοπικές μεταβλητές (local variables). Μια τοπική μεταβλητή δημιουργείται κατά την προσπέλαση της συνάρτησης και καταστρέφεται κατά την έξοδο από την συνάρτηση. Η μοναδική εξαίρεση στον κανόνα συμβαίνει όταν στην δήλωση της μεταβλητής προτάσσεται η λέξη κλειδί static. Σε αυτή την περίπτωση ο compiler αντιλαμβάνεται την μεταβλητή σαν καθολική μεταβλητή για σκοπούς αποθήκευσης, αλλά περιορίζει το πεδίο εφαρμογής της εντός της συνάρτησης [1].

1.3 Ορίσματα συνάρτησης

Όταν μια συνάρτηση χρησιμοποιεί ορίσματα, τότε πρέπει να έχει δηλωμένες μεταβλητές στην λίστα παραμέτρων που δέχονται τις τιμές των ορισμάτων. Αυτές οι μεταβλητές ονομάζονται επίσημες παράμετροι (formal parameters) της συνάρτησης και συμπεριφέρονται ακριβώς όπως οι τοπικές μεταβλητές [1].

1.4 Κλήση συνάρτησης με τιμή και αναφορά

Σε μια γλώσσα προγραμματισμού υπάρχουν δύο τρόποι για να περαστούν τα ορίσματα σε μια υπορουτίνα, η κλήση με τιμή και η κλήση με αναφορά. Κατά την κλήση με τιμή, αντιγράφεται η τιμή του ορίσματος στην επίσημη παράμετρο της υπορουτίνας και οι αλλαγές της επίσημης παραμέτρου δεν επηρεάζουν το όρισμα. Κατά την κλήση με αναφορά, η επίσημη παράμετρος αντιγράφει την διεύθυνση του ορίσματος και χρησιμοποιεί την διεύθυνση αυτή για να αποκτήσει πρόσβαση στο όρισμα, με αποτέλεσμα οι αλλαγές που γίνονται στην επίσημη παράμετρο να επηρεάζουν το όρισμα [1].

```c
#include <stdio.h>

int sqr(int x);

int main(void)
{
int t=10;
printf("%d %d", sqr(t), t);
return 0;
}

int sqr(int x)
{
x = x*x;
return(x);
}
```

Εικόνα 2 Παράδειγμα κλήσης συνάρτησης με τιμή στην C++ [1]

```c
void swap(int *x, int *y)
{
int temp;
temp = *x; /* save the value at address x */
*x = *y; /* put y into x */
*y = temp; /* put x into y */
}

void swap(int *x, int *y);

int main(void)
{
int i, j;
i = 10;
j = 20;
printf("%d %d", i, j);
swap(&i, &j); /* pass the addresses of i and j */
printf("%d %d", i, j);
return 0;
}
```

Εικόνα 3 Παράδειγμα κλήσης συνάρτησης με αναφορά στην C++ [1]

1.5 Κλήση συνάρτησης με πίνακα

Όταν ένας πίνακας χρησιμοποιείται σαν όρισμα στην κλήση μιας συνάρτησης, αντιγράφεται η διεύθυνσή του στην συνάρτηση που καλείται. Αν και η συνάρτηση καλείται όπως στην περίπτωση της κλήσης με τιμή, το περιεχόμενο του πίνακα αλλάζει όπως στην περίπτωση της κλήσης με αναφορά [1].

```c
#include <stdio.h>
#include <ctype.h>

void print_upper(char *string);

int main(void)
{
    char s[80];
    gets(s);
    print_upper(s);
    printf("\ns is now uppercase: %s", s);
    return 0;
}

void print_upper(char *string)
{
    register int t;
    for(t=0; string[t]; ++t) {
        string[t] = toupper(string[t]);
        putchar(string[t]);
    }
}
```

Εικόνα 4 Παράδειγμα κλήσης συνάρτησης με πίνακα στην C++ [1]

1.6 Ορίσματα argc και argv της συνάρτησης main()

Υπάρχουν δύο ειδικά ορίσματα, argc και argv, που χρησιμοποιούνται για να λαμβάνουν ορίσματα γραμμής εντολών (command line arguments). Το argc είναι integer και είναι ο αριθμός των ορισμάτων της γραμμής εντολών. Το argv είναι pointer που αντιστοιχεί σε ένα πίνακα χαρακτήρων pointers και κάθε στοιχείο αυτού του πίνακα δείχνει σε ένα όρισμα γραμμής εντολών. Όλα τα ορίσματα της γραμμής εντολών είναι τύπου String [1].

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    if(argc!=2) {
        printf("You forgot to type your name.\n");
        exit(1);
    }
    printf("Hello %s", argv[1]);
return 0;
}
```

Εικόνα 5 Παράδειγμα προγράμματος στην C++ με ορίσματα argc και argv [1]

Στο παράδειγμα παραπάνω αν το όνομα του προγράμματος είναι program και στην γραμμή εντολών τρέξει η εντολή *program John*, η έξοδος του προγράμματος θα είναι *Hello John*.

1.7 Εντολή return και είδη συναρτήσεων

Η εντολή return έχει ως αποτέλεσμα την έξοδο από την συνάρτηση και την επιστροφή στο σημείο του κώδικα από το οποίο έγινε η κλήση της συνάρτησης. Επιπλέον, η εντολή return επιστρέφει μία τιμή ίδιου τύπου με αυτόν της συνάρτησης, εκτός αν η συνάρτηση είναι τύπου void [1].

Τα είδη των συναρτήσεων στην C++ είναι τα εξής [1]:

- ✓ **Υπολογιστικές συναρτήσεις:** Εκτελούν υπολογισμούς και διεργασίες με βάση τα ορίσματα της συνάρτησης και επιστρέφουν μία τιμή. Παραδείγματα αυτού του είδους είναι οι συναρτήσεις βιβλιοθήκης sqrt() και sin(), οι οποίες υπολογίζουν την τετραγωνική ρίζα και το ημίτονο των ορισμάτων.
- ✓ **Ενδεικτικές συναρτήσεις:** Επεξεργάζονται πληροφορίες και επιστρέφουν μία τιμή που υποδηλώνει την επιτυχία ή την αποτυχία

της επεξεργασίας. Για παράδειγμα, η συνάρτηση βιβλιοθήκης fclose() που χρησιμοποιείται για να κλείσει ένα αρχείο, επιστρέφει 0 αν η διεργασία είναι επιτυχής και EOF αν η διεργασία είναι ανεπιτυχής.

- ✓ **Διαδικαστικές συναρτήσεις:** Είναι τύπου void και δεν επιστρέφουν τιμές. Για παράδειγμα, η συνάρτηση βιβλιοθήκης exit() προκαλεί την έξοδο από το πρόγραμμα.

Μία συνάρτηση μπορεί να επιστρέφει ένα pointer. Οι pointers που δείχνουν σε μεταβλητές δεν είναι ούτε τύπου integer ούτε unsigned integer, αλλά διευθύνσεις στην μνήμη συγκεκριμένου τύπου δεδομένων. Κάθε φορά που η τιμή ενός pointer αυξάνεται, δείχνει στο επόμενο αντικείμενο του ίδιου τύπου. Επειδή το μήκος των διαφορετικών τύπων δεδομένων διαφέρει, ο compiler πρέπει να γνωρίζει τον τύπο δεδομένων στον οποίο δείχνει ο pointer. Για τον λόγο αυτό, στην συνάρτηση που επιστρέφει pointer πρέπει να δηλώνεται σαφώς ο τύπος του pointer [1].

```
/* Return pointer of first occurrence of c in s. */
char *match(char c, char *s)
{
    while(c!=*s && *s) s++;
    return(s);
}

#include <stdio.h>
char *match(char c, char *s); /* prototype */
int main(void)
{
    char s[80], *p, ch;
    gets(s);
    ch = getchar();
    p = match(ch, s);
    if(*p) /* there is a match */
        printf("%s ", p);
    else
        printf("No match found.");
    return 0;
}
```

Εικόνα 6 Παράδειγμα συνάρτησης που επιστρέφει pointer στην C++ [1]

Στο προηγούμενο παράδειγμα η συνάρτηση match ελέγχει αν ο χαρακτήρας c υπάρχει στον πίνακα χαρακτήρων s, ο οποίος προσπελαύνεται μέσω του pointer *s. Αν βρεθεί στοιχείο στον πίνακα χαρακτήρων με τιμή ίση με τον χαρακτήρα c, επιστρέφεται ο pointer που δείχνει το στοιχείο στον πίνακα χαρακτήρων, ενώ αν δεν βρεθεί, επιστρέφεται pointer που δείχνει στην τιμή NULL [1].

1.8 Αναδρομικές συναρτήσεις

Μία συνάρτηση ονομάζεται αναδρομική αν στο σώμα της συνάρτησης καλεί τον εαυτό της. Χαρακτηριστικό παράδειγμα αναδρομικής συνάρτησης είναι η συνάρτηση που υπολογίζει το παραγοντικό ενός ακεραίου. Όταν μία συνάρτηση καλεί τον εαυτό της, ένα set καινούργιων τοπικών μεταβλητών και παραμέτρων κατανέμεται στην μνήμη και ο κώδικας εκτελείται από πάνω προς τα κάτω με τις νέες τιμές, ενώ το παλιό set μεταβλητών αφαιρείται από την μνήμη. Συνήθως οι αναδρομικές ρουτίνες δεν μειώνουν το μέγεθος του κώδικα και δεν βελτιώνουν την χρήση της μνήμης συγκριτικά με τους βρόγχους επανάληψης. Στην πραγματικότητα, αν η αναδρομική ρουτίνα εκτελείται πολύ γρήγορα, μπορεί να προκαλέσει αδυναμία εκτέλεσης της ρουτίνας λόγω εξάντλησης της διαθέσιμης μνήμης. Το βασικό πλεονέκτημα των αναδρομικών συναρτήσεων είναι η αναπαράσταση αρκετών αλγορίθμων με λιγότερο και πιο απλό κώδικα [1].

```
/* recursive */
int factr(int n) {
    int answer;
    if(n==1) return(1);
        answer = factr(n-1)*n; /* recursive call */
    return(answer);
}
```

Εικόνα 7 Παράδειγμα αναδρομικής συνάρτησης στην C++ [1]

1.9 Inline συναρτήσεις και Overloading συναρτήσεων

Ο compiler εισάγει τον κώδικα της inline συνάρτησης στην διεύθυνση όπου η συνάρτηση καλείται και με αυτό τον τρόπο αποφεύγει να μεταφερθεί σε μια υπορουτίνα. Ο ορισμός της inline συνάρτησης γίνεται με την λέξη-κλειδί inline. Ο κώδικας του προγράμματος επεκτείνεται κάθε φορά που η συνάρτηση καλείται. Για τον λόγο αυτό, οι inline συναρτήσεις δεν πρέπει να περιέχουν περισσότερες από δύο εντολές. Διαφορετικά, ο compiler μπορεί να αγνοήσει τη λέξη-κλειδί inline και να δημιουργήσει μήνυμα Warning [2].

```
inline int max( int x, int y)
{ return (x >= y ? x : y ); }
```

Εικόνα 8 Παράδειγμα inline συνάρτησης στην C++ [2]

Στις παραδοσιακές γλώσσες προγραμματισμού, οι συναρτήσεις που εκτελούν την ίδια διεργασία αλλά έχουν διαφορετικά ορίσματα, πρέπει να έχουν το ίδιο όνομα. Στην περίπτωση που έχουν το ίδιο όνομα, πρέπει να διαφέρει ο τύπος της συνάρτησης (Overloading) [2].

```
int max( int x, int y);
double max( double x, double y);
```

Εικόνα 9 Παράδειγμα Overloading συναρτήσεων στην C++ [2]

2. Συναρτήσεις στην Python

Σημαντικά προγράμματα κατασκευάζονται με πολλές συναρτήσεις για λόγους ευκολότερης αναβάθμισης και συντήρησης. Η Python ενσωματώνει μια σειρά χαρακτηριστικών γλωσσών συναρτησιακού προγραμματισμού (functional programming) καθιστώντας την χρήση συναρτήσεων ευέλικτη [3].

2.1 Γενική μορφή συνάρτησης

Η συνάρτηση στην Python ορίζεται με την λέξη-κλειδί *def.* Το σώμα της συνάρτησης είναι μία ακολουθία εντολών η οποία εκτελείται όταν καλείται η συνάρτηση. Η κλήση της συνάρτησης πραγματοποιείται γράφοντας το όνομα και τα ορίσματα της συνάρτησης. Η σειρά και ο αριθμός των ορισμάτων πρέπει να ταιριάζουν με τις παραμέτρους της συνάρτησης κατά τον ορισμό της. Στην περίπτωση που τα ορίσματα δεν ταιριάζουν με τις παραμέτρους της συνάρτησης, ανακύπτει η εξαίρεση *TypeError* [3]. Όταν η συνάρτηση ορίζει μία παράμετρο με προεπιλεγμένη τιμή, τότε αυτή η παράμετρος και όσες ακολουθούν είναι προαιρετικές. Αν δεν ανατεθούν τιμές στις προαιρετικές παραμέτρους, ανακύπτει η εξαίρεση *SyntaxError* [3].

Η χρήση μεταβλητών αντικειμένων (mutable objects) ως προεπιλεγμένες τιμές μπορεί να οδηγήσει σε μη προβλεπόμενο αποτέλεσμα. Για να αποφευχθεί αυτό, είναι καλύτερο να χρησιμοποιείται η τιμή *None.* Στην περίπτωση που η συνάρτηση δέχεται ορίσματα υπό τη μορφή keywords, η σειρά των παραμέτρων κατά την κλήση της συνάρτησης δεν παίζει ρόλο. Επιπλέον, μία συνάρτηση μπορεί να δεχθεί μεταβλητό αριθμό παραμέτρων αν το σύμβολο * προστεθεί στην τελευταία μεταβλητή. Σε αυτή την περίπτωση, τα εναπομείναντα ορίσματα αντιγράφονται σε αυτή την μεταβλητή με την μορφή tuple. Αν η τελευταία παραμέτρος της συνάρτησης ξεκινάει με το σύμβολο **, τότε όλα τα επιπλέον ορίσματα τύπου keyword κατά την κλήση της συνάρτησης αντιγράφονται σε αυτή τη παράμετρο με τη μορφή dictionary.

```
def add(x,y):
    return x + y
```

Εικόνα 10 Παράδειγμα συνάρτησης στην Python [3]

```
# Accept variable number of positional or keyword arguments
def spam(*args, **kwargs):
    # args is a tuple of positional args
    # kwargs is dictionary of keyword args
```

Εικόνα 11 Παράδειγμα συνάρτησης με παραμέτρους μορφής tuple και dictionary για ορίσματα τύπου keyword στην Python [3]

2.2 Παράμετροι και επιστρεφόμενες τιμές

Η σύνταξη και η κλήση συναρτήσεων στην Python διαφέρουν συγκριτικά με την C++. Όταν το όρισμα είναι μεταβλητό αντικείμενο, η κλήση της συνάρτησης είναι όμοια με την κλήση με τιμή, ενώ όταν το όρισμα είναι μεταβλητό αντικείμενο (π.χ. λίστα ή dictionary) τότε η κλήση της συνάρτησης είναι όμοια με την κλήση με αναφορά, καθώς είναι δυνατόν να μεταβληθεί η τιμή του ορίσματος λόγω της εκτέλεσης της συνάρτησης [3].

Η εντολή return επιστρέφει μία τιμή από την συνάρτηση. Στην περίπτωση που δεν οριστεί τιμή, επιστρέφεται το αντικείμενο *None*. Επίσης είναι εφικτό να επιστραφούν πολλές τιμές με την μορφή tuple [3].

```
def factor(a):
    d = 2
    while (d <= (a / 2)):
        if ((a / d) * d == a):
            return ((a / d), d)
        d = d + 1
    return (a, 1)
```

Εικόνα 12 Παράδειγμα συνάρτησης που επιστρέφει πολλαπλές τιμές στην Python [3]

2.3 Κανόνες συναρτήσεων

Κάθε φορά που η συνάρτηση εκτελείται, δημιουργείται ένα νέο τοπικό namespace. Το namespace αναπαριστά ένα τοπικό περιβάλλον που περιέχει τα ονόματα των παραμέτρων της συνάρτησης και τα ονόματα των μεταβλητών που ορίζονται εντός του σώματος της συνάρτησης. Ο interpreter αρχικά αναζητά το όνομα στο τοπικό namespace και στην συνέχεια το αναζητά στο καθολικό namespace. Αν δεν βρεθεί το όνομα, ανακύπτει η εξαίρεση *NameError*. Μία μεταβλητή εντός του σώματος της συνάρτησης ανήκει στο τοπικό namespace και ακόμα και αν μεταβληθεί η τιμή της, δεν επηρεάζεται μία μεταβλητή με το ίδιο

όνομα που βρίσκεται εκτός της συνάρτησης, γιατί η δεύτερη ανήκει στο καθολικό namespace [3].

Στην Python υποστηρίζεται ο ορισμός εμφωλευμένων συναρτήσεων. Η λέξη-κλειδί nonlocal υποδηλώνει ότι η μεταβλητή δεν είναι τοπική. Πρέπει να ανατίθενται τιμές στις τοπικές μεταβλητές προτού αυτές χρησιμοποιηθούν προκειμένου να αποφεύγεται η εξαίρεση *UnboundLocalError* [3].

```
def countdown(start):
    n = start
    def display():
        print('T-minus %d' % n)
    def decrement():
        nonlocal n # Bind to outer n (Python 3 only)
        n -= 1
    while n > 0:
        display()
        decrement()
```

Εικόνα 13 Παράδειγμα εμφωλευμένης συνάρτησης στην Python [3]

Οι συναρτήσεις, όπως οι περισσότερες οντότητες στην Python, είναι αντικείμενα και μπορούν να χρησιμοποιηθούν ως ορίσματα σε άλλες συναρτήσεις, να τοποθετηθούν εντός δομών δεδομένων (data structures) και να επιστραφούν ως αποτέλεσμα από μία συνάρτηση [3].

2.4 Συνάρτηση τύπου closure

Η συνάρτηση τύπου closure είναι ένα συναρτησιακό αντικείμενο που θυμάται τιμές εμφωλευμένων συναρτήσεων ακόμα και αυτές τις τιμές έχουν αφαιρεθεί από την μνήμη.

```
def countdown(n):
    def next():
        nonlocal n
        r = n
        n -= 1
    return r
return next
# Example use
next = countdown(10)
while True:
    v = next() # Get the next value
    if not v: break
```

Εικόνα 14 Παράδειγμα συνάρτησης τύπου closure στην Python [3]

2.5 Συνάρτηση τύπου decorator

Μία συνάρτηση τύπου decorator λαμβάνει μία άλλη συνάρτηση και επεκτείνει την συμπεριφορά της δεύτερης συνάρτησης, χωρίς όμως να την τροποποιεί ουσιαστικά [4]. Οι decorators δηλώνονται με το ειδικό σύμβολο @ [3].

```
@trace
def square(x):
    return x*x

def trace(func):
    if enable_tracing:
        def callf(*args,**kwargs):
            debug_log.write("Calling %s: %s, %s\n" %
                    (func.__name__, args, kwargs))
            r = func(*args,**kwargs)
            debug_log.write("%s returned %s\n" % (func.__name, r))
            return r
        return callf
    else:
        return func
```

Εικόνα 15 Παράδειγμα συνάρτησης τύπου decorator στην Python [3]

Στο προηγούμενο παράδειγμα, η συνάρτηση trace() δημιουργεί μία συνάρτηση – wrapper η οποία έχει ως έξοδο κώδικα debugging, και στην συνέχεια καλεί το αυθεντικό συναρτησιακό αντικείμενο square(x). Η συνάρτηση callif() που επιστρέφεται από την trace() είναι ένα closure που αντικαθιστά την αυθεντική συνάρτηση. Αν η μεταβλητή enabling_tracing έχει τιμή False, η συνάρτηση τύπου decorator trace() επιστρέφει την αυθεντική συνάρτηση square(x) ως έχει [3].

2.6 Συνάρτηση τύπου generator και coroutine

Με τη χρήση της λέξης-κλειδί *yield*, μία συνάρτηση στην Python ορίζει ένα αντικείμενο ως generator. Το generator είναι μία συνάρτηση που παράγει μια ακολουθία τιμών για χρήση σε βρόγχο επανάληψης. Όταν καλείται η συνάρτηση next() , η συνάρτηση τύπου generator εκτελεί εντολές μέχρι να φτάσει στην εντολή yield. Η εντολή yield παράγει ένα αποτέλεσμα και η εκτέλεση των εντολών σταματάει μέχρι να κληθεί ξανά η next(). Η εκτέλεση της συνάρτησης τύπου generator ολοκληρώνεται επιστρέφοντας τιμή ή την εξαίρεση StopIteration στο σημείο που σταματάει η επανάληψη [3].

```
def countdown(n):
    print("Counting down from %d" % n)
    while n > 0:
        yield n
        n -= 1
    return

c=countdown(10)
print(c.__next__())
print(c.__next__())
print(c.__next__())
```

Εικόνα 16 Παράδειγμα συνάρτησης τύπου yield στην Python [3]

Η λέξη-κλειδί yield μπορεί να δηλωθεί στην δεξιά πλευρά της πράξης ανάθεσης. Μία συνάρτηση που χρησιμοποιεί το yield με αυτό τον τρόπο λέγεται coroutine και εκτελείται ανάλογα με τις τιμές που λαμβάνει [3].

```
def receiver():
    print("Ready to receive")
    while True:
        n = (yield)
        print("Got %s" % n)

r = receiver()
r.__next__()
r.send(1)
r.send(2)
r.send("Hello")
```

Εικόνα 17 Παράδειγμα συνάρτησης τύπου coroutine στην Python [3]

2.7 Ανώνυμη συνάρτηση

Οι ανώνυμες συναρτήσεις μπορούν να δημιουργηθούν με τη χρήση του τελεστή lambda: *lambda args : expression* , όπου args λίστα ορισμάτων που χωρίζονται με κόμμα και expression η έκφραση που περιέχει τα ορίσματα. Ο κώδικας της ανώνυμης συνάρτησης πρέπει να περιέχει μία σαφή έκφραση και να μην περιλαμβάνει for ή while βρόγχους επανάληψης. Η κύρια χρήση του τελεστή lambda είναι η κλήση σύντομων συναρτήσεων [3].

```
a = lambda x,y : x+y
r = a(2,3)
print(r)
```

Εικόνα 18 Παράδειγμα ανώνυμης συνάρτησης στην Python [3]

3. Συναρτήσεις στην Matlab

Τόσο τα scripts όσο και οι συναρτήσεις επιτρέπουν την επαναχρησιμοποίηση ακολουθιών εντολών με το να τις αποθηκεύουν σε αρχεία προγράμματος [7]. Οι συναρτήσεις προσφέρουν μεγαλύτερη ευελιξία από τα scripts κυρίως γιατί μπορούν δεχθούν τιμές ως εισόδους και να επιστρέφουν τιμές στην έξοδο. Επιπρόσθετα, οι συναρτήσεις αποφεύγουν την αποθήκευση προσωρινών μεταβλητών και έτσι εκτελούνται γρηγορότερα από τα scripts [7].

3.1 Ορισμός συνάρτησης

Ο τύπος της συνάρτησης ορίζεται εντός ενός αρχείου και όχι στην γραμμή εντολών. Συνήθως οι συναρτήσεις αποθηκεύονται σε ξεχωριστά αρχεία και σε αυτή την περίπτωση είναι καλή πρακτική το όνομα του αρχείου να είναι ίδιο με το όνομα της συνάρτησης [7].

```
function f = fact(n)
    f = prod(1:n);
end


y=fact(4) % Κλήση συνάρτησης στην γραμμή εντολών
```

Εικόνα 19 Παράδειγμα ορισμού και κλήσης συνάρτησης στην Matlab [7]

```
% Αν η συνάρτηση επιστρέφει μία τιμή, τότε το όνομα της μεταβλητής
% μπορεί να οριστεί μετά τη λέξη - κλειδί function
function myOutput = myFunction(x)
% Αν η συνάρτηση επιστρέφει περισσότερες από μία τιμές, τα ονόματα
% των μεταβλητών ορίζονται εντός brackets
function [one,two,three] = myFunction(x)
If there is no output, you can omit it.
% Αν η συνάρτηση δεν επιστρέφει τιμές, παραλείπεται ο ορισμός μεταβλητής
% ή χρησιμοποιούνται brackets χωρίς περιεχόμενο
function myFunction(x)
function [] = myFunction(x)
% Αν η συνάρτηση δέχεται ορίσματα, χρησιμοποιείται παρένθεση και τα
% ορίσματα χωρίζονται με κόμμα
function y = myFunction(one,two,three)
```

Εικόνα 20 Διαφορετικοί τρόποι ορισμού συνάρτησης στην Matlab [7]

Τα έγκυρα ονόματα συναρτήσεων ακολουθούν τους ίδιους κανόνες που ακολουθούν τα ονόματα των μεταβλητών και μπορούν να περιέχουν γράμματα, ψηφία ή τον χαρακτήρα underscore. Το σώμα της συνάρτησης μπορεί να περιέχει εκφράσεις Matlab, εντολές ελέγχου ροής, σχόλια, κενές γραμμές και εμφωλευμένες συναρτήσεις. Οι μεταβλητές που δημιουργούνται εντός μίας συνάρτησης αποθηκεύονται σε συγκεκριμένο χώρο μνήμης που αντιστοιχεί στην συνάρτηση και είναι ανεξάρτητος από τον χώρο μνήμης που αντιστοιχεί στο πρόγραμμα [7].

Τα αρχεία προγράμματος μπορούν να περιέχουν πολλές συναρτήσεις. Αν το αρχείο περιέχει μόνο συναρτήσεις, τότε η πρώτη συνάρτηση είναι η κύρια συνάρτηση και η συνάρτηση την οποία Matlab συνδέει με το όνομα του αρχείου. Οι συναρτήσεις που έπονται της κύριας συνάρτησης ονομάζονται τοπικές συνάρτησης και είναι προσβάσιμες μόνο από το συγκεκριμένο αρχείο [7].

Οι συναρτήσεις τερματίζονται είτε με τη λέξη – κλειδί *end* είτε με τον ορισμό μιας τοπικής συνάρτησης. Η λέξη – κλειδί *end* απαιτείται αν μία συνάρτηση εντός αρχείου περιέχει εμφωλευμένη συνάρτηση και αν η συνάρτηση είναι τοπική συνάρτηση. Αν και η χρήση του *end* είναι προαιρετική, προτείνεται για λόγους εύκολης ανάγνωσης του κώδικα [7].

3.2 Τοπική συνάρτηση

Σε ένα αρχείο συναρτήσεων η πρώτη συνάρτηση στο αρχείο είναι η κύρια συνάρτηση. Αυτή η συνάρτηση είναι προσβάσιμη από άλλα αρχεία ή από τη γραμμή εντολών. Οι επιπρόσθετες συναρτήσεις στο ίδιο αρχείο είναι τοπικές συναρτήσεις και είναι προσβάσιμες μόνο εντός του αρχείου. Οι τοπικές συναρτήσεις είναι όμοιες με τις υπορουτίνες άλλων γλωσσών προγραμματισμού. Οι τοπικές συναρτήσεις δεν έχουν πρόσβαση σε μεταβλητές που χρησιμοποιούνται από άλλες συναρτήσεις εκτός αν περαστούν σε αυτές ως ορίσματα [7].

```
function [avg, med] = mystats(x)
n = length(x);
avg = mymean(x,n);
med = mymedian(x,n);
end
%Τοπική συνάρτηση
function a = mymean(v,n)
a = sum(v)/n;
end
%Τοπική συνάρτηση
function m = mymedian(v,n)
w = sort(v);
if rem(n,2) == 1
    m = w((n + 1)/2);
else
    m = (w(n/2) + w(n/2 + 1))/2;
end
end
```

Εικόνα 21 Παράδειγμα τοπικών συναρτήσεων στην Matlab [7]

3.3 Εμφωλευμένη συνάρτηση

Μία εμφωλευμένη συνάρτηση βρίσκεται εντός του σώματος μίας άλλης συνάρτησης. Η βασική διαφορά μεταξύ των εμφωλευμένων συναρτήσεων και των άλλων τύπων συνάρτησης είναι ότι οι πρώτες μπορούν να έχουν πρόσβαση και να τροποποιήσουν μεταβλητές που ορίζονται στις γονικές συναρτήσεις. Ως εκ τούτου, οι εμφωλευμένες συναρτήσεις μπορούν να χρησιμοποιήσουν μεταβλητές χωρίς απαραιτήτως να περαστούν σε αυτές ως ορίσματα. Επιπλέον, στη γονική συνάρτηση μπορεί να οριστεί ένας handler που περιέχει τα απαραίτητα δεδομένα προκειμένου να εκτελεστεί η εμφωλευμένη συνάρτηση. Οι εμφωλευμένες χρησιμοποιούν μεταβλητές από τρεις πηγές: ορίσματα, μεταβλητές που ορίζονται εντός της εμφωλευμένης συνάρτησης, μεταβλητές που ορίζονται στην γονική συνάρτηση [7].

Δεν επιτρέπεται ο ορισμός εμφωλευμένων συναρτήσεων εντός βρόγχων επανάληψης και εντολών ελέγχου ροής. Οι εμφωλευμένες συναρτήσεις καλούνται είτε απευθείας με το όνομά τους είτε με ένα handler με τη χρήση του τελεστή @ .

```
function A(x, y)              % Κύρια συνάρτηση
B(x,y)
D(y)

   function B(x,y)            % Εμφωλευμένη στην A
   C(x)
   D(y)

      function C(x)           % Εμφωλευμένη στην B
      D(x)
      end
   end

   function D(x)              % Εμφωλευμένη στην A
   E(x)

      function E(x)           % Εμφωλευμένη στην D
      disp(x)
      end
   end
end
```

Εικόνα 22 Παράδειγμα εμφωλευμένων συναρτήσεων στην Matlab [7]

```
function p = makeParabola(a,b,c)
p = @parabola;

   function y = parabola(x)
   y = a*x.^2 + b*x + c;
   end

end
```

Εικόνα 23 Παράδειγμα εμφωλευμένης συνάρτησης με τη χρήση handler στην Matlab [7]

3.4 Ιδιωτική συνάρτηση

Οι ιδιωτικές συναρτήσεις είναι χρήσιμες προκειμένου να μειωθεί η εμβέλεια της συνάρτησης. Μία συνάρτηση χαρακτηρίζεται ιδιωτική με το να αποθηκευτεί σε ένα υποφάκελο με το όνομα *private*. Η ιδιωτική συνάρτηση είναι προσβάσιμη μόνο από τα scripts και τις συναρτήσεις που βρίσκονται στον γονικό φάκελο του υποφακέλου private. Οι ιδιωτικές συναρτήσεις έχουν προτεραιότητα έναντι των κανονικών συναρτήσεων [7].

3.5 Ανώνυμη συνάρτηση

Μία ανώνυμη συνάρτηση δεν αποθηκεύεται σε αρχείο και συνδέεται με μία μεταβλητή τύπου *function_handle*. Οι ανώνυμες συναρτήσεις μπορούν να έχουν πολλές εισόδους και να επιστρέφουν μία τιμή. Επίσης μπορούν να περιέχουν μόνο μία εκτελέσιμη εντολή [7].

Η έκφραση σε μία ανώνυμη συνάρτηση μπορεί να περιλαμβάνει μία άλλη ανώνυμη συνάρτηση, το οποίο είναι χρήσιμο στην περίπτωση να υπολογιστεί ένα εύρος τιμών από την συνάρτηση. Στο παρακάτω παράδειγμα, με τη χρήση δύο ανώνυμων συναρτήσεων υπολογίζεται εξίσωση [7]:

$$g(c) = \int_0^1 x^2 + cx + 1\,dx$$

```
g = @(c) (integral(@(x) (x.^2 + c*x + 1),0,1));
```

Εικόνα 24 Παράδειγμα χρήσης δύο ανώνυμων συναρτήσεων στην Matlab [7]

```
myfunction = @(x,y) (x^2 + y^2 + x*y);

x = 1;
y = 10;
z = myfunction(x,y)
```

Εικόνα 25 Παράδειγμα ορισμού και κλήσης ανώνυμης συνάρτησης στην Matlab [7]

4. Διαφορές μεταξύ των γλωσσών προγραμματισμού C++, Python και Matlab

Οι διαφορές μεταξύ της C++ και της Python συνοψίζονται στον παρακάτω πίνακα [12]:

Παράμετρος	C++	Python
Εκτέλεση	Μεταγλωττιστής	Διερμηνευτής
Χρήση	Όχι εύκολη συγγραφή κώδικα	Εύκολη συγγραφή κώδικα
Φύση γλώσσας	Στατική	Δυναμική
Φορητότητα	Δεν υποστηρίζει	Υποστηρίζει
Συλλέκτης απορριμμάτων (garbage collector)	Δεν υποστηρίζει	Υποστηρίζει
Εγκατάσταση	Εύκολη	Δύσκολη
Τύποι δεδομένων	Συνδέονται με ονόματα μεταβλητών κατά την μεταγλώττιση	Συνδέονται με τιμές μεταβλητών κατά την εκτέλεση
Γρήγορο prototyping	Όχι εφικτό λόγω μεγάλου κώδικα	Εφικτό λόγω σύντομου κώδικα
Συναρτήσεις	Περιορισμοί στον τύπο των παραμέτρων και των επιστρεφόμενων τιμών	Δεν υπάρχουν περιορισμοί στον τύπο των παραμέτρων και των επιστρεφόμενων τιμών
Μεταβλητές	Η πρόσβαση περιορίζεται εντός των βρόγχων επανάληψης	Η πρόσβαση δεν περιορίζεται εντός των βρόγχων επανάληψης
Πολυπλοκότητα σύνταξης	Χρησιμοποιεί blocks και τελικά ερωτηματικά	Δεν χρησιμοποιεί blocks και τελικά ερωτηματικά
Ταχύτητα εκτέλεσης	Γρήγορη	Αργή
Επιδόσεις	Υψηλές	Χαμηλές

Δημοφιλία	Δημοφιλής για ενσωματωμένες ή επιχειρησιακές εφαρμογές	Δημοφιλής για εφαρμογές Machine Learning
Απλότητα και χρηστικότητα	Δύσκολη στην εκμάθηση και χρησιμοποιείται σε εφαρμογές χαμηλού επιπέδου	Εύκολη στην εκμάθηση και χρησιμοποιείται για διαδικτυακές εφαρμογές ή εφαρμογές machine learning

Πίνακας 1 Διαφορές μεταξύ C++ και Python

Ένα από τα κύρια πλεονεκτήματα της Python είναι ότι έχει απλή και καθαρή σύνταξη [12]. Για τους προγραμματιστές της C++, η Python φαίνεται οικεία αλλά ταυτόχρονα και πιο εύκολη λόγω της απουσίας blocks και των τελικών ερωτηματικών για τον τερματισμό των εντολών [12]. Επιπλέον, η Python έχει μία τεράστια βιβλιοθήκη που περιλαμβάνει readers και writers αρχείων ZIP και CSV, XML parsers, βιβλιοθήκες για κάθε διαδικτυακό πρωτόκολλο και τύπο δεδομένων [12]. Λόγω της απόδοσης και της απλότητάς της ενδείκνυται για την ανάπτυξη διαδικτυακών εφαρμογών και εφαρμογές τύπου Machine Learning [12]. Ένα άλλο μεγάλο πλεονέκτημα της Python είναι το duck typing, δηλαδή η κλήση οποιουδήποτε αντικειμένου μέσα στο πρόγραμμα χωρίς να ενδιαφέρει ο τύπος του [12].

Από την άλλη πλευρά, το κυριότερο πλεονέκτημα της C++ είναι οι επιδόσεις της [12]. Η C++ έχει γρηγορότερους χρόνους εκτέλεσης σε σχέση με την Python [12]. Η C++ είναι κατάλληλη για σχεδόν κάθε πλατφόρμα και για ενσωματωμένα συστήματα, ενώ η Python μπορεί να χρησιμοποιηθεί μόνο σε συγκεκριμένες πλατφόρμες που υποστηρίζουν γλώσσες υψηλού επιπέδου [12]. Οι επιδόσεις της C++ ενισχύονται από το γεγονός ότι είναι στατική γλώσσα, το οποίο την καθιστά πιο προβλέψιμη, ενώ η Python είναι η δυναμική γλώσσα προγραμματισμού [12]. Επιπλέον, η C++ μπορεί να χρησιμοποιηθεί για τον προγραμματισμό συστημάτων π.χ. λειτουργικών συστημάτων, καθώς ως γλώσσα χαμηλού επιπέδου βρίσκεται πιο κοντά στο hardware [12].

Οι διαφορές μεταξύ της C++ και της Matlab συνοψίζονται στον παρακάτω πίνακα [13]:

Παράμετρος	C++	Matlab
Ορισμός	Αντικειμενοστραφής γλώσσα προγραμματισμού γενικού σκοπού	Γλώσσα υψηλών επιδόσεων (Matrix Laboratory)
Χρήση	Ενσωματωμένα συστήματα, προβλήματα πραγματικού χρόνου	Τεχνικά προβλήματα σε δύο και τρεις διαστάσεις

Σύνταξη	#include <iostream>
	#include <cmath>
	Using namespace std;
	int main() {
	cout<<pow(3,3)<<endl;
	return 0;
	}
		3^3

Παραγωγή ενσωματωμένου κώδικα	Είναι case-sensitive, τρέχει σε οποιοδήποτε συσκευή και λειτουργικό σύστημα	Παράγει φορητό κώδικα C και C++, δεν είναι case-sensitive
Prototyping	Ο κώδικας είναι μεγάλος, επομένως το prototyping είναι αργό	Ταχύτερο prototyping σε σύγκριση με την C++
Επιδόσεις	Έχει βιβλιοθήκες με ενσωματωμένες συναρτήσεις κάνοντας τα πράγματα ευκολότερα για τους χρήστες	Για βελτιωμένες επιδόσεις απαιτεί εγκατάσταση, μεταγλώττιση κ.ο.κ.
Λογισμικό	Ανοιχτού κώδικα	Εμπορική έκδοση

Πίνακας 2 Διαφορές μεταξύ C++ και Matlab

Το πλεονέκτημα της Matlab είναι ότι επιτρέπει στον χρήστη να τεστάρει τους αλγορίθμους άμεσα χωρίς να χρειάζεται μεταγλώττιση [13]. Το πλεονέκτημα της C++ ως γλώσσα αντικειμενοστραφούς προγραμματισμού είναι ότι ενδείκνυται για προβλήματα του πραγματικού κόσμου λόγω της σύνθετης δομής της, η οποία περιλαμβάνει κλάσεις, τον πολυμορφισμό κ.α. [13].

Η Python θεωρείται καλύτερη επιλογή γλώσσας προγραμματισμού από την Matlab για τους ακόλουθους λόγους [14]:

- ✓ **Είναι δωρεάν**: Το Matlab παραμένει ακριβό λογισμικό, ενώ η Python ως δωρεάν εναλλακτική λύση αποτελεί δελεαστική πρόταση για τις επιχειρήσεις.
- ✓ **Είναι ανοιχτού κώδικα**: Η Python έχει μεγάλη κοινότητα προγραμματιστών με αποτέλεσμα να επιδιορθώνονται τα bugs και να προστίθενται νέα χαρακτηριστικά σε τακτική βάση.
- ✓ **Έχει μέλλον**: Η δημοφιλία της Python αυξάνεται εκθετικά τα τελευταία χρόνια. Αυτό σημαίνει ότι ο προγραμματιστής μπορεί να βρει απαντήσεις σε ερωτήματα που έχει, όπως και παραδείγματα κώδικα για προβλήματα που αντιμετωπίζει.
- ✓ **Έχει περισσότερα χαρακτηριστικά**: Σε αντίθεση με την Matlab, η Python δεν είναι απλά μία scripting γλώσσα προγραμματισμού για προβλήματα μαθηματικών. Η Python είναι μία γλώσσα προστακτικού (imperative) και συναρτησιακού προγραμματισμού, της οποίας το εύρος εφαρμογών εκτείνεται από το crawling διαδικτυακών servers και τον έλεγχο εξωτερικών συσκευών έως την δημιουργία διεπαφών χρήστη.

- ✓ **Είναι φορητή**: Η Python μπορεί να τρέξει σε όλα τα λειτουργικά συστήματα, ακόμα και σε ενσωματωμένα συστήματα με πυρήνα Linux. Απαιτεί απλά την εγκατάσταση των πακέτων Python, τα οποία συνήθως είναι προεγκατεστημένα στο λειτουργικό σύστημα. Αυτό καθιστά εύκολη την ανάπτυξη κώδικα Python σε servers.
- ✓ **Είναι κατάλληλη για Machine Learning**: Με την ανάπτυξη της Τεχνητής Νοημοσύνης (Artificial Intelligence) η Python βρίσκεται έτη φωτός μπροστά από την Matlab. Όλα τα frameworks Τεχνητής Νοημοσύνης, όπως Tensorflow, Keras, PyTorch, Scikit-learn, βασίζονται στην Python.
- ✓ **Είναι ευέλικτη**: Η Python παρέχει την δυνατότητα να επιτευχθεί το ίδιο αποτέλεσμα με πολλούς διαφορετικούς τρόπους.
- ✓ **Επιτρέπει την χρήση πολλών IDEs**: Σε αντίθεση με την Python, η Matlab χρησιμοποιεί μόνο το Ολοκληρωμένο Περιβάλλον Ανάπτυξης λογισμικού (Integrated Development Environment) της Matlab.
- ✓ **Χρησιμοποιεί απλό κώδικα**: Η Python μπορεί να παράξει πολύ πιο απλό και όμορφο κώδικα σε σύγκριση με την Matlab. Για παράδειγμα, στους βρόγχους επανάληψης for παρέχει τη δυνατότητα να προσπελαστούν τόσο ο δείκτης i όσο και η τιμή στην θέση i ενός πίνακα.
- ✓ **Ονοματοδοτεί τα ορίσματα των συναρτήσεων**: Για παράδειγμα, στην Python είναι mean(X, axis=1), ενώ στην Matlab είναι mean(X,1). Αυτό βοηθάει στην ανάγνωση του κώδικα και στο debugging.

Από την άλλη πλευρά, το λογισμικό Matlab ως κλειστού τύπου λογισμικό εγγυάται την ποιότητα και την ορθότητα του προϊόντος [14]. Επιπλέον, το Simulink σε συνδυασμό με δεκάδες εργαλειοθήκες είναι κατάλληλο για πλήθος εφαρμογών στο πεδίο της Μηχανικής [14]. Η Matlab μπορεί να εκτελεστεί αρκετές φορές ταχύτερα από την Python, διότι αρκετές συναρτήσεις στην Matlab είναι προ-μεταγλωττισμένες [14]. Αξίζει να σημειωθεί ότι στην Matlab όλες οι βιβλιοθήκες και τα πακέτα που είναι απαραίτητα για την εκτέλεση του προγράμματος είναι προεγκατεστημένα και δεν απαιτείται η εισαγωγή ή η διαχείρισή τους από τον χρήστη, όπως συμβαίνει στην Python [14].

5. Ανάπτυξη εφαρμογών

5.1 Εύρεση ν+1 όρου ακολουθίας

Το πρόβλημα ορίζεται ως εξής [11]: Να βρεθεί ο όρος n της ακολουθίας που ξεκινάει με S(0) = start και διέπετε από τον κανόνα «Δοθέντος ενός όρου S(i) της ακολουθίας, ο επόμενος όρος της ακολουθίας S(i+1) μπορεί να βρεθεί μετρώντας τα γράμματα (αγνοώντας τα κενά) που προκύπτουν από την αναπαράσταση της ακολουθίας σε δυαδική μορφή. Για παράδειγμα, αν S(0)=5, τότε η δυαδική μορφή του 5 είναι 101 και η έκφραση σε γράμματα είναι “one zero one”. Μετρώντας τα γράμματα προκύπτει ότι ο επόμενος όρος της ακολουθίας είναι S(1) = 10.

Η ορθότητα των προγραμμάτων επιβεβαιώνεται με την εκτέλεσή τους για συγκεκριμένες εισόδους. Οι τρεις υλοποιήσεις του προβλήματος εκτελέστηκαν για τις εισόδους του παρακάτω πίνακα και σε κάθε περίπτωση τα αποτελέσματα ήταν ίδια.

Input (start, n)	Output
(5,2)	14
(891,3)	16
(263,6)	18
(480,16)	18
(481,32)	13
(2466262232, 131212325)	18
(99999999999999999, 99999999999999999)	18

Πίνακας 3 Τιμές εισόδου και εξόδου προγράμματος «Εύρεση νιοστού όρου ακολουθίας»

Η υλοποίηση σε C++ χρησιμοποιεί περισσότερες σειρές κώδικα λόγω του γεγονότος ότι δηλώνονται όλες οι μεταβλητές και χρησιμοποιούνται περισσότεροι χαρακτήρες για τη σύνταξη του κώδικα. Και στις τρεις περιπτώσεις η συνάρτηση count υλοποιείται με παρόμοιο τρόπο. Αρχικά ο ακέραιος αριθμός μέσω ενός loop μετατρέπεται σε δυαδικό αριθμό και τα δυαδικά ψηφία αποθηκεύονται σε μονοδιάστατο πίνακα – διάνυσμα, ο οποίος αντιστρέφεται. Για κάθε δυαδικό ψηφίο, προστίθεται στην μεταβλητή αθροίσματος το 4 αν πρόκειται για το 0 και 3 αν πρόκειται για το 1. Τελικά επιστρέφεται η μεταβλητή αθροίσματος. Η συνάρτηση count καλείται τόσες φορές όσες ορίζει η παράμετρος n, προκειμένου να βρεθεί ο ν+1 όρος της ακολουθίας.

Από την υλοποίηση του προγράμματος σε Matlab φαίνεται η διαφορά αναφορικά με την επιστροφή τιμής από συνάρτηση στην Matlab. Ενώ στην C++ και στην Python η επιστρεφόμενη τιμή ορίζεται με την εντολή return, στην Matlab επιστρέφεται η τιμή της μεταβλητής στην οποία ανατίθεται η τιμή της συνάρτησης στην πρώτη σειρά, όπου ορίζεται το όνομα της συνάρτησης.

Τόσο η Matlab όσο και η Python δεν χρησιμοποιούν τύπους δεδομένων κατά την δήλωση των μεταβλητών ή των παραμέτρων των συναρτήσεων. Αντίθετα, στην C++ είναι απαραίτητο να καθοριστεί ο τύπος δεδομένων κάθε μεταβλητής και παραμέτρου συνάρτησης.

5.2 Το πρόβλημα του περιοδεύοντος πωλητή

Το πρόβλημα του περιοδεύοντος πωλητή απαντά στο ερώτημα [11]: «Δοθέντων μίας λίστας με πόλεις και αποστάσεις μεταξύ κάθε ζεύγους πόλεων, ποια είναι η μικρότερη δυνατή διαδρομή που διαπερνά κάθε πόλη και επιστρέφει στην αρχική πόλη;». Η μικρότερη απόσταση βρίσκεται μέσω ενός άπληστου αλγορίθμου (greedy algorithm). Ο άπληστος αλγόριθμος δεν βρίσκει πάντα τη βέλτιστη λύση, αλλά μπορεί να προσεγγίσει τη βέλτιστη λύση σε λογικό χρονικό διάστημα. Ο άπληστος αλγόριθμος ξεκινά από την πρώτη είσοδο και πάντα επιλέγει την κοντινότερο σημείο από το τρέχον σημείο. Αυτό συνεχίζεται μέχρι να μην απομείνουν σημεία και το τελευταίο σημείο να συνδεθεί με το αρχικό σημείο. Χρησιμοποιείται η Ευκλείδια απόσταση ως η απόσταση μεταξύ δύο σημείων. Στην περίπτωση που υπάρχουν σημεία με την ίδια απόσταση, επιλέγεται το σημείο που προσπελαύνεται πρώτο στη λίστα.

Το πρόγραμμα δέχεται ως εισόδους τον αριθμό των πόλεων και τις συντεταγμένες κάθε πόλης και επιστρέφει στρογυλλοποιημένη την τιμή της μικρότερης διαδρομής.

Η ορθότητα των προγραμμάτων επιβεβαιώνεται με την εκτέλεσή τους για συγκεκριμένες εισόδους. Οι τρεις υλοποιήσεις του προβλήματος εκτελέστηκαν για τις εισόδους του παρακάτω πίνακα και σε κάθε περίπτωση τα αποτελέσματα ήταν ίδια.

Input (N, XY[N][2])	Output(Dmin)
5 9 12 24 15 12 30 4 3 13 27	71
12 4 5 12 80 65 18 39 29 99 11 84 31 9 9 54 49 16 27 31 67 0 71 60 0	403
14 62 6 21 19 23 25 11 29 73 10 14 55 47 3 18 71 4 7 52 93 12 31 76 60 81 72 59 34	327
10 0 26 18 27 33 15 25 11 41 36 80 41 9 10 34 15 71 11 84 6	252

Πίνακας 4 Τιμές εισόδου και εξόδου προγράμματος «Το πρόβλημα του περιοδεύοντος πωλητή»

Το πρόγραμμα στην C++ διαβάζει τις συντεταγμένες των πόλεων, οι οποίες δηλώνονται ως καθολική μεταβλητή με τη δομή δεδομένων pair. Με την συνάρτηση nth_element ταξινομούνται οι συντεταγμένες των πόλεων έτσι ώστε σε κάθε επανάληψη η επόμενη πόλη να απέχει από την τρέχουσα (city) την μικρότερη απόσταση. Αυτό επιτυγχάνεται θέτοντας ως iterator το αντικείμενο city, το οποίο λαμβάνει όλες τις τιμές της δομής δεδομένων cities, και καλώντας την συνάρτηση distance(const coords &a, const coords &b) για διαδοχικά στοιχεία της δομής δεδομένων cities. Η συνάρτηση distance(const coords &a, const coords &b) επιστρέφει την καρτεσιανή απόσταση δύο πόλεων. Σε κάθε επανάληψη και αφού γίνει η ταξινόμηση της δομής δεδομένων cities, προστίθεται στην μεταβλητή total η απόσταση μεταξύ της τρέχουσας πόλης και της επόμενης. Τελικά, προστίθεται η απόσταση μεταξύ της τελευταίας και της αρχικής πόλης, και στρογγυλοποιείται το αποτέλεσμα.

Το πρόγραμμα στην Matlab χρησιμοποιεί δύο συναρτήσεις: την distance(c1x,c1y,c2x,c2y) και την findmin(c,cx,m). Η πρώτη υπολογίζει και επιστρέφει την καρτεσιανή απόσταση μεταξύ δύο πόλεων. Η συνάρτηση findmin(c,cx,m) δέχεται ως όρισματα τις συντεταγμένες των πόλεων, τις συντεταγμένες της τρέχουσας πόλης και των αριθμό των πόλεων, και επιστρέφει την ελάχιστη απόσταση από την τρέχουσα πόλη και τις συντεταγμένες της πόλης (η οποία τίθεται ως η επόμενη τρέχουσα πόλη) από την οποία η τρέχουσα πόλη απέχει λιγότερο. Στη συνέχεια, στην κύρια συνάρτηση, η νέα τρέχουσα πόλη αφαιρείται από τον δισδιάστατο πίνακα συντεταγμένων points. Τελικά, όπως και στο πρόγραμμα στην C++, προστίθεται στη συνάρτηση total η απόσταση μεταξύ της αρχικής και της τελευταίας πόλης, και στρογγυλοποιείται το αποτέλεσμα.

Το πρόγραμμα στην Python ακολουθεί την ίδια λογική με το πρόγραμμα στην Matlab. Λόγω των πλεονεκτημάτων της Python, ο κώδικας είναι πιο περιεκτικός π.χ. αντί για δισδιάστατο πίνακα για τις συντεταγμένες των πόλεων χρησιμοποιείται λίστα και η αφαίρεση της τρέχουσας πόλης γίνεται με την συνάρτηση remove() αντί για τη χρήση τεσσάρων γραμμών κώδικα στην Matlab. Επιπλέον, η χρήση της συνάρτησης enumerate() στον βρόγχο επανάληψης for καθιστά τον κώδικα πιο ευέλικτο, γιατί με αυτόν τον τρόπο, τόσο ο δείκτης του βρόγχου επανάληψης όσο και η τιμή του είναι διαθέσιμα, χωρίς να απαιτείται ο ορισμός του.

Λόγω της συνάρτησης nth_element στο πρόγραμμα στην C++, το πρόγραμμα στην C++ θεωρείται πιο αποδοτικό σε σχέση με τις άλλες δύο υλοποιήσεις, γιατί με αυτό τον τρόπο αποφεύγεται η χρήση της συνάρτησης findmin, το οποίο συνεπάγεται την χρήση ενός επιπλέον βρόγχου επανάληψης, για την εύρεση της ελάχιστης απόστασης.

5.3 Συνδυασμοί στο παιχνίδι Mölkky

Τα τελευταία χρόνια γίνεται όλο και πιο δημοφιλές ένα πολύ παλιό Φιλανδικό παιχνίδι που συνδυάζει ικανότητα και τύχη, το Mölkky [11]. Οι κανόνες του παιχνιδιού είναι οι εξής [11]: Οι παίκτες ρίχνουν ένα ξύλινο κύλινδρο για ανατρέψουν αριθμημένες κορίνες (από 1 εώς 12), οι οποίες είναι τοποθετημένες 3 με 4 μέτρα μακριά, με σκοπό να συγκεντρώσουν 50 πόντους. Οι αριθμοί συγκεντρώνονται ως εξής [11]: Αν ο παίκτης ανατρέψει μόνο μία κορίνα, τότε σκοράρει τους πόντους που αναγράφονται πάνω στην κορίνα. Αν ο παίκτης ανατρέψει περισσότερες από μία κορίνες, τότε λαμβάνει πόντους ίσους με τον αριθμό των κορίνων που ανέτρεψε.

Ο σκοπός του συγκεκριμένου προβλήματος είναι, δοθέντος ενός αρχικού σκορ, να βρεθεί ο αριθμός των συνδυασμών που υπάρχει έτσι ώστε ο παίκτης να συγκεντρώσει 50 πόντους. Για παράδειγμα, αν το αρχικό σκορ είναι 47, τότε οι πιθανοί συνδυσμοί ώστε ο παίκτης να συγκεντρώσει τους 50 πόντους είναι παρακάτω (με P παριστάνεται η περίπτωση στην οποία ανατρέπεται μία κορίνα και με αριθμό παριστάνεται η περίπτωση στην οποία ανατρέπονται περισσότερες από μία κορίνες) [11]: [(P1,P1,P1) (P2,P1) (P1,P2) (2,P1) (P1,2) (P3) (3)].

Το πρόγραμμα δέχεται ως είσοδο το αρχικό σκορ και εκτυπώνει τον ζητούμενο αριθμό συνδυασμών. Ο μέγιστος αριθμός ρίψεων για τον υπολογισμό του αριθμού των συνδυασμών είναι 4 [11].

Η ορθότητα των προγραμμάτων επιβεβαιώνεται με την εκτέλεσή τους για συγκεκριμένες εισόδους. Οι τρεις υλοποιήσεις του προβλήματος εκτελέστηκαν για τις εισόδους του παρακάτω πίνακα και σε κάθε περίπτωση τα αποτελέσματα ήταν ίδια.

Input(n)	Output(Combinations)
47	7
38	1776
49	1
43	176
0	0
2	16
25	16148
24	16552

Πίνακας 5 Τιμές εισόδου και εξόδου προγράμματος «Συνδυασμοί στο παιχνίδι Mölkky»

Και στις τρεις υλοποιήσεις χρησιμοποιήθηκε η ίδια λογική με τη χρήση αναδρομικής συνάρτησης για τη σύνταξη των προγραμμάτων . Στην κύρια

συνάρτηση διαβάζεται το αρχικό σκορ. Κατόπιν καλείται η συνάρτηση combinations με σκοπό να βρεθεί ο αριθμός των πιθανών συνδυασμών που επιτυγχάνει το σκορ των 50 πόντων. Όταν το σκορ γίνεται 50 η συνάρτηση combinations επιστρέφει την τιμή 1, ενώ όταν το σκορ γίνεται μεγαλύτερο από 50 ή αριθμός των ρίψεων ξεπεράσει το 4, επιστρέφει την τιμή 0. Σε κάθε άλλη περίπτωση, δοκιμάζει όλους τους πιθανούς συνδυασμούς ξεκινώντας από το 1 και το 2, καλώντας αναδρομικά τον εαυτό της και μεταβάλλοντας το όριο των ρίψεων κάθε φορά κατά μία μονάδα.

5.4 Υπολογισμός τετραγώνων σε αγροτεμάχιο

Το συγκεκριμένο πρόβλημα περιγράφεται ως εξής [11]: «Ένας αγρότης θέλει να ορίσει τετράγωνα στο αγροτεμάχιό του προκειμένου να κατασκευάσει ένα φράκτη. Για τον σκοπό αυτό, χρειάζεται να τοποθετήσει 4 πασσάλους σε διακριτά σημεία που αντιστοιχούν στις γωνίες του τετραγώνου. Οι πάσσαλοι μπορούν να τοποθετηθούν μόνο σε συγκεκριμένα σημεία όπου το έδαφος είναι τόσο μαλακό ώστε να μπορούν οι πάσσαλοι να στερεωθούν σε κατάλληλο βάθος. Δίνονται οι συντεταγμένες των θέσεων στις οποίες μπορούν να τοποθετηθούν οι πάσσαλοι και ζητείται να βρεθεί ο αριθμός των διαφορετικών τετραγώνων που μπορούν να σχηματιστούν. »

Το πρόγραμμα δέχεται ως εισόδους τον αριθμό N των διακριτών θέσεων και τις συντεταγμένες (X,Y) των θέσεων και έχει ως έξοδο ένα ακέραιο που αντιστοιχεί στον αριθμό των τετραγώνων που σχηματίζονται για τις συγκεκριμένες εισόδους.

Η ορθότητα των προγραμμάτων επιβεβαιώνεται με την εκτέλεσή τους για συγκεκριμένες εισόδους. Οι τρεις υλοποιήσεις του προβλήματος εκτελέστηκαν για τις εισόδους του πίνακα 4 (Παράρτημα Β) και σε κάθε περίπτωση τα αποτελέσματα ήταν ίδια.

Το πρόγραμμα στην C++ αποθηκεύει τις συντεταγμένες των θέσεων σε ένα πίνακα δομών. Η συνάρτηση squares(position x[], int m) επιστρέφει τον αριθμό των δυνατών τετραγώνων. Αρχικά η συνάρτηση προσπελαύνει τις συντεταγμένες όλων των θέσεων υποθέτοντας ότι πρόκειται για αντιδιαμετρικά σημεία τετραγώνου, και υπολογίζει τις υπόλοιπες δύο κορυφές που λείπουν. Στον τρίτο βρόγχο επανάληψης εξετάζει αν οι δύο κορυφές του τετραγώνου που υπολογίστηκαν ανήκουν στα δοθέντα σημεία. Αν ανήκουν, τότε σημαίνει ότι βρέθηκε ένα τετράγωνο που μπορεί να σχηματιστεί από τα δοθέντα σημεία. Επειδή κάθε τετράγωνο που μπορεί να σχηματιστεί από τα δοθέντα σημεία υπολογίζεται δύο φορές, η επιστρεφόμενη τιμή της συνάρτησης διαιρείται με το 2 κατά την εκτύπωση της εξόδου.

Το πρόγραμμα στην Matlab είναι όμοιο με το πρόγραμμα στην C++ χρησιμοποιώντας στην συνάρτηση ένα πίνακα δομών τύπου struct για την

αποθήκευση των συντεταγμένων των πασσάλων και την συνάρτηση squares(p,n) για τον υπολογισμό των τετραγώνων που μπορούν να σχηματιστούν.

Η υλοποίηση στην Python μειώνει την τάξη πολυπλοκότητας του προγράμματος από $O(n^3)$ σε $O(n^2)$ καθώς χρησιμοποιεί μόνο δύο βρόγχους επανάληψης αντί για τρεις, μειώνοντας τον χρόνο εκτέλεσης του προγράμματος. Ορίζονται τρεις καθολικοί πίνακες, για τις συντεταγμένες x και y, και ο λογικός πίνακας t που καταχωρεί την τιμή true για τα δοθέντα σημεία. Η συνάρτηση pegs(i,j) εφαρμόζει διαφορετικά κριτήρια για την εύρεση των υπόλοιπων δύο κορυφών του τετραγώνου και επιβεβαιώνει ότι οι δύο κορυφές ανήκουν στα δοθέντα σημεία μέσω της συνάρτησης examine(x,y).

5.5 Σκοτεινά σημεία δωματίου

Το συγκεκριμένο πρόβλημα έχει ως εξής [11]: «Σε ένα τετράγωνο δωμάτιο πλευράς N μέτρων τοποθετούνται κεριά. Κάθε κερί παράγει φωτισμό έντασης L στο σημείο που βρίσκεται και κάθε σημείο του τετραγώνου έχει ένταση φωτισμού μία μονάδα λιγότερη από τα διπλανά σημεία. Κάθε σημείο χωρίς φωτισμό έχει την τιμή 0. Αν ένα σημείο συνορεύει με δύο κεριά, τότε έχει το μεγαλύτερο δυνατό φωτισμό. Ο χάρτης του δωματίου δίνεται ως πίνακας σημείων με τον χαρακτήρα “C” στα κελιά που έχουν κερί και τον χαρακτήρα “X” στα υπόλοιπα. Το ζητούμενο είναι να βρεθεί ο αριθμός των σκοτεινών σημείων του δωματίου».

Για παράδειγμα, ένα δωμάτιο με N=5 και L=3 έχει την παρακάτω απεικόνιση σύμφωνα με τα δεδομένα του προβλήματος:

X	X	X	X	X
X	C	X	X	X
X	X	X	X	X
X	X	X	X	X
X	X	X	X	X

2	2	2	1	0
2	3	2	1	0
2	2	2	1	0
1	1	1	1	0
0	0	0	0	0

Συνεπώς ο αριθμός των σκοτεινών σημείων του δωματίου ή των κελιών του πίνακα με τη τιμή 0 είναι 9.

Η ορθότητα των προγραμμάτων επιβεβαιώνεται με την εκτέλεσή τους για συγκεκριμένες εισόδους. Οι τρεις υλοποιήσεις του προβλήματος εκτελέστηκαν για τις εισόδους του πίνακα 4 (Παράρτημα Β) και σε κάθε περίπτωση τα αποτελέσματα ήταν ίδια.

Στο πρόγραμμα στην C++ ορίζονται ως καθολικές μεταβλητές, το μήκος της πλευράς του τετραγώνου N και το δισδιάστατο διάνυσμα ακεραίων x. Στη συνάρτηση main() διαβάζονται οι μεταβλητές N, L και ο δισδιάστατος πίνακας-pointer string *map[N], ο οποίος παριστάνει τα σημεία του δωματίου. Οι συντεταγμένες των σημείων, τα οποία έχουν κερί, αποθηκεύονται στο διάνυσμα x. Κατόπιν καλείται η συνάρτηση darkspots(L,map) για τον υπολογισμό του αριθμού των σκοτεινών σημείων. Με τη χρήση τριών βρόγχων επανάληψης, εντοπίζονται τα σημεία του δωματίου που έχουν τιμή διαφορετική του μηδενός (“0”), συγκρίνοντας την απόσταση των σημείων με κερί με τις τρέχουσες συντεταγμένες έτσι ώστε να είναι μικρότερη από την τιμή L. Σε αυτά τα φωτεινά σημεία ανατίθεται η τιμή “1”. Στη συνέχεια μετρούνται τα σημεία με τιμή “Χ”, δηλαδή τα σκοτεινά σημεία, και επιστρέφεται το σύνολο τους.

Το πρόγραμμα στην Matlab υλοποιείται με όμοιο τρόπο. Φαίνεται ότι ο κώδικας στην Matlab είναι πιο απλός καθώς απουσιάζουν οι δηλώσεις των μεταβλητών που υπάρχουν στην C++. Αυτό έχει ως αποτέλεσμα ο κώδικας να είναι πιο κατανοητός και να είναι μικρότερος, π.χ. η αρχικοποίηση του δισδιάστατου πίνακα map γίνεται απευθείας στον βρόγχο επανάληψης, ενώ στην C++ απαιτείται η δήλωση του pointer και ένας επιπλέον βρόγχος επανάληψης για τη δήλωση – δημιουργία των στηλών του πίνακα.

Ο κώδικας του προγράμματος στην Python είναι ακόμα πιο περιληπτικός. Σε αυτό συντελούν οι δυνατότητες που παρέχει η Python σχετικά με τη σύνταξη των βρόγχων επανάληψης και με τον ορισμό των δομών δεδομένων.

5.6 Μέγιστη επιτρεπόμενη κλίση

Το συγκεκριμένο πρόβλημα Φυσικής ορίζεται ως εξής [11]: «Κατά την διάρκεια διαγωνισμούς ταχύτητας μοτοσυκλετών, οι ταχύτητες που επιτυγχάνονται είναι πολύ υψηλές. Οι υψηλές ταχύτητες είναι συχνά η αιτία ατυχημάτων, ειδικά στις στροφές. Όταν η μοτοσυκλέτα εισέρχεται σε μία στροφή, ασκείται πάνω της η φυγόκεντρος δύναμη η οποία ωθεί την μοτοσυκλέτα εκτός στροφής. Προκειμένου οι οδηγοί να αντιμετωπίσουν αυτό το φαινόμενο, χρησιμοποιούν την τεχνική της κεντρομόλου κλίσης, δηλαδή δίνουν κλίση στην μοτοσυκλέτα προς την φορά της στροφής για να αντισταθμιστεί η φυγόκεντρος δύναμη. Αυτή η τεχνική φτάνει στα όρια της όταν η ταχύτητα είναι πολύ υψηλή, καθώς η μοτοσυκλέτα ολισθαίνει με συνέπεια την πτώση του οδηγού.»

Ο σκοπός αυτού του προβλήματος είναι η εύρεση της μέγιστης σταθερής ταχύτητας της μοτοσυκλέτας ούτως ώστε να περάσει όλες τις στροφές της διαδρομής χωρίς να ολισθήσει και ταυτόχρονα να τερματίσει πρώτη. Η κατάταξη του διαγωνισμού διαμορφώνεται με τον εξής τρόπο: Αν V είναι ταχύτητα και

Va>Vb>Vc, τότε πρώτος είναι ο a, δεύτερος ο b και τρίτος ο c. Αν Va>Vb>Vc και ο a πέσει τότε πρώτος είναι ο b, δεύτερος ο c και τρίτος ο a. Αν Va>Vb>Vc, ο b πέσει στη $2^{η}$ στροφή και ο a πέσει στη $5^{η}$ στροφή, τότε πρώτος είναι ο c, δεύτερος a και τρίτος ο b. Αν Va>Vb>Vc και ο a πέσει στην ίδια στροφή με τον b, τότε πρώτος είναι ο c, δεύτερος ο a και τρίτος ο b. Στην τελική κατάταξη πρώτος πρέπει να είναι ο y.

Μία μοτοσυκλέτα ολισθαίνει αν η γωνία ως προς το έδαφος είναι μικρότερη από 30^{o}. Για τους απαραίτητους υπολογισμούς δίνεται η σχέση $\tan(\theta) = v^2 / (r*g)$, όπου v η ταχύτητα (m/s) της μοτοσυκλέτας, r η ακτίνα (m) της στροφής, g (m/s^2) η επιτάχυνση της βαρύτητας και θ η γωνία ως προς τον κάθετο άξονα σε μοίρες.

Το πρόγραμμα έχει ως εισόδους τον αριθμό των μοτοσυκλετών n, τον αριθμό των στροφών της διαδρομής v, την σταθερή ταχύτητα κάθε μοτοσυκλέτας σε m/s και την ακτίνα κάθε στροφής της διαδρομής σε m. Οι έξοδοι του προγράμματος είναι η μέγιστη δυνατή σταθερή ταχύτητα για να τη αποφυγή της ολίσθησης στις στροφές της διαδρομής και η κατάταξη του διαγωνισμού.

Η ορθότητα των προγραμμάτων επιβεβαιώνεται με την εκτέλεσή τους για συγκεκριμένες εισόδους. Οι τρεις υλοποιήσεις του προβλήματος εκτελέστηκαν για τις εισόδους του πίνακα 4 (Παράρτημα Β) και σε κάθε περίπτωση τα αποτελέσματα ήταν ίδια.

Όλες οι υλοποιήσεις βασίζονται στις κύριες συναρτήσεις των προγραμμάτων. Το πρόγραμμα στην C++ περιλαμβάνει την κλάση Motorcycle, η οποία περιέχει ένα constructor και τις μεταβλητές της ταχύτητας και του αριθμού του διαγωνιζόμενου/κάθε μοτοσυκλέτας. Επιπλέον, η συνάρτηση comp1 τύπου comparator έχει ως σκοπό την ταξινόμηση των αντικειμένων Motorcycle κατά φθίνουσα σειρά με βάση την ταχύτητα. Η κύρια συνάρτηση main() διαβάζει τις απαραίτητες μεταβλητές n και v, ορίζει ένα πίνακα string με τους χαρακτήρες του αλφαβήτου, δημιουργεί το διάνυσμα mc με αντικείμενα Motorcycle και το διάνυσμα bends με τις ακτίνες των στροφών. Στη συνέχεια, υπολογίζει και αποθηκεύει στο διάνυσμα minSpeed τις μέγιστες δυνατές ταχύτητες σε κάθε στροφή ούτως ώστε να μην ολισθήσει μία μοτοσυκλέτα. Η ελάχιστη ταχύτητα από τις προηγούμενες είναι η μέγιστη σταθερή ταχύτητα που μπορεί να έχει μία μοτοσυκλέτα προκειμένου να διανύσει όλη την διαδρομή χωρίς να ολισθήσει. Στον αμέσως επόμενο βρόγχο επανάληψης, προσπελαύνονται τα αντικείμενα του πίνακα mc για κάθε στροφή και τροποποιούνται οι ταχύτητες τους αφαιρώντας από αυτές το 100 επί την μεταβλητή st (counter των περιπτώσεων ολίσθησης) στην περίπτωση που προκαλούν ολίσθηση, καθώς η ελάχιστη γωνία ως προς το έδαφος πρέπει να είναι 30^{o} και η μέγιστη γωνία ως προς τον κάθετο άξονα πρέπει να είναι 60^{o}. Η συνάρτηση sort() ταξινομεί τον πίνακα mc με βάση την συνάρτηση comp1. Τελικά εκτυπώνονται τα ζητούμενα όπως ορίζεται στην εκφώνηση του προβλήματος.

Το πρόγραμμα στην Matlab ακολουθεί την ίδια φιλοσοφία με το πρόγραμμα στην C++. Η ειδοποιός διαφορά είναι η χρήση του τύπου δεδομένων struct στην Matlab αντί για κλάση όπως συμβαίνει στην C++, με αποτέλεσμα να μειώνονται σημαντικά οι γραμμές κώδικα, καθώς παραλείπεται η κλάση και συνάρτηση τύπου comparator για την ταξινόμηση των αντικειμένων κατά φθίνουσα σειρά με βάση την ταχύτητα.

Το πρόγραμμα στην Python αποτελεί την πιο σύντομη και περιεκτική λύση, αναδεικνύοντας τα συγκριτικά πλεονεκτήματα της Python. Στην λίστα speeds αποθηκεύονται οι ταχύτητες των μοτοσυκλετών, οι οποίες ταξινομούνται κατά φθίνουσα σειρά με χρήση της συνάρτησης βιβλιοθήκες sort(), ενώ παραλείπεται η δημιουργία λίστας για τις ακτίνες των στροφών. Στον πίνακα άγνωστου μεγέθους names χρησιμοποιείται ως δείκτης η ταχύτητα και ως τιμή, η τιμή του χαρακτήρα (όνομα μοτοσυκλέτας/διαγωνιζόμενου) που αντιστοιχεί στον αριθμό της μοτοσυκλέτας κάνοντας χρήση casting. Στον δεύτερο βρόγχο επανάληψης του προγράμματος διαβάζονται οι ακτίνες των στροφών και αποθηκεύονται στην λίστα ms οι μέγιστες επιτρεπόμενες ταχύτητες σε κάθε στροφή. Για κάθε μία από αυτές τις ταχύτητες, ταξινομείται η λίστα speeds με κριτήριο κάθε στοιχείο της να είναι μεγαλύτερο από την μέγιστη επιτρεπόμενη ταχύτητα στη συγκεκριμένη στροφή, με αποτέλεσμα υψηλές ταχύτητες που υπερβαίνουν τις μέγιστες επιτρεπόμενες ταχύτητες στις στροφές (με άλλα λόγια οι μοτοσυκλέτες με αυτές τις ταχύτητες ολισθαίνουν στις στροφές και δεν ολοκληρώνουν την διαδρομή) να μετακινούνται στο τέλος της ουράς της λίστας speeds. Τελικά εκτυπώνονται τα ζητούμενα σύμφωνα με την εκφώνηση του προβλήματος.

6. Συμπεράσματα

Στην παρούσα πτυχιακή εργασία αναλύθηκαν εις βάθος οι συναρτήσεις στις γλώσσες προγραμματισμού C++, Python και Matlab. Είναι σαφές ότι η συγγραφή κώδικα στην Python είναι ευκολότερη και η εκμάθησή της είναι επίσης πιο ταχεία, πλεονεκτήματα τα οποία συντελούν στο γρήγορο prototyping. Επιπλέον, η Python, όπως και η Matlab, δεν απαιτεί την δήλωση του τύπου δεδομένων των μεταβλητών, χαρακτηριστικό που προσθέτει ευελιξία στην ανάπτυξη του προγράμματος. Στην C++ χρειάζεται να δηλώνεται τόσο ο τύπος δεδομένων των μεταβλητών όσο και των συναρτήσεων και των ορισμάτων τους. Η Python δεν χρησιμοποιεί τελικά ερωτηματικά στις εντολές και blocks στις συναρτήσεις, η Matlab χρησιμοποιεί τελικά ερωτηματικά στις εντολές, ενώ η C++ χρησιμοποιεί και τα δύο. Ένα ακόμα συγκριτικό πλεονέκτημα της Python σε σχέση με τις άλλες δύο γλώσσες προγραμματισμού είναι το duck typing ή δυναμική σύνταξη κώδικα, το οποίο επιτρέπει την κλήση οποιοδήποτε αντικειμένου ή την μεταβολή του τύπου δεδομένων σε οποιοδήποτε σημείο του κώδικα, καθώς στην Python δίνεται περισσότερη έμφαση στο τι μπορεί να κάνει το αντικείμενο και όχι στο τι είναι το αντικείμενο. Επιπρόσθετα, η ονοματοδοσία των ορισμάτων των συναρτήσεων που παρέχει η Python καθιστά πιο ευανάγνωστο τον κώδικα και αποτελεσματικότερο το debugging.

Αν και η C++ προσφέρει υψηλότερες επιδόσεις και ταχύτητα εκτέλεσης και η Matlab ενδείκνυται για την επίλυση πλήθους τεχνικών προβλημάτων, η Python παραμένει μία γλώσσα προγραμματισμού ανοιχτού, η οποία υποστηρίζεται από μία μεγάλη κοινότητα, φορητή και συμβατή με πολλά Ολοκληρωμένα Περιβάλλοντα Ανάπτυξης λογισμικού. Συνεπώς, οι επιδόσεις της διαρκώς αυξάνονται, τα bugs της επιδιορθώνονται και το εύρος εφαρμογών της επεκτείνεται.

Η ανάπτυξη των ίδιων προγραμμάτων στις C++, Python και Matlab συνετέλεσε στην εμβάθυνση στην θεωρία των συναρτήσεων και στην συγγραφή κώδικα σε πρακτικό επίπεδο και ανέδειξε τις διαφορές στις τρεις γλώσσες προγραμματισμού.

Ειδικότερα, στο μαθηματικό πρόβλημα «Εύρεση v+1 όρου ακολουθίας» έγινε σαφές ότι η υλοποίηση των συναρτήσεων σε Matlab είναι πιο απλή και ευέλικτη σε σχέση με την C++, ενώ τόσο στην Python όσο και στην Matlab το πρόγραμμα είναι πιο λιτό λόγω της απουσίας δήλωσης του τύπου των δεδομένων των μεταβλητών και των παραμέτρων των συναρτήσεων.

Στο αλγοριθμικό «Πρόβλημα του περιοδεύοντος πωλητή» αναδείχθηκε η σημασία των συναρτήσεων βιβλιοθήκης. Εν προκειμένω, η υλοποίηση σε C++ θεωρήθηκε η καλύτερη δυνατή λόγω του γεγονότος ότι η χρήση της συνάρτησης nth_element μείωσε κατά πολύ τον απαιτούμενο κώδικα.

Στο μαθηματικό πρόβλημα «Συνδυασμοί στο παιχνίδι Mölkky» παρουσιάστηκε η χρήση των αναδρομικών συναρτήσεων, η οποία έχει την ίδια εφαρμογή και στις τρεις γλώσσες προγραμματισμού.

Στο μαθηματικό πρόβλημα «Υπολογισμός τετραγώνων σε αγροτεμάχιο» επιδεικνύονται τα σημαντικά αποτελέσματα που μπορεί να επιφέρει η ευελιξία της σύνταξης κώδικα στην Python, όπως η μείωση της πολυπλοκότητας του προγράμματος από $O(n^3)$ σε $O(n^2)$, και τελικά η μείωση του χρόνου εκτέλεσης του προγράμματος.

Στο αλγοριθμικό πρόβλημα «Σκοτεινά σημεία δωματίου» αναδεικνύεται ακόμη μία φορά το συγκριτικό πλεονέκτημα της Python έναντι των άλλων δύο γλωσσών προγραμματισμού σχετικά με την σύνταξη των βρόγχων επανάληψης και τον ορισμό των δομών δεδομένων.

Στο πρόβλημα Φυσικής «Μέγιστη επιτρεπόμενη κλίση» γίνεται περισσότερο σαφές το συγκριτικό πλεονέκτημα της Python σχετικά με τον ορισμό και τη χρήση των δεδομένων και το γεγονός ότι η Python είναι η κατάλληλη για προγράμματα με πολλές και σύνθετες δομές δεδομένων.

Επειδή η Python είναι μία διαρκώς εξελισσόμενη γλώσσα προγραμματισμού με μεγάλο εύρος εφαρμογών, ως θέμα για περαιτέρω μελέτη, προτείνεται η σύγκριση της ανάπτυξης διαδικτυακών εφαρμογών με frameworks σε Python όπως το Django, το Pyramid και το Web.py, με τις υπάρχουσες δημοφιλείς λύσεις σε CMS (WordPress, Joomla, Drupal) και με δημοφιλή JavaScript frameworks (React, Angular, Vue.js).

7. Βιβλιογραφία

[1] H. Schildt, *C++: The Complete Reference, Fourth Edition*. 2003.

[2] U. Kirch-Prinz and P. Prinz, *A Complete Guide to Programming in C++*. JONES AND BARTLETT PUBLISHER, 2002.

[3] Beazley., *Python Essential Reference*. Addison-Wesley Professional, 2010.

[4] R. Python, "Python Tutorials – Real Python", *Realpython.com*, 2021. [Online]. Available: https://realpython.com .

[5] "W3Schools Online Web Tutorials", *W3schools.com*, 2021. [Online]. Available: https://www.w3schools.com/ .

[6] "Python - Functions - Tutorialspoint", *Tutorialspoint.com*, 2021. [Online]. Available: https://www.tutorialspoint.com/python/python_functions.htm .

[7] "MathWorks - Makers of MATLAB and Simulink", *Mathworks.com*, 2021. [Online]. Available: https://www.mathworks.com/ .

[8] *MATLAB The Language of Technical Computing: Language Reference Manual*, 5th ed. 1997.

[9] *Hkn.umn.edu*, 2021. [Online]. Available: http://www.hkn.umn.edu/ .

[10] "All C++ Functions", *Doc.bccnsoft.com*, 2021. [Online]. Available: https://doc.bccnsoft.com/docs/cppreference_en/all_cpp_functions.html .

[11] *CodinGame*, 2021. [Online]. Available: https://www.codingame.com/

[12] 2021. [Online]. Available: https://www.softwaretestinghelp.com/python-vs-cpp/

[13] "WHAT IS THE DIFFERENCE BETWEEN MATLAB AND C++?", *Issuu*, 2021. [Online]. Available: https://issuu.com/matlabassignmenthelp3/docs/what_is_the_difference_between_matlab_and_c__ .

[14] "10 Reasons Why Python is Better than Matlab", *Medium*, 2021. [Online]. Available: https://medium.com/swlh/python-for-matlab-users-part-1-why-python-python-vs-matlab-959d92d702ef .

8. Παράρτημα Α

8.1 Βασικές συναρτήσεις στην Matlab

Cell Array Functions	
`cell`	Creates cell array.
`celldisp`	Displays cell array.
`cellplot`	Displays graphical representation of cell array.
`num2cell`	Converts numeric array to cell array.
`deal`	Matches input and output lists.
`iscell`	Identifies cell array.

Structure Functions	
`fieldnames`	Returns field names in a structure array.
`getfield`	Returns field contents of a structure array.
`isfield`	Identifies a structure array field.
`isstruct`	Identifies a structure array.
`rmfield`	Removes a field from a structure array.
`setfield`	Sets contents of field.
`struct`	Creates structure array.

Exponential and Logarithmic Functions	
`exp(x)`	Exponential; e^x.
`log(x)`	Natural logarithm; ln(x).
`log10(x)`	Common (base 10) logarithm; log(x)= $\log_{10}(x)$.
`sqrt(x)`	Square root; $\sqrt{x}$.

Trigonometric Functions	
`acos(x)`	Inverse cosine; arcos x = $\cos^{-1}(x)$.
`acot(x)`	Inverse cotangent; arccot x = $\cot^{-1}(x)$.
`acsc(x)`	Inverse cosecant; arcs x = $\csc^{-1}(x)$.
`asec(x)`	Inverse secant; arcsec x = $\sec^{-1}(x)$.
`asin(x)`	Inverse sine; arcsin x = $\sin^{-1}(x)$.
`atan(x)`	Inverse tangent; arctan x = $\tan^{-1}(x)$.
`atan2(y,x)`	Four-quadrant inverse tangent.
`cos(x)`	Cosine; cos(x).
`cot(x)`	Cotangent; cot(x).
`csc(x)`	Cosecant; csc(x).
`sec(x)`	Secant; sec(x).
`sin(x)`	Sine; sin(x).
`tan(x)`	Tangent; tan(x).

Hyperbolic Functions	
`acosh(x)`	Inverse hyperbolic cosine; $\cosh^{-1}(x)$.
`acoth(x)`	Inverse hyperbolic cotangent; $\coth^{-1}(x)$.
`acsch(x)`	Inverse hyperbolic cosecant; $\mathrm{csch}^{-1}(x)$.
`asech(x)`	Inverse hyperbolic secant; $\mathrm{sech}^{-1}(x)$.
`asinh(x)`	Inverse hyperbolic sine; $\sinh^{-1}(x)$.
`atanh(x)`	Inverse hyperbolic tangent; $\tanh^{-1}(x)$.
`cosh(x)`	Hyperbolic cosine; cosh(x).
`coth(x)`	Hyperbolic cotangent; cosh(x)/sinh(x).
`csch(x)`	Hyperbolic cosecant; 1/sinh(x).
`sech(x)`	Hyperbolic secant; 1/cosh(x).
`sinh(x)`	Hyperbolic sine; sinh(x).
`tanh(x)`	Hyperbolic tangent; sinh(x)/cosh(x).

Complex Functions	
`abs(x)`	Absolute value; |x|.
`angle(x)`	Angle of a complex number x.
`conj(x)`	Complex conjugate of x.
`imag(x)`	Imaginary part of a complex number x.
`real(x)`	Real part of a complex number x.

Statistical Functions	
`erf(x)`	Computes the error function *erf* (x).
`mean`	Calculates the average.
`median`	Calculates the median.
`std`	Calculates the standard deviation.

Random Number Functions	
`rand`	Generates uniformly distributed random numbers between 0 and 1.
`randn`	Generates normally distributed random numbers.

Numeric Functions	
`ceil`	Rounds to the nearest integer toward ∞.
`fix`	Rounds to the nearest integer toward zero.
`floor`	Rounds to the nearest integer toward - ∞.
`round`	Rounds towards the nearest integer.
`sign`	Signum function.

String Functions	
`findstr`	Finds occurrences of a string.
`strcmp`	Compares strings.
`char`	Creates character string array

Polynomial and Regression Functions	
`conv`	Computes product of two polynomials
`deconv`	Computes ratio of polynomials.
`eig`	Computes the eigenvalues of a matrix.
`poly`	Computes polynomial from roots.
`polyfit`	Fits a polynomial to data.
`polyval`	Evaluates polynomial and generates error estimates.
`roots`	Computes polynomial roots.

Interpolation Functions	
`interp1`	Linear and cubic-spline interpolations of a function of one variable.
`interp2`	Linear interpolation of a function of two variables.
`spline`	Cubic-spline interpolation.
`unmkpp`	Computes the coefficients of cubic-spine polynomials.

Root Finding and Minimization	
`fmin`	Finds minimum of single-variable function.
`fmins`	Finds minimum of multivariable function.
`fzero`	Finds zero of single-variable function.

Numerical Integration Functions	
`quad`	Numerical integration with adaptive Simpson's rule.
`quadl`	Numerical integration with adaptive Lobatto quadrature.
`trapz`	Numerical integration with the trapezoidal rule.

Numerical Differentiation Functions	
`diff(x)`	Computes the difference between adjacent elements in the vector x.
`polyder`	Differentiates a polynomial, a polynomial product, or a polynomial quotient.

Predefined Input Functions	
gensig	Generates a periodic sine, square, or pulse input.
sawtooth	Generates a periodic sawtooth input.
square	Generates a square wave input.
stepfun	Generates a step function input.

Functions for Creating and Evaluating Symbolic Expressions	
class	Returns the class of an expression.
digits	Sets the number of decimal digits used to do variable precision arithmetic.
double	Converts an expression to numeric form.
ezplot	Generates a plot of a symbolic expression.
findsym	Finds the symbolic variables in a symbolic expression.
numden	Returns the numerator and denominator of an expression.
sym	Creates a symbolic variable.
syms	Creates one or more symbolic variables.
vpa	Sets the number of digits used to evaluate expressions.

Functions for Manipulating Symbolic Expressions	
collect	Collects coefficients of like powers in an expression.
expand	Expands an expression by carrying out jpowers.
factor	Factors an expression.
poly2sym	Converts a polynomial coefficient vector to a symbolic polynomial.
pretty	Displays an expression in a form that resembles typeset mathematics.
simple	Searches for the shortest form of an expression.
simplify	Simplifies an expression using Maple's simplification rules.
subs	Substitutes variables or expressions.
sym2poly	Converts an expression to a polynomial coefficient vector.

Symbolic Calculus Functions	
diff	Returns the derivative of an expression.
Dirac	Dirac delta function (unit impulse).
Heaviside	Heaviside function (unit step).
int	Returns the integral of an expression.
limit	Returns the limit of an expression.
symsum	Returns the symbolic summation of an expression.
taylor	Returns the Taylor series of a function.

Laplace Transform Functions	
ilaplace	Returns the inverse Laplace transform.
laplace	Returns the Laplace transform.

Symbolic Linear Algebra Functions	
det	Returns the determinant of a matrix.
eig	Returns the eigenvalues (characteristic roots) of a matrix.
inv	Returns the inverse of a matrix.
poly	Returns the characteristic polynomial of a matrix.

8.2 Βασικές συναρτήσεις στην C++

Bitset Constructors (C++ Bitsets)	create new bitsets
Bitset Operators (C++ Bitsets)	compare and assign bitsets
Vector constructors	create vectors and initialize them with some data
Container constructors (C++ Double-ended Queues)	create containers and initialize them with some data
Container constructors (C++ Lists)	create containers and initialize them with some data
Container constructors & destructors (C++ Sets)	default methods to allocate, copy, and deallocate containers
Container constructors & destructors (C++ Multisets)	default methods to allocate, copy, and deallocate multisets
Map constructors & destructors (C++ Maps)	default methods to allocate, copy, and deallocate maps
Multimap constructors & destructors (C++ Multimaps)	default methods to allocate, copy, and deallocate containers
Container operators (C++ Lists)	assign and compare containers
Container operators (C++ Sets)	assign and compare containers
Container operators (C++ Multisets)	assign and compare containers
Multimap operators (C++ Multimaps)	assign and compare containers
Vector operators	compare, assign, and access elements of a vector
Container operators (C++ Double-ended Queues)	compare, assign, and access elements of a container
I/O Constructors (C++ I/O)	constructors
Map operators (C++ Maps)	assign, compare, and access elements of a map
Priority queue constructors (C++ Priority Queues)	construct a new priority queue
Queue constructor (C++ Queues)	construct a new queue

Stack constructors (C++ Stacks)	construct a new stack
String constructors (C++ Strings)	create strings from arrays of characters and other strings
String operators (C++ Strings)	concatenate strings, assign strings, use strings for I/O, compare strings
accumulate (C++ Algorithms)	sum up a range of elements
adjacent_difference (C++ Algorithms)	compute the differences between adjacent elements in a range
adjacent_find (C++ Algorithms)	finds two items that are adjacent to eachother
any (C++ Bitsets)	true if any bits are set
append (C++ Strings)	append characters and strings onto a string
assign (C++ Vectors)	assign elements to a container

assign (C++ Lists)	assign elements to a container
assign (C++ Strings)	give a string values from strings of characters and other C++ strings
at (C++ Vectors)	returns an element at a specific location
at (C++ Double-ended Queues)	returns an element at a specific location
at (C++ Strings)	returns an element at a specific location
auto_ptr (Miscellaneous C++)	create pointers that automatically destroy objects
back (C++ Vectors)	returns a reference to last element of a container
back (C++ Double-ended Queues)	returns a reference to last element of a container
back (C++ Lists)	returns a reference to last element of a container
back (C++ Queues)	returns a reference to last element of a container
bad (C++ I/O)	true if an error occurred
begin (C++ Strings)	returns an iterator to the beginning of the container
begin (C++ Vectors)	returns an iterator to the beginning of the container
begin (C++ Double-ended Queues)	returns an iterator to the beginning of the container
begin (C++ Lists)	returns an iterator to the beginning of the container
begin (C++ Sets)	returns an iterator to the beginning of the container
begin (C++ Multisets)	returns an iterator to the beginning of the container
begin (C++ Maps)	returns an iterator to the beginning of the container
begin (C++ Multimaps)	returns an iterator to the beginning of the container
binary_search (C++ Algorithms)	determine if an element exists in a certain range
c_str (C++ Strings)	returns a standard C character array version of the string
capacity (C++ Vectors)	returns the number of elements that the container can hold
capacity (C++ Strings)	returns the number of elements that the container can hold
clear (C++ I/O)	clear and set status flags
clear (C++ Strings)	removes all elements from the container
clear (C++ Vectors)	removes all elements from the container
clear (C++ Double-ended Queues)	removes all elements from the container
clear (C++ Lists)	removes all elements from the container
clear (C++ Sets)	removes all elements from the container
clear (C++ Multisets)	removes all elements from the container
clear (C++ Maps)	removes all elements from the container
clear (C++ Multimaps)	removes all elements from the container
close (C++ I/O)	close a stream

compare (C++ Strings)	compares two strings
copy (C++ Strings)	copies characters from a string into an array
copy (C++ Algorithms)	copy some range of elements to a new location
copy_backward (C++ Algorithms)	copy a range of elements in backwards order
copy_n (C++ Algorithms)	copy N elements
count (C++ Sets)	returns the number of elements matching a certain key
count (C++ Multisets)	returns the number of elements matching a certain key
count (C++ Maps)	returns the number of elements matching a certain key
count (C++ Multimaps)	returns the number of elements matching a certain key
count (C++ Bitsets)	returns the number of set bits
count (C++ Algorithms)	return the number of elements matching a given value
count_if (C++ Algorithms)	return the number of elements for which a predicate is true
data (C++ Strings)	returns a pointer to the first character of a string
empty (C++ Strings)	true if the container has no elements
empty (C++ Vectors)	true if the container has no elements
empty (C++ Double-ended Queues)	true if the container has no elements
empty (C++ Lists)	true if the container has no elements
empty (C++ Sets)	true if the container has no elements
empty (C++ Multisets)	true if the container has no elements
empty (C++ Maps)	true if the container has no elements
empty (C++ Multimaps)	true if the container has no elements
empty (C++ Stacks)	true if the container has no elements
empty (C++ Queues)	true if the container has no elements
empty (C++ Priority Queues)	true if the container has no elements
end (C++ Strings)	returns an iterator just past the last element of a container
end (C++ Vectors)	returns an iterator just past the last element of a container
end (C++ Double-ended Queues)	returns an iterator just past the last element of a container
end (C++ Lists)	returns an iterator just past the last element of a container
end (C++ Sets)	returns an iterator just past the last element of a container
end (C++ Multisets)	returns an iterator just past the last element of a container
end (C++ Maps)	returns an iterator just past the last element of a container
end (C++ Multimaps)	returns an iterator just past the last element of a container
eof (C++ I/O)	true if at the end-of-file
equal (C++ Algorithms)	determine if two sets of elements are the same
equal_range (C++ Sets)	returns iterators to the first and just past the last elements matching a specific key
equal_range (C++ Multisets)	returns iterators to the first and just past the last elements matching a specific key
equal_range (C++ Maps)	returns iterators to the first and just past the last elements matching a specific key
equal_range (C++ Multimaps)	returns iterators to the first and just past the last elements matching a specific key
equal_range (C++ Algorithms)	search for a range of elements that are all equal to a certain element
erase (C++ Strings)	removes elements from a string
erase (C++ Vectors)	removes elements from a container

erase (C++ Double-ended Queues)	removes elements from a container
erase (C++ Lists)	removes elements from a container
erase (C++ Sets)	removes elements from a container
erase (C++ Multisets)	removes elements from a container
erase (C++ Maps)	removes elements from a container
erase (C++ Multimaps)	removes elements from a container
fail (C++ I/O)	true if an error occurred
fill (C++ I/O)	manipulate the default fill character
fill (C++ Algorithms)	assign a range of elements a certain value
fill_n (C++ Algorithms)	assign a value to some number of elements
find (C++ Algorithms)	find a value in a given range
find (C++ Sets)	returns an iterator to specific elements

find (C++ Multisets)	returns an iterator to specific elements
find (C++ Maps)	returns an iterator to specific elements
find (C++ Multimaps)	returns an iterator to specific elements
find (C++ Strings)	find characters in the string
find_end (C++ Algorithms)	find the last sequence of elements in a certain range
find_first_not_of (C++ Strings)	find first absence of characters
find_first_of (C++ Strings)	find first occurrence of characters
find_first_of (C++ Algorithms)	search for any one of a set of elements
find_if (C++ Algorithms)	find the first element for which a certain predicate is true
find_last_not_of (C++ Strings)	find last absence of characters
find_last_of (C++ Strings)	find last occurrence of characters
flags (C++ I/O)	access or manipulate io stream format flags
flip (C++ Bitsets)	reverses the bitset
flush (C++ I/O)	empty the buffer
for_each (C++ Algorithms)	apply a function to a range of elements
front (C++ Vectors)	returns a reference to the first element of a container
front (C++ Double-ended Queues)	returns a reference to the first element of a container
front (C++ Lists)	returns a reference to the first element of a container
front (C++ Queues)	returns a reference to the first element of a container
gcount (C++ I/O)	number of characters read during last input
generate (C++ Algorithms)	saves the result of a function in a range
generate_n (C++ Algorithms)	saves the result of N applications of a function
get (C++ I/O)	read characters
getline (C++ I/O)	read a line of characters
getline (C++ Strings)	read data from an I/O stream into a string
good (C++ I/O)	true if no errors have occurred
ignore (C++ I/O)	read and discard characters
includes (C++ Algorithms)	returns true if one set is a subset of another
inner_product (C++ Algorithms)	compute the inner product of two ranges of elements
inplace_merge (C++ Algorithms)	merge two ordered ranges in-place
insert (C++ Strings)	insert characters into a string
insert (C++ Vectors)	inserts elements into the container
insert (C++ Double-ended Queues)	inserts elements into the container

insert (C++ Lists)	inserts elements into the container
insert (C++ Sets)	insert items into a container
insert (C++ Multisets)	inserts items into a container
insert (C++ Multimaps)	inserts items into a container
insert (C++ Maps)	insert items into a container
is_heap (C++ Algorithms)	returns true if a given range is a heap
is_sorted (C++ Algorithms)	returns true if a range is sorted in ascending order
iter_swap (C++ Algorithms)	swaps the elements pointed to by two iterators
key_comp (C++ Sets)	returns the function that compares keys
key_comp (C++ Multisets)	returns the function that compares keys
key_comp (C++ Maps)	returns the function that compares keys
key_comp (C++ Multimaps)	returns the function that compares keys
length (C++ Strings)	returns the length of the string

lexicographical_compare (C++ Algorithms)	returns true if one range is lexicographically less than another
lexicographical_compare_3way (C++ Algorithms)	determines if one range is lexicographically less than or greater than another
lower_bound (C++ Sets)	returns an iterator to the first element greater than or equal to a certain value
lower_bound (C++ Multisets)	returns an iterator to the first element greater than or equal to a certain value
lower_bound (C++ Maps)	returns an iterator to the first element greater than or equal to a certain value
lower_bound (C++ Multimaps)	returns an iterator to the first element greater than or equal to a certain value
lower_bound (C++ Algorithms)	search for the first place that a value can be inserted while preserving order
make_heap (C++ Algorithms)	creates a heap out of a range of elements
max (C++ Algorithms)	returns the larger of two elements
max_element (C++ Algorithms)	returns the largest element in a range
max_size (C++ Strings)	returns the maximum number of elements that the container can hold
max_size (C++ Vectors)	returns the maximum number of elements that the container can hold
max_size (C++ Double-ended Queues)	returns the maximum number of elements that the container can hold
max_size (C++ Lists)	returns the maximum number of elements that the container can hold
max_size (C++ Sets)	returns the maximum number of elements that the container can hold
max_size (C++ Multisets)	returns the maximum number of elements that the container can hold
max_size (C++ Maps)	returns the maximum number of elements that the container can hold
max_size (C++ Multimaps)	returns the maximum number of elements that the container can hold
merge (C++ Lists)	merge two lists
merge (C++ Algorithms)	merge two sorted ranges
min (C++ Algorithms)	returns the smaller of two elements
min_element (C++ Algorithms)	returns the smallest element in a range
mismatch (C++ Algorithms)	finds the first position where two ranges differ

next_permutation (C++ Algorithms)	generates the next greater lexicographic permutation of a range of elements
none (C++ Bitsets)	true if no bits are set
nth_element (C++ Algorithms)	put one element in its sorted location and make sure that no elements to its left are greater than any elements to its right
open (C++ I/O)	create an input stream
partial_sort (C++ Algorithms)	sort the first N elements of a range
partial_sort_copy (C++ Algorithms)	copy and partially sort a range of elements
partial_sum (C++ Algorithms)	compute the partial sum of a range of elements
partition (C++ Algorithms)	divide a range of elements into two groups
peek (C++ I/O)	check the next input character
pop (C++ Stacks)	removes the top element of a container

pop (C++ Queues)	removes the top element of a container
pop (C++ Priority Queues)	removes the top element of a container
pop_back (C++ Vectors)	removes the last element of a container
pop_back (C++ Double-ended Queues)	removes the last element of a container
pop_back (C++ Lists)	removes the last element of a container
pop_front (C++ Double-ended Queues)	removes the first element of the container
pop_front (C++ Lists)	removes the first element of the container
pop_heap (C++ Algorithms)	remove the largest element from a heap
power (C++ Algorithms)	compute the value of some number raised to the Nth power
precision (C++ I/O)	manipulate the precision of a stream
prev_permutation (C++ Algorithms)	generates the next smaller lexicographic permutation of a range of elements
push (C++ Stacks)	adds an element to the top of the container
push (C++ Queues)	adds an element to the end of the container
push (C++ Priority Queues)	adds an element to the end of the container
push_back (C++ Vectors)	add an element to the end of the container
push_back (C++ Double-ended Queues)	add an element to the end of the container
push_back (C++ Lists)	add an element to the end of the container
push_back (C++ Strings)	add an element to the end of the container
push_front (C++ Double-ended Queues)	add an element to the front of the container
push_front (C++ Lists)	add an element to the front of the container
push_heap (C++ Algorithms)	add an element to a heap
put (C++ I/O)	write characters
putback (C++ I/O)	return characters to a stream
random_sample (C++ Algorithms)	randomly copy elements from one range to another
random_sample_n (C++ Algorithms)	sample N random elements from a range
random_shuffle (C++ Algorithms)	randomly re-order elements in some range
rbegin (C++ Vectors)	returns a reverse_iterator to the end of the container
rbegin (C++ Strings)	returns a reverse_iterator to the end of the container
rbegin (C++ Double-ended Queues)	returns a reverse_iterator to the end of the container
rbegin (C++ Lists)	returns a reverse_iterator to the end of the container

rbegin (C++ Sets)	returns a reverse_iterator to the end of the container
rbegin (C++ Multisets)	returns a reverse_iterator to the end of the container
rbegin (C++ Maps)	returns a reverse_iterator to the end of the container
rbegin (C++ Multimaps)	returns a reverse_iterator to the end of the container
rdstate (C++ I/O)	returns the state flags of the stream
read (C++ I/O)	read data into a buffer
remove (C++ Lists)	removes elements from a list
remove (C++ Algorithms)	remove elements equal to certain value
remove_copy (C++ Algorithms)	copy a range of elements omitting those that match a certian value
remove_copy_if (C++ Algorithms)	create a copy of a range of elements, omitting any for which a predicate is true
remove_if (C++ Lists)	removes elements conditionally
remove_if (C++ Algorithms)	remove all elements for which a predicate is true
rend (C++ Vectors)	returns a reverse_iterator to the beginning of the container
rend (C++ Strings)	returns a reverse_iterator to the beginning of the container
rend (C++ Double-ended Queues)	returns a reverse_iterator to the beginning of the container
rend (C++ Lists)	returns a reverse_iterator to the beginning of the container
rend (C++ Sets)	returns a reverse_iterator to the beginning of the container
rend (C++ Multisets)	returns a reverse_iterator to the beginning of the container
rend (C++ Maps)	returns a reverse_iterator to the beginning of the container
rend (C++ Multimaps)	returns a reverse_iterator to the beginning of the container
replace (C++ Strings)	replace characters in the string
replace (C++ Algorithms)	replace every occurrence of some value in a range with another value
replace_copy (C++ Algorithms)	copy a range, replacing certain elements with new ones
replace_copy_if (C++ Algorithms)	copy a range of elements, replacing those for which a predicate is true
replace_if (C++ Algorithms)	change the values of elements for which a predicate is true
reserve (C++ Vectors)	sets the minimum capacity of the container
reserve (C++ Strings)	sets the minimum capacity of the container
reset (C++ Bitsets)	sets bits to zero
resize (C++ Vectors)	change the size of the container
resize (C++ Double-ended Queues)	change the size of the container
resize (C++ Lists)	change the size of the container
resize (C++ Strings)	change the size of the container
reverse (C++ Lists)	reverse the list
reverse (C++ Algorithms)	reverse elements in some range
reverse_copy (C++ Algorithms)	create a copy of a range that is reversed
rfind (C++ Strings)	find the last occurrence of a substring
rotate (C++ Algorithms)	move the elements in some range to the left by some amount
rotate_copy (C++ Algorithms)	copy and rotate a range of elements
search (C++ Algorithms)	search for a range of elements
search_n (C++ Algorithms)	search for N consecutive copies of an element in some range
seekg (C++ I/O)	perform random access on an input stream
seekp (C++ I/O)	perform random access on output streams

set (C++ Bitsets)	sets bits
set_difference (C++ Algorithms)	computes the difference between two sets
set_intersection (C++ Algorithms)	computes the intersection of two sets
set_symmetric_difference (C++ Algorithms)	computes the symmetric difference between two sets
set_union (C++ Algorithms)	computes the union of two sets
setf (C++ I/O)	set format flags
size (C++ Strings)	returns the number of items in the container
size (C++ Vectors)	returns the number of items in the container
size (C++ Double-ended Queues)	returns the number of items in the container
size (C++ Lists)	returns the number of items in the container
size (C++ Sets)	returns the number of items in the container
size (C++ Multisets)	returns the number of items in the container
size (C++ Maps)	returns the number of items in the container
size (C++ Multimaps)	returns the number of items in the container
size (C++ Stacks)	returns the number of items in the container
size (C++ Queues)	returns the number of items in the container
size (C++ Priority Queues)	returns the number of items in the container
size (C++ Bitsets)	number of bits that the bitset can hold
sort (C++ Lists)	sorts a list into ascending order
sort (C++ Algorithms)	sort a range into ascending order
sort_heap (C++ Algorithms)	turns a heap into a sorted range of elements
splice (C++ Lists)	merge two lists in constant time
stable_partition (C++ Algorithms)	divide elements into two groups while preserving their relative order
stable_sort (C++ Algorithms)	sort a range of elements while preserving order between equal elements
substr (C++ Strings)	returns a certain substring
swap (C++ Strings)	swap the contents of this container with another
swap (C++ Vectors)	swap the contents of this container with another
swap (C++ Double-ended Queues)	swap the contents of this container with another
swap (C++ Lists)	swap the contents of this container with another
swap (C++ Sets)	swap the contents of this container with another
swap (C++ Multisets)	swap the contents of this container with another
swap (C++ Maps)	swap the contents of this container with another
swap (C++ Multimaps)	swap the contents of this container with another
swap (C++ Algorithms)	swap the values of two objects
swap_ranges (C++ Algorithms)	swaps two ranges of elements
sync_with_stdio (C++ I/O)	synchronize with standard I/O
tellg (C++ I/O)	read input stream pointers
tellp (C++ I/O)	read output stream pointers
test (C++ Bitsets)	returns the value of a given bit
to_string (C++ Bitsets)	string representation of the bitset
to_ulong (C++ Bitsets)	returns an integer representation of the bitset
top (C++ Stacks)	returns the top element of the container
top (C++ Priority Queues)	returns the top element of the container
transform (C++ Algorithms)	applies a function to a range of elements

unique (C++ Lists)	removes consecutive duplicate elements
unique (C++ Algorithms)	remove consecutive duplicate elements in a range
unique_copy (C++ Algorithms)	create a copy of some range of elements that contains no consecutive duplicates
unsetf (C++ I/O)	clear io stream format flags
upper_bound (C++ Sets)	returns an iterator to the first element greater than a certain value
upper_bound (C++ Multisets)	returns an iterator to the first element greater than a certain value
upper_bound (C++ Maps)	returns an iterator to the first element greater than a certain value
upper_bound (C++ Multimaps)	returns an iterator to the first element greater than a certain value

upper_bound (C++ Algorithms)	searches for the last possible location to insert an element into an ordered range
value_comp (C++ Sets)	returns the function that compares values
value_comp (C++ Multisets)	returns the function that compares values
value_comp (C++ Maps)	returns the function that compares values
value_comp (C++ Multimaps)	returns the function that compares values
width (C++ I/O)	access and manipulate the minimum field width
write (C++ I/O)	write characters

8.3 Βασικές συναρτήσεις στην Python

abs()	Returns the absolute value of a number
all()	Returns True if all items in an iterable object are true
any()	Returns True if any item in an iterable object is true
ascii()	Returns a readable version of an object. Replaces none-ascii characters with escape character
bin()	Returns the binary version of a number
bool()	Returns the boolean value of the specified object
bytearray()	Returns an array of bytes
bytes()	Returns a bytes object
callable()	Returns True if the specified object is callable, otherwise False
chr()	Returns a character from the specified Unicode code.
classmethod()	Converts a method into a class method
compile()	Returns the specified source as an object, ready to be executed
complex()	Returns a complex number
delattr()	Deletes the specified attribute (property or method) from the specified object
dict()	Returns a dictionary (Array)
dir()	Returns a list of the specified object's properties and methods
divmod()	Returns the quotient and the remainder when argument1 is divided by argument2

enumerate()	Takes a collection (e.g. a tuple) and returns it as an enumerate object
eval()	Evaluates and executes an expression
exec()	Executes the specified code (or object)
filter()	Use a filter function to exclude items in an iterable object
float()	Returns a floating point number
format()	Formats a specified value
frozenset()	Returns a frozenset object
getattr()	Returns the value of the specified attribute (property or method)
globals()	Returns the current global symbol table as a dictionary

hasattr()	Returns True if the specified object has the specified attribute (property/method)
hash()	Returns the hash value of a specified object
help()	Executes the built-in help system
hex()	Converts a number into a hexadecimal value
id()	Returns the id of an object
input()	Allowing user input
int()	Returns an integer number
isinstance()	Returns True if a specified object is an instance of a specified object
issubclass()	Returns True if a specified class is a subclass of a specified object
iter()	Returns an iterator object
len()	Returns the length of an object
list()	Returns a list
locals()	Returns an updated dictionary of the current local symbol table
map()	Returns the specified iterator with the specified function applied to each item
max()	Returns the largest item in an iterable
memoryview()	Returns a memory view object
min()	Returns the smallest item in an iterable
next()	Returns the next item in an iterable
object()	Returns a new object
oct()	Converts a number into an octal

open()	Opens a file and returns a file object
ord()	Convert an integer representing the Unicode of the specified character
pow()	Returns the value of x to the power of y
print()	Prints to the standard output device
property()	Gets, sets, deletes a property
range()	Returns a sequence of numbers, starting from 0 and increments by 1 (by default)
repr()	Returns a readable version of an object
reversed()	Returns a reversed iterator
round()	Rounds a numbers
set()	Returns a new set object
setattr()	Sets an attribute (property/method) of an object
slice()	Returns a slice object
sorted()	Returns a sorted list
@staticmethod()	Converts a method into a static method
str()	Returns a string object
sum()	Sums the items of an iterator
super()	Returns an object that represents the parent class
tuple()	Returns a tuple

type()	Returns the type of an object
vars()	Returns the __dict__ property of an object
zip()	Returns an iterator, from two or more iterators

8.4 Είσοδοι και έξοδοι για τα προγράμματα του προβλήματος «Υπολογισμός τετραγώνων σε αγροτεμάχιο»

Input(N, (X,Y))	Output(counter)
5 0 0 0 2 1 0 2 0 2 2	1

```
50
0 2
0 3
0 5
0 7
0 9
1 0
1 1
1 2
1 3
1 4
1 6
1 8
1 9
2 3
2 4
2 6
2 8
3 2
3 4
3 5
3 7
4 2
4 5
4 7
4 8                    46
4 9
5 2
5 5
5 7
6 1
6 2
6 4
6 5
6 6
6 7
6 9
7 0
7 6
8 0
8 1
8 4
8 6
8 7
8 8
8 9
9 1
9 3
9 4
9 5
9 7
```

```
100
0 2
0 5
0 7
0 8
0 10
1 1
1 2
1 3
1 4
1 5
1 6
1 9
1 10
1 11
1 12
1 13
2 0
2 1
2 3
2 5
2 6
2 7
2 9
2 10
2 11                198
3 0
3 3
3 6
3 9
3 10
4 0
4 1
4 2
4 3
4 4
4 5
4 6
4 8
4 10
4 13
5 0
5 3
5 4
5 6
5 9
5 10
6 5
6 8
6 10
6 13
```

```
7 4
7 5
7 8
8 0
8 2
8 3
8 4
8 5
8 13
9 0
9 1
9 3
9 4
9 8
9 9
9 10
9 11
9 12
10 3
10 4
10 6
10 8
10 11
10 13
11 2
11 4
11 5
11 6
11 11
11 12
12 1
12 2
12 3
12 6
12 8
12 9
12 10
12 11
12 12
12 13
13 0
13 1
13 2
13 3
13 5
13 6
13 7
13 8
13 11
13 12
```

```
500
0 1
0 2
0 5
0 6
0 8
0 9
0 14
0 15
0 17
0 18
0 19
0 22
0 23
0 24
0 30
1 5
1 7
1 9
1 11
1 12
1 13
1 14
1 17
1 18
1 19                    6114
1 21
1 24
1 25
1 29
1 30
2 2
2 4
2 5
2 7
2 8
2 9
2 10
2 12
2 13
2 14
2 15
2 18
2 19
2 20
2 21
2 23
2 26
2 27
2 28
2 29
```

3 0
3 3
3 7
3 9
3 12
3 14
3 16
3 17
3 18
3 20
3 21
3 23
3 24
3 26
3 27
3 30
4 0
4 1
4 2
4 4
4 5
4 8
4 10
4 12
4 13
4 14
4 15
4 16
4 17
4 19
4 21
4 23
4 24
4 25
4 27
4 29
5 0
5 1
5 2
5 4
5 5
5 6
5 8
5 9
5 11
5 18
5 22
5 26
5 30
6 2
6 5

6 8
6 9
6 10
6 12
6 13
6 14
6 16
6 20
6 22
6 23
6 24
6 29
7 3
7 4
7 5
7 9
7 10
7 11
7 12
7 13
7 14
7 15
7 16
7 18
7 20
7 22
7 23
7 24
7 25
7 26
7 27
7 28
7 29
7 30
8 0
8 1
8 6
8 7
8 9
8 10
8 11
8 13
8 14
8 16
8 18
8 19
8 21
8 23
8 25
8 26
8 27

8 28
8 30
9 0
9 1
9 7
9 10
9 11
9 14
9 15
9 17
9 19
9 20
9 21
9 22
9 23
9 24
9 25
9 27
9 28
9 29
9 30
10 2
10 3
10 5
10 7
10 8
10 10
10 11
10 12
10 13
10 21
10 23
10 24
10 25
10 26
10 27
10 29
11 0
11 2
11 3
11 7
11 9
11 10
11 11
11 13
11 15
11 16
11 18
11 19
11 21
11 25

```
11 27
12 2
12 7
12 8
12 9
12 12
12 13
12 14
12 17
12 18
12 19
12 20
12 22
12 27
12 30
13 1
13 2
13 3
13 4
13 8
13 9
13 10
13 11
13 14
13 15
13 16
13 18
13 20
13 22
13 27
14 1
14 2
14 3
14 8
14 9
14 10
14 11
14 14
14 16
14 17
14 20
14 21
14 22
14 24
14 25
14 28
14 29
14 30
15 2
15 6
15 7
```

15 8
15 9
15 14
15 16
15 20
15 22
15 25
15 26
15 27
15 29
16 1
16 2
16 3
16 4
16 6
16 12
16 16
16 17
16 19
16 20
16 21
16 22
16 24
16 29
16 30
17 0
17 1
17 2
17 3
17 4
17 5
17 6
17 8
17 10
17 11
17 12
17 13
17 14
17 15
17 16
17 21
17 22
17 23
17 24
17 26
17 28
17 30
18 3
18 4
18 6
18 9

```
18 10
18 11
18 12
18 13
18 15
18 17
18 19
18 20
18 21
18 23
18 30
19 0
19 1
19 2
19 4
19 5
19 8
19 9
19 12
19 13
19 14
19 15
19 16
19 19
19 21
19 22
19 23
19 25
19 27
20 0
20 1
20 2
20 3
20 6
20 8
20 10
20 11
20 14
20 17
20 18
20 19
20 20
20 21
20 22
20 25
20 27
20 30
21 0
21 1
21 3
21 4
```

21 7
21 8
21 10
21 11
21 12
21 16
21 18
21 19
21 21
21 23
21 24
21 25
21 26
21 27
21 29
22 2
22 5
22 7
22 9
22 11
22 14
22 15
22 19
22 20
22 24
22 26
22 27
22 30
23 2
23 3
23 4
23 9
23 10
23 13
23 17
23 18
23 23
23 24
23 25
23 28
23 30
24 1
24 4
24 5
24 6
24 10
24 12
24 13
24 15
24 16
24 18

24 19
24 20
24 22
24 23
24 24
24 25
24 30
25 3
25 5
25 7
25 9
25 10
25 11
25 12
25 16
25 17
25 18
25 19
25 20
25 22
25 28
25 29
25 30
26 0
26 4
26 9
26 10
26 14
26 15
26 18
26 19
26 20
26 21
26 22
26 24
26 27
27 0
27 3
27 4
27 5
27 6
27 9
27 10
27 13
27 14
27 15
27 16
27 18
27 20
27 21
27 23

27 24
27 27
28 1
28 5
28 7
28 8
28 13
28 20
28 24
28 25
28 28
28 30
29 2
29 3
29 8
29 9
29 11
29 14
29 19
29 21
29 22
29 23
29 25
29 26
29 27
29 28
30 1
30 3
30 4
30 9
30 10
30 11
30 12
30 15
30 17
30 18
30 23
30 24
30 25
30 27
30 29
30 30

Πίνακας 6 Τιμές εισόδου και εξόδου προγράμματος «Το πρόβλημα του περιοδεύοντος πωλητή»

8.5 Είσοδοι και έξοδοι για τα προγράμματα του προβλήματος «Σκοτεινά σημεία δωματίου»

Input(N,L,πίνακας δωματίου)	Output(Αριθμός σκοτεινών σημείων)
5 3 X X X X X X C X X X X X X X X X X X X X X X X X X	9
5 3 C X X X C X X X X X X X X X X X X X X X C X X X C	0
5 3 X X X X X X C X X X X X X X X X X X C X X X X X X	2
6 3 X X X X X X X C X X X X X X X X X X X X X C X X X X X X X X X X X X X X	4

15 3 X X X X X X X X X X X X C X X X X X X X X X X X X X X C X X X X X X X X X X X X C X X X X X X X X X C X X X X X X X X X X X X X C X X X X X X X X X C X X X X X X X C X X C X X X X X X X X C X X X X X X X C X C X X X X X X X X X X X X X X X X X X X C X C X X X X X X X X X X X X X X X X C X X X X X X X X X X X X X X X X X X X C X C X X X X X X X X C X X X X X X C X X X C X X X X X C X X X X X X X X X X X X X X X X X X X X X X X X X X	14
20 3 X X X C X C X X X X X X X X X X X X X X C X X X C X X X X C X X X X C X X X X X X X X X X X X X X X X C C X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X C X X X X X X X C X X X X X X X C X X X X X X X X C X X X X X X X C X X X X C X X X X X X C X X C C C X X X X X X X X X C X X X X X X X X C X X X C X X X X X X X X X C X X C X X X X X X X X C X X X X X X X X X X X C X C X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X C X X X X X X X X X C X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X C C X X X X C X X X X X X X X X X X X X X X C C C X X X X X X X X X X X X X C X X X X X X X C C X X X X X C X X X X X X X X X X X X C X X X X X X X C X X X X X X X X X C X X X X X C X C X X X X X X X X X X X X X X X X	34
3 3 X X X X X X X X X	9

25 2 X X C C C C X X X X C C X X X C X X X X X C X X X C X X X X X X X X X X X X X C X X X X X X X X X X X X X X X X X X X X C X X X X C X X X C C X X C X X X X C X X X X X X X X X X X C X X X X X X X C X C X X X X X X X X C X X C X X X C X X X X X X X X C X X X X X X X C X C X X X X X X X X X C X X X X C C X X X X X X X X X C X X X X X X C X C X C C X X C X X C X X C C X X C X X X X X X X X X X X X X X X X X X X X X X X X X C X X X X X C X X C X X X X X X X X X X X X X X X X X C C X X X X C X X X X X X C X C X X X X C X X X X X C X X X X X C X X C X X X X X X X X X C X X X X X C X X X C X X X X X X X X X C X X C X X X X X X X X C X X X X C X C X X X X X C X X X X X X C C X X X C X C C X C C X X X X X X X X X C X X X X X C X X X X C X X C X C X X C C X X X C C X X X X X X X C C X C X X X X X X X X X X X X X X X X X X X C X X X X X X X X X X X X C X X X X X X X C X X X X X X X C X X X C X C X X C X X X X X C X X X X X X X C C X C X X X X C X X C C C X X C X X X X X C X X X X X X X X C X X X X X X X X X X X C X X X X X X X X X X X X X X C C X X C X C X X X X X X X X X C C X X X X X X C X X X X C X C C X X X X X C X X X X C C X X X X X X C X X X X X C X X X X X C X X X C X X C X X X X C X X C X X X X X X X X X X X C X X X X X X	90
5 4 C X X X X X X X X X X X X X X X X X X X X X X X X	9

Πίνακας 7 Τιμές εισόδου και εξόδου προγράμματος «Σκοτεινά σημεία δωματίου»

8.6 Είσοδοι και έξοδοι για τα προγράμματα του προβλήματος «Μέγιστη επιτρεπόμενη κλίση»

Input(n,v,speeds[n],bends[v])	Output(max_speed, κατάταξη)
8 2 56 6 2 23 9 25 41 15 40 30	22 y h e b c f d a g
11 3 20 15 48 13 53 33 3 37 22 42 17 160 80 20	18 y k b d g f i a c j h e

11 7 25 48 35 41 26 15 32 37 9 46 6 130 120 110 100 90 80 50	29 y e a f i k c g h d j b
11 1 25 17 35 11 2 15 22 56 8 46 6 5	9 y i k e h j c a g b f d
1 1 32 60	31 y a

Πίνακας 8 Τιμές εισόδου και εξόδου προγράμματος «Μέγιστη επιτρεπόμενη κλίση»

Παράρτημα Κώδικα

Εύρεση v+1 όρου ακολουθίας

Πρόγραμμα σε C++

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
using namespace std;

long long count(long long x) {
    long long s = 0;
    vector<long long>num;
    while(x!=0) {
        num.push_back(x%2);
        x=x/2;
    }
    reverse(num.begin(),num.end());
    for (long long y:num) {
        if(y==0) {
            s+=4;
        }
        if(y==1) {
            s+=3;
        }
    }
    return s;
}

int main()
{
    long long start;
    long long n;
    long long temp,i;
    cin >> start >> n; cin.ignore();

    for(i=0; i<n; i++) {
        temp = count(start);
        if(start==temp) {
            break;
        }
        start = temp;
    }

    cout << temp << endl;
}
```

Πρόγραμμα σε Python

```python
import sys
import math

def count(x):
    s = 0
    num = []
    while x!=0:
        num.append(x%2)
        x = int(x/2)
    num.reverse()
    for j in num:
        if j==0:
            s+=4
        if j==1:
            s+=3
    return s

start, n = [int(i) for i in input().split()]
for i in range(0,n):
    temp = count(start)
    if temp==start:
        break
    start = temp
print(temp)
```

Πρόγραμμα σε Matlab

```
function m = numbers()
    start = input("");
    n = input("");
    for v = 1:n
        temp = count(start);
        if temp==start
            break
        end
        start = temp;
    end
    disp(temp);
end


function c = count(x)
    s = 0;
    i = 1;
    num = [];
    while x~=0
        num(i) = mod(x,2);
        x = fix(x/2);
        i = i+1;
    end
    x = flip(x);
    for j = 1:length(num)
        if num(j)==0
            s = s + 4;
```

```
        end
        if num(j)==1
            s = s + 3;
        end
    end
    c = s;
end
```

Το πρόβλημα του περιοδεύοντος πωλητή

Πρόγραμμα σε C++

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <cmath>
using namespace std;

using coords = pair<int, int>;

double distance(const coords &a, const coords &b) {
    const auto x = a.first - b.first;
    const auto y = a.second - b.second;
    return sqrt(pow(x,2) + pow(y,2));
}

int main()
{
    int N; cin >> N; cin.ignore();
    vector<coords> cities(N);
    for (auto &&city : cities) {
        cin >> city.first >> city.second; cin.ignore();
    }

    float total = 0;
    for (auto city = begin(cities); prev(end(cities)) != city; city++) {
        nth_element(next(city), next(city), end(cities),
                [&city](const auto &a, const auto &b) {
                    return distance(*city, a) < distance(*city, b);
                });
        total += distance(*city, *next(city));
    }

    total += distance(*begin(cities), *rbegin(cities));
    cout<<round(total)<<endl;
}
```

Πρόγραμμα σε Python

```python
import sys
import math

def distance(c1, c2):
    x1, y1 = c1
    x2, y2 = c2
    return math.sqrt((x1-x2)**2 + (y1-y2)**2)

def findmin(coord, cur):
    min = 9999999
    curpos = -1
    for x, coordx in enumerate(coord):
        if min > distance(cur, coordx):
            min = distance(cur, coordx)
            curpos = x
    return min, coord[curpos]


def main():
    n = int(input())
    c = []
    first = None
    for i in range(n):
        x, y = map(int, input().split())
        if i == 0:
            first = (x,y)
        else:
            c.append((x,y))
    total = 0
    current = first
    curp = 0
    for j in range(n-1):
        dist, current = findmin(c, current)
        c.remove(current)
        total+= dist

    total+=distance(first,current)
    print(round(total))

main()
```

Πρόγραμμα σε Matlab

```matlab
function m = traveling_salesman()
    n = input("");
    points = zeros(n-1,2);
    first = zeros(1,2);
    for i=1:n
        x = input("");
        y = input("");
        if i == 1
            first(1,1) = x;
            first(1,2) = y;
        end
```

```
        points(i,1) = x;
        points(i,2) = y;

    end
    total = 0;
    current1 = first;
    nn = n;

    for j = 1:nn
        [dist,current,cposition] = findmin(points, current1, nn);
        current1 = current;
        for z =1:nn
            if points(z,1)==current1(1,1) && points(z,2)==current1(1,2)
                points(z,:) = [];
                nn = nn - 1;
                break;
            end
        end

        total = total + dist;
        disp(total);
    end

    m = total + distance(current1(1,1), current1(1,2), first(1,1), first(1,2));
    disp(round(m));
end



function d = distance(c1x, c1y, c2x, c2y)
    d = ((c1x-c2x).^2 + (c1y-c2y).^2).^0.5;
end

function [min,cp,curpos] = findmin(c, cx, m)
    min = 9999999;
    cp = zeros(1,2);
    curpos = -1;
    for x = 1:m
        if min > distance(c(x,1),c(x,2),cx(1,1),cx(1,2))
            min = distance(c(x,1),c(x,2),cx(1,1),cx(1,2));
            curpos = x;
        end
    end
    cp(1,1) = c(curpos,1);
    cp(1,2) = c(curpos,2);
end
```

Συνδυασμοί στο παιχνίδι στο Mölkky

Πρόγραμμα σε C++

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
```

```cpp
using namespace std;

// https://www.geeksforgeeks.org/find-all-combinations-that-adds-upto-given-number-2/
int combinations(int x, int turns=1) {
    if(x == 50) {
        return 1;
    }
    if(x > 50 || turns > 4) {
        return 0;
    }
    int counter = 0;
    for(int i = 1; i<=2; i++) {
        for(int j = i; j<=12; j++) {
            counter+=combinations(x+j, turns+1);
        }
    }
    return counter;

}



int main()
{
    int n;
    cin >> n; cin.ignore();
    cout<<combinations(n)<<endl;
}
```

Πρόγραμμα σε Python

```python
import sys
import math

def combinations(x, turns=1):
    if x == 50:
        return 1
    if x > 50 or turns > 4:
        return 0
    counter = 0
    for i in range(1,3):
        for j in range(i,13):
            counter+=combinations(x+j,turns+1)
    return counter

n = int(input())
print(combinations(n))
```

Πρόγραμμα σε Matlab

```matlab
function combs()
    n = input("");
    disp(combinations(n,1));
```

```
end

function counter = combinations(x,turns)
   if x == 50
      counter = 1;
   elseif x>50 || turns>4
      counter = 0;
   else
      counter = 0;
      for i=1:2
         for j=i:12
            counter = counter + combinations(x+j,turns+1);
         end
      end
   end
end
```

Υπολογισμός τετραγώνων σε αγροτεμάχιο

Πρόγραμμα σε C++

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

struct position {
   int X;
   int Y;
};

int squares(position x[], int m) {
   int counter = 0;
   for(int i = 0; i<m; i++) {
      for(int j = i + 1; j<m; j++) {
         bool flag1 = false;
         bool flag2 = false;
         int x1 = x[i].X, y1 = x[i].Y;
         int x2 = x[j].X, y2 = x[j].Y;
         int x3 = x1 - y1 + x2 + y2;
         int x4 = x1 + y1 + x2 - y2;
         int y3 = x1 + y1 - x2 + y2;
         int y4 = -x1 + y1 + x2 + y2;

         for(int z = 0; z<m; z++) {
            if(2*x[z].X == x3 && 2*x[z].Y == y3) {
               flag1 = true;
            }
            if(2*x[z].X == x4 && 2*x[z].Y == y4) {
               flag2 = true;
            }
         }
         if(flag1 == true && flag2 == true) {
```

```
                counter+=1;
            }
        }
    }
    return counter;

}


int main()
{
    int N;
    cin >> N; cin.ignore();
    position* p = new position[N];


    for (int i = 0; i < N; i++) {
        int X;
        int Y;
        cin >> X >> Y; cin.ignore();
        p[i].X = X;
        p[i].Y = Y;
    }

    cout <<squares(p,N)/2<< endl;
}
```

Πρόγραμμα σε Python

```
import sys
import math

t = [False for x in range(2000)]
cx = [0 for x in range(2000)]
cy = [0 for x in range(2000)]

def examine(x, y):
    return t[y] and t[y][x]

def pegs(i, j):
    dx = cx[j] - cx[i]
    dy = cy[j] - cy[i]
    return examine(cx[j] + dy, cy[j] - dx) and examine(cx[i] + dy, cy[i] - dx)

def main():
    n = int(input())
    for i in range(n):
        x, y = [int(j) for j in input().split()]
        if(t[y] == False):
            t[y] = [False for x in range(1000)]
        t[y][x] = True
        cx[i] = x
        cy[i] = y
        squares = 0
    for i in range(n):
        for j in range(n):
```

```
            if(i!=j and pegs(i,j)):
                squares+=1
    print(int(squares/4))

main()
```

Πρόγραμμα σε Matlab

```
function r = countS()
    n = input("");
    pos(1:2000) = struct('x',[],'y',[]);
    for i=1:n
        pos(i).x = input("");
        pos(i).y = input("");
    end
    r = (squares(pos,n))/2;
    disp(round(r));
end

function counter = squares(p,n)
    counter = 0
    for i = 1:n
        for j = i+1:n
            flag1 = 0;
            flag2 = 0;
            x1 = p(i).x;
            y1 = p(i).y;
            x2 = p(j).x;
            y2 = p(j).y;
            x3 = x1 - y1 + x2 + y2;
            x4 = x1 + y1 + x2 - y2;
            y3 = x1 + y1 - x2 + y2;
            y4 = -x1 + y1 + x2 + y2;
            for z = 1:n
                if 2*p(z).x == x3 && 2*p(z).y == y3
                    flag1 = 1;
                end
                if 2*p(z).x == x4 && 2*p(z).y == y4
                    flag2 = 1;
                end
            end
            if flag1 == 1 && flag2 == 1
                counter = counter + 1;
            end
        end
    end
end
```

Σκοτεινά σημεία δωματίου

Πρόγραμμα σε C++

```
#include <iostream>
#include <string>
```

```
#include <vector>
#include <algorithm>
#include <cmath>

using namespace std;
int N;
vector<vector<int>> x;

int darkspots(int l, string** m) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < x.size(); k++) {
                if (abs(x[k][0] - i) < l && abs(x[k][1] - j) < l) {
                    m[i][j] = 1;
                }
            }
        }
    }

    int sum = 0;

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (m[i][j] == nolight) {
                sum += 1;
            }
        }
    }
    return sum;
}

int main()
{
    cin >> N; cin.ignore();
    int L;
    cin >> L; cin.ignore();

    string* map = new string[10, 10];

    delete[] map;
    map = new string[N, N];

    for (int i = 0; i < N; i++) {
        map[i].resize(N);
    }
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            char cell;
```

```
            cin >> cell; cin.ignore();
            map[i][j] = cell;
            if (map[i][j] == light) {
               x.push_back({ i,j });
            }
         }
      }

      int res = darkspots(L, map);
      cout << res << endl;
   }
```

Πρόγραμμα σε Python

```
import sys
import math

def darkspots(n,l,x,s):
   for i in range(n):
      for j in range(n):
         for z in x:
            if abs(z[0]-i) < l and abs(z[1]-j) < l:
               s[i][j] = '1'
   sum = 0
   for i in range(n):
      for j in range(n):
         if s[i][j] == 'X':
            sum+=1
   return sum

n = int(input())
l = int(input())
x = []
spots = [input().split() for i in range(n)]
for i in range(n):
   for j in range(n):
      if spots[i][j] == 'C':
         x+=[[i,j]]
print(darkspots(n,l,x,spots))
```

Πρόγραμμα σε Matlab

```
function ds = main()
   n = input("");
   l = input("");
   k = 1;
   for i = 1:n
      for j = 1:n
         if i == n+1 && j == n+1
            break
         end
         map(i,j) = input("", 's');
```

```
        if map(i,j) == "C"
            x(k,1) = i;
            x(k,2) = j;
            k = k + 1;
        end
    end
  end
  res = darkspots(l,map,n,x);
  disp(res);
end


function counter = darkspots(l, map, n, x)
  for i = 1:n
    for j = 1:n
      for k = 1:size(x)
        if abs(x(k,1)- i) < l && abs(x(k,2) - j) < l
          map(i,j) = "1";
        end
      end
    end
  end
  sum = 0;
  for i = 1:n
    for j = 1:n
      if map(i,j) == "X"
        sum = sum + 1;
      end
    end
  end
  counter = sum;
end
```

Μέγιστη επιτρεπόμενη κλίση

Πρόγραμμα σε C++

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <cmath>

using namespace std;

class Motorcycle {
  public:
    int speed;
    int num;
    Motorcycle(int speed, int num):speed(speed),num(num) {}
};

static bool comp1(const Motorcycle& a,const Motorcycle& b) {
  return a.speed > b.speed;
```

```
}

int main()
{
    int n;
    cin >> n; cin.ignore();
    int v;
    cin >> v; cin.ignore();
    string names[] = {"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u",
"v","w","x","y","z"};
    vector<Motorcycle> mc;

    for (int i = 0; i < n; i++) {
        int speed;
        cin >> speed; cin.ignore();
        mc.push_back(Motorcycle(speed,i));
    }
    vector<int> bends;
    int radius;
    for(int i = 0; i < v; i++) {
        cin >> radius; cin.ignore();
        bends.push_back(radius);
    }

    vector<int> minSpeed;
    for(int i = 0; i < v; i++) {
        int u = sqrt((tan(60*3.14159265359/180)*bends[i]*9.81));
        minSpeed.push_back(u);
    }
    int min = 999999;
    for(int i = 0; i < v; i++) {
        if(min > minSpeed[i]) {
            min = minSpeed[i];
        }
    }

    for(int i = 0; i < n; i++) {
        int st = 0;
        for(int j = 0; j < v; j++) {
            if(pow(mc[i].speed,2)/(bends[j]*9.81) >= tan(60*3.14159265359/180)) {
                st = st + 1;
            }
        }
        mc[i].speed = mc[i].speed - st*100;
    }
    sort(mc.begin(), mc.end(), comp1);
    cout<<min<<endl;
    cout<<"y"<<endl;
    for(int i = 0; i < n; i++) {
        cout<<names[mc[i].num]<<endl;
    }
}
```

Πρόγραμμα σε Python

```
import sys
```

```python
import math

n = int(input())
v = int(input())
speeds=[]
names={}
for i in range(n):
    speed = int(input())
    speeds.append(speed)
    names[speed]=chr(97+i)
speeds.sort(reverse=True)

ms=[]
for i in range(v):
    bends=int(input())
    maxSpeed=(math.tan(60* math.pi/180)*bends*9.81)**0.5
    ms.append(maxSpeed)
    speeds.sort(key=lambda x: x > maxSpeed)

print(int(min(ms)))
print("y")
for i in speeds:
    print(names[i])
```

Πρόγραμμα σε Matlab

```
function s = stallTilt()
    n = input("");
    v = input("");
    names = ["a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w"
"x" "y" "z"];
    mc(1:n) = struct('sp',[],'num',[]);
    for i = 1:n
        speed = input("");
        mc(i).sp = speed;
        mc(i).num = i;
    end
    for i = 1:v
        radius = input("");
        bends(i) = radius;
    end
    for i = 1:v
        u = sqrt((tan(60*3.14159265359/180)*bends(i)*9.81));
        minSpeed(i) = u;
    end
    min = 999999;
    for i = 1:v
        if min > minSpeed(i);
            min = minSpeed(i);
        end
    end
    for i = 1:n
        st = 0;
        for j = 1:v
            if (mc(i).sp)^2/(bends(j)*9.81) >= tan(60*3.14159265359/180)
                st = st + 1;
```

```
            end
        end
        mc(i).sp = mc(i).sp - st*100;
    end
    [x,idx] = sort([mc.sp],'descend');
    mc = mc(idx);
    disp(fix(min));
    disp("y");
    for i = 1:n
        disp(names(mc(i).num));
    end
end
```