תרגיל בית 4

<u>מועד הגשת התרגיל:</u> עד יום שלישי 15/01/19 בחצות.

הבהרות (28/12): א. בעמוד 10, נוספה הבהרה שהתכונה מחזירה 0 במקרה של סיום מוצלח, אם היא במוד לקוח (השרת לא אמור להסתיים).

ב. נאמר ששם המשתמש במוד לקוח מתקבל דרך קובץ הקלט או שהמשתמש מזין אותו. לכן, ארגומנט ה-cmd ששמו username הינו מיותר, והוא *<u>הוסר</u>*.

מטרת התרגיל

- העמקת ההבנה במושגי החוט (Thread) במערכות הפעלה בכלל וב-Windows בפרט.
- עבודה עם מספר חוטים במקביל.
- העמקת ההבנה בתקשורת מחשבים.
- שימוש ב-TCP Sockets:
  - WSAStartup
  - WSACleanup
  - socket
  - bind
  - listen
  - accept
  - recv
  - connect
  - closesocket
  - send

הנחיות הגשה

צורת ההגשה מפורטת במסמך "הנחיות להגשת תרגילי בית – תשע"ט" שבאתר המודל. אנא הקפידו למלא אחר ההוראות.

בנוסף, אנא הקפידו על ההנחיות הבאות:

- הגישו פרויקט מלא, כולל קבצי פרויקט (sln, *.vcxproj, *.vcxproj.filters.*) של Visual Studio 2017, באופן שיאפשר לבודק התרגילים לפתוח את הפרויקט על ידי לחיצה כפולה על קובץ ה-solution ולקמפל את הפרויקט ללא התראות או שגיאות.
- שם הפרויקט צריך להיות ex4. שם ה-executable צריך להיות ex4.exe.
- הגישו בנוסף את תיקיית ה-Debug עם קובץ ה-exe. הקוד לא צריך לתמוך ב-Unicode. וודאו בהגדרות הפרויקט, שאתם לא מקמפלים ל-UNICODE: בהגדרות הפרויקט תחת General בשורה Character Set יש לבחור Use Multi-Byte Character Set ולא Use Unicode Character Set. לאחר קביעת ההגדרה הזאת, אפשר להתייחס ל-TCHAR כמו ל-char.

דגשים

- הקפידו על קוד קריא ומתועד.
- עבדו באיטרציות – בדקו את הקוד שלכם לעתים תכופות בעת הקידוד, ולא לאחר כתיבת התוכנה כולה.
- רשמו לעצמכם את מבנה התוכנה הכללי לפני שאתם מתחילים לקודד.
  - חשבו איזה מודולים ופונקציות אתם צריכים. מתוך הפונקציות, איזה יהיו סטטיות ואיזה פומביות. אל תכתבו את כל התוכנה בקובץ אחד!

- זכרו כי כל קטע קוד שאתם משתמשים בו יותר מפעם אחת, צריך להיכתב כפונקציה נפרדת. כאשר פונקציה נעשית גדולה ומסובכת, פצלו אותה למספר פונקציות.

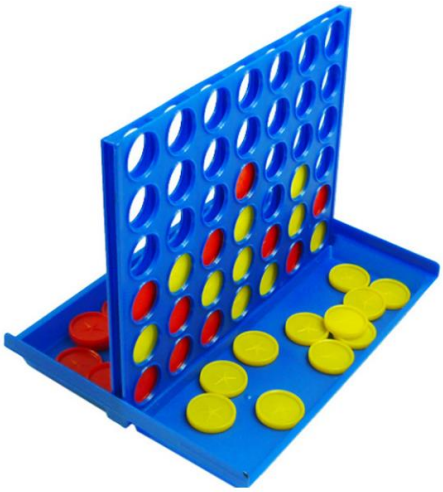
- זכרו להשתמש בכלי הדיבוג שה-IDE מספק.
- אין דרך אחת נכונה לפתור את התרגיל והתרגיל לא כוון לפתרון ספציפי.
- השתמשו בזיכרון דינמי לאחסן מידע שגודלו אינו ידוע בזמן הקומפילציה. אינכם רשאים להניח חסם עליון שרירותי לגודל המידע. השתמשו בקבועים ושימו לב לשחרור זיכרון דינמי.
- אתחלו את כל הפוינטרים ל-NULL. כל פונקציה שמקבלת מצביע צריכה לבדוק שהוא שונה מ-NULL לפני שהיא עושה dereference (אופרטור *).
- בדקו את ערך החזרה של כל פונקציה, שיכולה להחזיר שגיאה (malloc, WaitForSingleObject וכו'). פעלו בהתאם לערך.
- שחררו זיכרון דינאמי ו-handles בהקדם האפשרי (באמצעות Free ו-CloseHandle בהתאמה).
- לפני שאתם משתמשים בפקודת API בפעם הראשונה, רצוי לקרוא את התיעוד שלה ב-MSDN שלה. באופן כללי, רצוי גם לקרוא את הפונקציות שמופיעות ב-MSDN תחת Related Functions.
- הפורום עומד לשירותכם. אנו מעודדים אתכם לנסות תחילה לחפש תשובות באינטרנט, כאשר מדובר בשאלות תכנות כללית.

בהצלחה!

סקירה כללית

בתרגיל זה, תממשו גרסת online למשחק הילדים האהוב: "ארבע בשורה" הכולל צ'ט בין השחקנים. במשחק זה משתתפים 2 שחקנים, כאשר כל שחקן מקבל דסקיות בצבע אחד – צהוב או אדום. לוח המשחק מכיל 7 עמודות ו-6 שורות, כאשר הכנסת דסקית לראש עמודה מסוימת, מפילה אותו לתחתית העמודה, במקום הפנוי הבא. כלומר, אם ישנה דסקית כבר בתחתית, הדיסקית הנוספת תתמקם באותה העמודה, בשורה שמעל השורה התחתית, וכך הלאה. השחקן המנצח הוא זה שמצליח ליצור רצף של 4 דסקיות לאורך, לרוחב או באלכסון.

הסבר על המשחק ניתן לקרוא בויקיפדיה.



עליכם לכתוב תוכנית אחת עם שני מצבי ריצה (רק אחד מהם כל פעם):

1. מצב שרת – מנהל את המשחק. מקבל תקשורת נכנסת מאפליקציות הלקוח, מחשב את מצב הלוח, ומפיץ הודעות בין הלקוחות. תוכנת השרת צריכה להיות מופעלת לפני תוכנת הלקוח.
2. מצב לקוח – הממשק של הלקוח למשחק. התוכנה מקבלת פקודות מהשחקן, ושולחת אותן לשרת. התוכנה מקבלת עדכונים מהשרת, ומציגה אותם לשחקן. היא מהווה גשר בין השחקן לתוכנת השרת.

שורת הרצה

תוכנת שרת

הפעלת התוכנית במצב השרת נראית כך:

ex4.exe server <logfile> <server port>

- server - מציין שהתוכנה תופעל במוד שרת.
- logfile - נתיב לקובץ לוג שהתוכנה תכתוב אליו.
- server port – כתובת port של תוכנת השרת.

לדוגמא:

```
ex4.exe server c:\documents\ex4_server_log.txt 4444
```

תוכנת לקוח

הפעלת התוכנית במצב הלקוח נראית כך:

ex4.exe client <logfile> <server port> <input_mode> <input file>

- client – מציין שהתוכנה תופעל במוד לקוח.
- logfile – נתיב לקובץ לוג שהתוכנה תכתוב אליו.
- server port – כתובת port של תוכנת השרת.
- input mode – ארגומנט שיכול לקבל שני ערכים בלבד. human או file. במוד human, הקלטים יוזנו ע"י משתמש אנושי. במוד file, הקלטים יקראו מתוך קובץ.
- input file – ארגומנט זה ישלח רק אם ה-input mode הוא file. זהו נתיב לקובץ הקלט.

לדוגמא:

```
ex4.exe client c:\documents\ex4_client1_log.txt 4444 file c:\ex4_input.txt
```

שימו לב שמדובר באותה תוכנת exe לשרת וללקוח, ושמספר הארגומנטים עלול להשתנות בהתאם למוד, ולפרמטרים עצמם.

הנחות

- לא ניתן להניח שפתיחת הקבצים מצליחה תמיד.
- לא ניתן להניח שפתיחת ה-socket מצליחה תמיד.
- ניתן להניח ששם המשתמש מכיל רק אותיות, מספרים וקווים תחתונים, ללא רווחים. ניתן להניח אורך מקסימלי של 30 תווים.
- אפשר להניח שיינתן קובץ לוג אחר לכל תוכנה.
- אפשר להניח שמספר הארגומנטים שהוזן מתאים למוד הרלוונטי.

מהלך ריצה כאשר אין תקלות (Happy Path)

להלן מהלך הריצה אידאלי. זהו תיאור של התנהגות התוכנה, כאשר אין אף תקלה. נושא התקלות ידון בהמשך.

למען פשטות, הושמטו פרטי ממשק המשתמש (קלט פלט למסך ולקבצים) מהטבלה.

נושא הודעות התקשורת ידון בהמשך.

	מבצע	פעולה	סוג הודעה
1	שרת	מחכה ל-connection.	
2	לקוח 1	מתחבר לשרת.	
3	לקוח 1	שולח את שמו לשרת.	NEW_USER_REQUEST
4	שרת	מאשר את הלקוח. שולח ללקוח את מספר השחקנים הנוכחי (1).	NEW_USER_ACCEPTED
5	לקוח 2	מתחבר לשרת.	
6	לקוח 2	שולח את שמו לשרת.	NEW_USER_REQUEST
7	שרת	מאשר את הלקוח. שולח ללקוח את מספר השחקנים הנוכחי (2).	NEW_USER_ACCEPTED
8	שרת	מודיע לכל השחקנים שהמשחק התחיל.	GAME_STARTED
9	שרת	מודיע לכל השחקנים את מצב הלוח הנוכחי.	BOARD_VIEW
10	שרת	מודיע לכל השחקנים, שתורו של לקוח 1 (תמיד הלקוח הראשון שהתחבר מתחיל).	TURN_SWITCH
11	לקוח 1	משחק: מודיע לשרת באמצעות הודעה על המהלך שהוא מעוניין לבצע.	PLAY_REQUEST
12	שרת	מאשר את המהלך.	PLAY_ACCEPTED
13	שרת	מודיע לכל השחקנים את מצב הלוח הנוכחי.	BOARD_VIEW
14	שרת	מודיע לכל השחקנים, שתורו של לקוח 2.	TURN_SWITCH
15	חזרה על שלבים 11-14 (בכל פעם שחקן אחר) עד שנגמר המשחק		
16	שרת	מודיע לכל השחקנים על סיום המשחק, ועל התוצאה.	GAME_ENDED
17	לקוח 1	מתנתק מהשרת. תוכנת הלקוח מסתיימת.	

18	לקוח 2	מתנתק מהשרת. תוכנת הלקוח מסתיימת. (הערה: אין חשיבות לסדר בין שלב 17 לשלב 18).	
19	שרת	השרת ממתין להתנתקות הלקוחות, על מנת לוודא שהם אכן קיבלו את הודעת סיום המשחק.	
20	שרת	השרת סוגר את החיבורים לכל הלקוחות.	
21	שרת	השרת חוזר לשלב 1.	

הודעות התקשורת

בתרגיל הזה, אתם תצטרכו לממש פרוטוקול מעל TCP.

מבנה ההודעות יהיה מבוסס מחרוזות טקסט, שיועברו בין כל לקוח לשרת. תזכורת: מחרוזת היא מערך תווים המסתיים בתו '\0' .

ההודעה מורכבת משני שדות, שמופרדים באמצעות התו נקודותיים (':').

<message_type>:<param_list>\0

1. message_type – סוג ההודעה. השדה הזה משמש את התוכנה כדי להבחין בין הודעות. כך ניתן להפעיל לוגיקה מתאימה לכל סוג הודעה.
2. param_list – רשימת פרמטרים מופרדים על ידי התו נקודה-פסיק (';').
   <param0>;<param1>;<param2>
   מספר הפרמטרים אינו קבוע.
3. '\0' – התו שמציין את סיום ההודעה. השדה הזה משמש את התוכנה כדי לזהות את סוף ההודעה.

אם יש 0 פרמטרים, ישלח השדה <message_type> ללא פרמטרים וללא נקודותיים (':'), אך עם '\0' בסוף.

שימו לב שגם כאשר הפרמטר מציין מספר, ישלח התו שמציין את המספר הזה. לדוגמה, עבור הפרמטר 0, ישלח התו '0' (שבמקרה ערך ה-ascii שלו שווה למספר 48 בבסיס עשרוני). להלן ההודעות אותן תידרשו להגדיר. אין להגדיר הודעות נוספות.

שולח	message_type	תיאור
לקוח	NEW_USER_REQUEST	בקשה להצטרף למשחק. ההודעה מכילה את שם המשתמש החדש.
שרת	NEW_USER_ACCEPTED	תגובה חיובית להודעת NEW_USER_REQUEST. שולחת כפרמטר את מספר השחקנים הנוכחי (1 או 2).
שרת	NEW_USER_DECLINED	תגובה להודעת NEW_USER_REQUEST. ההודעה מציינת שהשחקן לא אושר להצטרף למשחק.
שרת	GAME_STARTED	הודעה על תחילת המשחק.
שרת	BOARD_VIEW	מכילה את המצב המלא של הדיסקיות על הלוח.
שרת	TURN_SWITCH	הודעה על התחלת תורו של שחקן. ההודעה מכילה את שם השחקן.
לקוח	PLAY_REQUEST	מכילה את העמודה שבוחר השחקן, ולתוכה תוכנס הדיסקית שלו אם יש מקום.
שרת	PLAY_ACCEPTED	נשלחת בתגובה להודעת PLAY_REQUEST, כאשר הבקשה חוקית – כלומר נשלחה בתור הרלוונטי, עמודה בטווח 0-6, והעמודה אינה מלאה.
שרת	PLAY_DECLINED	נשלחת כתגובה ל-PLAY_REQUEST , אם לא התקיימו הקריטריונים לשליחת PLAY_ACCEPTED. להודעה יצורפו פרמטרי הסבר כמפורט בהמשך.
שרת	GAME_ENDED	הודעה על סיום המשחק. ההודעה מכילה כפרמטר את תוצאת המשחק – איזה שחקן ניצח, או תיקו.

לקוח	SEND_MESSAGE	בקשה להעביר הודעה לשחקן האחר. כל מילה בהודעה וכל רווח יהוו פרמטרים נפרדים. ההודעה לא תכיל את התו ';' כחלק מהמשפט, אלא רק לשם הפרדה.
שרת	RECEIVE_MESSAGE	הועברה הודעה מהשחקן האחר. הפרמטר הראשון יהיה שמו של השחקן ששלח. מעבר לכך כל מילה בהודעה וכל רווח יהוו פרמטרים נפרדים.

דוגמא להודעה מסוג GAME_STARTED:

"GAME_STARTED"

דוגמא להודעה מסוג NEW_USER_REQUEST:

"NEW_USER_REQUEST:isp_student"

דוגמה להודעת מסוג SEND_MESSAGE:

"SEND_MESSAGE:Prepare; ;to; ;lose!"

כאן הגרשיים הכפולים נועדו להבהיר שמדובר במחרוזת המסתיימת ב-'\0'.

תיאור מפורט של התוכנה

הסעיף הזה מכיל את פרטי התוכנות, כולל ממשק המשתמש וטיפול בשגיאות.

תוכנת הלקוח – מהלך ריצה מפורט

1. תוכנת הלקוח תתחבר לשרת בפרוטוקול TCP ב-port שצוין בארגומנט הקלט.
2. לאחר חיבור מוצלח, תירשם השורה הבאה למסך ולקובץ הלוג:
   Connected to server on 127.0.0.1:<port>
3. במידה והחיבור נכשל, תירשם למסך וקובץ הלוג ההודעה הבאה:
   Failed connecting to server on 127.0.0.1:<port>. Exiting
   תוכנת הלקוח תסיים את פעולתה באופן מידי. השרת ישאר באותו מצב שהיה לפני ניסיון החיבור.
   שימו לב שבמקום <port>, אמור להופיע המספר בן-4 הספרות שהופיע כארגומנט הקלט.
4. לאחר החיבור לשרת, תוכנת הלקוח תבקש מהמשתמש להזין שם משתמש אם אנו ב- human input mode (הבקשה תודפס על המסך). אם אנו ב-file input mode, התוכנה תקרא את שם המשתמש מהשורה הראשונה בקובץ הקלט (ראו מבנה קובץ הקלט בהמשך). תוכנת הלקוח תשלח לשרת הודעת NEW_USER_REQUEST עם שם המשתמש המבוקש.
5. אם השרת קיבל את המשתמש, השרת ישלח הודעה NEW_USER_ACCEPTED, שמודיע על קבלת המשתמש ועל מספר השחקנים הנוכחי. השחקן הראשון שמתחבר מקבל את הצבע האדום, והשני צהוב (רלוונטי לצורך הדפסת הלוח).
   אם השרת סרב לקבל את השחקן (שם משתמש תפוס או שיש כבר שני שחקנים מחוברים), הוא ישלח את ההודעה NEW_USER_DECLINED. תירשם למסך הלקוח ההודעה הבאה:
   Request to join was refused
   תוכנת הלקוח תסגור את המשאבים שפתחה, ותסיים את פעולתה באופן מידי.
6. מעתה ואילך המשתמש יוכל להקליד פקודות. פקודה היא שורת טקסט שהזין ומסתיימת ב-enter.
   כל הפקודות הן case-sensitive.
   הפקודות הנתמכות הן:
   a. play <column num>
      בקשה מהשרת לשחק את התור. column num הוא מספר העמודה שבה רוצה השחקן להכניס דיסקית. העמודות נספרות מ-0 עד 6, כאשר העמודה השמאלית היא 0.
      לדוגמה:
      play 1

שפירושו להכניס דיסקית לעמודה השנייה משמאל.

תוכנת הלקוח תשלח לשרת הודעת PLAY_REQUEST עם העמודה המבוקשת.
אם הבקשה מתקבלת, השרת יחזיר הודעת PLAY_ACCEPTED.
תירשם למסך ההודעה הבאה:

Well played

אם הבקשה נדחית, השרת יחזיר הודעת PLAY_DECLINED עם הודעת שגיאה.
תירשם למסך ההודעה הבאה:

Error: <msg>

היא הודעת טקסט, שהתקבלה מהשרת. שגיאות אפשריות כוללות מהלך עם קואורדינטות לא חוקיות, או מהלך של שחקן שלא בתורו – ראו בהמשך. אם התקבלה מהשרת ההודעה "Illegal move", השרת ימתין שהלקוח ישלח הודעת play נוספת (התור נשאר אצל אותו לקוח).

השגיאות האלה **לא** גורמות לסיום התוכנה.

b. message <message body>

השחקן יכול לשלוח הודעות לשחקן השני. כל הודעה תהיה באורך של עד 100 תווים (כולל הפקודה message עצמה). שימו לב, ה-enter שמסמן את סוף הפקודה, נכלל בהודעה עצמה. תוכנת הלקוח תשלח לשרת את ההודעה SEND_MESSAGE.

c. exit

פקודת יציאה מהתוכנה. התוכנה תיסגר באופן מסודר (ראו את הסעיף "סיום התוכנה כאשר אין שגיאות"). אין לשלוח הודעה לשרת.

7. השחקן יוכל להקליד פקודות עד לקבלת הודעת GAME_ENDED (ראו את החלק הבא).
8. במידה ועובדים ב-file input mode, הפקודות שהוגדרו בסעיף 6 יכולות להופיע בקובץ הקלט. בכל פעם שמגיע תורו של שחקן, תוכנת הלקוח שלו תקרא את השורה הבאה מהקובץ (כל שורה מכילה פקודה אחת ורק אחת, פרט לשורה הראשונה שהכילה את שם המשתמש). אם קראנו הודעת message, אז נעביר את ההודעה לשרת ונקרא את ההוראה הבאה, עד שנגיע להוראת play, שלאחריה התור יעבור ללקוח השני.

תוכנת הלקוח – הודעות מהשרת

בחלק הקודם, כל ההודעות שהגיעו מהשרת היו תגובות להודעות של הלקוח. השרת יכול גם ליזום הודעות ללקוח. להלן הודעות שהשרת יכול לשלוח, והתגובה הנדרשת לכל פקודה.

1. GAME_STARTED

הודעה שמציינת שהמשחק התחיל. בקבלת ההודעה, תירשם למסך ההודעה הבאה:

Game is on!

2. TURN_SWITCH

עדכון על שינוי תור. הודעה מציינת תור איזה שחקן כעת. ההודעה מכילה גם את שם המשתמש. כאשר מתקבלת, תירשם למסך ולקובץ הלוג ההודעה:

<username>'s turn

3. BOARD_VIEW

עדכון על מצב הלוח. ההודעה מכילה את מצב הלוח העדכני. כאשר מתקבלת הודעה כזו, תירשם למסך (אך לא לקובץ הלוג) תמונת הלוח. אתם חופשיים לבחור באיזה פורמט (כלומר אילו פרמטרים) השרת יעדכן את הלקוח במצב הלוח החדש. תמונת הלוח שיש להדפיס (לדוגמה) נראית כך:

כדי להדפיס את הלוח, היעזרו בקובץ PrintBoard.c שנמצא באתר הקורס. האפסים מייצגים חורים, והריבועים הצעבוניים מייצגים את הדיסקיות.

4. RECEIVE_MESSAGE

השחקן השני שלח הודעה. תרשם למסך ההודעה הבאה:

<username>: <message>

כאשר <username> הוא שמו של השחקן השני, ו-<message> זה תוכן ההודעה שנשלחה.

5. GAME_ENDED

עדכון על סיום המשחק. ההודעה מכילה את שם השחקן המנצח, או מדווחת על תיקו. כאשר מתקבלת ההודעה, תירשם למסך ולקובץ הלוג ההודעה הבאה:

Game ended. The winner is <username>!

במקרה של תיקו תירשם למסך ולקובץ הלוג ההודעה הבאה:

Game ended. Everybody wins!

תוכנת הלקוח תיסגר באופן מסודר (ראו את הסעיף "סיום התוכנה כאשר אין שגיאות").

תוכנת הלקוח – פרטים נוספים

6. אם יש שגיאה בהכנסת הפקודה (פקודה לא קיימת או פורמט לא נכון), תירשם למסך ולקובץ הלוג ההודעה הבאה:

Error: Illegal command

השגיאה הזו **לא** גורמת לסיום התוכנה. הלקוח יכול לנסות פקודות נוספות.

שימו לב: בסעיף הזה מדובר רק על שגיאות שמונעות את שליחת ההודעה. שגיאות הנוגעות לתוכן ההודעה מטופלות על ידי השרת.

7. כל הודעה שנשלחת לשרת, וכל הודעה שמתקבלת מהשרת, תתועד בקובץ הלוג (אבל לא תודפס למסך) בפורמט הבא:

Sent to server: <raw message>

Received from server: <raw message>

raw message הוא הטקסט הגולמי של ההודעה, כולל כל השדות, כפי שהתקבל מהשולח.

8. כאשר החיבור לשרת מתנתק, תירשם למסך ולקובץ הלוג ההודעה הבאה:

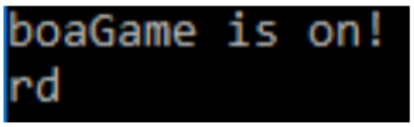
Server disconnected. Exiting.

התוכנה תסיים את פעולתה באופן מידי אחרי שחרור המשאבים הרלוונטיים.

9. התוכנה תורכב ממספר חוטים. החוט הראשי (main thread) אחראי רק לבדיקת הקלט לתוכנה, לפתיחת החוטים האחרים ולסגירת התוכנה. קיימים לפחות החוטים הבאים:
   a. חוט לתקשורת יוצאת – החוט הזה מטפל בשליחת הודעות
   b. חוט לתקשורת נכנסת – החוט הזה מטפל בקבלת הודעות
   c. חוט ממשק משתמש – החוט הזה מטפל בקליטת תווים מהמשתמש או מקובץ הקלט.

אתם רשאים להוסיף חוטים נוספים כראות עינכם.

10. ייתכן מצב שבו מדפיסים למסך באמצע הקלדה של פקודה. מותר שהתווים של ההדפסה יוצגו ב-cmd באמצע התווים של ההקלדה. כלומר, אין צורך לסנכרן את ההקלדות וההדפסות. דוגמא לפלט חוקי למסך:



11. אם מתקבלת הודעת תקשורת לא תקנית, התוכנה תדפיס הודעת שגיאה למסך ולקובץ הלוג, ותסיים את פעולתה באופן מידי.

תוכנת השרת – מהלך ריצה מפורט

1. השרת יאזין לתקשורת נכנסת בפרוטוקול TCP על הפורט שצוין בארגומנט הקלט.
2. השרת יתמוך במקסימום 2 לקוחות. לאחר שני לקוחות, הוא יסרב לחיבורים.
3. לצורך כך, יש ליצור לפחות 2 חוטי תקשורת, כל אחד מנהל את הקשר מול לקוח אחד. זה בנוסף לחוט הראשי של השרת. ניתן ליצור חוטים נוספים.
4. לאחר חיבור ללקוח חדש, השרת יקבל מהלקוח הודעת NEW_USER_REQUEST, המכילה את שם המשתמש המבוקש.
   אם שם המשתמש פנוי, השרת ישלח בתגובה NEW_USER_ACCEPTED. ההודעה תכיל את מספר השחקנים העדכני. אם שם המשתמש תפוס, השרת ישלח בתגובה NEW_USER_DECLINED.
5. לאחר שהתחברו שני לקוחות שאושרו, השרת ישלח לשניהם את הודעות הבאות, בסדר הזה:
   GAME_STARTED, הודעה שמציינת שהמשחק התחיל
   BOARD_VIEW, הודעה שמכילה את מצב הלוח הנוכחי (בשלב הזה ריק)
   TURN_SWITCH, הודעה שמכילה את שם השחקן שתורו. הלקוח הראשון שהתחבר מתחיל את המשחק.
6. השרת יגיב להודעות שמגיעות מתוכנת הלקוח, על פי מה שמוגדר בתוכנת הלקוח.
7. כאשר מתקבלת הודעה ממשתמש על מהלך שהוא רוצה לבצע, השרת יבדוק אם המהלך חוקי (תור השחקן, התקבלה עמודה בטווח 0-6, העמודה אינה מלאה).
   אם המהלך חוקי, הוא ישלח לכל השחקנים הודעת BOARD_VIEW ולאחר מכן הודעת TURN_SWITCH. שימו לב שהדיסקיות "נופלות" למקום הנמוך ביותר בעמודה.
   אם המהלך לא חוקי, השרת ישלח לשחקן דחייה, עם הודעת הסבר. ההודעות הן:
   "Not your turn"
   "Game has not started"
   "Illegal move"
   ההודעות יעברו כפרמטרים, בדומה להודעת message.
8. כאשר השרת מזהה שהסתיים המשחק, הוא ישלח לכל השחקנים הודעת GAME_ENDED. אם יש ניצחון, ההודעה מכילה את שם השחקן שניצח. אם יש תיקו, ההודעה תציין שיש תיקו.
9. השרת יסגור את החיבורים לכל השחקנים באופן שמבטיח שהם קיבלו את ההודעה על סיום המשחק.
10. השרת יהיה מוכן להתחיל משחק חדש.

תוכנת השרת – פרטים נוספים

1. כאשר שחקן מתנתק, תירשם למסך ולקובץ הלוג ההודעה הבאה:
   Player disconnected. Ending communication.
   לאחר מכן השרת יסגור את החיבורים הנותרים.
   השרת יהיה מוכן להתחיל משחק חדש.

סיום התוכנה כאשר אין שגיאות

התוכנה תחזיר 0 אם הסתיימה ללא שגיאות במוד לקוח.

כאשר אין שגיאות, התוכנה צריכה להסתיים באופן מסודר:

- אין לצאת מהתוכנה כאשר יש חוטים משניים פתוחים. מותר להשתמש ב-TerminateThread על מנת לסגור את החוטים.
- אין לצאת מהתוכנה כאשר יש handles פתוחים.
- אין לצאת מהתוכנה כאשר יש זיכרון דינאמי שלא שוחרר.
- אין לצאת מהתוכנה כאשר יש קבצים פתוחים.
- אין לצאת מהתוכנה כאשר יש sockets פתוחים.

- אין לצאת מהתוכנה לפני שקוראים ל-WSACleanup.

טיפול בשגיאות

יש לבדוק את הצלחה של כל פונקציה שעלולה להיכשל (הקצאת זיכרון, פתיחת קבצים, יצירת חוט, פעולות על sockets וכו').

במקרה של שגיאה כלשהי, כגון כישלון בהקצאה דינאמית או כישלון בפתיחת חוט, התוכנה תדפיס למסך ולקובץ הלוג הודעת שגיאה בעלת משמעות. אם מדובר בשגיאה שלא הוגדרה במסמך הזה, השתמשו בפורמט שמוגדר בהמשך להודעות לוג נוספות.

לאחר הדפסת הודעת השגיאה לקובץ הלוג, התוכנה שבה נגרמה השגיאה (לקוח/שרת) תסיים את פעולתה מיד - השתמשו בפונקציה exit. עשו זאת מתוך המקום בו קרתה התקלה, גם אם מדובר בחוט משני. אין צורך לסגור את כל המשאבים.

קובץ לוג

הנחה

אם קובץ הלוג קיים לפני ריצת התוכנית, על התוכנית לדרוס את הקבצים הקיימים בקבצים חדשים.

הודעות לוג נוספות

אתם רשאים להוסיף הודעות ללוג כרצונכם, בפורמט הבא:

Custom message: <message>

message היא ההודעה החדשה שהגדרתם.

קובץ הקלט

קובץ הקלט (במידה והוא קיים), תמיד יכיל את שם המשתמש בשורה הראשונה, ולאחר מכן הוראות play ו-message. ניתן להניח שיש עד 50 הוראות play בקובץ, ובין 2 הוראות play עוקבות, תופענה לכל היותר 10 הוראות message. כל הוראה מסתיימת בירידת שורה, כולל האחרונה. ההוראה האחרונה בקובץ תהיה תמיד exit. לדוגמה:

awesome_student

message Hi there!

play 5

play 3

play 3

message I think I will quit

exit

כתובות

כדי להתחבר לתוכנת שרת שרצה באותו מחשב כמו תוכנת הלקוח, השתמשו בכתובת ה-IP ל-localhost: 127.0.0.1.

inet_addr

הוסיפו הגדרה של _WINSOCK_DEPRECATED_NO_WARNING כדי לקמפל עם הפונקציה inet_addr.