Installer

יום שני 23 דצמבר 2024 10:02

נממש installer ב++C שיבצע את המשימות הבאות:

- התוכנה מקבלת לקבל כקלט שם של תיקייה קיימת, שמות של שלושה קבצים בתוך התיקייה, ותיקייה חדשה שאליה צריך להעתיק אותם.
- במידה ופעולה כלשהיא נכשלת (לדוגמה אחד הקבצים לא קיים), על התוכנה למחוק את הקבצים החדשים שנוצרו.
- במידה והתיקייה שאליה מעתיקים את הקבצים לא קיימת, צריך ליצור אותה. במידה והתיקייה לא הייתה קיימת לפני, יש למחוק אותה במידה ויש כישלון.

ניצור פרוייקט חדש בVisual Studio 2019 ונממש את הפתרון מעל C++ 14.
ניצור קובץ Installer.h שיהווה קובץ הheader של המחלקה Installer שתייצג את הinstaller כמובן.
למחלקה יהיו 4 members:

- m_SourceDir - מחרוזת המייצגת את תיקיית המקור ממנה נעתיק את הקבצים.
- m_DestDir - מחרוזת המייצגת את תיקיית היעד אליה נעתיק את הקבצים.
- m_Files- וקטור מחרוזות המכיל את שמות הקבצים אותם נרצה להעתיק מתיקיית המקור לתיקיית היעד.
- m_DestDirCreated - בוליאני המציין אם היישום הינו זה שיצר את התיקייה, שימושי למקרה ונצטרך למחוק את התיקייה לאחר כישלון (באמור - יש למחוק את התיקייה במקרה של כישלון רק במידה והתיקייה לא הייתה קיימת לפני), מאותחל לfalse, ערכו ישתנה לtrue במידה והתהליך הנוכחי ייצור את התיקייה.

```
/// <summary>
/// Represents the source directory which installer copy files from
/// </summary>
string m_SourceDir;

/// <summary>
/// Represents the destination directory which installer copy files to
/// </summary>
string m_DestDir;

/// <summary>
/// Represents the files to copy from the source directory
/// </summary>
vector<string> m_Files;

/// <summary>
/// Indicates if the destination directory has been created by the current process
/// </summary>
bool m_DestDirCreated = false;
```

ובעת נסקור את הפונקציות של המחלקה -

- בנאי המחלקה - בונה אובייקט מטיפוס Installer.

```
// Constructor
Installer(const string& source, const string& destination, const vector<string>& fileList)
    : m_SourceDir(source), m_DestDir(destination), m_Files(fileList) {}
```

- הפונקציה שתבצע את תהליך ההתקנה (העתקה למעשה) שתואר מעלה - install.
  - זוהי הפונקציה הראשית של המחלקה.
  - שלבי פעולת הפונקציה:
    - שלב 1 : בדיקה אם תיקיית המקור קיימת.
    - שלב 2 : ניסיון לייצר תיקייה חדשה באמצעות קריאה לפונקציה מתאימה.
    - שלב 3 : העתקת קבצים באיטרציה על וקטור המחרוזות שמייצג את הקבצים להעתקה, באמצעות קריאה מתאימה לפונקציה שמעתיקה קובץ בודד.
    - במידה ואחד השלבים נכשל - יימחקו הקבצים שיצרנו בתיקיית היעד, ותיקיית היעד תימחק במידה ואינה הייתה קיימת טרם ריצת היישום.

```
/// <summary>
/// Main operation function, launching the installation process
/// </summary>
void install();
```

- פונקציה להעתקת קובץ בודד - copyFile
  - מחזירה ערך בוליאני כתלות בסטטוס ההצלחה של העתקת הקובץ שניתן כפרמטר.

```
/// <summary>
/// Copy a single file
/// </summary>
/// <param name="fileName">file name to copy</param>
/// <returns>copy operation success/failure</returns>
bool copyFile(const string& fileName);
```

- פונקציה ליצירת תיקייה חדשה - createDestDir
  - הפונקציה יוצרת תיקייה חדשה לפי ערכו של m_DestDir, התיקייה תיווצר במידה ואינה הייתה קיימת קודם.
  - מחזירה ערך בוליאני כתלות בסטטוס ההצלחה של יצירת התיקייה\קיום התיקייה.
  - במקרה של הצלחה חשוב להדגיש נקודה חשובה - ערך הבוליאני m_DestDirCreated יתעדכן לtrue רק אם התיקייה נוצרה כתוצאה מפעולת הinstaller.

```
/// <summary>
/// Create the destination folder
/// </summary>
/// <returns>directory creation success/failure</returns>
bool createDestDir();
```

- פונקציה למחיקת קובץ בודד - deleteFile

```
/// <summary>
/// Delete a single file
/// </summary>
/// <param name="fileName">file name to be deleted</param>
void deleteFile(string fileName);
```

- פונקציה למחיקת תיקיית היעד - deleteDestDir

```
/// <summary>
/// Delete the destination folder
/// </summary>
void deleteDestDir();
```

- פונקציה להמרת מחרוזת (string) להיות מחרוזת רחבה (wstring) - convertToWstring
  - מחזירה את הייצוג של המחרוזת שניתנת כפרמטר כמחרוזת רחבה.
  - תו של מחרוזת רחבה תופס יותר מקום בזיכרון מאשר "תו קלאסי" (תו קלאסי תופס 8 סיביות).
  - ההמרה נחוצה מאחר ומס' פונקציות מWINAPI כמו CreateDirectory, CopyFile, DeleteFile, RemoveDirectory מצפות לקבל LPCWSTR (Long Pointer Const Wide String) כפרמטר.
  - על מנת לקרוא באופן תקין לפונקציות האלו מWINAPI נצטרך להעביר wide string עם קריאה ל(c_str() - שמחזירה מצביע למערך שמכיל את התווים של הwide string שהעברנו.

```
/// <summary>
/// Function for casting a string to a wide string
/// </summary>
/// <param name="str">a string to be converted</param>
/// <returns>wide string represents the given string</returns>
wstring convertToWstring(const string& str);
```

- פונקציה לבדיקת קיום של תיקייה - isDirExists
  - מקבלת כפרמטר מחרוזת שמייצגת את נתיב התיקייה לבדיקת קיומה.
  - לשם בדיקת קיום של התיקייה, נאתחל DWORD שיציג לנו את מאפייני הקובץ (File Attributes) ע"י קריאה לפונקציה GetFileAttributes מWINAPI.
  - נרצה לבדוק אם הDWORD מכיל את הדגל שמציין שהHandle הנוכחי הינו תיקייה.

FILE_ATTRIBUTE_DIRECTORY 16 (0x00000010)	The handle that identifies a directory.

- במקרה של כישלון, הפונקציה תחזיר INVALID_FILE_ATTRIBUTES.
- לכן הבדיקה של קיום התיקייה צריכה להיות עפ"י הלולאה הבאה:

```
/// <summary>
/// checks directory existence status
/// </summary>
/// <param name="filename">file's name</param>
/// <returns></returns>
bool Installer::isDirExists(const string& fileName) {
    wstring file = convertToWstring(fileName);
    // Retrieves attributes for a specified file or directory
    DWORD fileAttributes = GetFileAttributes(file.c_str());
    // Checking if file attributes contains the flag of FILE_ATTRIBUTE_DIRECTORY
    return (fileAttributes != INVALID_FILE_ATTRIBUTES && (fileAttributes & FILE_ATTRIBUTE_DIRECTORY));
}
```

- נחזיר true במקרה זה שכן התיקייה קיימת.

- מבחינת Access Modifiers -
  - נרצה לייחצן את הבנאי ואת הפונקציה הראשית install בלבד, ולכן הן יהיו תחת public.
  - שאר הפונקציות שהוגדרו הינן לצורך שימוש פנימי בלבד, install קוראת לכל אחת מהן בשלב הרלוונטי בתהליך, ואין לנו עניין לייחצן אותן למשתמש, ולכן הן יהיו תחת private.
- ניצור קובץ Installer.cpp בו נממש את כלל הפונקציות שהצהרנו עליהן בקובץ הheader.
  - מבחינת פרטי מימוש של copyFile, deleteFile, deleteDestDir :
    - הלוגיקה די דומה במהותה - המרת הפרמטרים לwstring מתאימ\ים על מנת לקרוא לפונקציה הרלוונטית ב WINAPI מיד לאחר מכן:
      - עבור copyFile נקרא לCopyFile.
      - עבור deleteFile נקרא לDeleteFile.
      - עבור deleteDestDir נקרא לRemoveDirecory.
      - במובן שבמקרה של כשלון תירשם השגיאה לcerr בכל אחד מן הפונקציות.
  - createDestDir -
    - נרצה לייצר תיקייה חדשה בנתיב של m_DestDir, במידה ואינה קיימת כזאת.
    - אחרת, ננסה ליצור תיקייה חדשה -
      - במידה והיצירה נכשלה - נרשום לcerr את השגיאה והפוקנציה תחזיר false.
      - אחרת, נעדכן את הבוליאני m_DestDirCreated להיות true ונחזיר true.
        - עדכון ערך הבוליאני קריטי לשם שאלת מחיקת תיקיית היעד במקרה של כישלון!

```
if (!CreateDirectory(wideStrDestDir.c_str(), nullptr)) {
    // The directory creation has been failed
    cerr << "Failed to create directory:" << m_DestDir << " Error:" << GetLastError() << endl;
    return false;
}

m_DestDirCreated = true;
return true;
```

  - convertToWstring - מעבר איטרטיבי על כל תו מחרוזת כדי להפוך את המחרוזת מטיפוס string לwstring.

```
/// <summary>
/// Function for casting a string to a wide string
/// </summary>
/// <param name="str">a string to be converted</param>
/// <returns>wide string represents the given string</returns>
wstring Installer::convertToWstring(const string& str) {
    return wstring(str.begin(), str.end());
}
```

  - ניצור קובץ main.cpp בו תרוץ הפונקציה (main() - נקלוט את כלל הארגומנטים, ובמידה ולא נקבל את מס' הארגומנטים המצופה נדפיס הודעה שתציג למשתמש את

הפורמט הנדרש להעברת הארגומנטים - באמצעות קריאה לפונקציה PrintUsage.

```
if (argc != 6) {
    printUsage();
    return 1;
}
```

```
void printUsage() {
    cerr << "Usage: Installer.exe <sourceDir> <destDir> <file1> <file2> <file3>" << endl;
    cerr << "Example: Installer.exe C:\\Source C:\\Dest file1.txt file2.txt file3.txt" << endl;
}
```

- במידה וקיבלנו את הפורמט התקין, נכניס את הארגומנטים למקום המתאים להם (נתיב לתיקיות המקור והיעד למחרוזת שרלוונטית להן, ואת שמות הקבצים לוקטור המחרוזות). לאחר מכן, ניצור installer חדש.

```
string sourceDir = argv[1];
string destDir = argv[2];
vector<string> files = { argv[3], argv[4], argv[5]};

Installer installer(sourceDir, destDir, files);
```

- לאחר מכן, נפעיל את install על הinstaller שיצרנו כדי לבצע את תהליך העתקת הקבצים, נעטוף את הקריאה בtry על מנת להתריע על חריגה במידה ותתרחש שגיאה במהלך תהליך ההתקנה.

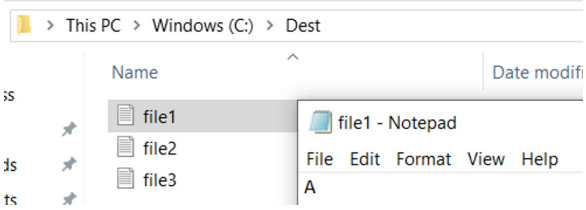
```
try {
    installer.install();
} catch (exception& ex) {
    cerr << "Error during installation: " << ex.what() << endl;
    return 1;
}

return 0;
```

- ערך החזרה 1 יציין שגיאה בתהליך ההתקנה, ערך ההחזרה 0 יציין שתהליך ההתקנה יתבצע באופן תקין.

- ננסה להריץ כעת דרך שורת הפקודה את היישום, אבל לפני כן, ניצור תיקייה בנתיב C:\\Source שתכיל 3 קבצים :
  - file1.txt, file2.txt, file3.txt
  - בכל הקבצים הכנסתי את התו 'A' כתוכן.
- נתיב היעד שנרצה להעתיק אליו את 3 הקבצים הוא C:\\Destination.
- נריץ בשורת הפקודה ונקבל פלט שפעולת ההעתקה של 3 הקבצים מתיקיית המקור לתיקיית היעד הצליחה!

```
C:\Users\Hanich\source\repos\Installer\x64\Debug>Installer.exe C:\\Source C:\\Destination file1.txt file2.txt file3.txt
Creating new directory at path C:\\Destination
Copying file1.txt..
Copying file1.txt Success!
Copying file2.txt..
Copying file2.txt Success!
Copying file3.txt..
Copying file3.txt Success!
Copy files operation completed!
```

- נוודא זאת:



כל הקבצים הועתקו עם התוכן המצופה להם!

- מקרים נוספים שנבדקו:
  - ניסיון העתקת קבצים קיימים בתיקיית המקור אל תיקייה קיימת - הצלחה בהעתקה.
  - העברת כמות פרמטרים שאינה מספיקה - הדפסת הודעה מתאימה.

```
C:\Users\Hanich\source\repos\Installer\x64\Debug>Installer.exe
Usage: Installer.exe <sourceDir> <destDir> <file1> <file2> <file3>
Example: Installer.exe C:\Source C:\Dest file1.txt file2.txt file3.txt
```

  - ניסיון העתקת קבצים מתיקיית מקור שאינה קיימת - הדפסת הודעה שמתריעה על כך.

```
C:\Users\Hanich\source\repos\Installer\x64\Debug>Installer.exe C:\\BLA C:\\Destination file1.txt file2.txt file3.txt
Installation failed: Source directory doesn't exists at path C:\\BLA
Cleanup - Deletion of copied files
```

  - נמחק מתיקיית המקור את file2.txt וכעת נבצע ניסיון העתקת קבצים שאינם קיימים בתיקיית המקור אל תיקייה קיימת - מחיקת הקבצים שנוצרו בתיקיית היעד, תיקיית היעד אינה נמחקת.

```
C:\Users\Hanich\source\repos\Installer\x64\Debug>Installer.exe C:\\Source C:\\Destination file1.txt file2.txt file3.txt
Copying file1.txt..
Copying file1.txt Success!
Copying file2.txt..
Installation failed: Failed to copy file: file2.txt
Error: 2

Cleanup - Deletion of copied files
```

  - ניסיון העתקת קבצים שאינם קיימים בתיקיית המקור אל תיקייה חדשה - מחיקת הקבצים שנוצרו בתיקיית היעד, תיקיית היעד נמחקת.

```
C:\Users\Hanich\source\repos\Installer\x64\Debug>Installer.exe C:\\Source C:\\Desti file1.txt file2.txt file3.txt
Creating new directory at path C:\\Desti
Copying file1.txt..
Copying file1.txt Success!
Copying file2.txt..
Installation failed: Failed to copy file: file2.txt
Error: 2

Cleanup - Deletion of copied files
Cleanup - Deletion of destination file at path C:\\Desti
```

- שינוי סדר בהעברת הפרמטרים כל שאופן העברת הפרמטרים אינו תקין - כשל בהעתקת הקבצים, כמצופה.

```
C:\Users\Hanich\source\repos\Installer\x64\Debug>Installer.exe C:\\Source file1.txt C:\\Destination file2.txt file3.txt
Creating new directory at path file1.txt
Copying C:\\Destination..
Installation failed: Failed to copy file: C:\\Destination
Error: 123

Cleanup - Deletion of copied files
Cleanup - Deletion of destination file at path file1.txt
```

- ניסיון העתקה חוזר של אותם קבצים לאותה תיקיית יעד (עם אותו התוכן) - העתקה מצליחה.

```
C:\Users\Hanich\source\repos\Installer\x64\Debug>Installer.exe C:\\Source C:\\Destination file1.txt file2.txt file3.tx
Copying file1.txt..
Copying file1.txt Success!
Copying file2.txt..
Copying file2.txt Success!
Copying file3.txt..
Copying file3.txt Success!
Copy files operation completed!
```

- ניסיון העתקה חוזר של אותם קבצים לאותה תיקיית יעד (עם תוכן קובץ\קבצים שונה) - העתקה מצליחה, תוכן הקבצים בתיקיית היעד מתעדכן.

```
C:\Users\Hanich\source\repos\Installer\x64\Debug>Installer.exe C:\\Source C:\\Destination file1.txt file2.txt file3.tx
Copying file1.txt..
Copying file1.txt Success!
Copying file2.txt..
Copying file2.txt Success!
Copying file3.txt..
Copying file3.txt Success!
Copy files operation completed!
```

- חריגות - במימוש הפונקציות הפנימיות (Access modifier = private) בInstaller.cpp כל חריגה פוטנציאלית נזרקה בשגיאת זמן ריצה באופן הבא:

```
throw runtime_error("Failed to create directory: " + m_DestDir + "\nError: " + to_string(GetLastError()) + "\n");
```

```
throw runtime_error("Source destination directory doesn't exists at path " + m_SourceDir);
```

כך באשר יש שגיאה בהתקנה בmain.cpp ונגיע לcatch ונרצה ליידע את המשתמש אודות החריגה באמצעות קריאה לפונקציה what על האובייקט מטיפוס exception אז המשתמש יקבל את המידע האינפורמטיבי אודות סיבת כישלון ההתקנה.

```
try {
    installer.install();
} catch (exception& ex) {
    cerr << "Error during installation: " << ex.what() << endl;
    return 1;
}
```

- קצת העשרה - מה הייתי מממש שונה בגרסאות מתקדמות יותר של C++?
  - ב17 C++ יצאה לאור הספריה FileSystem שמבצעת פעולות על מערכות קבצים והרכיבים שלהם, כמו נתיבים, קבצים ותיקיות.
    - אחת המתודות שהספריה מספקת היא is_directory - שבודקת אם הנתיב שניתן כפרמטר מייצג תיקייה.
    - במקרה כזה, ניתן היה לחסוך את המימוש של isDirExists ולקרוא ישירות לis_directory.