תרגיל בית 4

מועד הגשת התרגיל: עד 09.01.2022.

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"ב" 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 בת.ז. של המגישים. שמות הפרויקטים צריכים להיות client ו- server. שמות ה-executables צריכים להיות client.exe ו- server.exe בהתאמה.
- שם ה-zip שאתם מגישים צריך להיות Ex4_id1_id2.zip, כאשר id1/id2 מוחלפים בת.ז. של המגישים.
- שימו את הקבצים המשותפים לשני הפרויקטים בתיקייה בשם Share, שממוקמת ביחד עם תיקיות הפרויקט בתיקיית ה-solution.
- קמפלו את הקוד לגרסאת x86 Debug.
- 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.
- 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.
- 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-Studio Visual תוכלו לדבג במקביל גם את התכנית השנייה לאחר שהיא התחילה לרוץ: Debug -> process to Attach ולבחור את התהליך.
- 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.

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 Control Version 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 גרסת online למשחק הילדים האהוב והנדוש: שבע בום.

הסבר על המשחק ניתן לקרוא בויקיפדיה.

**__הערה__: למען הסר ספק נדגיש כי פסילה במשחק נגרמת במצבים הבאים:

1. המילה boom **לא** נאמרת כאשר המספר הצפוי הנוכחי הוא **כפולה של 7 או מכיל את הספרה 7.**
2. המילה boom נאמרת כאשר המספר הצפוי הנוכחי הוא **__לא__ כפולה של 7 וגם __לא__ מכיל את הספרה 7.**
3. כאשר המספר שנאמר הוא לא המספר הצפוי הנוכחי.

דוגמה ראשונה:	**דוגמה שנייה:**	**דוגמה שלישית:**
שחקן 1: 1	שחקן 1: 1	שחקן 1: 1
שחקן 2: 2	שחקן 2: 2	שחקן 2: 2
שחקן 1: 3	שחקן 1: 3	שחקן 1: 3
שחקן 2: **boom**	שחקן 2: **6**	שחקן 2: 4
		שחקן 1: 5
		שחקן 2: 6
		שחקן 1: **7**
*נאמר boom בזמן הלא נכון	*נאמר מספר שונה מהמספר הצפוי של התור	*לא נאמר boom במקום המתאים

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 ואת מצב הלוח. 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

- לא ניתן להניח שפתיחת הקבצים מצליחה תמיד.
- לא ניתן להניח שפתיחת ה-socket מצליחה תמיד.
- ניתן להניח ששם המשתמש מכיל רק אותיות ומספרים ללא רווחים. ניתן להניח אורך מקסימלי של 20 תווים.
- שם המשתמש ייחודי (לא יתחברו שני לקוחות עם אותו שם).

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, שולח את שמו לשרת.	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, שולח את שמו לשרת.	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
	לקוח 1	בוחר לשחק נגד שחקן אחר	CLIENT_VERSUS
	לקוח 2	בוחר לשחק נגד שחקן אחר	CLIENT_VERSUS
8	שרת	מודיע לשחקן 1 שהמשחק התחיל.	GAME_STARTED
	שרת	מודיע לשחקן 2 שהמשחק התחיל.	GAME_STARTED
9	שרת	מודיע לשחקן 2 שזה תורו של שחקן 1.	TURN_SWITCH
	שרת	מודיע לשחקן 1 שזה תורו.	TURN_SWITCH
10	שרת	מבקש משחקן 1 את המהלך שלו.	SERVER_MOVE_REQUEST
11	לקוח 1	משחק: מודיע לשרת באמצעות הודעה על המהלך שהוא מעוניין לבצע (הקלט שניתן, במהלך תקין יהיה מספר או boom).	CLIENT_PLAYER_MOVE
12	שרת	מודיע לשחקן 2 על מצב המשחק הנוכחי (מה השחקן 1 ביצע והאם הוא נפסל או שהמשחק ממשיך).	GAME_VIEW
13	שרת	מודיע לשחקן 1, שתורו של לקוח 2.	TURN_SWITCH
	שרת	מודיע לשחק 2 שזה תורו.	TURN_SWITCH
14	שרת	מבקש משחקן 2 את המהלך שלו.	SERVER_MOVE_REQUEST
15	לקוח 2	משחק: מודיע לשרת באמצעות הודעה על המהלך שהוא מעוניין לבצע (הקלט שניתן, במהלך תקין יהיה מספר או boom).	CLIENT_PLAYER_MOVE
16	שרת	מודיע לשחקן 1 על מצב המשחק הנוכחי (מה השחקן 2 ביצע והאם הוא נפסל או שהמשחק ממשיך).	GAME_VIEW
17	חזרה על שלבים 9-15 עד שנגמר המשחק (פסילה של אחד השחקנים)		
18	שרת	מודיע לשחקן 1 על סיום המשחק, ועל התוצאה.	GAME_ENDED
	שרת	מודיע לשחקן 2 על סיום המשחק, ועל התוצאה.	GAME_ENDED
19	שרת	מציג ללקוח 1 את האפשרויות הבאות: 1. לשחק נגד שחקן אחר 2. להתנתק מהשרת	SERVER_MAIN_MENU
	שרת	מציג ללקוח 2 את האפשרויות הבאות: 1. לשחק נגד שחקן אחר 2. להתנתק מהשרת	SERVER_MAIN_MENU
20	לקוח 1	בוחר לשחק נגד שחקן אחר	CLIENT_VERSUS

21	לקוח 2	בוחר להתנתק מהשרת	CLIENT_DISCONNECT
22	שרת	אין שחקנים אחרים לשחק מולם אז שולח ללקוח 1 שהוא לא הצליח למצוא מול מי לשחק	SERVER_NO_OPPONENTS
23	שרת	מציג לו (לקוח 1) את האפשרויות הבאות: 1. לשחק נגד שחקן אחר 2. להתנתק מהשרת	SERVER_MAIN_MENU
24	לקוח 1	בוחר להתנתק מהשרת	CLIENT_DISCONNECT
25	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. המערך אינו מחרוזת, משום שהוא אינו מסתיים בתו '\0', ורשאי להכיל '\0'.

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. אין להגדיר הודעות נוספות.

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_PLAYER_MOVE	תגובה להודעת SERVER_MOVE_REQUEST. ההודעה מציינת את מהלך השחקן.	-המהלך (מספר או boom).
	CLIENT_DISCONNECT	הלקוח מעוניין להתנתק.	
שרת	SERVER_APPROVED	השרת אישר את התחברותו של הלקוח	
	SERVER_DENIED	השרת דחה את התחברות הלקוח, נשלח כאשר יש 2 שחקנים במשחק נוכחי.	
	SERVER_MAIN_MENU	השרת רוצה שהלקוח יציג למשתמש את התפריט הראשי	
	GAME_STARTED	הודעת התחלת משחק.	
	TURN_SWITCH	הודעה המסמלת החלפת תור בין שחקן מסוים לשחקן הנגדי.	-שם השחקן שמקבל את התור

	SERVER_MOVE_REQUEST	בקשה לקבל את קלט מהלך השחקן.	
	GAME_ENDED	הודעת סיום משחק. ההודעה תציג איזה שחקן ניצח.	-השחקן שניצח
	SERVER_NO_OPPONENTS	השרת מודיע ללקוח שאין לקוחות הפנויים למשחק כרגע	
	GAME_VIEW	הודעה זו תציג את מהלך המשחק הנוכחי והאם המשחק ממשיך או שהשחקן הנוכחי נפסל.	-המהלך -האם המשחק נגמר
	SERVER_OPPONENT_QUIT	השרת מודיע ללקוח שהשחקן השני התנתק והמשחק מופסק	

דוגמא להודעה מסוג SERVER_MAIN_MENU:

"SERVER_MAIN_MENU"

דוגמא להודעה מסוג GAME_ENDED:

"GAME_ENDED:username_player "

דוגמה להודעה מסוג GAME_VIEW:

GAME_VIEW:<other player's name>;<the number or boom><END/CONT>

תיאור מפורט של התוכנה

הסעיף הזה מכיל את פרטי התוכנות, כולל ממשק המשתמש וטיפול בשגיאות. תוכנת הלקוח תפתח עבור כל לקוח בנפרד.

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

תוכנת הלקוח – מהלך ריצה מפורט

1. תוכנת הלקוח תתחבר לשרת בפרוטוקול TCP בכתובת שצוינה בארגומנטי הקלט.
2. לאחר חיבור מוצלח, תירשם השורה הבאה למסך ולקובץ הלוג (קובץ זה יפורט בהמשך):

   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

   ולקובץ הלוג יירשם:

   Failed connecting to server on <ip>:<port>.

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

   ולקובץ הלוג יירשם:

   Failed connecting to server on <ip>:<port>.

   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(לפי הזמן שהוגדר לעיל) 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 לכל אחד מהם הודעת GAME_STARTED 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 GAME_STARTED 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. במידה והמשחק התחיל, כל אחד מהשחקנים יקבל הודעת TURN_SWITCH המסמלת את התחלת התור של השחקן הראשון. אם התור הוא של השחקן הנוכחי, תוכנת הלקוח תציג את הפלט:

   Your turn!

   אם תורו של השחקן הנגדי, יוצג הפלט:

   <other player username>'s turn!
10. לאחר מכן עם קבלת הודעת SERVER_MOVE_REQUEST מהשרת, הלקוח יציג למשתמש את ההודעה הבאה:

    Enter the next number or boom:

    וימתין לקבלת הספרות או boom.

הניחוש יישלח לשרת בהודעת CLIENT_PLAYER_MOVE.

11. לאחר שליחת המספר לשרת, הלקוח שלא היה תורו יקבל את ההודעה GAME_VIEW מהשרת ויציג למשתמש את ההודעה הבאה:

    <other player's name> move was <the number he chose>
    <if the game ended>

    במידה והמשחק לא נגמר, ישלחו הודעות TURN_SWITCH אל 2 הלקוחות והלקוח שעכשיו תורו יקבל SERVER_MOVE_REQUEST וכך ימשיך המשחק.

במידה והמשחק נגמר, תתקבל הודעת GAME_ENDED ותוצג ההודעה הבאה:

<winner> won!

12. לאחר סיום המשחק תתקבל הודעת MENU_MAIN_SERVER (חוזרים לשלב 6).
13. במקרה של התנתקות לא צפויה של השחקן השני, תתקבל הודעת QUIT_OPPONENT_SERVER. יש להציג למשתמש הודעה מתאימה:

    Opponent quit .

תוכנת הלקוח – פרטים נוספים

1. 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. במידה והוכנס קלט לא תקין, תוכנת הלקוח תבקש שוב קלט מהמשתמש.
   שימו לב: בסעיף הזה לא מדובר על רק על שגיאות שמונעות את שליחת ההודעה. שגיאות הנוגעות לתוכן ההודעה מטופלות על ידי השרת.
2. כל הודעה שנשלחת לשרת, וכל הודעה שמתקבלת מהשרת, תתועד בקובץ הלוג (אבל לא תודפס למסך) בפורמט המתואר למטה בהסבר על קובץ הלוג.
3. כאשר החיבור לשרת מתנתק, תירשם למסך ולקובץ הלוג ההודעה הבאה:

   Server disconnected. Exiting.

   התוכנה תסיים את פעולתה באופן מידי.

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. השרת יאזין לתקשורת נכנסת בפרוטוקול TCP על הפורט שצוין בארגומנט הקלט.
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. השרת יתמוך במקסימום 2 לקוחות. לאחר שני לקוחות, הוא יסרב לחיבורים.
4. עם התחברות לקוח, השרת יקבל את ההתחברות וייצור thread חדש עבור אותו לקוח שיטפל בו.
5. 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(.
6. .לאחר מכן השרת ישלח ללקוח הודעת MENU_MAIN_SERVER .
   הלקוח יציג למשתמש את התפריט הראשי כפי שתואר קודם לכן. השרת ימתין (ללא הגבלת זמן) להחלטת הלקוח.
7. .אם הלקוח בוחר להתנתק השרת יסיים (שוב, הכוונה ל-thread הספציפי שנוצר לאותו לקוח ולא לתכנית השרת כולה).
8. במידה והלקוח מעוניין לשחק מול לקוח אחר (התקבלה הודעת CLIENT_VERSUS) השרת צריך לבדוק אם יש עוד לקוח שרוצה לשחק מול לקוח אחר. עליכם לממש מנגנון תקשורת בין ה-thread-ים של השרת המתקשרים עם התהליכים 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 ישלח TURN_SWITCH עם שם השחקן, וישלח SERVER_MOVE_REQUEST לאותו לקוח שעכשיו תורו.
   השרת יחכה למהלך השחקן (להודעת CLIENT_PLAYER_MOVE).
   לאחר מכן ישלח ל-2 הלקוחות GAME_VIEW, וישלח TURN_SWITCH לשני הלקוחות עם שם הלקוח שעכשיו תורו וכך חוזר כל מהלך המשחק.
   השרת ישלח GAME_ENDED ל-2 הלקוחות כאשר אחד השחקנים נפסל, ולאחר מכן ישלח ל-2 הלקוחות הודעת SERVER_MAIN_MENU.
9. במקרה של התנתקות לא צפויה של אחד הלקוחות, יש לשלוח ללקוח השני הודעת SERVER_OPPONENT_QUIT, ולאחריה הודעת SERVER_MAIN_MENU המחזירה לתפריט הראשי.

תוכנת השרת – פרטים נוספים

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

תקשורת בין thread-ים שונים של השרת

כאשר שני לקוחות רוצים לשחק זה עם זה ה-thread-ים שמטפלים בהם בשרת צריכים לתקשר.

מומלץ לפעול בצורה של משאב משותף (ומוגן) של מבנה נתונים שיכיל את מצב המשחק הנוכחי.

חשוב לדאוג שלא תהיה גישה לכתיבת המשאב בו זמנית, כפי שלמדתם במהלך הסמסטר.

קובץ לוג

הנחה

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

הודעות לוג

יש להדפיס כל הודעה המתקבלת מתכנית הלקוח אל חוט השרת המטפל בו או נשלחת מחוט השרת אל הלקוח שבו הוא מטפל לקובץ לוג ייחודי. סה"כ ייפתחו 4 קבצי לוג - קובץ לוג עבור כל לקוח, וקובץ לוג עבור כל חוט שמטפל בלקוח. קובץ הלוג שנוצר ע"י חוטי השרת ייקרא Thread_log_<the client name>.txt בהתאם לשם הלקוח בו הוא מטפל.

קובץ הלוג שנוצר ע"י תכנית הלקוח ייקרא Client_log_<the client name>.txt בהתאם לשם הלקוח בו הוא מטפל.

לדוגמה, אם חוט השרת מטפל בלקוח בשם yossi, קובץ הלוג ייקרא thread_log_yossi.txt.

לדוגמה, אם יש לקוח בשם yossi, קובץ הלוג ייקרא Client_log_yossi.txt.

פורמט ההודעות שהמודפסות לקובץ יהיה כך:

<sent/received from server>-<raw message>

לדוגמה:

sent from server-GAME_ENDED:yossi

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 בעצמם לאחר 15 שניות מותר להשתמש ב-TerminateThread על מנת לסגור את החוטים (זה תקף רק לתוכנת השרת).
- אין לצאת מהתוכנה כאשר יש handles פתוחים.
- אין לצאת מהתוכנה כאשר יש זיכרון דינאמי שלא שוחרר.
- 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 ולקובץ הלוג 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.

לאחר הדפסת הודעת השגיאה לקובץ הלוג, התוכנה שבה נגרמה השגיאה (לקוח/שרת) תסיים את פעולתה מיד.

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!