Τεχνολογικό Εκπαιδευτικό Ίδρυμα (ΤΕΙ) Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Ηλεκτρολόγων Μηχανικών

Πτυχιακή Εργασία

Ανάπτυξη εκπαιδευτικού εργαλείου για την διδασκαλία της γλώσσας προγραμματισμού C

Του
Σώζων Κιαγιά
Αρ. Μητρώου: 5791

Επιβλέπων Καθηγητής
Κώστας Βασιλάκης

Ηράκλειο 2022

ΠΕΡΙΛΗΨΗ

Τίτλος: Ανάπτυξη εκπαιδευτικού εργαλείου για την διδασκαλία της γλώσσας προγραμματισμού C.

Στόχος αυτής της πτυχιακής εργασίας ήταν η δημιουργία ενός λογισμικού το οποίο θα χρησιμοποιείται από καθηγητές προγραμματισμού σε εργαστηριακές ασκήσεις της γλώσσας C. Τροποποιώντας το λογισμικό κατάλληλα θα μπορούν οι καθηγητές να δημιουργούν εργαστηριακές ασκήσεις για να τις αναθέτουν στους φοιτητές για την ευκολότερη εκμάθηση και κατανόηση διάφορων μεθόδων που χρησιμοποιούνται στην γλώσσα προγραμματισμού C για έλεγχο συσκευών.
Το λογισμικό είναι γραμμένο στην γλώσσα C++ και απεικονίζει μέσω γραφικών δύο εικονικούς βραχίονες οι οποίοι κάνουν κινήσεις (δεξιά , αριστερά , πάνω , κάτω κλπ) μέσω functions που πρέπει να καλούν οι φοιτητές μέσα από προγράμματα τους. Το λογισμικό τρέχει μέσω ενός εικονικού δικτύου δημιουργημένου με sockets με πρωτόκολλο TCP και μέσω επικοινωνίας του client και του server. Η επιθυμητή εντολή στέλνεται με κλήση της αντίστοιχής function που έχει αναπτυχθεί και ο βραχίονας θα εκτελέσει την κατάλληλη κίνηση.

ABSTRACT

Title: Development of an educational tool for teaching the C programming language.

The aim of this thesis was to create a software that will be used by programming teachers. By modifying the software appropriately, teachers will be able to create exercises to assign to students for easier learning and understanding of various methods used in the C programming language for controling devices.
The software is written in the C++ language and graphically illustrates two virtual arms that make movements (right, left, up, down, etc.) through functions that the students must call from their programms. The software runs through a virtual network created with sockets with TCP protocol and through client and server communication. By calling the appropriate function in student's program, the arm wil do the desired movement.

Πίνακας περιεχομένων

ΚΕΦΑΛΑΙΟ 1:ΕΙΣΑΓΩΓΗ

1.1 Γενική Περιγραφή

Το λογισμικό που αναπτύχθηκε έχει σκοπό την διευκόλυνση στην εκμάθηση διαφόρων μεθόδων που χρησιμοποιούνται στην γλώσσα προγραμματισμού C για έλεγχο ψηφιακών διατάξεων. Εκτελώντας ασκήσεις με το συγκεκριμένο λογισμικό , οι φοιτητές θα κατανοούν πολύ πιο εύκολα κάποιες βασικές μεθόδους που χρησιμοποιούνται πολύ συχνά στην C, αλλά και πιο εξειδικευμένες μεθόδους που πιθανόν να χρειαστεί να χρησιμοποιήσουν στο μέλλον για κάποιο άλλο μάθημα. Με αυτόν τον τρόπο, το λογισμικό θα βοηθήσει τους φοιτητές να εμβαθύνουν τις γνώσεις τους πάνω στην γλώσσα C και θα τους προετοιμάσει και για τεχνολογίες και μεθόδους που θα συναντήσουν στο μέλλον. Θα μάθουν τεχνολογίες όπως την εφαρμογή γραφικού περιβάλλοντος στην C++, καθώς και την δημιουργία και εφαρμογή ενός εικονικού δικτύου με sockets.

1.2 Δομή Πτυχιακής Εργασίας

Η πτυχιακή εργασία,αποτελείται από 6 Κεφάλαια,και θα έχει την παρακάτω δομή:

Κεφάλαιο 2 : Γενική Περιγραφή Θεωρητικού Μέρους του Λογισμικού

Κεφάλαιο 3 : Περιγραφή Ειδικού Μέρους του Λογισμικού

Κεφάλαιο 4 : Εργαστηριακές Ασκήσεις

Κεφάλαιο 5 : Εγχειρίδιο Χρήσης

Κεφάλαιο 6 : Συμπεράσματα

ΚΕΦΑΛΑΙΟ 2: ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ ΘΕΩΡΙΑΣ

2.1 Εισαγωγή στον δικτυακό προγραμματισμό

Το internet παρέχει έναν γενικό μηχανισμό επικοινωνίας, τον οποίο μπορούμε να εκμεταλλευτούμε για την δημιουργία διάφορων υπηρεσιών. Η δημιουργία των μεμονωμένων υπηρεσιών οι οποίες παρέχονται από προγράμματα εφαρμογών, εκτελούνται σε υπολογιστές οι οποίοι είναι συνδεδεμένοι στο internet. Οι υπηρεσίες αυτές στηρίζονται σε δύο βασικά υποδείγματα επικοινωνιών:

- το υπόδειγμα ρευμάτων (stream)
- το υπόδειγμα μηνυμάτων.

2.1.1 Υπόδειγμα ρευμάτων

Το υπόδειγμα ρευμάτων χρησιμοποιεί το πρωτόκολλο TCP για την αποστολή μίας μεμονωμένης ακολουθίας byte, η οποία ρέει από ένα πρόγραμμα εφαρμογής σε ένα άλλο. Διαθέτει αυθαίρετο μήκος μεταφοράς και χρησιμοποιείται από τις περισσότερες εφαρμογές. Η υπηρεσία ρευμάτων είναι συνδεσμική, το οποίο σημαίνει ότι πρέπει να υπάρξει πρώτα σύνδεση μεταξύ πελάτη-διακομιστή για την επιτυχής αποστολή δεδομένων και η επικοινωνία να είναι αμφιμονοσήμαντη (ένα-προς-ένα).

2.1.2 Υπόδειγμα μηνυμάτων

Το υπόδειγμα μηνυμάτων χρησιμοποιεί τα πρωτόκολλα UDP/TCP για την αποστολή μίας ακολουθίας διακριτών μηνυμάτων, δηλαδή το δίκτυο δέχεται και παραδίδει μηνύματα. Σε αντίθεση με το υπόδειγμα ρευμάτων, το υπόδειγμα μηνυμάτων περιορίζει το κάθε μήνυμα να μην υπερβαίνει τα 64Kbyte. Η υπηρεσία μηνυμάτων είναι ασυνδεσμική οπότε δεν απαιτείται η σύνδεση πελάτη-διακομιστή (client-server) για την επιτυχής αποστολή μηνυμάτων και η επικοινωνία είναι πολυσήμαντη (πολλά-προς-πολλά).Χρησιμοποιείται κυρίως από εφαρμογές πολυμέσων όπως μία εφαρμογή επεξεργασία ήχου , επεξεργασία εικόνας κλπ.

2.2 Πρωτόκολλο UDP (Αυτοδύναμων Πακέτων Χρήστη)

Ένα από τα δύο κύρια πρωτόκολλα μεταφοράς της οικογένειας πρωτοκόλλων TCP/IP είναι το πρωτόκολλο UDP. Το πρωτόκολλο UDP είναι απλό στην χρήση σε σχέση με το πρωτόκολλο TCP αλλά για αυτόν τον λόγο δεν παρέχει τον ίδιο αριθμό υπηρεσιών που παρέχει το TCP σε μία εφαρμογή. Το UDP έχει τα εξής χαρακτηριστικά:

- Δεν απαιτείται σύνδεση για την αποστολή μηνυμάτων.
- Μπορεί να διακρίνει διαφορετικά προγράμματα εφαρμογών που εκτελούνται στον ίδιο υπολογιστή.
- Μπορεί να στέλνει και να λαμβάνει μεμονωμένα μηνύματα.
- Προσφέρει στις εφαρμογές την βέλτιστη προσπάθεια παράδοσης.
- Δίνει την δυνατότητα στην εφαρμογή να είναι αυθαίρετης αλληλεπίδρασης.
- Μπορεί να αναγνωρίσει εφαρμογές ανεξάρτητα του λειτουργικού συστήματος.

Το πρωτόκολλο UDP είναι ιδανικό όταν θέλουμε μια εφαρμογή να στέλνει και να λαμβάνει μηνύματα σε ξεχωριστά αυτοδύναμα πακέτα και θέλουμε να έχει αποκλειστική επικοινωνία με μία η περισσότερες εφαρμογές. Δυστυχώς όμως , το πρωτόκολλο UDP έχει και τα μειονεκτήματα του, τα οποία περιγράφουμε παρακάτω:

- Μπορεί το μήνυμα να χαθεί κατά την μεταφορά του.
- Μπορεί το μήνυμα να επαναλαμβάνετε.
- Μπορεί το μήνυμα να καθυστερήσει.
- Μπορεί το μήνυμα να παραδοθεί εκτός σειράς.
- Μπορεί το μήνυμα να αλλοιωθεί κατά την μεταφορά του.

Για αυτόν το λόγο πρέπει να επιλέγουμε το πρωτόκολλο UDP σε εφαρμογές οι οποίες είναι πιο ανεκτικές σε αυτά τα σφάλματα όπως είναι για παράδειγμα μία εφαρμογή μεταφοράς ήχου η μια εφαρμογή μεταφοράς εικόνας.

2.3 Πρωτόκολλο Έλεγχου Μετάδοσης (TCP)

Το δεύτερο κύριο πρωτόκολλο μεταφοράς της οικογένειας πρωτοκόλλων TCP/IP είναι το πρωτόκολλο έλεγχου μετάδοσης (TCP), το οποίο είναι το κύριο πρωτόκολλο που χρησιμοποιείται στις περισσότερες εφαρμογές του internet, λόγω της αξιοπιστίας που προσφέρει στην παράδοση δεδομένων. Τα χαρακτηριστικά για το πρωτόκολλο έλεγχου μετάδοσης είναι τα εξής:

- Απαιτείται σύνδεση για την μεταφορά δεδομένων.
- Εγγυάται ότι όλα τα δεδομένα που στάλθηκαν , παραδίδονται χωρίς καμία αλλοίωση, θα είναι πλήρης και στην σωστή σειρά.
- Οποιαδήποτε σύνδεση TCP θα έχει ακριβώς δύο ακραία σημεία.
- Επιτρέπει την ροή δεδομένων να υπάρχει και στις δύο κατευθύνσεις δίνοντας την δυνατότητα να επικοινωνούν οι δύο εφαρμογές μεταξύ τους στέλνοντας δεδομένα οποιαδήποτε στιγμή (όσο υπάρχει σύνδεση).
- Στέλνει δεδομένα μέσω μίας ακολουθίας byte.
- Εγγυάται ότι οι δύο εφαρμογές θα ξεκινήσουν την επικοινωνία με αξιόπιστο τρόπο.
- Εγγυάται ότι θα γίνει ομαλό κλείσιμο της σύνδεσης αφού εξασφαλίσει ότι όλα τα δεδομένα έχουν μεταφερθεί σωστά.

Το TCP είναι ιδανικό για εφαρμογές που χρειάζονται πλήρης αξιοπιστία και η ταχύτητα μεταφοράς δεδομένων δεν είναι τόσο κρίσιμη. Οι συνδέσεις που γίνονται μέσω TCP ονομάζονται εικονικές (virtual connections) διότι γίνονται μέσω λογισμικού.

Γενικά ,τα πρωτόκολλα μεταφοράς χρησιμοποιούν διάφορους μεθόδους και μηχανισμούς για την εξασφάλιση μια αξιόπιστης υπηρεσίας. Το TCP χρησιμοποιεί έναν περίπλοκο συνδυασμό διάφορων μεθόδων και μηχανισμών για την εξασφάλιση της αξιοπιστίας του και για αυτόν τον λόγο θεωρείται το καλύτερο πρωτόκολλο για εφαρμογές του internet.

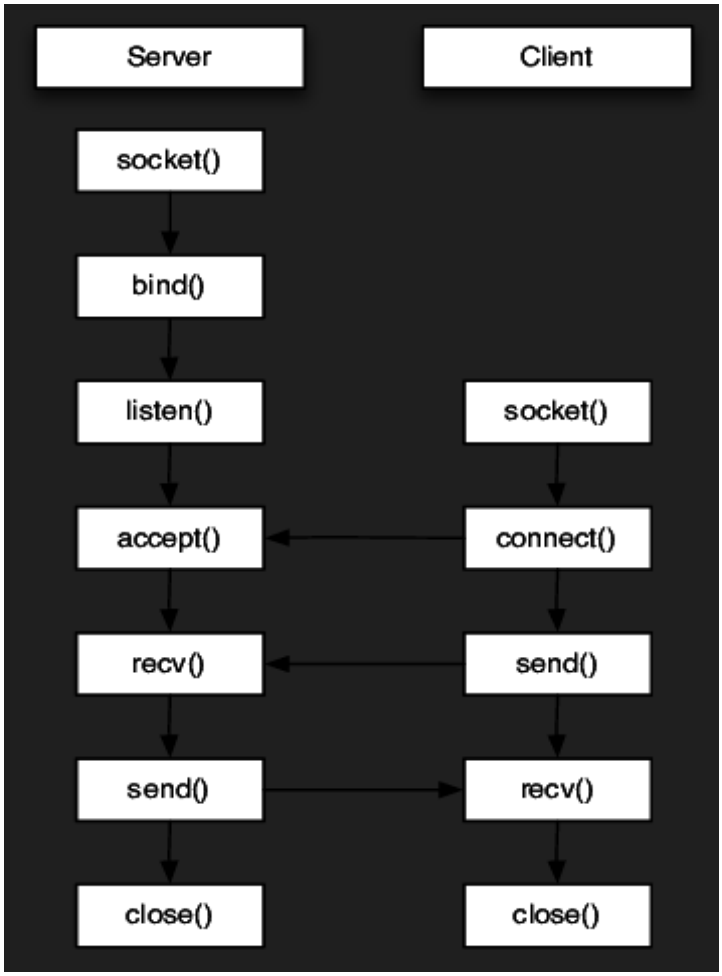
2.4 Διασύνδεση Προγραμματισμού Εφαρμογών (Application Program Interface,API)

Η διασύνδεση προγραμματισμού εφαρμογών είναι η διαδικασία της διασύνδεσης μιας εφαρμογής για τον καθορισμό της επικοινωνίας της στο internet. Οι λεπτομέρειες της διασύνδεσης εξαρτώνται από το λειτουργικό σύστημα που χρησιμοποιούμε, αλλά γενικά η διασύνδεση API είναι το πρότυπο για την επικοινωνία στο internet μέσω λογισμικού. Το API υποδοχών (socket API) είναι διαθέσιμο για χρήση από τα περισσότερα λειτουργικά συστήματα όπως τα windows , τα linux , το android κλπ. Επειδή στον προγραμματισμό υποδοχών χρειαζόμαστε πολλές λεπτομέρειες όπως την διεύθυνση ενός υπολογιστή , έναν αριθμό θύρας πρωτοκόλλου κτλπ, οι σχεδιαστές των API υποδοχών αποφάσισαν να δημιουργήσουν πολλές συναρτήσεις για περισσότερη ευκολία και ευκρίνεια. Αρχικά πρέπει να δημιουργήσουμε μία υποδοχή (socket) με την συνάρτηση socket() η οποία θα μας επιστρέψει έναν ακέραιο περιγραφέα και έπειτα να καλέσουμε τις συναρτήσεις για τον καθορισμό των λεπτομερειών που αναφέραμε. Η εφαρμογή που μπορούμε να δημιουργήσουμε με το API υποδοχών θα μπορεί να ενεργήσει είτε ως πελάτης (client), είτε ως διακομιστής (server). Κάποιες από τις συναρτήσεις μπορούν να χρησιμοποιηθούν μόνο αν η εφαρμογή ενεργηθεί ως πελάτης, κάποιες μόνο αν η εφαρμογή ενεργηθεί ως διακομιστής (server), και κάποιες συναρτήσεις μπορούν να χρησιμοποιηθούν και από τους δύο.

Οι βασικές συναρτήσεις του API υποδοχών είναι οι εξής:

Συνάρτηση	Χρησιμοποιείται από	Χρήση
socket()	Διακομιστή και Πελάτη	Αυτή η συνάρτηση δημιουργεί μία υποδοχή για χρήση από τις παρακάτω συναρτήσεις
accept()	Διακομιστή	Αυτή η συνάρτηση αποδέχεται μία εισερχόμενη κλήση
bind()	Διακομιστή	Αυτή η συνάρτηση καθορίζει την διεύθυνση IP και την θύρα πρωτοκόλλου
connect()	Πελάτη	Αυτή η συνάρτηση δημιουργεί μια σύνδεση με μία απομακρυσμένη εφαρμογή
listen()	Διακομιστή	Αυτή η συνάρτηση προετοιμάζει την υποδοχή για χρήση από τον διακομιστή
recv()	Διακομιστή και Πελάτη	Αυτή η συνάρτηση λαμβάνει τα εισερχόμενα δεδομένα η το μήνυμα
send()	Διακομιστή και Πελάτη	Αυτή η συνάρτηση κάνει αποστολή εξερχόμενων δεδομένων η μηνυμάτων

shutdown()	Διακομιστή και Πελάτη	Αυτή η συνάρτηση τερματίζει την σύνδεση
close()	Διακομιστή και Πελάτη	Αυτή η συνάρτηση τερματίζει την επικοινωνία



Εικόνα 1: Επικοινωνία Διακομιστή-Πελάτη

Στην παραπάνω εικόνα (Εικόνα 1) βλέπουμε την ακολουθία των συναρτήσεων υποδοχών που καλούνται από έναν διακομιστή και έναν πελάτη χρησιμοποιώντας το υπόδειγμα ρευμάτων.

Το API υποδοχών που θα χρησιμοποιήσουμε εμείς θα είναι η βιβλιοθήκη <winsoch2.h> για τα λειτουργικά συστήματα windows η οποία περιλαμβάνει τις παραπάνω και τις υπόλοιπες συναρτήσεις του API υποδοχών. Οι συναρτήσεις παραμένουν ίδιες σε όλα τα λειτουργικά συστήματα.

2.5 Γραφικό περιβάλλον υπολογιστή

Τα γραφικά υπολογιστών είναι η κατασκευή και η επεξεργασία μιας πληροφορίας η οποία βρίσκεται σε μορφή εικόνας. Με την χρήση τελιών, κουκίδων, καμπύλων, σχεδίων κλπ μπορούμε να εισάγουμε, να δημιουργήσουμε και να επεξεργαστούμε γραφικά υπολογιστών με την χρήση κατάλληλων εφαρμογών όπως λογισμικά επεξεργασίας εικόνας η λογισμικά τρισδιάστατης μοντελοποίησης. Υπάρχουν δύο βασικές κατηγορίες που χωρίζονται τα γραφικά : τα δισδιάστατα (2D) γραφικά και τα τρισδιάστατα (3D) γραφικά.

2.6 Δισδιάστατα γραφικά (2D)

Τα δισδιάστατα γραφικά αποτελούν προσπάθειες απεικόνισης γραφικών των δύο διαστάσεων μέσα σε μία οθόνη μιας συσκευής. Χρησιμοποιούνται κυρίως για την απεικόνιση μιας εικόνας , ενός έντυπου κλπ και χωρίζονται και αυτά σε δύο κατηγορίες:

- Διανυσματικά γραφικά (Vector graphics)
- Γραφικά ψηφίδων (Bitmap graphics)

Εικόνα 2: Δισδιάστατα Γραφικά

2.6.1 Διανυσματικά Γραφικά (Vector Graphics)

Τα διανυσματικά γραφικά χρησιμοποιούνται για την δημιουργία εικόνων που αποθηκεύονται ως μαθηματικοί τύποι της αναλυτικής γεωμετρίας και περιγράφουν γεωμετρικά σχήματα (τετράγωνα, τρίγωνα κλπ). Επομένως μπορούμε να χρησιμοποιήσουμε δισδιάστατους γεωμετρικούς μετασχηματισμούς για την επεξεργασία και την τροποποίηση των σχημάτων που θα δημιουργήσουμε. Μερικοί από τους γεωμετρικούς μετασχηματισμούς που μπορούμε να χρησιμοποιήσουμε είναι:

- Μεταφορά (Translation) για την μετακίνηση του σχήματος μας.
- Περιστροφή (Rotation) για την περιστροφή του σχήματος μας.
- Κλιμάκωση (Scaling) για την μεγέθυνση η σμίκρυνση του σχήματος μας.

2.6.2 Γραφικά Ψηφίδων (Bitmap Graphics)

Τα γραφικά ψηφίδων είναι τα γραφικά τα οποία δημιουργούνται από την ψηφιοποίηση διάφορων υπαρκτών αντικειμένων όπως μια φωτογραφία ενός τοπίου η μια ζωγραφιά. Στα γραφικά ψηφίδων πρέπει να λαμβάνουμε υπόψιν μας και την ανάλυση του αντικειμένου διότι αν για παράδειγμα μια φωτογραφία είναι μικρότερη ανάλυση από αυτήν της οθόνης , τότε αν δεν προβληθεί σε μικρότερες διαστάσεις, η εικόνα θα φαίνεται θολή και χωρίς ευκρίνεια.

2.7 Τρισδιάστατα Γραφικά (3D)

Τα τρισδιάστατα γραφικά αποτελούν προσπάθεια απεικόνισης γραφικών τριών διαστάσεων σε μία δισδιάστατη οθόνη μίας συσκευής. Είναι ένας συνδυασμός των μεθόδων που χρησιμοποιούνται στα διανυσματικά γραφικά και στα γραφικά ψηφίδων και χρησιμοποιούνται κυρίως σε εφαρμογές όπως παιχνίδια υπολογιστών. Υπάρχουν δύο κατηγορίες τρισδιάστατων γραφικών οι οποίες είναι η εξής:

- Στατικά γραφικά υπολογιστών (Pre-rendered graphics).
- Γραφικά υπολογιστών πραγματικού χρόνου(Rendered graphics).

Εικόνα 3: Τρισδιάστατα Γραφικά

2.7.1 Στατικά Γραφικά Υπολογιστών (Pre-renderd Graphics)

Τα στατικά γραφικά υπολογιστών είναι τα τρισδιάστατα γραφικά τα οποία δεν αποδίδονται σε πραγματικό χρόνο. Τα αντικείμενα των γραφικών είναι προϋπολογισμένα και προεπεξεργασμένα και δεν αλλάζουν επομένως δεν είναι αλληλεπιδραστικά. Αυτά τα γραφικά αναπαράγονται κυρίως σαν κάποιο βίντεο όπως αυτά τα βίντεο που παίζουν σε παιχνίδια υπολογιστών.

2.7.2 Γραφικά Υπολογιστών Πραγματικού Χρόνου (Rendered Graphics)

Τα γραφικά υπολογιστών πραγματικού χρόνου είναι τα αντικείμενα γραφικών τα οποία αποδίδονται οπτικά την στιγμή που εκτελείται το πρόγραμμα και μπορούν να είναι και αλληλεπιδραστικά. Παράδειγμα ενός τέτοιου προγράμματος είναι τα παιχνίδια του υπολογιστή.

2.8 Γραφικό Περιβάλλον στην C/C++

Τα γραφικά τα οποία θα χρησιμοποιήσουμε στην γλώσσα προγραμματισμού C ,θα είναι δισδιάστατα και θα χρησιμοποιήσουμε την μέθοδο των διανυσματικών γραφικών. Θα χρησιμοποιήσουμε την βιβλιοθήκη graphics.h η οποία είναι από τις πιο απλές και εύκολες βιβλιοθήκες γραφικών και θα μας δώσει πρόσβαση σε συναρτήσεις που θα μας επιτρέψουν την απεικόνιση διάφορων γεωμετρικών σχημάτων όπως μία γραμμή , έναν κύκλο , ένα τετράγωνο κτλπ. Αρχικά , θα παρουσιάσουμε τις βασικές συναρτήσεις που θα χρειαστούμε για την δημιουργία του γραφικού περιβάλλοντος. Οι συναρτήσεις είναι οι εξής:

Συνάρτηση	Χρήση
intigraph()	Αυτή η συνάρτηση χρησιμοποιείται για την εκκίνηση της λειτουργίας των γραφικών
detectgraph()	Αυτή η συνάρτηση χρησιμοποιείται για τον καθορισμό του hardware των γραφικών στο σύστημα
cleardevice()	Αυτή η συνάρτηση ανανεώνει τα γραφικά
closegraph()	Αυτή η συνάρτηση χρησιμοποιείται για τον τερματισμό της λειτουργίας γραφικών

Ας δούμε τώρα κάποιες συναρτήσεις για την δημιουργία γεωμετρικών σχημάτων:

Συνάρτηση	Χρήση
line(int starx , int starty , int endx , int endy)	Αυτή η συνάρτηση ζωγραφίζει μία γραμμή στο γραφικό περιβάλλον
arc(int x , int y , int start , int end , int radius)	Αυτή η συνάρτηση ζωγραφίζει ένα τόξο στο γραφικό περιβάλλον
bar(int left , int top , int right , int bottom)	Αυτή η συνάρτηση ζωγραφίζει μία τετράγωνη μπάρα στο γραφικό περιβάλλον
bar3d(int left , int top , int right , int bottom, int depth , int top flag)	Αυτή η συνάρτηση ζωγραφίζει μία τρισδιάστατη τετράγωνη μπάρα στο γραφικό περιβάλλον
circle(int x , int y , int radius)	Αυτή η συνάρτηση ζωγραφίζει έναν κύκλο στο γραφικό περιβάλλον
drawpoly(int numpoint , int far *points)	Αυτή η συνάρτηση ζωγραφίζει ένα πολύγωνο στο γραφικό περιβάλλον
ellipse(int x , int y , int start , int end , int xradius , int yradius)	Αυτή η συνάρτηση ζωγραφίζει μία έλλειψη στο γραφικό περιβάλλον
rectangle(int left , int top , int right , int bottom)	Αυτή η συνάρτηση ζωγραφίζει ένα ορθογώνιο στο γραφικό περιβάλλον

Η βιβλιοθήκη μας δίνει επίσης την δυνατότητα να εισάγουμε και κείμενο στο γραφικό περιβάλλον με την προϋπόθεση ότι έχουμε συμπεριλάβει και την βιβλιοθήκη <conio.h>. Μερικές συναρτήσεις για την εισαγωγή κειμένου στο γραφικό περιβάλλον είναι οι εξής:

Συνάρτηση	Χρήση
outtext(char far *str)	Αυτή η συνάρτηση εμφανίζει ένα κείμενο στο γραφικό περιβάλλον
outtext(int x , int y , char *str)	Αυτή η συνάρτηση εμφανίζει ένα κείμενο στο γραφικό περιβάλλον σε σημείο που θα ορίσουμε.

Τέλος , υπάρχουν και συναρτήσεις οι οποίες μπορούν να αλλάξουν το χρώμα των σχημάτων που ζωγραφίζονται στο γραφικό περιβάλλον. Μία από αυτές είναι η setcolor() και μπορούμε να βάλουμε χρώματα όπως μαύρο , μπλε , πράσινο κτλπ αν το επιθυμούμε. π.χ. setcolor(BLUE); για μπλε χρώμα.

Η βιβλιοθήκη graphics.h είναι από τις παλαιότερες βιβλιοθήκες γραφικών που υπάρχουν και τρέχει σε περιβάλλον MS-DOS οπότε δεν έχει πάρα πολλές δυνατότητες. Δεν χρησιμοποιείται σχεδόν καθόλου πια, αλλά η απλότητα της την καθιστά ιδανική ως εισαγωγή στα γραφικά και η παλαιότητα της εξασφαλίζει ότι όλα τα σύγχρονα συστήματα θα μπορούν να την τρέξουν χωρίς κανένα πρόβλημα.

ΚΕΦΑΛΑΙΟ 3:ΠΕΡΙΓΡΑΦΗ ΕΙΔΙΚΟΥ ΜΕΡΟΥΣ ΤΟΥ ΛΟΓΙΣΜΙΚΟΥ

Η πτυχιακή αποτελείται από τρία διαφορετικά προγράμματα. Το πρώτο εκτελέσιμο πρόγραμμα αποτελεί την δημιουργία του γραφικού περιβάλλοντος,που θα περιλαμβάνει τους εικονικούς βραχίονες και την δημιουργία των συναρτήσεων που θα καλούνται όταν θα θέλουμε οι βραχίονες να κάνουν κάποια κίνηση. Το δεύτερο εκτελέσιμο πρόγραμμα θα αποτελεί τον διακομιστή της υποδοχής που θα επικοινωνούν, ενώ το τρίτο εκτελέσιμο πρόγραμμα θα αποτελεί τον πελάτη της υποδοχής που θα επικοινωνούν.

3.1 Γραφικό Περιβάλλον

Το λογισμικό δημιουργήθηκε με σκοπό να μπορεί να εκτελεστεί στους υπολογιστές που παρέχονται στα εργαστήρια του ΤΕΙ. Για αυτόν τον λόγο το λογισμικό δημιουργήθηκε στο πρόγραμμα Dev-C++ το οποίο δουλεύει με gcc compiler. Επειδή όμως το πρόγραμμα Dev-C++ είναι αρκετά παλιό , τα γραφικά που μπορούμε να δημιουργήσουμε μπορούν να είναι μόνο σε δισδιάστατη μορφή σε περιβάλλον MS-DOS.

3.1.1 Δημιουργία και περιστροφή βραχιόνων

Έχουμε ορίσει μια συνάρτηση στροφής εν ονόματι rectangleRotate() την οποία θα χρησιμοποιήσουμε για να σχεδιάσουμε ένα ορθογώνιο το οποίο θα περιστρέφεται σε οποιαδήποτε γωνία του δώσουμε. Η συνάρτηση rectangleRotate() έχει τα εξής ορίσματα:

- Το cx το οποίο είναι η συντεταγμένη του x από το κέντρο του ορθογωνίου.
- Το cy το οποίο είναι η συντεταγμένη του y από το κέντρο του ορθογωνίου.
- Το w το οποίο είναι το πάχος του ορθογώνιου.
- Το h το οποίο είναι το ύψος του ορθογώνιου.
- Το angle το οποίο είναι η γωνία περιστροφής του ορθογωνίου.

Αρχικά πρέπει να μετατρέψουμε την τιμή της γωνίας από degrees σε radian οπότε ορίζουμε μια μεταβλητή theta τύπου double και την χρησιμοποιούμε για να κάνουμε την μετατροπή με τον τύπο όπως φαίνεται παρακάτω:

```
double theta = (double)(angle%180)*M_PI/180.0;
```

Έπειτα, ορίζουμε άλλες δύο μεταβλητές τύπου integer, τις dx και dy.

- Το dx ισούται με το μισό του πάχους του ορθογωνίου (dx = w/2).
- Το dy ισούται με το μισό του ύψους του ορθογωνίου (dy = h/2).

Το ορθογώνιο έχει ως ιδιότητα όλες οι κορυφές του ορθογωνίου να έχουν ίσες αποστάσεις από το κέντρο του. Οπότε αν θεωρήσουμε το κέντρο του ορθογωνίου ως την αρχή των αξόνων , τότε μπορούμε να χρησιμοποιήσουμε το dx και το dy για να υπολογίσουμε τις συντεταγμένες x και y όλων των κορυφών του ορθογωνίου.

Στην συνέχεια ορίζουμε έναν πίνακα τύπου integer 8 θέσεων τον οποίο θα τον χρησιμοποιήσουμε για να αποθηκεύουμε τις συντεταγμένες x και y όλων των κορυφών του ορθογωνίου μετά την περιστροφή του.

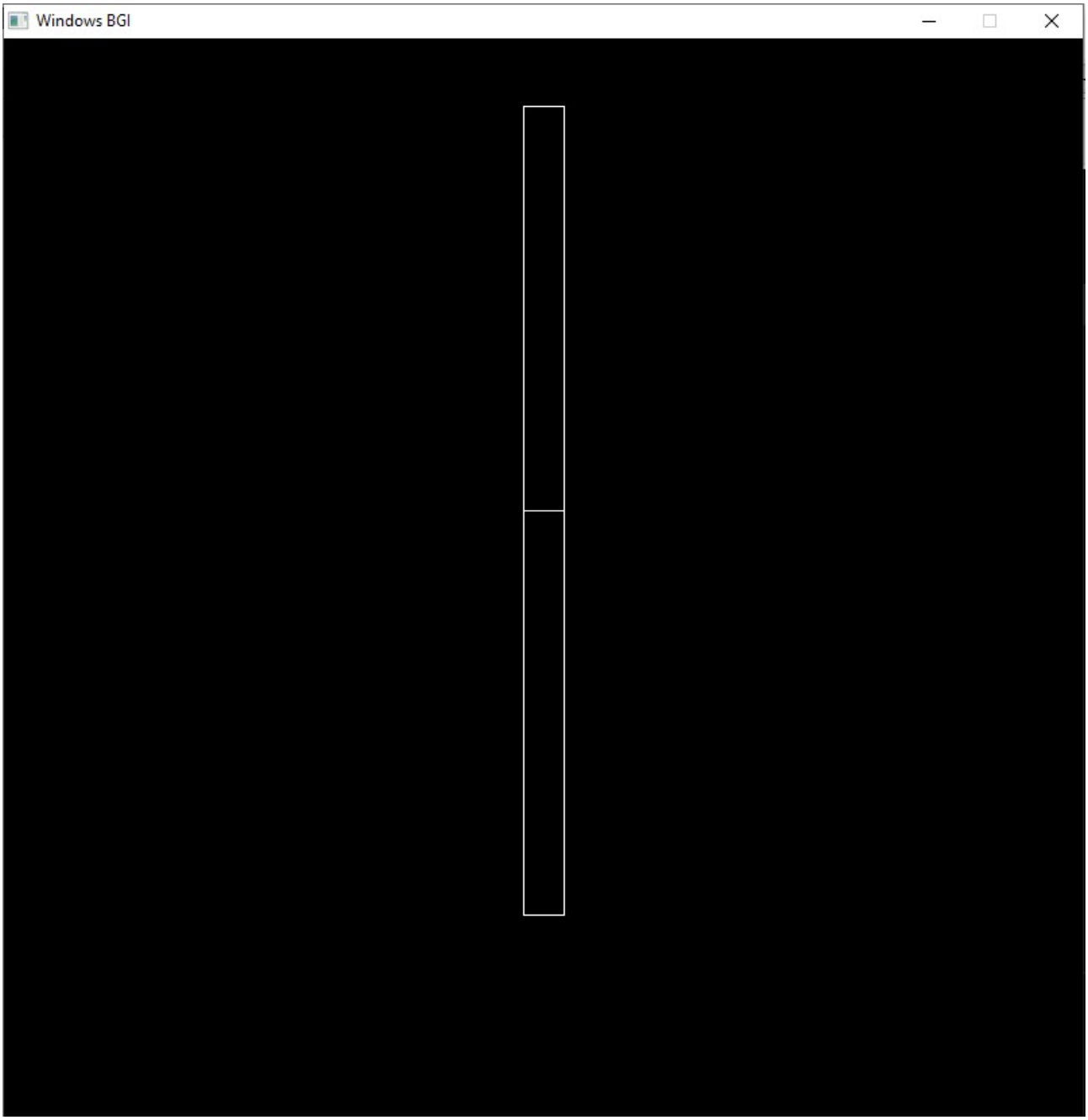
```
int point[8] = {
(int)(-dx*cos(theta) - dy*sin(theta)+cx) ,
(int)(-dx*sin(theta) + dy*cos(theta)+cy),
(int)(dx*cos(theta) - dy*sin(theta)+cx),
(int)(dx*sin(theta) + dy*cos(theta)+cy),
(int)(dx*cos(theta) + dy*sin(theta)+cx),
(int)(dx*sin(theta) - dy*cos(theta)+cy),
(int)(-dx*cos(theta) + dy*sin(theta)+cx),
(int)(-dx*sin(theta) - dy*cos(theta)+cy)

};
```

Όλες οι τιμές που βρίσκονται σε θέση ζυγού αριθμού (γνωρίζοντας ότι η C++ είναι Zero-based γλώσσα) εκπροσωπούν τις συντεταγμένες του x και όλες οι τιμές που βρίσκονται σε θέση περιττού αριθμού εκπροσωπούν τις συντεταγμένες του y. Στην πρώτη γραμμή χρησιμοποιούμε τον μαθηματικό τύπο περιστροφής ορθογωνίου για να βρούμε την συντεταγμένη x θεωρώντας το κέντρο του ορθογωνίου ως την αρχή των αξόνων και προσθέτουμε το cx για να το μεταφέρουμε κατά cx αριστερά τοποθετώντας το κέντρο του ορθογωνίου στην αρχή των αξόνων. Ακολουθώντας την ίδια συλλογιστική πορεία, πράττουμε όμοια και για το cy. Με τον ίδιο τρόπο, βρίσκουμε τις υπόλοιπες τιμές του x και του y μετά την περιστροφή όπως φαίνεται στις υπόλοιπες γραμμές.

Έχουμε τώρα τις συντεταγμένες του x και του y όλων των τεσσάρων κορυφών του περιστρεφόμενου ορθογωνίου και μπορούμε να χρησιμοποιήσουμε αυτές τις τιμές για την σχεδίαση του.
Σε μία for loop θα χρησιμοποιήσουμε την line() συνάρτηση από την βιβλιοθήκη graphics.h για να σχεδιάσουμε όλες τις τέσσερις πλευρές του ορθογώνιου.

```
for (int i=0; i<8; i+=2)
   line(point[i],point[i+1],point[(i+2)%8],point[(i+3)%8]);
```

Θα καλέσουμε την συνάρτηση rectangleRotate() που δημιουργήσαμε δύο φορές για την δημιουργία δύο εικονικών βραχιόνων. Θεωρούμε ότι ο πρώτος βραχίονας βρίσκεται αρχικά σε όρθια θέση (angle = 0) και ο δεύτερος σε κάθετη θέση (angle = 180) και τους εμφανίζουμε σε ένα παράθυρο με διαστάσεις 800x800 χρησιμοποιώντας την συνάρτηση initwindow(800,800) η οποία βρίσκεται επίσης στην βιβλιοθήκη graphics.h.



Εικόνα 4: Οι εικονικοί βραχίονες στην αρχική τους θέση

Με τον μαθηματικό τύπο που χρησιμοποιήσαμε στην συνάρτηση rectangleRotate() ο βραχίονας θα περιστρεφόταν γύρω από το σημείο που αντιστοιχεί στον κέντρο του ορθογωνίου αυτού. Για λόγους σαφήνειας, θέτω δύο πίνακες τύπου integer 12 θέσεων, τους movex και movey οι οποίοι αντιστοιχούν στις συνθήκες αντιστοίχησης cx και cy οι οποίες διέπουν την στροφή του βραχίονα με κέντρο αναφοράς της στροφής να είναι το σημείο που αντιστοιχεί στην μέση της κάτω πλευράς του ορθογωνίου , όταν ο βραχίονας βρίσκεται στην αρχική (όρθια) θέση.

```
int movex[12] = {400,470,530,550,530,470,400,330,270,250,270,330};
int movey[12] = {200,215,260,335,410,460,500,460,410,335,260,215};
```

Οι συνθήκες αντιστοίχισης που φαίνονται στην παραπάνω εικόνα έχουν βρεθεί χειρονακτικά οπότε πιθανώς να υπάρχουν αποκλίσεις από το σημείο που θέλαμε να κάνει την περιστροφή.

Οι δύο πίνακες που δημιουργήσαμε έχουν 12 θέσεις διότι θέλουμε κάθε φορά που κινείται ο βραχίονας μία θέση, να κινείται κατά 30 μοίρες έτσι ώστε να έχει εύρος από 0 έως 330 μοίρες για να μπορεί να κάνει έναν ολόκληρο κύκλο.

3.1.2 Δημιουργία Συναρτήσεων Για Τις Κινήσεις Των Βραχιόνων

Έχουμε δημιουργήσει συνολικά 16 συναρτήσεις, 14 για τις κινήσεις των βραχιόνων (7 για τον καθένα) και δύο επιπλέον οι οποίες η μία επαναφέρει τους βραχίονες στην αρχική τους θέση και η άλλη τερματίζει το πρόγραμμα.

Οι δημιουργημένες functions είναι οι εξής:

moveRight1()

Η συνάρτηση moveRight1() κινεί τον 1ο βραχίονα μια θέση στα δεξιά.

moveRight2()

Η συνάρτηση moveRight1() κινεί τον 2ο βραχίονα μια θέση στα δεξιά.

moveLeft1()

Η συνάρτηση moveLeft1() κινεί τον 1ο βραχίονα μια θέση στα αριστερά.

moveLeft2()

Η συνάρτηση moveLeft1() κινεί τον 2ο βραχίονα μια θέση στα αριστερά.

moveUp1()

Η συνάρτηση moveUp1() κινεί τον 1ο βραχίονα στην ακριβώς πάνω θέση.

moveUp2()

Η συνάρτηση moveUp2() κινεί τον 2ο βραχίονα στην ακριβώς πάνω θέση.

Παρατήρηση: Αν κληθεί η συνάρτηση moveUp1() ή η συνάρτηση moveUp2() όταν ο πρώτος ή ο δεύτερος βραχίονας βρίσκεται στο πάνω ημισφαίριο του κύκλου , τότε θα μεταφερθεί στην πιο υψηλή θέση η οποία είναι η αρχική θέση του πρώτου βραχίονα.

moveDown1()

Η συνάρτηση moveDown1() κινεί τον 1ο βραχίονα στην ακριβώς κάτω θέση.

moveDown2()

Η συνάρτηση moveDown2() κινεί τον 2ο βραχίονα στην ακριβώς κάτω θέση.

*σημείωση: Αν κληθεί η συνάρτηση moveDown1() ή η συνάρτηση moveDown2() όταν ο πρώτος ή ο δεύτερος βραχίονας βρίσκεται στο κάτω ημισφαίριο του κύκλου τότε θα μεταφερθεί στην πιο χαμηλή θέση η οποία είναι η αρχική θέση του δεύτερου βραχίονα.

rotate90right1()

Η συνάρτηση rotate90right1() μεταφέρει την θέση του 1ου βραχίονα 3 θέσεις (η 90 μοίρες) προς τα δεξιά.

rotate90right2()

Η συνάρτηση rotate90right2() μεταφέρει την θέση του 2ου βραχίονα 3 θέσεις (η 90 μοίρες) προς τα δεξιά.

rotate90left1()

Η συνάρτηση rotate90left1() μεταφέρει την θέση του 1ου βραχίονα 3 θέσεις (η 90 μοίρες) προς τα αριστερά.

rotate90left2()

Η συνάρτηση rotate90left2() μεταφέρει την θέση του 2ου βραχίονα 3 θέσεις (η 90 μοίρες) προς τα αριστερά.

rotate180first()

Η συνάρτηση rotate180first() μεταφέρει την θέση του 1ου βραχίονα 6 θέσεις (η 180 μοίρες).

rotate180second()

Η συνάρτηση rotate180second() μεταφέρει την θέση του 2ου βραχίονα 6 θέσεις (η 180 μοίρες).

reset()

Η συνάρτηση reset() επαναφέρει τους δύο βραχίονες στην αρχική τους θέση.

exit()

Η συνάρτηση exit() τερματίζει το πρόγραμμα.

Μέσα σε κάθε συνάρτηση έχει προστεθεί η συνάρτηση cleardevice() της βιβλιοθήκης graphics.h έτσι ώστε κάθε φορά που κάνει κίνηση ένας από τους δύο βραχίονες, να γίνεται ανανέωση των γραφικών διαφορετικά θα εμφανιζόντουσαν οι βραχίονες προηγούμενων καταστάσεων. Επίσης μέσα σε κάθε συνάρτηση καλούμε δύο φορές την function rotateRectangle() και έχουμε αντικαταστήσει τον αριθμό του πίνακα movex και movey με μετρητές. Αυτό το κάναμε για να μπορεί το πρόγραμμα να θυμάται τις προηγούμενες καταστάσεις των βραχιόνων.

```
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
```

Οι αρχικές καταστάσεις των βραχιόνων έχουν οριστεί ως global μεταβλητές.

3.2 Δημιουργία Εικονικού Δικτύου Με Υποδοχή (Socket)

Για την δημιουργία μίας υποδοχής χρειάζεται να δημιουργήσουμε δύο ξεχωριστά προγράμματα, ένα για τον πελάτη (client) ο οποίος θα στέλνει το μήνυμα που θέλει ο χρήστης και ένα για τον διακομιστή (server) ο οποίος θα λαμβάνει το μήνυμα από τον πελάτη (client). Το μήνυμα από τον πελάτη θα μεταφράζεται στον διακομιστή και θα καλεί την αντίστοιχη συνάρτηση. Για λόγους ευελιξίας, ο κώδικας που οπτικοποιεί τα γραφικά του βραχίονα έχει τοποθετηθεί ως header στον κώδικα του διακομιστή.

3.2.1 Υποδοχή-Διακομιστής

Για την δημιουργία μίας υποδοχής στο λειτουργικό σύστημα των windows, πρέπει να προσθέσουμε την βιβλιοθήκη winsock2.h. Αρχικά, πριν καλέσουμε οποιαδήποτε function πρέπει να αρχικοποιήσουμε την βιβλιοθήκη του winsock2. Αυτό θα επιτευχθεί με τον παρακάτω κώδικα:

```
WSADATA WinSockData;
WSAStartup(MAKEWORD(2,2),&WinSockData);
```

Η παράμετρος MAKEWORD(2,2) κάνει αίτημα για την νεότερη έκδοση του winsock στο σύστημα.

Έπειτα καλούμε την συνάρτηση socket() για την δημιουργία της υποδοχής μας.

```
network_socket = socket(AF_INET , SOCK_STREAM,0);
```

Η συνάρτηση socket() έχει 3 παραμέτρους:

- Η **Domain** η οποία περιγράφει με πιο πρωτόκολλο θα δημιουργηθεί η υποδοχή. Επιλέξαμε AF_INET για πρωτόκολλο στην IP v4
- Το **Type** το οποίο καθορίζει τον τύπο της υποδοχής. Επιλέξαμε το SOCK_STREAM (TCP) για περισσότερη αξιοπιστία στην ροή των δεδομένων (υπόδειγμα ρεύματων).
- Το **Πρωτόκολλο** το οποίο καθορίζει το πιο πρωτόκολλο θα χρησιμοποιηθεί. Το 0 σε αυτήν την παράμετρο υποδεικνύει στο σύστημα να χρησιμοποιεί το προκαθορισμένο πρωτόκολλο για αυτό το domain και type.

Έπειτα πρέπει να ορίσουμε την διεύθυνση του server.

```
struct sockaddr_in server_address;
server_address.sin_family = AF_INET;
server_address.sin_port = htons(9005);
server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
```

- Ορίζουμε το πρωτόκολλο AF_INET για IP v4
- Ορίζουμε την θύρα που θα χρησιμοποιηθεί μετατρέποντας τον αριθμό από host byte integer σε network byte integer χρησιμοποιώντας την συνάρτηση htons().
- Ορίζουμε την διεύθυνση IP 127.0.0.1. Η συγκεκριμένη IP διεύθυνση συνδέεται με τον υπολογιστή που τρέχει το πρόγραμμα.

Τώρα πρέπει να χρησιμοποιηθεί η συνάρτηση bind() για να αναθέσει την διεύθυνση και την θύρα που επιλέξαμε στο socket.

```
bind(server_socket, (struct sockaddr*) &server_address, sizeof(server_address));
```

Μετά , καλείται την συνάρτηση listen() για να προετοιμάσουμε την υποδοχή για εισερχόμενες συνδέσεις.

```
listen(server_socket, 5);
```

Στην πρώτη παράμετρο επιλέγουμε την υποδοχή που θέλουμε να κάνει listen και στην δεύτερη παράμετρο ορίζουμε τον αριθμό των επιτρεπόμενων εκκρεμείς συνδέσεων.
Θα καλέσουμε την συνάρτηση accept() για να επιτραπεί μία εισερχόμενη προσπάθεια σύνδεσης.

```
client_socket = accept(server_socket, NULL, NULL);
```

Τέλος , θα δημιουργήσουμε έναν πίνακα χαρακτήρων και μία μεταβλητή τύπου integer. Η πίνακας χαρακτήρων θα είναι ο buffer μας ο οποίος θα λαμβάνει και θα αποθηκεύει το μήνυμα από τον client μέσω της συνάρτησης recv().

```
length = recv(client_socket, server_message, sizeof(server_message),0);
```

Το server_message θα είναι το μήνυμα που θα λαμβάνεται από τον client.
Θα χρησιμοποιήσουμε την μεταβλητή τύπου integer (length) για να ελέγχουμε το μήκος του buffer μας έτσι ώστε κάθε φορά που θα στέλνει ο χρήστης κάποιο μήνυμα , να αδειάζει ο buffer μας.

```
if ( length > 0 )
{
    server_message[length] = '\0' ;
```

Ανάλογα με το μήνυμα που θα στέλνει ο χρήστης από τον πελάτη, θα καλείται και η αντίστοιχη συνάρτηση:

```
if (strcmp( server_message,"moveright1") == 0)
{
    moveRight1();
}
```

- Αν ο χρήστης πληκτρολογήσει “moveright1” τότε θα κληθεί η συνάρτηση moveRight1() η οποία θα κινήσει τον πρώτο βραχίονα μια θέση δεξιά.

```
if (strcmp(server_message,"moveright2") == 0)
{
        moveRight2();
}
```

- Αν ο χρήστης πληκτρολογήσει “moveright2” τότε θα κληθεί η συνάρτηση moveRight2() η οποία θα κινήσει τον δεύτερο βραχίονα μια θέση δεξιά.

```
if (strcmp(server_message,"moveleft1") == 0)
{
        moveLeft1();
}
```

- Αν ο χρήστης πληκτρολογήσει “moveleft1” τότε θα κληθεί η συνάρτηση moveLeft1() η οποία θα κινήσει τον πρώτο βραχίονα μια θέση αριστερά.

```
if (strcmp(server_message,"moveleft2") == 0)
{
        moveLeft2();
}
```

- Αν ο χρήστης πληκτρολογήσει “moveleft2” τότε θα κληθεί η συνάρτηση moveLeft2() η οποία θα κινήσει τον δεύτερο βραχίονα μια θέση αριστερά.

```
if (strcmp(server_message,"moveup1") == 0)
{
        moveUp1();
}
```

- Αν ο χρήστης πληκτρολογήσει “moveup1” τότε θα κληθεί η συνάρτηση moveUp1() η οποία θα κινήσει τον πρώτο βραχίονα στην ακριβώς πάνω θέση.

```
if (strcmp(server_message,"moveup2") == 0)
{
        moveUp2();
}
```

- Αν ο χρήστης πληκτρολογήσει “moveup2” τότε θα κληθεί η συνάρτηση moveUp2() η οποία θα κινήσει τον δεύτερο βραχίονα στην ακριβώς πάνω θέση.

```
if (strcmp(server_message,"movedown1") == 0)
{
        moveDown1();
}
```

- Αν ο χρήστης πληκτρολογήσει “movedown1” τότε θα κληθεί η συνάρτηση moveDown1() η οποία θα κινήσει τον πρώτο βραχίονα στην ακριβώς κάτω θέση.

```
if (strcmp(server_message,"movedown2") == 0)
{
        moveDown2();
}
```

- Αν ο χρήστης πληκτρολογήσει “movedown2” τότε θα κληθεί η συνάρτηση moveDown2() η οποία θα κινήσει τον δεύτερο βραχίονα στην ακριβώς κάτω θέση.

```
if (strcmp(server_message,"rotate90right1") == 0)
{
        rotate90right1();
}
```

- Αν ο χρήστης πληκτρολογήσει “rotate90right1” τότε θα κληθεί η συνάρτηση rotate90right1() η οποία θα κινήσει τον πρώτο βραχίονα 3 θέσεις (η 90 μοίρες) δεξιά.

```
if (strcmp(server_message,"rotate90right2") == 0)
{
        rotate90right2();
}
```

- Αν ο χρήστης πληκτρολογήσει “rotate90right2” τότε θα κληθεί η συνάρτηση rotate90right2() η οποία θα κινήσει τον δεύτερο βραχίονα 3 θέσεις (η 90 μοίρες) δεξιά.

```
if (strcmp(server_message,"rotate90left1") == 0)
{
        rotate90left1();
}
```

- Αν ο χρήστης πληκτρολογήσει “rotate90left1” τότε θα κληθεί η συνάρτηση rotate90left1() η οποία θα κινήσει τον πρώτο βραχίονα 3 θέσεις (η 90 μοίρες) αριστερά.

```
if (strcmp(server_message,"rotate90left2") == 0)
{
        rotate90left2();
}
```

- Αν ο χρήστης πληκτρολογήσει “rotate90left2” τότε θα κληθεί η συνάρτηση rotate90left2() η οποία θα κινήσει τον δεύτερο βραχίονα 3 θέσεις (η 90 μοίρες) αριστερά.

```
if (strcmp(server_message,"rotate180first") == 0)
{
        rotate180first();
}
```

- Αν ο χρήστης πληκτρολογήσει “rotate180first” τότε θα κληθεί η συνάρτηση rotate180first() η οποία θα κινήσει τον πρώτο βραχίονα 6 θέσεις (η 180 μοίρες).

```
if (strcmp(server_message,"rotate180second") == 0)
{
        rotate180second();
}
```

- Αν ο χρήστης πληκτρολογήσει “rotate180second” τότε θα κληθεί η συνάρτηση rotate180second() η οποία θα κινήσει τον δεύτερο βραχίονα 6 θέσεις (η 180 μοίρες).

```
if (strcmp(server_message,"reset") == 0)
{
        reset();
}
if (strcmp(server_message,"exit") == 0)
{
        exit();
        break;
}
```

- Αν ο χρήστης καλέσει “reset” τότε θα κληθεί η συνάρτηση reset() η οποία θα επαναφέρει τους βραχίονες στην αρχική τους θέση.
- Αν ο χρήστης καλέσει “exit” τότε θα κληθεί η συνάρτηση exit() η οποία θα τερματίσει το πρόγραμμα.

Έχουμε προσθέσει και δύο while loops ως ατέρμονες βρόγχους έτσι ώστε το πρόγραμμα να μην τερματίζεται ποτέ μέχρι ο χρήστης να καλέσει “exit”. Η μία while βρίσκεται πριν την συνάρτηση accept() για να υπάρχει συνεχώς σύνδεση με τον client και η δεύτερη while βρίσκεται πριν την συνάρτηση recv() για να λαμβάνει συνεχώς τα μηνύματα που του στέλνει ο χρήστης.

Τέλος , μετά τους ατέρμονες βρόγχους , καλούμε την συνάρτηση close()* η οποία τερματίζει την υποδοχή του διακομιστή και απελευθερώνει όλους τους πόρους που χρησιμοποίησε και την συνάρτηση WSACleanup() για τον τερματισμό του winsock2.

```
close(server_socket);
WSACleanup();
```

*Σημείωση: Για να χρησιμοποιήσουμε την συνάρτηση close() πρέπει να κάνουμε include μαζί με όλες τις άλλες βιβλιοθήκες, την βιβλιοθήκη unistd.h. Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε την συνάρτηση closesocket() της βιβλιοθήκης winsock2.h.

3.2.2 Υποδοχή-Πελάτης

Για την δημιουργία του client, πρέπει αρχικά να επαναλάβουμε κάποια βήματα όπως στον διακομιστή. Για να γίνει επιτυχής η σύνδεση με τον server, πρέπει να δημιουργήσουμε πάλι ένα socket και να ξανακαλέσουμε κάποιες συναρτήσεις με τις ίδιες παραμέτρους με πριν.

```
WSADATA WinSockData;
WSAStartup(MAKEWORD(2,2),&WinSockData);
//create a socket
int network_socket;

network_socket = socket(AF_INET , SOCK_STREAM,0);

//Specify an address for the socket
struct sockaddr_in server_address;
server_address.sin_family = AF_INET;
server_address.sin_port = htons(9005);
server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
```

Έπειτα, αντί να καλέσουμε την συνάρτηση bind() όπως πριν , θα καλέσουμε την συνάρτηση connect() η οποία δημιουργεί μια σύνδεση στην υποδοχή που του ορίσαμε.

```
int connection_status = connect(network_socket, (struct sockaddr *) &server_address, sizeof(server_address));
```

Θα δημιουργήσουμε πάλι έναν πίνακα χαρακτήρων ως buffer για να μπορεί να αποθηκεύσει το μήνυμα που θα στείλει ο χρήστης και θα καλέσουμε την συνάρτηση send() για να στείλουμε το μήνυμα στον διακομιστή.

```
char server_response[1024] = "";
int length = strlen(server_response);
char *buffer = server_response;
send(network_socket, buffer, length,0);
```

Θα χρησιμοποιήσουμε πάλι μία while loop ως ατέρμονα βρόγχο έτσι ώστε να μπορεί τα στέλνει ο χρήστης μηνύματα όσο υπάρχει σύνδεση μεταξύ πελάτη και διακομιστή.

```
if (strcmp(server_response,"exit") == 0)
        {
                break;
        }
```

Αν ο χρήστης καλέσει “exit” τότε το πρόγραμμα θα τερματίζει.

Τέλος, όπως και πριν θα χρησιμοποιήσουμε τις συναρτήσεις close() για το κλείσιμο της υποδοχής και εκκαθάριση των πόρων που χρησιμοποίησε και την συνάρτηση WSACleanup() για τον τερματισμό του winsock2.

```
close(server_socket);
WSACleanup();
```

ΚΕΦΑΛΑΙΟ 4: ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ

Οι γλώσσες προγραμματισμού είναι τεχνητές γλώσσες οι οποίες χρησιμοποιούνται για την επικοινωνία με μία μηχανή, συνήθως ενός υπολογιστή. Για την ευκολότερη κατανόηση τους, βασική προϋπόθεση είναι η επίλυση ασκήσεων για να μπορέσει ο φοιτητής να εξοικειωθεί με τους κανόνες του συντακτικού της γλώσσας και τους κανόνες που αφορούν το λεξιλόγιο αλλά και την πλήρης κατανόηση των μεθόδων που υπάρχουν σε μία γλώσσα προγραμματισμού. Γι αυτόν τον λόγο, έχουμε συντάξει τις παρακάτω ασκήσεις οι οποίες κάνουν χρήση του λογισμικού που δημιουργήσαμε για την εκμάθηση και κατανόηση κάποιων βασικών και πιο ανεπτυγμένων μεθόδων της γλώσσας προγραμματισμού C. Με την επιτυχής επίλυση των παρακάτω ασκήσεων ο φοιτητής θα έχει κατανοήσει μεθόδους όπως:

- Δημιουργία υποδοχής με πρωτόκολλο TCP για πελάτη και διακομιστή
- Κάλεσμα συναρτήσεων μέσω strings
- Κάλεσμα συναρτήσεων μέσω πινάκων
- Κάλεσμα συναρτήσεων μέσω γραμμή εντολών
- Κάλεσμα συναρτήσεων μέσω αρχείου (txt)
- Κάλεσμα συναρτήσεων μέσω δομής δεδομένων (λίστες)

Άσκηση 1

Στα εργαστήρια υπολογιστών παρέχονται διάφορες βιβλιοθήκες και προγράμματα. Ένα από αυτά τα προγράμματα μπορεί και δημιουργεί ένα γραφικό περιβάλλον δύο εικονικών βραχιόνων. Το πρόγραμμα αυτό αποτελεί τον διακομιστή μιας υποδοχής με πρωτόκολλο TCP. Καλείστε να δημιουργήσετε τον πελάτη της υποδοχής ώστε να επικοινωνήσουν τα δύο προγράμματα μεταξύ τους.

Για την επικοινωνία θα χρειαστούν οι εξής παράμετροι για τον πελάτη:

- Το IP του τρέχοντα υπολογιστή
- Η πόρτα επικοινωνίας να είναι 9005
- Ορίστε πρωτόκολλο TCP

Στόχος της άσκησης είναι να δημιουργήσετε το κανάλι επικοινωνίας μεταξύ πελάτη και διακομιστή.

Χρήσιμες βιβλιοθήκες: stdlib.h, stdio.h, winsock2.h , unistd.h.

Υπόδειξη: Χρησιμοποιήστε το struct sockaddr_in της βιβλιοθήκης winsock2.h για να ορίσετε τις κατάλληλες παραμέτρους , και έπειτα χρησιμοποιήστε τις συναρτήσεις της βιβλιοθήκης winsock2.h για τον πελάτη. Ξεκινήστε με την συνάρτηση connect() για να γίνει η σύνδεση με τον διακομιστή και στην συνέχει την συνάρτηση send() για να στείλετε το επιθυμητό μήνυμα(char array[]). Τέλος τερματίστε την σύνδεση με την συνάρτηση close().

Σκοπός της Άσκησης 1

Σκοπός της πρώτης άσκησης είναι η εξοικείωση των φοιτητών με τις υποδοχές κάνοντας μία μικρή εισαγωγή στον διαδικτυακό προγραμματισμό και το πρωτόκολλο TCP.

Άσκηση 2

Οι εικονικοί βραχίονες του γραφικού περιβάλλοντος του διακομιστή της Άσκησης 1 μπορεί να κινηθούν με τις κατάλληλες συναρτήσεις. Καλείστε μέσω του πελάτη να καλέσετε τις συγκεκριμένες συναρτήσεις και να κινήσετε τους εικονικούς βραχίονες. Ο τρόπος που θα κινήσετε τους δύο εικονικούς βραχίονες θα είναι μέσω strings. Στόχος είναι ο πάνω βραχίονας να κάνει μια κίνηση δεξιά και ο κάτω μία κίνηση αριστερά ώστε να κινηθούν και οι δύο.

Αντιστοιχία εντολών:

Βραχίονας	**Κίνηση**	**Συνάρτηση**
1	Δεξιά	moveright1
2	Αριστερά	moveleft2

Υπόδειξη: Χρησιμοποιήστε την υλοποίηση της προηγούμενης άσκησης για τον πελάτη και δημιουργήστε δύο ξεχωριστά strings τα οποία θα έχουν ως τιμή το όνομα των συναρτήσεων που φαίνονται στον παραπάνω πίνακα και στείλτε τα με την συνάρτηση send() σαν μήνυμα στον διακομιστή.

Σκοπός της Άσκησης 2

Σκοπός της δεύτερης άσκησης είναι να εξοικειωθούν οι φοιτητές με την μεταβλητή string και να δούνε στην πράξη την επικοινωνία πελάτη-διακομιστή μέσω πρωτοκόλλου TCP.

Άσκηση 3

Σε συνέχεια των προηγούμενων εργαστηρίων, καλείστε να κινήσετε τους εικονικούς βραχίονες καλώντας τις αντίστοιχες συναρτήσεις μέσω strings τα οποία θα είναι αποθηκευμένα σε έναν πίνακα. Ο πίνακας αυτός θα διατρέχεται στο τέλος ώστε να κληθούν οι κατάλληλες συναρτήσεις.

Αντιστοιχία εντολών:

Βραχίονας	Κίνηση	Συνάρτηση
1	Δεξιά	moveright1
2	Δεξιά	moveright2
1	Αριστερά	moveleft1
2	Αριστερά	moveleft2
1	Πάνω	moveup1
2	Πάνω	moveup2
1	Κάτω	movedown1
2	Κάτω	movedown2
1	90 μοίρες δεξιά	rotate90right1
2	90 μοίρες δεξιά	rotate90right2
1	90 μοίρες αριστερά	rotate90left1
2	90 μοίρες αριστερά	rotate90left2
1	180 μοίρες	rotate180first
2	180 μοίρες	rotate180second

Υπόδειξη: Δημιουργήστε έναν πίνακα στο πρόγραμμα του πελάτη και γεμίστε τον με strings τα οποία θα ως τιμή τις συναρτήσεις. Διαβάστε ολόκληρο τον πίνακα με την βοήθεια μίας for και στείλτε σε μήνυμα στον διακομιστή το περιεχόμενο του.

Σκοπός της Άσκησης 3

Σκοπός της τρίτης άσκησης είναι η εξοικείωση του φοιτητή με τους πίνακες που αποτελούν strings και το πως μπορούν να αποκτήσουν πρόσβαση και να διαχειριστούν την πληροφορία μέσα σε έναν πίνακα.

Άσκηση 4

Σε συνέχεια των προηγούμενων δύο εργαστηρίων, καλείστε να κινήσετε μέσω των συναρτήσεων του διακομιστή ξανά μέσω του πελάτη. Ο τρόπος με τον οποίο θα επιτευχθεί αυτό θα είναι καλώντας τις συναρτήσεις μέσω γραμμή εντολών. Η επικοινωνία τερματίζετε όταν ο χρήστης δώσει στην γραμμή εντολών την εντολή exit.

Αντιστοιχία εντολών:

Βραχίονας	Κίνηση	Συνάρτηση
1	Δεξιά	moveright1
2	Δεξιά	moveright2
1	Αριστερά	moveleft1
2	Αριστερά	moveleft2
1	Πάνω	moveup1
2	Πάνω	moveup2
1	Κάτω	movedown1
2	Κάτω	movedown2
1	90 μοίρες δεξιά	rotate90right1
2	90 μοίρες δεξιά	rotate90right2
1	90 μοίρες αριστερά	rotate90left1
2	90 μοίρες αριστερά	rotate90left2
1	180 μοίρες	rotate180first
2	180 μοίρες	rotate180second

Υπόδειξη: Δημιουργήστε μία μεταβλητή η οποία θα διαβάζεται από το πληκτρολόγιο με την χρήση της scanf() και χρησιμοποιήστε την για να καλέσετε τις συναρτήσεις μέσω μηνυμάτων σε πραγματικό χρόνο. Δημιουργήστε και έναν ατέρμονα βρόγχο όπως η while() για να μπορέσετε να καλείτε πάνω από μία συνάρτηση χωρίς να τερματίσει το πρόγραμμα.

Σκοπός της Άσκησης 4

Σκοπός της τέταρτης άσκησης είναι η περαιτέρω εξοικείωση της επικοινωνίας μεταξύ πελάτη-διακομιστή μέσω πρωτοκόλλου TCP.

Άσκηση 5

Στην προηγούμενη άσκηση, κληθήκατε να κινήσετε τους δύο εικονικούς βραχίονες μέσω γραμμή εντολών. Ωστόσο υπάρχουν και άλλοι τρόποι που μπορείτε να κινήσετε τους δύο βραχίονες. Πλέον καλείστε να κινήσετε τους δύο εικονικούς βραχίονες μέσω ενός txt αρχείου. Πιο συγκεκριμένα κάθε φοιτητής θα πρέπει να δημιουργήσει ένα txt αρχείο στο οποίο θα περιέχεται μια ακολουθία εντολών. Κάθε γραμμή του αρχείου θα περιέχει από μία εντολή (Άρα η τελευταία γραμμή θα πρέπει να περιέχει την εντολή “exit”). Σκοπός είναι ο πελάτης να διαβάζει line-by-line από το txt αρχείο τις εντολές του χρήστη ώστε να καλέσει τις αντίστοιχες συναρτήσεις του διακομιστή. Χρησιμοποιήστε τις γνώσεις που έχετε από το κεφάλαιο “Αρχείο”.

Αντιστοιχία εντολών:

Βραχίονας	Κίνηση	Συνάρτηση
1	Δεξιά	moveright1
2	Δεξιά	moveright2
1	Αριστερά	moveleft1
2	Αριστερά	moveleft2
1	Πάνω	moveup1
2	Πάνω	moveup2
1	Κάτω	movedown1
2	Κάτω	movedown2
1	90 μοίρες δεξιά	rotate90right1
2	90 μοίρες δεξιά	rotate90right2
1	90 μοίρες αριστερά	rotate90left1
2	90 μοίρες αριστερά	rotate90left2
1	180 μοίρες	rotate180first
2	180 μοίρες	rotate180second

Υπόδειξη:Χρησιμοποιήστε την συνάρτηση fopen() για να προσπελάσετε το αρχείο που δημιουργήσατε και έπειτα την συνάρτηση fgets() ώστε να διαβάσετε line-by-line το αρχείο. Τέλος , καλέστε την συνάρτηση fclose() για τον τερματισμό του αρχειού.

Σκοπός της Άσκησης 5

Σκοπός της πέμπτης άσκησης είναι η εξοικείωση του φοιτητή με τα αρχεία και τις διάφορες διεργασίες που μπορούμε να κάνουμε με αυτά.

Άσκηση 6

Στην προηγούμενη άσκηση, κληθήκατε να κινήσετε τους δύο εικονικούς βραχίονες μέσω ενός txt αρχείου. Σε αυτήν την άσκηση καλείστε να κινήσετε τους δύο εικονικούς βραχίονες μέσω δομής δεδομένων. Πιο συγκεκριμένα , κάθε φοιτητής θα πρέπει να ορίσει μία δομή δεδομένων στην οποία θα αποθηκεύονται οι εντολές που θα δίνει ο χρήστης μέσω γραμμής εντολών. Τέλος ο χρήστης θα διατρέχει την δομή δεδομένων και θα καλεί μέσω των εντολών τις αντίστοιχες συναρτήσεις.

Ακολουθία εντολών:

Βραχίονας	Κίνηση	Συνάρτηση
1	Δεξιά	moveright1
2	Δεξιά	moveright2
1	Αριστερά	moveleft1
2	Αριστερά	moveleft2
1	Πάνω	moveup1
2	Πάνω	moveup2
1	Κάτω	movedown1
2	Κάτω	movedown2
1	90 μοίρες δεξιά	rotate90right1
2	90 μοίρες δεξιά	rotate90right2
1	90 μοίρες αριστερά	rotate90left1
2	90 μοίρες αριστερά	rotate90left2
1	180 μοίρες	rotate180first
2	180 μοίρες	rotate180second

Υπόδειξη: Ορίστε ένα struct στο οποίο θα υπάρχουν δύο πεδία: Ένας πίνακας χαρακτήρων και ένας δείκτη για την συγκεκριμένη δομή ο οποίος θα λειτουργεί ως συνδετικός κρίκος των κόμβων της λίστας (next). Έπειτα ορίστε έναν δείκτη ο οποίος θα αποτελεί το header της λίστας. Δημιουργήστε μία συνάρτηση μέσω της οποίας θα εισάγονται οι νέοι κόμβοι μέσα στην λίστα (δεσμεύοντας και την απαραίτητη μνήμη π.χ. malloc()). Τέλος προσπελάστε την λίστα και εκτελέστε τις κινήσεις.

Σκοπός της Άσκησης 6

Σκοπός της έκτης άσκησης είναι η εξοικείωση του φοιτητή με την δομή δεδομένων και πως μπορεί να διαχειριστή μία πληροφορία μέσω αυτών.

Άσκηση 7

Σε συνέχεια του προηγούμενου εργαστηρίου, καλείστε να αποθηκεύσετε τις εντολές που περιέχονται στο txt αρχείο σε μία δομή δεδομένων (λίστα). Θα πρέπει να ορίσετε κατάλληλα την δομή δεδομένων, και αφού αποθηκευτούν σε αυτήν οι εντολές του txt αρχείου, θα πρέπει να διατρέξετε την δομή δεδομένων ώστε να κληθούν οι αντίστοιχες συναρτήσεις και να κινήσουν τους εικονικούς βραχίονες.

Αντιστοιχία εντολών:

Βραχίονας	Κίνηση	Συνάρτηση
1	Δεξιά	moveright1
2	Δεξιά	moveright2
1	Αριστερά	moveleft1
2	Αριστερά	moveleft2
1	Πάνω	moveup1
2	Πάνω	moveup2
1	Κάτω	movedown1
2	Κάτω	movedown2
1	90 μοίρες δεξιά	rotate90right1
2	90 μοίρες δεξιά	rotate90right2
1	90 μοίρες αριστερά	rotate90left1
2	90 μοίρες αριστερά	rotate90left2
1	180 μοίρες	rotate180first
2	180 μοίρες	rotate180second

Υπόδειξη: Συνδυάστε τις τεχνικές και τις μεθόδους που χρησιμοποιήσατε στις ασκήσεις 5 και 6.

Σκοπός της Άσκησης 7

Σκοπός της έβδομης άσκησης είναι ο φοιτητής να συνδυάσει τις γνώσεις που απέκτησε από τις δύο προηγούμενες ασκήσεις.

Άσκηση 8

Σε συνέχεια των προηγούμενων ασκήσεων, καλείστε πλέον να συνδυάσετε τις γνώσεις που αποκτήσατε μέσα από αυτές. Σκοπός είναι ο χρήστης μέσω γραμμής εντολών να πληκτρολογεί την εντολή που θέλει να εκτελεστεί ώστε να κινηθεί ο αντίστοιχος βραχίονας. Στην συνέχεια η εντολή που πληκτρολόγησε ο χρήστης θα πρέπει να αποθηκεύετε σε ένα txt αρχείο και ο πελάτης διατρέχοντας αυτό το αρχείο , να αποθηκεύει τις εντολές σε μία δομή δεδομένων που θα έχει οριστεί κατάλληλα , και τέλος διατρέχοντας την δομή δεδομένων να καλεί την αντίστοιχη συνάρτηση του διακομιστή.

Αντιστοιχία εντολών:

Βραχίονας	Κίνηση	Συνάρτηση
1	Δεξιά	moveright1
2	Δεξιά	moveright2
1	Αριστερά	moveleft1
2	Αριστερά	moveleft2
1	Πάνω	moveup1
2	Πάνω	moveup2
1	Κάτω	movedown1
2	Κάτω	movedown2
1	90 μοίρες δεξιά	rotate90right1
2	90 μοίρες δεξιά	rotate90right2
1	90 μοίρες αριστερά	rotate90left1
2	90 μοίρες αριστερά	rotate90left2
1	180 μοίρες	rotate180first
2	180 μοίρες	rotate180second

Υπόδειξη: Συνδυάστε τις τεχνικές και τις μεθόδους που χρησιμοποιήσατε στις ασκήσεις 4,5 και 6.

Σκοπός της Άσκησης 8

Σκοπός της όγδοης άσκησης είναι οι φοιτητές να δούνε συνδυαστικά τις μεθόδους που διδάχθηκαν από τις προηγούμενες ασκήσεις.

ΚΕΦΑΛΑΙΟ 5: ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ

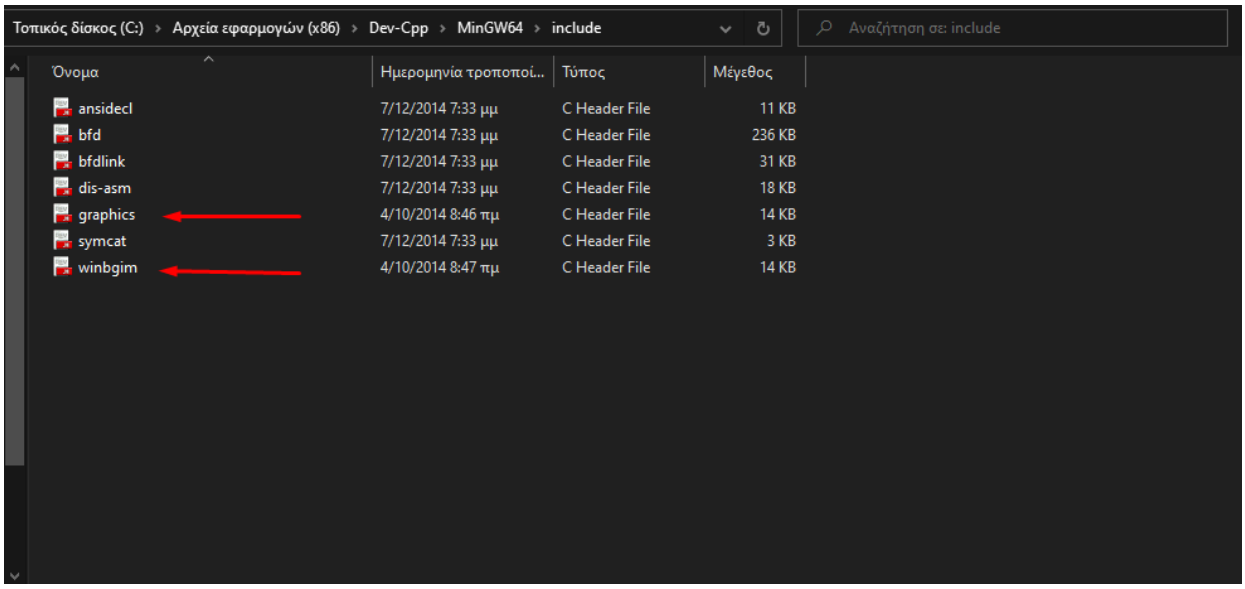
Για την βιβλιοθήκη graphics.h

Το πρόγραμμα dev-C++ που χρησιμοποιούμε στα εργαστήρια του ΤΕΙ, δεν διαθέτει από μόνο του την βιβλιοθήκη graphics.h οπότε πρέπει να την εγκαταστήσουμε εμείς για να μπορέσουμε να χρησιμοποιήσουμε τις functions που διαθέτει. Τα βήματα για την εγκατάσταση της βιβλιοθήκης είναι τα εξής:

Βήμα 1:Προσθήκη των header source files στο directory του Dev-C++

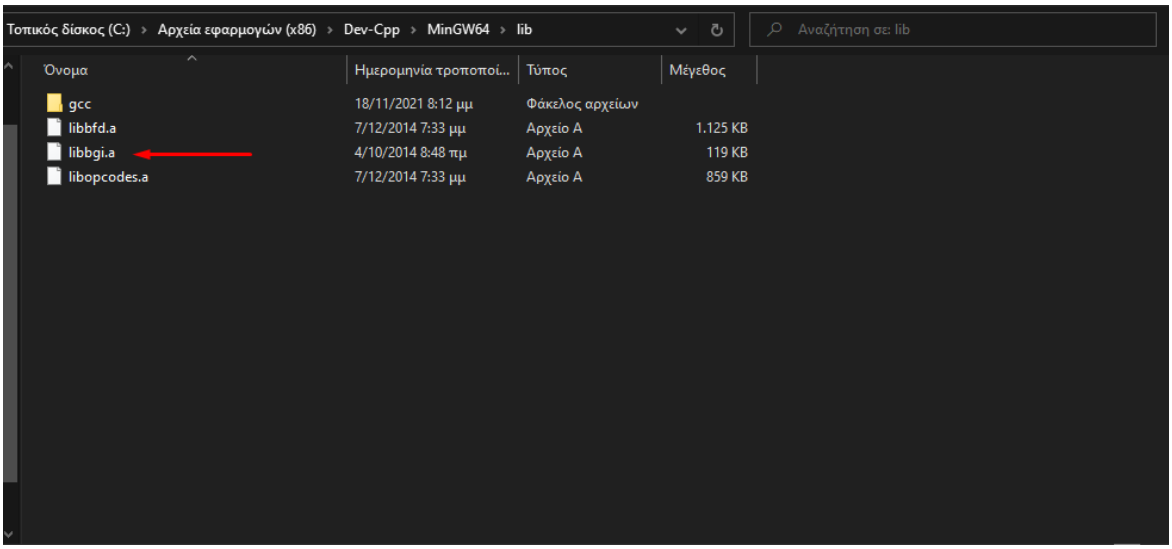
Αρχικά πρέπει να κατεβάσουμε τα αρχεία της βιβλιοθήκης graphics.h και να τα τοποθετήσουμε στους κατάλληλους φακέλους στο directory MinGW64.

1. Κατεβάζουμε τον φάκελο με τα αρχεία από τον σύνδεσμο http://winbgim.codecutter.org και τον κάνουμε extract.
2. Ο φάκελος περιέχει 3 αρχεία, το graphics.h, το winbgim.h και το libbgi.a.
3. Κάνουμε αντιγραφή το graphics.h και το winbgim.h και θα τα τοποθετήσουμε στον φάκελο include του dev-c++ που βρίσκεται στο directory address: C:\Program Files (x86)\Dev-Cpp\MinGW64\include όπως φαίνεται στην εικόνα (Εικόνα 4).



Εικόνα 5

Κάνουμε αντιγραφή το libbgi.a και θα το τοποθετήσουμε στον φάκελο lib του dev-c++ που βρίσκεται στο directory address: C:\Program Files (x86)\Dev-Cpp\MinGW64\lib όπως φαίνεται στην παρακάτω εικόνα(Εικόνα 5).

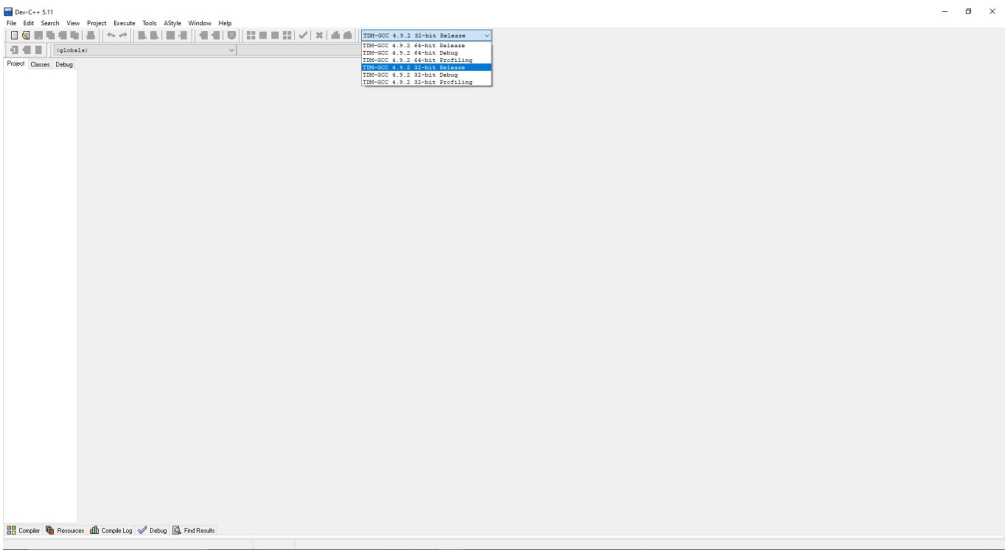


Εικόνα 6

Βήμα 2: Αλλαγή του compiler

Όπως προαναφέραμε , η βιβλιοθήκη graphics.h είναι αρκετά παλιά που σημαίνει ότι μπορεί να τρέξει μόνο σε συστήματα 32bit. Για αυτόν τον λόγο , πρέπει να αλλάξουμε τον τωρινό compiler που είναι 64bit σε compiler 32bit. Συγκεκριμένα πρέπει να επιλέξουμε τον compiler TDM – GCC 4.9.2 32-bit release .

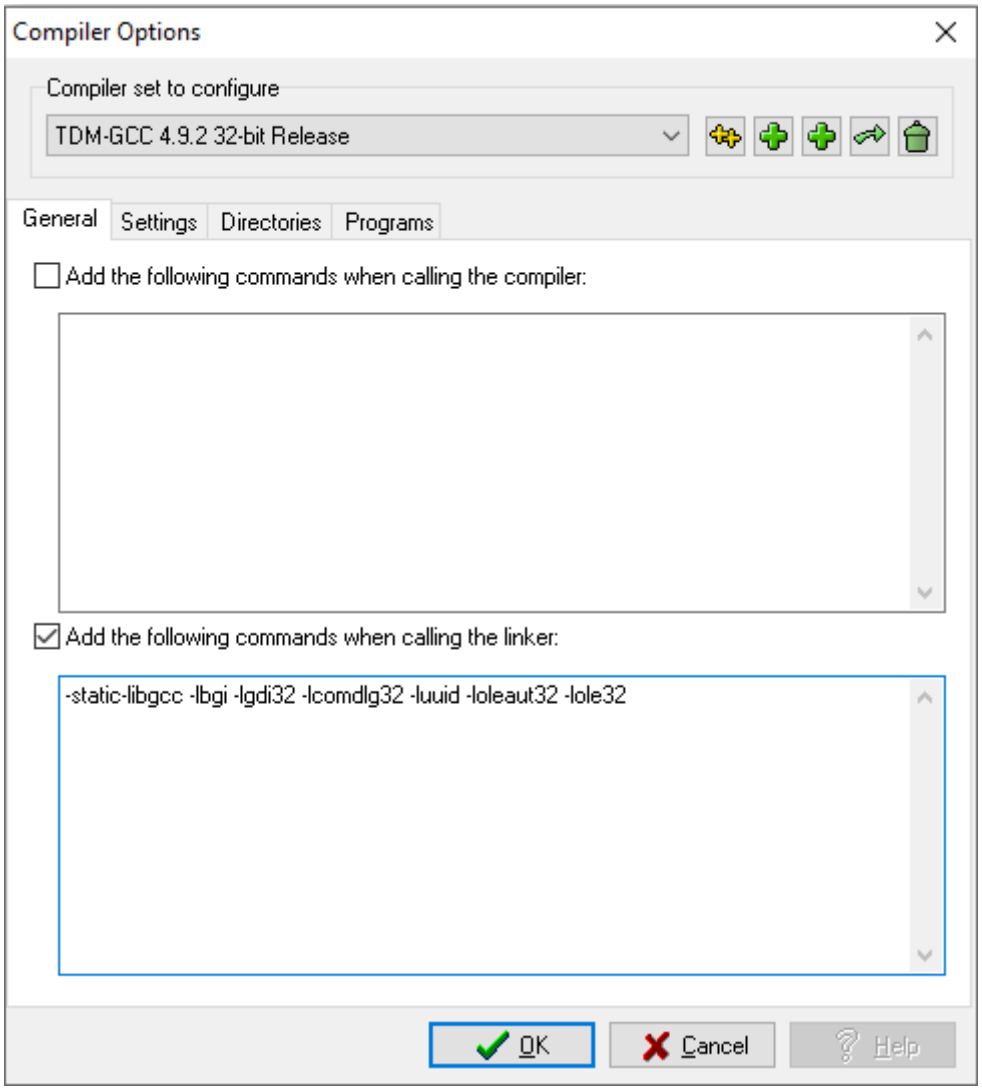
1. Αρχικά ανοίγουμε το πρόγραμμα dev-c++
2. Στην δεξιά μεριά του προγράμματος μετά τα εργαλεία υπάρχει ένα drop-down μενού που μας επιτρέπει να αλλάξουμε τον compiler.
3. Επιλέγουμε το TDM – GCC 4.9.2 32-bit release όπως φαίνεται στην παρακάτω εικόνα(Εικόνα 6).



Εικόνα 7

Βήμα 3:Διαμόρφωση των απαιτούμενων linkers για τα γραφικά

1. Στο dev-c++ πρέπει να πάμε στα “Εργαλεία” και να κάνουμε κλικ στις “Επιλογές μεταγλώτισης”
2. Στα “Γενικά” επιλέγουμε το κουτάκι που λέει “Παράμετροι γραμμής εντολών του συνδέτη (linker)” σε περίπτωση που δεν είναι επιλέγουμε.
3. Γράφουμε τις εξής εντολές όπως φαίνεται στην παρακάτω εικόνα(Εικόνα 7): -libgcc -lbgi -lgdi32 -lcomdlg32 -luuid -loleaut32 -lole32
4. Πατάμε OK για αποθήκευση.



Εικόνα 8

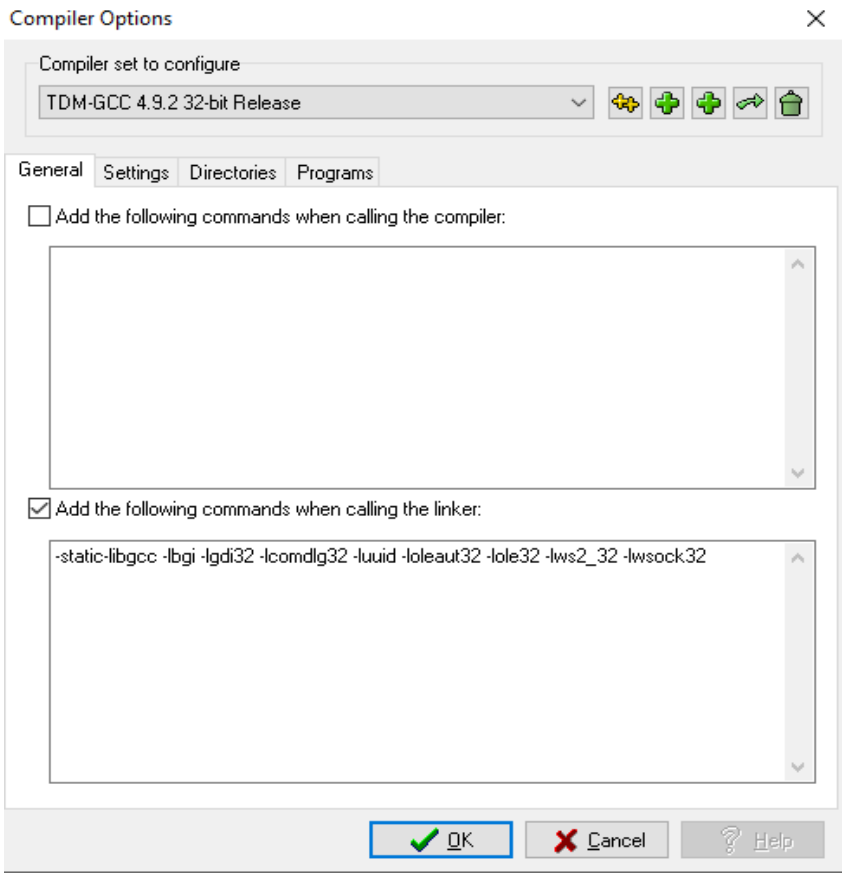
Τώρα είμαστε έτοιμοι να χρησιμοποιήσουμε την βιβλιοθήκη graphics.h.

Για την βιβλιοθήκη winsock2.h

Η βιβλιοθήκη winsock2.h περιέχει functions οι οποίες είναι απαραίτητες για την δημιουργία του εικονικού μας δικτύου μέσω sockets. Για να την χρησιμοποιήσουμε στο πρόγραμμα dev-c++ πρέπει να κάνουμε το έξης:

Διαμόρφωση των απαιτούμενων linkers για την winsock2.h

1. Στο dev-c++ πρέπει να πάμε στα “Εργαλεία” και να κάνουμε κλικ στις “Επιλογές μεταγλώτισης”
2. Στα “Γενικά” επιλέγουμε το κουτάκι που λέει “Παράμετροι γραμμής εντολών του συνδέτη (linker) σε περίπτωση που δεν είναι επιλεγμένο.
3. Γράφουμε τις εξής εντολές όπως φαίνεται στην παρακάτω εικόνα:
   -lws2_32 -lwsock32
4. Πατάμε OK για αποθήκευση.



Εικόνα 9

Τώρα μπορούμε να χρησιμοποιήσουμε την βιβλιοθήκη winsock2.h. Κάλο θα ήταν να κάνουμε μία επανεκκίνηση το dev-c++ πριν αρχίσουμε να χρησιμοποιούμε τις συναρτήσεις του winsock.

ΚΕΦΑΛΑΙΟ 6: ΣΥΜΠΕΡΑΣΜΑΤΑ

Σκοπός της πτυχιακής εργασίας ήταν να δημιουργήσουμε ένα λογισμικό για την ευκολότερη εκμάθηση της γλώσσας προγραμματισμού C. Για αυτόν τον λόγο δημιουργήσαμε ένα ευέλικτο λογισμικό το οποίο μπορεί να χρησιμοποιηθεί ως βάση για ασκήσεις που μπορούν να συνδυάσουν διάφορες βασικές μεθόδους της C αλλά και πιο ανεπτυγμένες όπως είναι τα γραφικά και ο προγραμματισμός δικτύων. Προσωπικά η εργασία μου φάνηκε αρκετά ενδιαφέρουσα διότι ασχολήθηκα και κατάλαβα περισσότερα πράγματα για το πως λειτουργούν τα γραφικά και τις διάφορες μεθόδους που υπάρχουν για την απεικόνιση, τον σχεδιασμό και την διαχείριση των γραφικών. Επίσης μου έκανε μια εισαγωγή για το πως δουλεύει ο διαδικτυακός προγραμματισμός και τα δίκτυα γενικότερα, κάτι το οποίο με ενδιαφέρει μιας και θα ήθελα μελλοντικά να ασχοληθώ κυρίως με δίκτυα και τον διαδικτυακό προγραμματισμό.

6.1 Αδυναμίες Συστήματος

Επέλεξα να δημιουργήσω το λογισμικό στο πρόγραμμα dev-c++ διότι ξέρω ότι αυτό το πρόγραμμα χρησιμοποιείται ακόμα στα εισαγωγικά μαθήματα. Δυστυχώς, το πρόγραμμα dev-c++ είναι αρκετά παλιό οπότε ο μόνος τρόπος απεικόνισης γραφικών ήταν μέσω γραφικού περιβάλλοντος MS-DOS. Το γραφικό περιβάλλον του MS-DOS ήταν πάρα πολύ χρήσιμο πριν πολλές δεκαετίες αλλά τώρα δεν χρησιμοποιείται καθόλου και αμφιβάλλω αν υπάρξει λόγος να το επισκεφθώ στο μέλλον. Για εκπαιδευτικούς σκοπούς όμως πιστεύω ότι είναι κατάλληλο διότι είναι πάρα πολύ απλό στην χρήση του επομένως είναι κατάλληλο για μία μικρή εισαγωγή στον κόσμο των γραφικών. Η παλαιότητα του επίσης εγγυάται ότι οποιοδήποτε σύγχρονο σύστημα θα μπορεί να το τρέξει χωρίς κανένα πρόβλημα σε αντίθεση με τις πιο σύγχρονες μηχανές γραφικών οι οποίες χρειάζονται παραπάνω πόρους για να τρέξουν σωστά και χωρίς καθυστερήσεις.

6.2 Μελλοντικές Βελτιώσεις

Η εργασία σίγουρα χρήζει βελτιώσεις κάποιες από τις οποίες είναι οι εξής:

- Θα μπορούσαν τα γραφικά να δημιουργηθούν πιο δυναμικά

- Θα μπορούσε ο χρήστης να δημιουργήσει παραπάνω από δύο βραχίονες δυναμικά
- Σε περίπτωση αλλαγής του προγράμματος dev-C++, θα μπορούσε να χρησιμοποιηθεί μία διαφορετική βιβλιοθήκη για την δημιουργία πιο σύγχρονων γραφικών

ΠΑΡΑΡΤΗΜΑ

Στο παρακάτω παράρτημα μπορείτε να δείτε το source code των δημιουργημένων συναρτήσεων για τις κινήσεις του βραχίονα:

moveRight1():

```
void moveRight1(){
                cleardevice();
                if (i==11 && j==11 && x==330){
        i=0;
        j=0;
                x=0;}
        else{
                x=x+30;
                i=i+1;
                j=j+1;}
           rectangleRotate(movex[i],movey[j],30,300,x);
           rectangleRotate(movex[i2],movey[j2],40,300,x2);
         }
```

moveRight2():

```
void moveRight2(){
                cleardevice();
                if (x2==0 && i2==0 && j2==0){
                        x2=330;
                        i2=11;
                        j2=11;
                }
                else{
                x2=x2-30;
                i2=i2-1;
                j2=j2-1;}
              rectangleRotate(movex[i],movey[j],30,300,x);
              rectangleRotate(movex[i2],movey[j2],40,300,x2);
}
```

moveLeft1():

```
void moveLeft1(){
                cleardevice();
                if (x==0 && i==0 && j==0){
                        x=330;
                        i=11;
                        j=11;
                }
                else{
                x=x-30;
                i=i-1;
                j=j-1;}
                rectangleRotate(movex[i],movey[j],30,300,x);
                rectangleRotate(movex[i2],movey[j2],30,300,x2);
}
```

moveLeft2():

```
void moveLeft2(){
                cleardevice();
                if (i2==11 && j2==11 && x2==330){
        i2=0;
        j2=0;
                x2=0;}
      else{
                x2=x2+30;
                i2=i2+1;
                j2=j2+1;}
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
}
```

exit():

```
void exit(){
                closegraph();
                exit(0);
}
```

moveUp1():

```
void moveUp1(){
        cleardevice();
        switch(i){
        case 0:{
        i=0;
        j=0;
        x=0;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
        case 1:{
        i=0;
        j=0;
        x=0;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
        case 2:{
        i=0;
        j=0;
        x=0;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 3:{
                i=0;
                j=0;
                x=0;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 4:{
                i=10;
                j=10;
                x=300;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 5:{
                i=11;
                j=11;
                x=330;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 6:{
                i=0;
```

```
                j=0;
                x=0;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 7:{
                i=1;
                j=1;
                x=30;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 8:{
                i=2;
                j=2;
                x=60;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 9:{
                i=0;
                j=0;
                x=0;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 10:{
                i=0;
                j=0;
                x=0;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 11:{
                i=0;
                j=0;
                x=0;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
}
}
```

moveUp2:()

```
void moveUp2(){
		cleardevice();
	switch(i2){
		case 0:{
		i2=0;
		j2=0;
		x2=0;
		rectangleRotate(movex[i],movey[j],30,300,x);
		rectangleRotate(movex[i2],movey[j2],30,300,x2);
		break;}
		case 1:{
		i2=0;
		j2=0;
		x2=0;
		rectangleRotate(movex[i],movey[j],30,300,x);
		rectangleRotate(movex[i2],movey[j2],30,300,x2);
		break;}
		case 2:{
		i2=0;
		j2=0;
		x2=0;
		rectangleRotate(movex[i],movey[j],30,300,x);
		rectangleRotate(movex[i2],movey[j2],30,300,x2);
			break;}
		case 3:{
			i2=0;
			j2=0;
			x2=0;
		rectangleRotate(movex[i],movey[j],30,300,x);
		rectangleRotate(movex[i2],movey[j2],30,300,x2);
			break;}
		case 4:{
			i2=10;
			j2=10;
			x2=300;
		rectangleRotate(movex[i],movey[j],30,300,x);
		rectangleRotate(movex[i2],movey[j2],30,300,x2);
			break;}
		case 5:{
			i2=11;
			j2=11;
			x2=330;
		rectangleRotate(movex[i],movey[j],30,300,x);
		rectangleRotate(movex[i2],movey[j2],30,300,x2);
			break;}
		case 6:{
			i2=0;
```

```
            j2=0;
            x2=0;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
            break;}
        case 7:{
            i2=1;
            j2=1;
            x2=30;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
            break;}
        case 8:{
            i2=2;
            j2=2;
            x2=60;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
            break;}
        case 9:{
            i2=0;
            j2=0;
            x2=0;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
            break;}
        case 10:{
            i2=0;
            j2=0;
            x2=0;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
            break;}
        case 11:{
            i2=0;
            j2=0;
            x2=0;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
            break;}

}
}
```

moveDown1():

```
void moveDown1(){
```

```
        cleardevice();
switch(i){
        case 0:{
        i=6;
        j=6;
        x=180;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
        case 1:{
        i=7;
        j=7;
        x=210;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
        case 2:{
        i=8;
        j=8;
        x=240;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 3:{
                i=6;
                j=6;
                x=180;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 4:{
                i=6;
                j=6;
                x=180;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 5:{
                i=6;
                j=6;
                x=180;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 6:{
                i=6;
                j=6;
                x=180;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 7:{
```

```
                i=6;
                j=6;
                x=180;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 8:{
                i=6;
                j=6;
                x=180;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 9:{
                i=6;
                j=6;
                x=180;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 10:{
                i=4;
                j=4;
                x=120;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 11:{
                i=5;
                j=5;
                x=150;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
}
}
```

moveDown2():

```
moveDown2(){
```

```
        cleardevice();
switch(i2){
        case 0:{
        i2=6;
        j2=6;
        x2=180;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
        case 1:{
        i2=7;
        j2=7;
        x2=210;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
        case 2:{
        i2=8;
        j2=8;
        x2=240;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 3:{
                i2=6;
                j2=6;
                x2=180;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 4:{
                i2=6;
                j2=6;
                x2=180;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 5:{
                i2=6;
                j2=6;
                x2=180;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 6:{
                i2=6;
                j2=6;
                x2=180;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 7:{
```

```
                i2=6;
                j2=6;
                x2=180;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 8:{
                i2=6;
                j2=6;
                x2=180;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 9:{
                i2=6;
                j2=6;
                x2=180;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 10:{
                i2=4;
                j2=4;
                x2=120;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 11:{
                i2=5;
                j2=5;
                x2=150;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
}
}
```

rotate90right1():

```
void rotate90right1(){
```

```
cleardevice();
switch(i){
case 0:{
i=3;
j=3;
x=90;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
break;}
case 1:{
i=4;
j=4;
x=120;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
break;}
case 2:{
i=5;
j=5;
x=150;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 3:{
        i=6;
        j=6;
        x=180;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 4:{
        i=7;
        j=7;
        x=210;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 5:{
        i=8;
        j=8;
        x=240;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 6:{
        i=9;
        j=9;
        x=270;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 7:{
```

```
                i=10;
                j=10;
                x=300;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 8:{
                i=11;
                j=11;
                x=330;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 9:{
                i=0;
                j=0;
                x=0;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 10:{
                i=1;
                j=1;
                x=30;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 11:{
                i=2;
                j=2;
                x=60;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
}
}
```

rotate90right2:

```
void rotate90right2(){
```

```
cleardevice();
switch(i2){
case 0:{
i2=9;
j2=9;
x2=270;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
break;}
case 1:{
i2=10;
j2=10;
x2=300;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
break;}
case 2:{
i2=11;
j2=11;
x2=330;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 3:{
        i2=0;
        j2=0;
        x2=0;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 4:{
        i2=1;
        j2=1;
        x2=30;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 5:{
        i2=2;
        j2=2;
        x2=60;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 6:{
        i2=3;
        j2=3;
        x2=90;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 7:{
```

```
                i2=4;
                j2=4;
                x2=120;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 8:{
                i2=5;
                j2=5;
                x2=150;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 9:{
                i2=6;
                j2=6;
                x2=180;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 10:{
                i2=7;
                j2=7;
                x2=210;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 11:{
                i2=8;
                j2=8;
                x2=240;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
}
}
```

rotate90left1():

```
void rotate90left1(){
```

```
cleardevice();
switch(i){
case 0:{
i=9;
j=9;
x=270;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
break;}
case 1:{
i=10;
j=10;
x=300;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
break;}
case 2:{
i=11;
j=11;
x=330;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 3:{
        i=0;
        j=0;
        x=0;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 4:{
        i=1;
        j=1;
        x=30;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 5:{
        i=2;
        j=2;
        x=60;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 6:{
        i=3;
        j=3;
        x=90;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 7:{
```

```
                i=4;
                j=4;
                x=120;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 8:{
                i=5;
                j=5;
                x=150;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 9:{
                i=6;
                j=6;
                x=180;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 10:{
                i=7;
                j=7;
                x=210;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 11:{
                i=8;
                j=8;
                x=240;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
}
}
```

rotate90left2():

```
void rotate90left2(){
```

```
cleardevice();
switch(i2){
case 0:{
i2=3;
j2=3;
x2=90;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
break;}
case 1:{
i2=4;
j2=4;
x2=120;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
break;}
case 2:{
i2=5;
j2=5;
x2=150;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 3:{
        i2=6;
        j2=6;
        x2=180;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 4:{
        i2=7;
        j2=7;
        x2=210;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 5:{
        i2=8;
        j2=8;
        x2=240;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 6:{
        i2=9;
        j2=9;
        x2=270;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 7:{
```

```
                i2=10;
                j2=10;
                x2=300;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 8:{
                i2=11;
                j2=11;
                x2=330;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 9:{
                i2=0;
                j2=0;
                x2=0;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 10:{
                i2=1;
                j2=1;
                x2=30;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
            case 11:{
                i2=2;
                j2=2;
                x2=60;
            rectangleRotate(movex[i],movey[j],30,300,x);
            rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
}
}
```

rotate180first():

```
void rotate180first(){
```

```
cleardevice();
switch(i){
case 0:{
i=6;
j=6;
x=180;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
break;}
case 1:{
i=7;
j=7;
x=210;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
break;}
case 2:{
i=8;
j=8;
x=240;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 3:{
        i=9;
        j=9;
        x=270;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 4:{
        i=10;
        j=10;
        x=300;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 5:{
        i=11;
        j=11;
        x=330;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 6:{
        i=0;
        j=0;
        x=0;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 7:{
```

```
                i=1;
                j=1;
                x=30;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 8:{
                i=2;
                j=2;
                x=60;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 9:{
                i=3;
                j=3;
                x=90;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 10:{
                i=4;
                j=4;
                x=120;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
        case 11:{
                i=5;
                j=5;
                x=150;
        rectangleRotate(movex[i],movey[j],30,300,x);
        rectangleRotate(movex[i2],movey[j2],30,300,x2);
                break;}
}
}
```

rotate180second():

```
rotate180second(){
```

```
cleardevice();
switch(i2){
case 0:{
i2=6;
j2=6;
x2=180;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
break;}
case 1:{
i2=7;
j2=7;
x2=210;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
break;}
case 2:{
i2=8;
j2=8;
x2=240;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 3:{
        i2=9;
        j2=9;
        x2=270;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 4:{
        i2=10;
        j2=10;
        x2=300;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 5:{
        i2=11;
        j2=11;
        x2=330;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 6:{
        i2=0;
        j2=0;
        x2=0;
rectangleRotate(movex[i],movey[j],30,300,x);
rectangleRotate(movex[i2],movey[j2],30,300,x2);
        break;}
case 7:{
```

```
                    i2=1;
                    j2=1;
                    x2=30;
                rectangleRotate(movex[i],movey[j],30,300,x);
                rectangleRotate(movex[i2],movey[j2],30,300,x2);
                    break;}
                case 8:{
                    i2=2;
                    j2=2;
                    x2=60;
                rectangleRotate(movex[i],movey[j],30,300,x);
                rectangleRotate(movex[i2],movey[j2],30,300,x2);
                    break;}
                case 9:{
                    i2=3;
                    j2=3;
                    x2=90;
                rectangleRotate(movex[i],movey[j],30,300,x);
                rectangleRotate(movex[i2],movey[j2],30,300,x2);
                    break;}
                case 10:{
                    i2=4;
                    j2=4;
                    x2=120;
                rectangleRotate(movex[i],movey[j],30,300,x);
                rectangleRotate(movex[i2],movey[j2],30,300,x2);
                    break;}
                case 11:{
                    i2=5;
                    j2=5;
                    x2=150;
                rectangleRotate(movex[i],movey[j],30,300,x);
                rectangleRotate(movex[i2],movey[j2],30,300,x2);
                    break;}
}
}

reset():
void reset(){
                cleardevice();
                i=0;
                i2=6;
                j=0;
                j2=6;
                x=0;
                x2=180;
                rectangleRotate(movex[i],movey[j],30,300,x);
                rectangleRotate(movex[i2],movey[j2],30,300,x2);
        }
```

ΑΝΑΦΟΡΕΣ:

Βιβλίο: ΔΙΚΤΥΑ ΚΑΙ ΔΙΑΔΙΚΤΥΑ ΥΠΟΛΟΓΙΣΤΩΝ (DOUGLAS E. COMER)

ΠΗΓΕΣ:

Zhigang Xiang, Roy A. Plastock, *Schaum's outline of computer graphics*, McGraw-Hill Professional, 2000 **https://el.wikipedia.org/wiki/Γραφικά_υπολογιστών** **(**τελευταία πρόσβαση 18 Αυγούστου 2022)

Richard Paul, 1981, Robot manipulators: mathematics, programming, and control : the computer control of robot manipulators, MIT Press, Cambridge, MA , W3C recommendation (2003), Scalable Vector Graphics -- the initial coordinate system , Durand; Cutler. "Transformations" (PowerPoint). Massachusetts Institute of Technology , Pile Jr, John (May 2013). 2D Graphics Programming for Games. New York, NY: CRC Press https://en.wikipedia.org/wiki/2D_computer_graphics (Τελευταία πρόσβαση 18 Αυγούστου 2022)

"3D computer graphics". ScienceDaily , Simmons, Bruce. "n-gon" MathWords **https://en.wikipedia.org/wiki/3D_computer_graphics** (Τελευταία πρόσβαση 18 Αυγούστου 2022)

Akshat Sinha https://www.geeksforgeeks.org/socket-programming-cc/ (Τελευταία πρόσβαση 18 Αυγούστου 2022)