תרגיל בית 4 – בול פגיעה (Bulls & Cows)

מועד הגשת התרגיל: עד יום ראשון 17/01/21 בשעה 23:55.

בודק אחראי: עופר בר oferbear@mail.tau.ac.il

עדכון אחרון – 27.12. כל השינויים נרשמו באדום.

מטרת התרגיל

- העמקת ההבנה במושגי החוט (Thread) במערכות הפעלה בכלל וב-Windows בפרט.
- עבודה עם מספר חוטים במקביל.
- שימוש ב-Mutex וב-Semaphore לסנכרון גישה למשאבים משותפים בין חוטים.
- שימוש ב-Mutex ו-Semaphore לסנכרון גישה לזיכרון משותף בין החוטים.
- 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 – תשפ"א" 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 2019, באופן שיאפשר לבודק התרגילים לפתוח את הפרויקט על ידי לחיצה כפולה על קובץ ה-solution ולקמפל את הפרויקט ללא התראות או שגיאות.
- הגישו בנוסף את תיקיית ה-Debug עם קובץ ה-exe.
- בפרויקט הזה תגישו שני פרויקטים בתוך solution אחד. שם ה-solution צריך להיות Ex4_id1_id2, כאשר id1/id2 מוחלפים בתעודות הזהות של זוג השותפים. שמות הפרויקטים צריכים להיות client ו-server. שמות ה-executables צריכים להיות client.exe -server.exe בהתאמה.
- שם ה-zip שאתם מגישים צריך להיות Ex4_id1_id2.zip. כאשר id1/id2 מוחלפים בתעודות הזהות של זוג השותפים.
- שימו את הקבצים המשותפים לשני הפרויקטים בתיקייה בשם Share, שממוקמת ביחד עם תיקיות הפרויקט בתיקיית ה-solution.
- קמפלו את הקוד לגרסאת 32-bit (Debug x86)
- 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

- מענה לשאלות בפורום – היות ומדובר בתרגיל ארוך שיתפרס מפרסומו עד סוף הסמסטר המענה בפורום ישתנה לאורך הזמן. קראו בקפידה את התרגיל בשבוע הראשון מפרסומו ותכננו את הקוד שלכם – חשבו מה אתם לא מבינים. מומלץ לקרוא גם שאלות של אחרים, שעשויות להיות רלוונטיות גם אליכם.
- הקפידו על קוד קריא ומתועד. בפרט הקפידו על חלוקה למודולים, פונקציות ומתן שמות משמעותיים לפונקציות ומשתנים והימנעו ממספרי קסם בקוד (השתמשו ב-define).
- 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.
- איטרציות – שימו לב שאתם מקדמים את שתי התכניות במקביל כדי שתוכלו לבדוק את עצמכם ולזהות בעיות בתקשורת ביניהם כמה שיותר מוקדם.
- אתם יכולים לדבג שני תהליכים, אחרי שהרצתם את אחת התכניות ב-Debug ב-Visual Studio תוכלו לדבאג במקביל גם את התכנית השנייה לאחר שהיא התחילה לרוץ: Debug -> Attach to process ולבחור את התהליך.
- 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. חפשו תחילה תשובות באינטרנט, כאשר מדובר בשאלות תכנות כללית.

הנחיות והנחות

- יש לבדוק את כל ערכי ההחזרה הרלוונטיים של כל הפונקציות שאתם משתמשים.
- יש לשחרר משאבים בכל תרחיש. משאבים הם זיכרון שהוקצה דינאמית HANDLE-ים של מערכת ההפעלה (סוקטים, קבצים, מיוטקסים וכו')
- אין להשתמש ב-TerminateProcess.
- השימוש ב-TerminateThread מותר אחר ורק לאחר ניסיון סגירה שנכשל בגלל Timeout.
- בכל מקרה של שגיאה יש להדפיס הודעת שגיאה ייחודית לאותה השגיאה שמפרטת מה קרה במלל ולסיים את ריצת התכנית בצורה מסודרת תוך שחרור כל המשאבים.
- זכרו – גם אם בתרגול לא ראיתם משהו אבל הוא כתוב בתיעוד המלא של הפונקציה – עליכם להתייחס אליו.
- אין לכתוב פונקציות ארוכות – לכל היותר 100~ שורות. מצד שני אל תכתבו פונקציה לכל 3 שורות תוודאו שהחלוקה לפונקציות היא מונחית מטרה, ממזערת שכפול קוד ו/או משפרת קריאות.
- יש לתת שמות משמעותיים לפונקציות ומשתנים וכן לבצע חלוקה למודולים.
- יש לתעד ב-Header-ים מעל כל פונקציה מה היא מקבלת, מה היא מחזירה ומה היא עושה. יש לתעד בגוף הפונקציות על מנת להקל על הבנת הקוד.

- עבדו לפי הקונבנציות שהוגדרו במסמך coding conventions שבאתר המודל.

טיפ: מומלץ **מאוד** לעבוד עם שירות Version Control כלשהו למשל כמו GitHub על מנת לשמור את ההתקדמות שלכם וזאת משום שהתרגיל הוא איטרטיבי. ביצוע commit לאחר השלמת מדרגה בתרגיל יעזור לכם לעקוב אחר ההתקדמות שלכם, לזהות בעיות ויאפשר לכם לחזור לנקודה האחרונה שבה הקוד שלכם עבד במקרה שהסתבכתם ואתם לא מצליחים למצוא את הבעיה. למי מכם שמעולם לא השתמש ב-Git ו-GitHub עדיף מאוחר מאשר אף פעם.

בהצלחה!

סקירה כללית

בתרגיל זה, תממשו גרסה מקוונת למשחק "בול פגיעה" (Bulls and Cows).

הסבר קצר על המשחק: בתחילת המשחק בוחר כל שחקן מספר בן 4 ספרות, אסורות חזרות על ספרות. המשחק מתנהל בתורות כאשר כל שחקן בתורו מנחש את מספרו של היריב. לאחר הניחוש מקבל השחקן רמזים המעידים על טיב הניחוש, הנקראים **בולים ופגיעות**.

בול - מצב בו אחת הספרות במספר המנוחש זהה לספרה במספר המקורי, וממוקמת נכון.

פגיעה - מצב בו אחת הספרות במספר המנוחש זהה לספרה במספר המקורי, אך ממוקמת במיקום שונה.

נצחון מתקבל כאשר השחקן מצליח לנחש את המספר של השחקן היריב.

הסבר נוסף על המשחק ניתן לקרוא בויקיפדיה.

עליכם יהיה לממש **שתי** תוכנות.

1. תוכנת שרת – מנהלת את המשחק. היא מקבלת תקשורת נכנסת מאפליקציית הלקוח, מחשבת את תוצאות המשחק, ומפיצה הודעות בין הלקוחות. תוכנת השרת צריכה להיות מופעלת לפני תוכנת הלקוח.
2. תוכנת לקוח – הממשק של הלקוח למשחק. התוכנה מקבלת פקודות מהשחקן, ושולחת אותן לשרת. התוכנה מקבלת עדכונים מהשרת, ומציגה אותם לשחקן. תוכנת הלקוח לא מבינה את חוקי המשחק. היא מהווה גשר בין השחקן לתוכנת השרת.

שורת הרצה

תוכנת שרת

קריאה לתוכנת השרת נראית כך:

Server.exe <server port>

- server port – כתובת port של תוכנת השרת

לדוגמא:

```
>C:\...\server.exe 8888
```

תוכנת לקוח

קריאה לתוכנת הלקוח נראית כך:

Client.exe <server ip> <server port> <username>

- server ip – כתובת ip של השרת
- server port – כתובת port של תוכנת השרת
- username – שם המשתמש

לדוגמא:

```
>C:\...\client.exe 127.0.0.1 8888 Thomas
```

הנחות

- שם המשתמש ייחודי (לא יתחברו שני לקוחות עם אותו שם)
- שם המשתמש מכיל רק אותיות ומספרים ללא רווחים. ניתן להניח אורך מקסימלי של 20 תווים.

מהלך ריצה כאשר אין תקלות (Happy Path)

להלן מהלך הריצה אידאלי. זהו תיאור של התנהגות התוכנה, כאשר אין אף תקלה.

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

נושא סוגי הודעות התקשורת ידון בהמשך.

	מבצע	פעולה	סוג הודעה
1	שרת	מחכה ל-connection.	
2	לקוח 1	מתחבר לשרת ושולח אליו את שם המשתמש	CLIENT_REQUEST
3	שרת	מאשר את הלקוח.	SERVER_APPROVED
4	שרת	מציג לו (לקוח 1) את האפשרויות הבאות: 1. לשחק נגד שחקן אחר 2. להתנתק מהשרת	SERVER_MAIN_MENU
5	לקוח 2	מתחבר לשרת ושולח אליו את שם המשתמש	CLIENT_REQUEST
6	שרת	מאשר את הלקוח.	SERVER_APPROVED
7	שרת	מציג לו (לקוח 2) את האפשרויות הבאות: 1. לשחק נגד שחקן אחר 2. להתנתק מהשרת	SERVER_MAIN_MENU
8	לקוח 1	בוחר לשחק נגד שחקן אחר	CLIENT_VERSUS
9	לקוח 2	בוחר לשחק נגד שחקן אחר	CLIENT_VERSUS
10	שרת	מודיע לשני השחקנים שהמשחק מתחיל.	SERVER_INVITE
11	שרת	מבקש מלקוח 1 ומלקוח 2 לבחור את רצף הספרות שלהם (אותו הלקוח השני יצטרך לנחש)	SERVER_SETUP_REQUSET
12	לקוח 1	בוחר רצף של 4 ספרות ושולח לשרת	CLIENT_SETUP
13	לקוח 2	בוחר רצף של 4 ספרות ושולח לשרת	CLIENT_SETUP
14	שרת	מבקש מלקוח 1 ומלקוח 2 לבחור את רצף ספרות הניחוש שלהם	SERVER_PLAYER_MOVE_REQUEST
15	לקוח 1	בוחר רצף של 4 ספרות ושולח לשרת	CLIENT_PLAYER_MOVE
16	לקוח 2	בוחר רצף של 4 ספרות ושולח לשרת	CLIENT_PLAYER_MOVE
17	שרת	בודק את הניחושים של הלקוחות, ומחזיר לכל אחד את מספר ה-"בול" ו-"פגיעה" שלו, ואת הרצף שהלקוח השני ניחש	SERVER_GAME_RESULTS
18	חזרה על שלבים 14-17 עד שמגיע רצף נכון מאחד הלקוחות		
19	שרת	מודיע ללקוח 1 ולקוח 2 מי ניצח, ומה הרצף הנכון של הלקוח השני	SERVER_WIN
20	שרת	מציג לשני הלקוחות את האפשרויות הבאות: 1. לשחק נגד שחקן אחר 2. להתנתק מהשרת	SERVER_MAIN_MENU
21	לקוח 1	בוחר לשחק נגד שחקן אחר	CLIENT_VERSUS
22	לקוח 2	בוחר להתנתק מהשרת	CLIENT_DISCONNECT
23	שרת	אין שחקנים אחרים לשחק מולם אז שולח ללקוח 1 שהוא לא הצליח למצוא מול מי לשחק	SERVER_NO_OPPONENTS
24	שרת	מציג ללקוח 1 את האפשרויות הבאות: 1. לשחק נגד שחקן אחר 2. להתנתק מהשרת	SERVER_MAIN_MENU
25	לקוח 2	בוחר להתנתק מהשרת	CLIENT_DISCONNECT
26	שרת	השרת ממשיך לרוץ עד שמי שהריץ אותו מכבה אותו	

הודעות התקשורת

בתרגיל הזה, אתם תצטרכו לממש פרוטוקול מעל TCP.

מבנה ההודעות יהיה מבוסס טקסט. הודעה היא מערך תווים.

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

<message_type>:<param_list>\n

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 (';').
   <param1>;<param2>;<param3>
   מספר הפרמטרים אינו קבוע.
3. '\n' – התו שמציין את סיום ההודעה. השדה הזה משמש את התוכנה כדי לזהות את סוף ההודעה.

אם יש 0 פרמטרים, ישלח השדה <message_type> בלבד, ללא נקודותיים (':').

הפרמטרים נשלחים בפורמט human readable. כלומר, גם כאשר הפרמטר מציין מספר, ישלח התו שמציין את המספר הזה, ולא תו שערך ה-ascii שלו שווה למספר. לדוגמא, אם הפרמטר הראשון הוא 1, ישלח התו '1' ולא התו '\1'.

להלן ההודעות אותן תידרשו להגדיר. ניתן לשלוח דברים נוספים שלא במסגרת הודעות.

שולח	message_type	תיאור	פרמטרים
לקוח	CLIENT_REQUEST	הלקוח שולח לשרת את שם המשתמש שלו	שם המשתמש
	CLIENT_VERSUS	לקוח רוצה לשחק נגד לקוח אחר	-
	CLIENT_SETUP	הלקוח בוחר את רצף 4 הספרות אותו השחקן השני ינחש	רצף 4 ספרות התחלתי
	CLIENT_PLAYER_MOVE	הלקוח בחר רצף 4 ספרות (ניחוש רצף השחקן השני)	רצף 4 ספרות (הניחוש)
	CLIENT_DISCONNECT	הלקוח מעוניין להתנתק מהשרת	-
שרת	SERVER_MAIN_MENU	השרת רוצה שהלקוח יציג למשתמש את התפריט הראשי	-
	SERVER_APPROVED	השרת אישר את התחברותו של הלקוח	-
	SERVER_DENIED	השרת דחה את בקשת ההתחברות של הלקוח מסיבה כלשהי (אולי מטפל כבר בשני לקוחות)	הסיבה שבגללה הבקשה נדחתה כמחרוזת
	SERVER_INVITE	השרת שולח ללקוח שעומד להתחיל משחק מול לקוח אחר ושולח בהודעה את שם המשתמש של היריב	שם הלקוח האחר
	SERVER_SETUP_REQUSET	השרת מבקש מהלקוח לבחור את רצף הספרות ההתחלתי שלו	-
	SERVER_PLAYER_MOVE_REQUEST	השרת מבקש מהלקוח לבחור ניחוש	-

	SERVER_GAME_RESULTS	תוצאת סבב הניחוש	מספר ה"בול" וה-"פגיעה", שם הלקוח האחר, הניחוש של הלקוח האחר
	SERVER_WIN	השרת מודיע ללקוח שיש מנצח למשחק	שם הלקוח המנצח, הרצף הנכון של הלקוח השני
	SERVER_DRAW	השרת מודיע ללקוח שהמשחק הסתיים בתיקו (שני הלקוחות ניחשו נכון באותו הסיבוב)	-
	SERVER_NO_OPPONENTS	השרת מודיע ללקוח שאין לקוחות הפנויים למשחק כרגע	-
	SERVER_OPPONENT_QUIT	השרת מודיע ללקוח שהשחקן השני התנתק והמשחק מופסק	-

דוגמא להודעה מסוג SERVER_APPROVED:

"SERVER_APPROVED"

דוגמא להודעה מסוג CLIENT_PLAYER_MOVE:

" CLIENT_PLAYER_MOVE:1234"

תיאור מפורט

תוכנת הלקוח – מהלך ריצה מפורט

אלא אם נאמר אחרת זמן ההמתנה לתגובה מהשרת יהיה 15 שניות.

1. תוכנת הלקוח תתחבר לשרת בפרוטוקול TCP בכתובת שצוינה בארגומנטי הקלט.
2. לאחר חיבור מוצלח, תירשם השורה הבאה למסך:

   Connected to server on <ip>:<port>

3. במידה והחיבור נכשל, תירשם למסך ההודעה הבאה:

   Failed connecting to server on <ip>:<port>.
   Choose what to do next:
   1. Try to reconnect
   2. Exit

עדכון 27.12 – נמחקה הדרישה להדפיס errorcode כלשהו.

4. במקרה של התנתקות פתאומית מהשרת או TIMEOUT בהמתנה לתגובה מהשרת לאחר ההתחברות יש להתנתק מהשרת ולהדפיס למסך את ההודעה הבאה:

   Failed connecting to server on <ip>:<port>.
   Choose what to do next:
   1. Try to reconnect
   2. Exit

במקרה שהמשתמש בוחר באופציה 1 הלקוח ינסה להתחבר מחדש לשרת. אם המשתמש בחר באופציה 2 הלקוח יסיים את ריצתו לאחר שיסגור וישחרר את כל המשאבים שהקצה במהלך הריצה.

5. לאחר החיבור לשרת, הלקוח ישלח לשרת את שם המשתמש בהודעת CLIENT_REQUEST וימתין לקבלת SERVER_APPROVED. אם אין מענה יש להתנתק מהשרת ולהציג את ההודעה מ-4. במידה ומתקבלת הודעת SERVER_DENIED יש להתנתק מהשרת ולהדפיס למסך את ההודעה הבאה:

   Server on <ip>:<port> denied the connection request.
   Choose what to do next:
   1. Try to reconnect
   2. Exit

6. לאחר חיבור מוצלח לשרת וקבלת אישור על שם המשתמש, הלקוח יציג למשתמש את התפריט של SERVER_MAIN_MENU (לאחר קבלת ההודעה המתאימה מהשרת):

   Choose what to do next:
   1. Play against another client
   2. Quit

7. במידה והמשתמש בחר באופציה 1 השרת יחפש עוד לקוח שבחר באופציה זו ויתחיל בין הלקוחות משחק. במידה והוא מוצא לקוח שכזה השרת שולח לשניהם הודעת SERVER_INVITE ומתחיל ביניהם משחק. במידה ואין לקוח כזה השרת ישלח SERVER_NO_OPPONENTS. יש להמתין לתשובה במשך 30 שניות זאת משום שהשרת עצמו ימתין 15 שניות כדי לראות אם לקוח כלשהו מתחבר ורוצה לשחק. במקרה שאין שחקן אחר הלקוח יציג שוב את התפריט הראשי.
   לאחר קבלת הודעת SERVER_INVITE הלקוח יציג למשתמש את ההודעה הבאה:

   Game is on!

8. במידה והמשתמש בחר באופציה 2 הלקוח ישלח לשרת הודעת CLIENT_DISCONNECT ויתנתק מהשרת. במקרה זה הלקוח יסיים את ריצתו לאחר שחרור כלל המשאבים שהקצה במהלך ריצתו.
9. בתחילת משחק מול לקוח אחר: עם קבלת הודעת SERVER_SETUP_REQUSET מהשרת, הלקוח יציג למשתמש את ההודעה הבאה:

Choose your 4 digits:

וימתין עד שהמשתמש יבחר 4 ספרות על ידי הקלדתן (ברצף, כמספר 4 ספרתי, לדוגמא 1234)

הלקוח ישלח לשרת את בחירת המשתמש בהודעת CLIENT_SETUP.

10. לאחר מכן עם קבלת הודעת SERVER_PLAYER_MOVE_REQUEST מהשרת, הלקוח יציג למשתמש את ההודעה הבאה:

Choose your guess:

וימתין עד שהמשתמש יבחר 4 ספרות. הניחוש ישלח לשרת בהודעת CLIENT_PLAYER_MOVE.

לאחר שליחת הניחוש הלקוח ימתין לתוצאות מהשרת (מספר הבולים והפגיעות) בהודעת SERVER_GAME_RESULTS. הלקוח יציג למשתמש את התוצאות באופן הבא:

Bulls: <bulls>
Cows: <cows>
<opponent_username> played: <opponent_move>

כאשר bulls הוא מספר ה"בולים", cows הוא מספר ה"פגיעות", opponent_username הוא שם השחקן היריב, opponent_move הוא ניחוש היריב.

11. כאשר יש מנצח למשחק, תתקבל מהשרת הודעת SERVER_WIN. הלקוח יציג למשתמש את המנצח באופן הבא:

<winner> won!
opponents number was <opponents_number>

כאשר winner הוא שם המנצח ו-opponents_number הוא רצף הספרות של היריב.

12. במקרה של תיקו תתקבל מהשרת הודעת SERVER_DRAW, ויוצג למשתמש:

It’s a tie

13. לאחר סיום המשחק תשלח שוב הודעת SERVER_MAIN_MENU (חוזרים לשלב 6).
14. במקרה של התנתקות לא צפויה של השחקן השני, תתקבל הודעת SERVER_OPPONENT_QUIT. יש להציג למשתמש הודעה מתאימה:

Opponent quit.

תוכנת השרת – מהלך ריצה מפורט

אלא אם נאמר אחרת ההמתנה להודעה מהלקוח היא 15 שניות. יוצאים מן הכלל הם התפריטים שבהם השרת ימתין ללא הגבלת זמן. באופן כללי בכל מקום שיש המתנה להחלטה של אדם יש לחכות ללא הגבלת זמן או זמן ארוך מאוד (10 דקות לפחות)

1. ה-thread הראשי של השרת יאזין לתקשורת נכנסת בפרוטוקול TCP על הפורט שצוין בארגומנט של שורת הפקודה.
2. השרת יבדוק באופן מחזורי אם נרשמה המחרוזת exit ב-console של השרת ואם כן ינסה לסגור את כל ה-thread-ים שיצר, לשחרר את כל המשאבים שהקצה ולצאת.
3. עם התחברות לקוח, השרת יקבל את ההתחברות וייצור thread חדש עבור אותו לקוח שיטפל בו.
4. ה-thread החדש שנוצר ממתין להודעת CLIENT_REQUEST. השרת ישמור את שם המשתמש שהלקוח שולח להמשך וישלח ללקוח הודעת SERVER_APPROVED. מעתה אלא אם נאמר אחרת, "השרת" מתייחס ל-thread שנוצר עבור אותו לקוח.
   כאשר לקוח מנסה להתחבר לשרת כאשר כבר יש שני לקוחות מחוברים השרת ידחה את בקשת התחברות השחקן. הדחיה תתבצע באמצעות הודעת SERVER_DENIED. ניתן להניח כי לא יתחברו יותר מ3 לקוחות בו זמנית (שניים משחקים ועוד אחד נדחה).
5. לאחר מכן השרת ישלח ללקוח הודעת SERVER_MAIN_MENU והלקוח יציג למשתמש את התפריט הראשי כפי שתואר קודם לכן. השרת ימתין (ללא הגבלת זמן) להחלטת הלקוח.
6. אם הלקוח בוחר להתנתק השרת יסיים (שוב, הכוונה ל-thread הספציפי שנוצר לאותו לקוח ולא לתכנית השרת כולה).
7. במידה והלקוח מעוניין לשחק מול לקוח אחר השרת צריך לבדוק אם יש עוד לקוח שרוצה לשחק מול לקוח אחר. עליכם לממש מנגנון תקשורת בין ה-thread-ים של שני הלקוחות (יפורט בהמשך).
   אם לא נמצא לקוח אחר לשחק מולו יש לשלוח הודעת SERVER_NO_OPPONENTS ואחריה הודעת SERVER_MAIN_MENU המחזירה לתפריט הראשי.
   לאחר שנמצא עוד שחקן השרת יבקש מכל אחד מהשחקנים את ארבעת המספרים אותם השחקן השני ינחש, בהודעת SERVER_SETUP_REQUSET, וימתין לתשובה משני השחקנים (בהודעת CLIENT_SETUP).
   לאחר קבלת התשובה ושמירת הרצף, השרת יבקש מכל אחד מהלקוחות את המהלך שלו בהודעת SERVER_PLAYER_MOVE_REQUEST וימתין לתשובה (בהודעת CLIENT_PLAYER_MOVE).
   השרת יחשב את תוצאות הסבב וישלח אותה ללקוחות בהודעת SERVER_GAME_RESULTS.
   במידה ויש מנצח למשחק (או תוצאת תיקו), השרת ישלח הודעת SERVER_WIN (או SERVER_DRAW), ולאחריה הודעת SERVER_MAIN_MENU המחזירה לתפריט הראשי.
8. במקרה של התנתקות לא צפויה של אחד הלקוחות, יש לשלוח ללקוח השני הודעת SERVER_OPPONENT_QUIT, ולאחריה הודעת SERVER_MAIN_MENU המחזירה לתפריט הראשי.

תקשורת בין thread-ים בשרת

כאשר שני לקוחות רוצים לשחק זה עם זה ה-thread-ים שמטפלים בהם בשרת צריכים לתקשר. התקשורת ביניהם תתבצע בעזרת קובץ זמני שיווצר עבור כל משחק וימחק בסופו.

אתם רשאים לעבוד עם קובץ זה כרצונכם על מנת להעביר מידע בין ה-thread-ים, כל עוד הפעולות מסונכרנות.

דרך פעולה אפשרית:

1. השרת יבדוק אם קיים קובץ GameSession.txt ואם לא אז ייצור אחד כזה (יש לסנכרן את היצירה והבדיקה עם mutex כדי שלא יהיה מצב שהקובץ יווצר פעמיים).
2. אם הוא קיים אז נדע שאפשר לשחק מול ה-thread שפתח אותו ונבקש מהלקוח את המהלך שלו
3. נרשום את המהלך לקובץ (יש לסנכרן את הפעולה עם mutex).
4. ה-thread השני יקרא את המהלך שלנו ויכתוב את שלו.
5. ה-thread שלנו יקרא את המהלך של היריב.
6. שני ה-thread-ים יחשבו את תוצאת המשחק וישלחו אותה ללקוחות
7. ה-thread שיצר את GameSession.txt ימחק אותו עם סיום המשחק.

ניתן להשתמש ב-Events כדי לסנכרן את פעולת ה-threads וכדי להמתין למשתמש השני.

ניתן לפתוח ולסגור את הקובץ בעבור כל כתיבה/קריאה, ובכך להמנע מפתיחת הקבצים באופן שיאפשר גישה במקביל, או לגשת במקביל בדומה לאופן בו ביצעתם זאת בתרגיל 2.

סיום התוכנה כאשר אין שגיאות

התוכנה תחזיר 0 אם הסתיימה ללא שגיאות.

כאשר אין שגיאות, התוכנה צריכה להסתיים באופן מסודר:

- אין לצאת מהתוכנה כאשר יש חוטים משניים פתוחים. יש לסמן להם סגירה ורק אם הם לא מסתיימים בעצם לאחר 15 שניות 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 תהליך, התוכנה תדפיס למסך הודעת שגיאה בעלת משמעות. היציאה מהתכנית תיעשה בצורה מסודרת ברמת ה-thread שבו השגיאה התרחשה – כלומר באותו thread שהשגיאה התרחשה יש לשחרר את כלל המשאבים שהקוצו כמיטב יכולתכם ולסיים את הריצה של שאר החוטים באמצעות TerminateThread. שימו לב שבמקרה של הצלחה כל החוטים צריכים לשחרר את המשאבים שלהם.

כתובות

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

אם תרצו, תוכלו להתחבר למחשב אחר. תוכלו לגלות מה כתובת ה-ip של מחשב על ידי הרצת הפקודה ipconfig ב-cmd באותו המחשב. בצעו ping ממחשב אחד לאחר, כדי לגלות אם הם רואים אחד את השני.

עבור מספר port, השתמשו במספר כלשהו בן 4 ספרות, למשל 8888.

ריבוי פרויקטים באותו solution

במודל יש מדריך, שמסביר איך לעבוד עם מספר פרויקטים באותו ה-solution. המדריך נמצא במדור How-to.

בהצלחה!