מדריך מהיר לתכנות בסביבת עבודה המגניבה
Borland C++ builder

מדריך זה נועד לאנשים שכבר יודעים לתכנת ב C++ אין כאן שום כוונה ללמד את השפה או להכין מטעמים מחרקים – לשם כך יש אלפי מדריכים אחרים ברשת.

המדריך יחולק לחלקים בהם אתמקד כל פעם בנושא אחר ואדגים אותו בצורה הכי פשוטה שאפשר (אפילו בלונדיות יבינו) , על הקוראים היקרים לעבוד במקביל... לקרוא את המדריך ולבצע את הוראות הזימה על סביבת העבודה כך שלבסוף תכירו אותה אינטימית כמעט כמוני.

כמו שבטח הבנתם , אין לי מצב רוח לכתוב בצורה "אקדמאית" יבשה , ואני מתכוון להוסיף את דעתי הממש לא אובייקטיבית , אך כל מה שקשור לחומר ייכתב בצורה הכי טובה שאני יכול מכיר על מנת לחסוך ממכם את הדם , זיעה ושנים של בידוד נפשי שעברתי כדי להבין איך הכי טוב לעבוד עם התוכנה.

נושאים לפי פרקים :

המדריך נכתב על ידי **עדי** - la_adi@hotmail.com.

הפעם הראשונה הכי מיוחדת

מטרת פרק זה היא להתחיל לעבוד בסביבת העבודה , ולכן כמו כל מדריך טוב נרשום את התוכנית המפורסמת ביותר בעולם "hello world" אך הפעם במקום הטקסט המשעמם נשתמש במשפט יותר מגניב ... נניח "ADi is the king!!" ... –לא לקחתי כדורים נגד מגלומניות היום...

מושגים בסיסים :

טופס – הדבר הגדול עם הנקודות שבמרכז המסך .

רכיב – אובייקט שנימצא בראש המסך , מחולקים לפי קטגוריות .

לעבודה :

1. הפעילו את הסביבה – אם כבר יש פרויקט פתוח תבחרו file->new application.

2. מתחת לתפריט ניתן להבחין ברשימת הרכיבים שניתן להוסיף לטופס הפתוח לכם במרכז החלון – כפי שניתן לראות תחת קטגורית Standard יש כפתור חמוד שכתוב עליו OK.
אל תבחרו אותו הוא מכוער !!!! עדיף לעבוד אם הכפתור שנמצא תחת Additional שגם עליו כתוב OK בצורף סימון V ... ביחרו אותו ותלחצו על הטופס.

3. תמונת העולם צריכה להיות טופס עם נקודות וכפתור שכתורתו BitBtn1 ... אם לא תסגרו את המחשב ותלכו למכור שניצלים .

4. לחיצה כפולה על הכפתור תוביל אתכם ישירות אל עורך הטקסט שם נרשום את פקודות התוכנית .

5. נא להקיש ללא שגיאות , את הפקודה הבאה ... כמובן העתק והדבק תמיד יהיה להיט ויכול לשמש אותנו גם כאן ; ("ShowMessage("ADi hate office")

6. ללחוץ על הרץ – F9 או כפתור ה PLAY הירוק שנראה כמו חץ מלא בראש המסך.

7. לאחר קומפילציה וקישור יופיע טופס שכל לחיצה על הכפתור תוביל להצגת הודעה על המסך .... ילדים אל תנסו את זה בבית !

8. כדי לצאת יפה יפה מהתוכנית תלחצו על סימן הבניין בפינה השמאלית עליונה ותבחרו בסגירה.

9. זהו !

היה פשוט... חגרו חגורות בטיחות אנו הולכים להמריא לעולם הנפלא של תוויות....

1. על אותו פרויקט המשיכו ועשו את הדברים הבאים.

2. הוסיפו עוד כפתור כמו מקודם . שמו יהיה בצורה טבעית BitBtn2

3. הוסיפו תיבת טקסט , נמצאת בקטגורית Standard ונראת כמו תיבת טקסט... שבה רשום ab (קיצור של ABA)

4. ועכשיו לרגע שכולם חיכו לו .... הוסיפו תווית , זה רכיב שכתוב עליו A .

5. מחלקה 3 להתפקד : 2 כפתורים מגניבים , תיבת טקסט אחת ותווית אחת.

6. לחצו לחיצה כפולה על הכפתור השני ובעורך הפקודות רשמו נא :
Label1->Caption = Edit1->Text;

7. כעת הריצו את התוכנית וקסם יקרה : על מה שתקלידו בתיבת הטקסט , לאחר הלחיצה על הכפתור השני יופיע כטקסט בתווית ... פשוט מדהים
8. זהו

הסבר לקוד : Caption זוהי תכונה של הרכיב תווית , Text זוהי תכונה של הרכיב Edit1. פשוט ביצענו השמת מחרוזות מתיבת הטקסט לתוך התווית . מחרוזת היא לא כמו ב C אלא אובייקט די מתוחכם (יוסבר עליו בהמשך) ולכן ניתן לבצע העתקה בעזרת אופרטור =

לסיום ההדגמה – כמו ששמתם לב הטופס ממש מכוער ולכן ניקח אותו לטיפול יופי אישי
1. לחצו במקום כלשהו בטופס - לא על רכיב כלשהו.

2. בצד שמאל יש את טבלת המאפיינים של הטופס . אם אתם לא רואים אותה לחצו על F11.

3. 3 . חפשו את המאפיין של כתב או בעברית FONT , אם יש לידו + אז לחצו על ה+ כדי לפתוח לרווחה את מאפייני הכתב. כעת שנו את הצבע , הגודל סוג הכתב כרצונכם . התפרעו רק חשוב לזכור שצבע עיני הוא כחול לכן על מנת להתחנף אלי נא לבחור בכחול.

4. 4 . כעת שנו את ה Caption של כל הרכיבים ... אני לא מפרט אתם ילדים גדולים ולכן תסתדרו (טיפ כדי לבחור ברכיב לעדכון פושט לחצו עליו בעזרת העכבר).

5. זהו

הסבר : מכוון שברירת המחדל של הרכיבים היא להתנהג כמו קופים , הם יקבלו את צורת האב שלהם כלומר מאפייני האב כמו כתב וצורתו יעברו מהאב לבנים....למה זה טוב ?? כך אפשר לשנות הרבה מאפיינם במכה אחת ולתת צורה אחידה לפרויקט .
לא טוב לכם ?? תבטלו, לרכיבים יש מאפיין בשם ParentFont קבעו אותו לשלילי...

הערות אחרונות :

- הגעתם לכאן ... כל הכבוד מקווה שהיה סבבה.
- זו הפעם הראשונה ואחרונה שעובדים ללא שמות משמעותים לאובייטים .
- כדי לתת שם לרכיב (כמה צפוי) יש לו מאפיין (PROPERTY) בשם NAME שנו אותו.
- יש כפתור חביב בשם SAVE ALL השתמשו בו בלי בושה
- הפרויקט שיצרתי ישמר במחיצה זו - אם היו בעיות עינו בו.
- העזרה של התוכנה ממש טובה אל תתביישו להשתמש בה על ידי לחיצה על F1
- השתעשעו בפרויקט כמו תאילנדי שהולך לאכול כלב.... מיתחו אותו , הוסיפו כפתורים או רכיבים אחרים (תראו מה הם עושים בעזרה) , רצוי לחממם על אש קטנה.
- הכי חשוב ספרו לכל החברים המסכנים שעובדים ב MFC כמה קל לכתוב תוכנית בעזרת BCB וצחקו להם בפנים .
- בדיחה : הייתה לי חברה שדיברה אלי בבינארית , כל הזמן אמרה לי אפס אחד אפס אחד...

סיכום :

- פונקציה להצגת הודעה פשוטה ShowMessage מקבלת מחרוזת ומציגה אותה על המסך בצורת הודעה , טוב בעיקר בזמן בדיקת התוכנית כשעושים DEBUG ורוצים להציג הודעות בסגנון ...עד כאן הכל בסדר .
- יש אובייקט מרכזי בשם טופס , הוא יכול להכיל רכיבים כמו כפתורים ותיבות טקסט ,
- לכולם יש מאפיינים שניתן לשנות אותם בזמן פיתוח התוכנית וגם בעזרת קוד
- כל הרכיבים בפרויקט יופיעו בצורת מצביעים – סיבה היסטורית - הסביבה התפתחה מדלפי ששם כל האובייקטים מוקצים דינאמית.
- לחיצה כפולה על רכיב שנמצא על הטופס יוביל למקום בו תיכתב הפונקציה המתאימה לאחד האירועים שהאובייקט יכול לקבל... הסברים נוספים בהמשך הפרקים.
- Parent – לכל רכיב יש הורה כך שבמידה ולא נאמר אחרת ישפיע על צורתו של הרכיב וזאת בנוסף לowner שאחראי לשחרור הזיכרון של הרכיב בסיום הפעולה (סתם לידע כללי בינתיים)
- חשוב לעבוד נכון : לתת שמות משמעותים , לשמור הרבה , לעשות גיבויים ולדאוג שהמשתמש יקבל טופס אסתטי שנוח לעבוד איתו .

פרטים טכנים

בחלק זה אסביר בקצרה מספר נושאים הקשורים לתוכנה , נבין מי נגד מי למה וכמה.
אנשי בורלנד המציאו לנו כמה מוסגים חדשים ומילות מפתח שאותם יש לשנן (הכתבה בעוד יומיים) וגם אותם אסביר . אל תשבו על הפרק הזה יותר מדי זמן – הוא מבוא להמשך , רצוי לחזור ולקרוא אותו שוב לאחר סיום קריאת המדריך כולו.

יש הרבה נושאים בפרק זה אז גם כאן יכנס אינדקס להקלת השימוש:

1. CodeGuard
2. קומפילציה .
3. המחשב כנביא – השלמת קוד.
4. סימניות .
5. הסייר האמיץ
6. קצת מסביב.
7. מילים חדשות לשפה .

CodeGuard - נושא זה הוא כה חשוב שהחלטתי לפתוח בו. סביבת העבודה מאפשרת לנו לשלב בפרויקט מעיין שומר , ולפי השם קל לראות שהבחורציק שומר על הקוד.
השאלה היא : מפני **מי** הוא שומר על הקוד ? והתשובה רבותיי וגבירותיי ממכם !
המתכנתים המוכשרים יותר או פחות... אני מודה ומתוודה שהשומר הציל את עורי מספר לא מבוטל של פעמים .
על ידי כך שמשלבים אותו בפרויקט (בקרוב אסביר איך) יחד עם התוכנית משתלבים מספר DLL שמוודאיים שאין כל מני בעיות כמו : כתיבה לזיכרון שלא הוקצע דינאמית , שימוש לא נכון בפונקציות ואי שחרור זיכרון בסיום התוכנית .
דוגמאות :

1.
```
int *vec
vec = new int [10];
delete vec;
```

2.
```
Node *ptr;
ptr = NULL;
ptr = ptr->data;
```

- במקרה הראשון לאופרטור delete שהוא בעצם כידוע פונקציה נשלך מצביע לint במקום מצביע למערך של int (הדרך הנכונה היא לשים [] לפני vec) , המגן יודיע לנו על כך.
- במקרה השני יש מצביע NULL שנמצא בצד שמאל של הקוד – בלגן .

את פעולתו נראה בצורת "חריג" שייזרק אלינו כמו אבן על אוטו שנוסע ביום שבת במאה שערים.

איך מפעילים את המגן: מהתפריט project -> options -> codeguard ולסמן את האפשרויות ב V חמוד.

הערות :

- לא לשכוח בסוף לפני הגרסה האחרונה להוריד את המגן שתופס הרבה זיכרון ומאט את התוכנית מאוד.
- יוצר קבצי clg שבהם מפרט את הבעיות שנתקלנו במהלך התוכנית , מה לא שוחרר , אילו פונקציות נקראות וכמה פעמים.
- אל תתביישו לעצור ברגע זה ולנסות לראות אם המגן שלכם לא יושן בשמירה.

בדיחה : מה זו בלונדינית אם פיאה ? בינה מלאכותית.....

קומפילציה : ברירת המחדל בגרסה 5 היא לבצע קומפילציה ברקע , ואין הכוונה לרקע של שקיעת החמה . פרוש הדבר שהמחשב יבצע קומפילציה בזמן שאתם ממשיכים לעבוד על הפרויקט , נשמע מגניב ואני חושב כמה קשה הם עבדו כדי לתת לנו את האופציה הזו **אבל** אם נחשוב על זה כמה זמן כבר לוקחת קומפלציה ... ובמקרה הגרוע של יותר מדקה הולכים לשתות קפה... מניסיוני זה סתם לא מועיל ואפילו קצת מפריע לכן אמליץ בחום להוריד את השימוש בקומפלציה ברקע - אלא אם כן באמת לוקח לכם הרבה זמן לקמפל .
איך מגיעים לשנות את האפשרות הזו : Tools -> Environment options
ואם אתם כבר שם הקדישו דקה או שתים מזמנכם היקר ועברו על יתר האפשרויות ,חלקן יעשו לכם את החיים הרבה יותר קלים .
אני עובד בגישה של בדיקת הקוד בחלקים קטנים , אני כותב משהו מוודא שעובד ואז ממשיך קדימה ...לכן זמני הקומפילציה שלי קטנים , חוץ מהמקרה שאני מאחד את החלקים הקטנים לפרויקט ואז בזמן הקומפלציה אני מדבר בטלפון עם כל העולם ואישתו.

בדיחה : בשבט קניבלים הבן אומר לאביו , אבא אני לא אוהב את אמא... אז האב אומר טוב בני תאכל רק את הירקות.

המחשב כנביא - כל מי שכתב תוכניות עם עורכי טקסט לא מתקופת ימי הביניים (כדוגמת EDIT של דוס הזכור לטוב) מכיר את האפשרות של השלמת קוד אוטומטית, כאילו המחשב יודע מראש איזה שטויות אנו הולכים לתקתק . גם כאן יש השלמת קוד אוטומטית שמופעלת על ידי לחיצה על { . או -> או רווח -ctrl} ונותנת רשימה של ביטוים חוקיים אותם אפשר לכתוב אחרי.

דוגמאות :

- לחיצת . אחרי שם של משתנה אובייקט תיתן את רשימת השדות (משתנים ופונקציות) שניתן להפעיל מאותו אובייקט.
- לחיצת חץ כלומר -> אחרי שם של מצביע לאובייקט ייתן אותו דבר.
- לחיצת רווח ו ctrl ייתן את רשימת המשתנים (כולל גלובלים) ופונקציות שניתן לכתוב בתוכנית.

כדי להגיע למאפייני אפשרות זו נא לגשת ל:
Tools->Editor Options ->Code insight

באותו מקום יש אפשרות לבקש עוד כמה אפשרויות מגניבות כמו השלמת פרמטרים , הערכת ביטויים (פרוש בעברית :כמה זה שווה??) ואפשרות לקבוע אחרי כמה זמן תופיע ההשלמה.

אין טוב בלי רע (המכוער תמיד מסתובב באזור) וגם לאפשרות זו יש **חסרונות** :

- תופס הרבה זיכרון ולכן אם המחשב שלכם סובל מבעיות זיכרון שקלו לבטל אפשרות זו.
- אם יש טעות בקוד שלכם , לא תתבצע השלמת הקוד. הטעות יכולה להיות כמה שורות לפני המקום שבו מבקשים את ההשלמה .
- הופך אתכם לעצלנים – התפקיד שלנו לתקתק אז תקתקו !!!

זהו – נא לגשת לשחק עם אפשרות זו - רק תהיו עדינים בבקשה.

בדיחה : יש 10 סוגים של אנשים בעולם , אילו שמבינים בינארית ואילו שלא.

סימניות

פעם הסמל בטירונות אמר לי : "סימנתי אותך" ובגלל זה לא יצאתי שבת – מאותו רגע לא אהבתי סימניות עד אשר נתקלתי בסימניות של עורך טקסט התוכנית ....
בתוכניות ארוכות יש נטייה ללכת לאיבוד , במיוחד אם לא כתבתם את הטקסט או שחוזרים אליו אחרי כמה ימים... יש כמה אפשרויות לפתור את הבעיה שהקלה ביניהם לדעתי היא סימון נקודות שאותם עובדים בעזרת סימניות.
לסמן נקודה : ללחוץ בימין עכבר על השורה אותה רוצים לסמן ולבחור מהתפריט את Toggle bookmarks ואז מספר .
לחזור לנקודה : ללחוץ ימין עכבר בעורך הטקסט ולבחור ב goto bookmarks ואת הנקודה אליה רוצים לחזור.

הסייר האמיץ

לצד הקוד שאותו כותבים ניתן לראות גם את סייר הפרויקט. אם אתם לא רואים אותו אז ניתן ללחוץ ימין עכבר בחלון הקוד ואז לבחור view explorer .

תפקידיו בחיים :

- בעזרתו ניתן לגשת להגדרות וקוד של פונקציות .
- להוסיף שדות , פונקציות מחלקה או מאפיינים לאובייקטים בפרויקט.
- לראות את ההירכיה של האובייקטים בפרויקט.

כדי להגיע לכל האפשרויות הנ"ל פשוט ללחוץ ימין עכבר ולבחור מתפריט. באפשרות של הוספת משתנים , פונקציות ומאפיינים לאובייקט יש אפשרות לבחור איזה טיפוס , הרשאות וכו וכו... בקיצור מאוד שימושי... רצוי להכיר .

בדיחה : מה ההבדל ביו עורך דין לאלוהים ? אלוהים לא חושב שהוא עורך דין.

קצת מסביב – (מה יש מולך? מסך , מאחורה? קיר)

כעת הסביר על 2 אוביקטיים עיקרים בסביבת העבודה :

- object inspector – כל אובייקט ויזואלי(נקרא גם רכיב או פקד) שנעבוד איתו ניתן לשנות לו את המאפיינים תוך כדי עיצוב התוכנית וגם בזמן הריצה. זה תפקידו של ה – object inspector , שם משנים את המאפיינים ובכרטיסיית האירועים אפשר לבחור אילו אירועים תתבצע פונקצית תגובה. אם הוא נעלם לכם מהמסך אפשר לקרוא לו לחזור בעזרת F11 .
- ליד התפריטים יש את הכרטיסיות בהם מוצגים הרכיבים שאותם אפשר להוסיף לפרויקט , יש שם רכיבים סטנדרטים , פקדי אינטרנט ואופיס ועוד . בעזרתם נעצב את הממשק משתמש של התוכנית – גשו וטילו שם ....

ילד : אמא מתי נגיע לאבא ? אמא :שתוק בני ותמשיך לחפור

מילים חדשות

יש מספר תוספות לשפה (ציפס סלט וכו) שהעיקריות הן :

- **__published** - בגלל המעבר לממשק גרפי נוצרה לחברת בורלנד בעיה – איך מייצגים אובייקט שאת ערך השדות שלו (כמו הכותרת של טופס) ניתן לשנות גם בזמן ריצה וגם בזמן כתיבת התוכנית –כלומר אובייקט בעל מאפיינים ויזואלים. הפתרון הוא להוסיף את מחלקת __published אשר כל מה שמופיע בה יוצג ב object inspector . ההרשאות של מחלקה זו הן כמו של public , ונכון להיום אין לכם מה לנגוע בכתוב שם – מטופל לגמרי לבד על ידי סביבת העבודה.
- **__fastcall** - מילה זו מופיע לפני פונקציות ופרושה הוא להעביר את הפרמטרים של הפונקציה דרך האוגרים – אם אפשר. יותר יעיל מאשר לעבוד עם המחסנית . כל פונקציה שמתרחשת כתוצאה של אירוע (לחיצת עכבר גרירה וכו,) חייבת להיות מוגדרת בעזרת **__fastcall** ותחת הכותרת **__published .**

<u>דוגמה</u> :

ניצור כפתור בפרויקט שיגיב לפונקציה שאנו כתבנו :

1. פתח או פתחי פרויקט חדש והכניסו לו כפתור – מי שלא יודע איך , יגש למוכר הבוטנים הקרוב – בטוח יש לו הסבר.
2. גשו לקובץ H של הטופס ורשמו תחת הכותרת **__published** את השורה הבאה :

void __fastcall WeDidIt (TObject *Sender);

3. עכשיו בחרו בכפתור מהטופס וב object inspector בחרו בכרטיסיית ה events ועבור כל אירוע כעת ניתן לבחור לביצוע את הפונקציה ששמה בישראל הוא WeDidIt.

למה זה טוב?? אם לדוגמה רוצים שמספר כפתורים יפעילו את אותה פונקציה , על נושא זה ארחיב בעתיד.

זו הייתה פחות או יותר סקירה על סביבת העבודה – מקווה שהיה כיף אותי באופן אישי היה קצת משעמם – אבל העיקר שקיבלתם רקע וטיפים על הסביבה כך שהעבודה תהיהי יותר מהירה וקלה ...

אם חשבתם שאין שעורי בית טעות!!! נא לגשת לעזרה ויש שם מדריך ב quickStart שמלמד איך לכתוב עורך טקסט (מי שבאמת משעמם לו שיעשה גם את המדריך לבסיסי נתונים) יש שם כמה טיפים טובים ולמרות שאני באופן אישי כבר לא משתמש בחלק מהטכניקות שמלמדים שם טוב לדעת שהן קימות.

Components

בפרק זה ארחיב קצת על הרכיבים או בשמם הלועזי Components , ואתאר מאפיינים מרכזים משותפים לרוב הרכיבים וקצת על הרכיבים הנפוצים איתם נעבוד הכי הרבה.

רקע - רכיב זהו בסך הכל אובייקט C++ שעבר מוטציה וקיבל את היכולת להיות מוצג על המסך גם בזמן תכנות . כמו שתיארתי בסוף הפרק הקודם אפשרות זו הגיע בעזרת הגדרות שבאות תחת הכותרת __published בקובץ ב H של האובייקט. אם לא ישבר לי וישאר לי זמן אני גם אכתוב פרק קצר איך אתם תוכלו ליצור רכיבים שלכם בהתבסס על רכיבים קיימים. לידע כללי האב של כל הרכיבים הוא TComponent .

קיימים 2 סוגי רכיבים עיקריים :

- כאלה שניתן לראות אותם בזמן ריצה כמו כפתורים , תוויות וכו.
- רכיבים ביישניים שלא ניתן לראות אותם בזמן ריצה כמו פקדי בסיס נתונים , רכיבי אינטרנט ותיבות הודאה כמו שמירה וטעינה.

עבור רכיבים גרפים שרואים אותם בזמן ריצה כמובן יש חשיבות למיקומם בטופס, ודרך התנהגותם כאשר מגיעים אירועים של שינוי גודל החלון , לעומת הרכיבים הלא גרפים שאין כל משמעות למיקומם , העיקר שהם נמצאים ועושים את תפקידם מאחורי הקלעים.

מבט מהיר על פלטת הרכיבים שנמצאת ליד התפריטים בראש המסך ניתן לראות שהם מחולקים לקטגוריות לפי כרטיסיות – נחלקם גם למספר נושאים כללים :

- כללים לתוכנית – שם נמצא את הפקדים שיבנו את הממשק משתמש ויהוו את המסגרת לתוכנית.
- בסיס נתונים – ניתן ליצור תוכנית התומכת בבסיס נתונים די בקלות , נוסיף רכיבים (שרובם לא גרפים) וקיבלנו תמיכה ב SQL וכו וכו.
- רכיבי אנטרנט – מאפשרים ליצור אפליקציות שיוצרות דפי HTML בצורה דינאמית (אין לי מוסג מה עוד במקום ללמוד את זה אני כותב כאן....)
- אופיס – רכיבים שמקשרים לתוכנות אופיס - בזה באמת אין לי שמץ של מוסג

כמובן יש עוד רכיבים שלא הזכרתי אבל אני מניח שהבנתם את העיקרון ואפשר להתחיל לגשת לעבודה.

מאפיינים משותפים לרכיבים :

- Tag – כמו לפרות שנותנים להן מספר כך לכל רכיב יש אפשרות לתת לו מספר, ברירת המחדל היא 0 . משמש כאשר מספר רכיבים מפעילים אותה פונקציה , כך אפשר להבחין ביניהם.
- לרכיבים גרפיים :

1. Anchors – או בעברית מדוברת: עוגן. קובע איזו פינה של הרכיב תישאר במקום כאשר החלון משנה את גודלו.

2. BiDiMode – קובע כיצד ייכתב הטקסט ברכיב , ימינה , שמאלה ויש עוד אפשרויות – עוזר ביישומים שבהם הכתב הוא מימין לשמאל .

3. Constrains - הגבלות, כידוע העולם לא חופשי וכל רכיב יכול לגדול עד גודל מסוים – שם נקבע את הגודל הזה.

4. Enabled – קובע אם ניתן בזמן ריצה לגשת ולהפעיל את הרכיב, כאשר הוא false לא ניתן לגשת לרכיב ואי אפשר להשתמש בו לצרכינו הזדוניים.

5. Font – כתב , כאן נשנה את גודל , סוג , צבע ומאפייני הכתב – אני ממליץ על Book Antiqua (בגלל יעל המבינה תבין)

6. Hint – רמז, מה הפקד עושה – ההיתם מתים שיהיה לכם אחד כזה במבחן! במאפיין זה נכניס את המחרוזת שתופיע כרמז , נשים לב שלא יעבוד אם ShowHint בעל ערך שלילי (כמו חשבון הבנק שלי)

7. Left – המרחק אופקי מהפינה השמאלית עליונה לדופן הרכיב שמכיל את הרכיב שלנו - צחוקים אבל הוא יכול לקבל גם ערך שלילי.

8. Name – שם הרכיב , טעות גדולה תהיהי בידכם אם לא תתנו שמות משמעותיים לרכיבים איתם אתם עובדים !! טעות עוד יותר גדולה אם תתנו שמות ארוכים מדי .

9. Parent – כל המאפיינים שמתחילים ב"הורה" קובעים אילו תכונות באות בירושה מהורה של הרכיב. טוב לכך שאם יש רכיב שמכיל הרבה אחרים אזי במכה אחת נשנה לכולם את הכתב אם משנים לאב את הכתב (ידוע שאב יש רק אחד)

10. Top - כמו Left רק לגבי מרחק לדופן עליון של הרכיב המכיל.

11. Visible- האם ניתן לראות את הרכיב בזמן ריצה, משמש מצבים בהם הטופס עמוס בפרטים ורוצים להקל על המשתמש ולצמצם את הרכיבים הטופס. (ברור שיש עוד שימושים – זו היתה רק דוגמא)

12. Width, Height – קובע את אורך וגובה הרכיב ולא בהתאמה. (יצא לי כך לא התכוונתי)

זהו פחות או יותר זקירה מהירה על מספר מאפיינים שימושים ,
נעבור לכמה טיפים כללים לפני שנמשיך :

- תנו אינדיקציה לסוג הרכיב בשמו ועשו זו בהתחלה לדוגמה שם של כפתור טיפוסי btnPlay ה btn מסמן לי שזה כפתור כמו כן lst,txt mem, lab מסמנים לי עבודה

אם רכיבים אחרים. זה טוב מכוון שאם אני רוצה לחפש רכיב ב object inspector כל הכפתורים ימצאו ביחד.

- כאשר בוחרים רכיב מפלטת הרכיבים תוך כדי לחיצה על SHIFT אז כל לחיצה על הטופס תוסיף מופע חדש של הרכיב. כך ניתן להוסיף הרבה רכיבים מאותו סוג. כדי לצאת מזה אפשר להוציא את המחשב מהחשמל או פשוט ללחוץ על החץ בפלטת הרכיבים.
- שמעו רוק כבד .
- כאשר משנים מאפיין של רכיב מסוים ובוחרים ברכיב אחר אז אם המאפיין ששינינו נמצא גם ברכיב החדש הוא כבר יהיה בחור בobject inspector (כשתתחילו לעבוד תראו שזה עוזר)
- לשנות גודל או מיקום של רכיב בצעדים קטנים השתמשו ב ctrl או shift והחצים.

הפסקה – לכו לשתות כוס קפה ותחזרו לעשות תרגיל. (אפשר גם נס – רצוי 24 קראט של עלית – הכי טעים , ולא אבא שלי לא עובד שם)

תרגיל : נועד לתרגל אתכם אז תעשו אותו בראש מורם

1 . צרו פרויקט חדש.

2.תנו לו כותרת (מאפיין caption)של הטופס ושילחו אותה לימין - כותרת כמובן בעברית - קצת מורל לאומי !

3 . הנחיתו Shape על הטופס (נמצא בכרטיסיית Additional) ושנו את הצורה שלה בעדינות וגם את הצבע בעזרת brush.

4.הוספו כמה כפתורים שישנו את המראה שלה בזמן ריצה .

- להגדיל
- להקטין.
- להעלים.
- להוזיז שמאלה וימינה.
- לתת רמזים .

5. לזכור לתת שמות משמעותיים לרכיבים , לשנות צבעים וכו, יש דוגמה בספריה למשהו דומה שעשיתי - זה עובד למרות שפחות או יותר מפר כל כלל בספר של עיצוב ממשק משתמש.

6. לשמור הכל יפה יפה – אני נותן קידומת U לפני שם הטופס לציין שזו יחידה – מורכב מ H ו CPP .

אין לי כוונה לפרט יותר מדי על כל רכיב - בשביל זה יש את העזרה (שהיא ממש ממש ממש טובה) אני נותן קצת מידע על חלק מהרכבים כדי שתכלו להתחיל לעבוד.

קצת על תכונות מיוחדות של רכיבים

- תפריטים – יש 2 סוגים , אחד יופיע מתחת לכותרת של האפליקציה , השני הוא כאשר לוחצים ימין עכבר . לטופס יש תכונה בשם Menu שקובעת באיזה תפריט ראשי הטופס משתמש – כלומר ניתן ליצור מספר תפריטים ואז לבחור בזמן ריצה איזה תפריט יהיה פעיל.

- Memo - ניתן שם לערוך מספר שורות טקסט . נשים לב למספר תכונות :
  1. ReadOnly – האם ניתן רק לקרוא או גם לערוך.
  2. WordWrap – האם לרדת לשורה חדשה כשאין מקום (הגדרה ממש לא מדויקת אבל מספיק קרובה)
  3. Lines – שם נשמרות שורות הטקסט , משתנה זה הוא מטיפוס TStrings משתנה זה מכיל 2 פונקציות מגניבות לשמירה וטעינה של קובץ טקסט שמותיהם SaveToFile, LoadFromFile ,

  רכיב זה דומה לרכיב RichEdit שגם הוא משמש לעריכת טקסט וכפי ששמו מעיד אליו הוא יותר עשיר – עובד עם טקסט עשיר RTF ויודע להדפיס . (RTF – Rich Text Format)

- תיבת טקסט – מאפשר לקלוט מחרוזת מהמשתמש , יש גרסה דומה בכרטסית הנוספים שקולטת סיסמאות .
- Label – נועד להציג טקסט , ואם תחפשו היטב תמצאו גם את הרכיב StaticText שהוא יצור די דומה ובד"כ משמש להצגת תווית שאנו משנה בזמן ריצה.
- כפתורי רדיו ותיבות סימון – מכפתורי רדיו (העגולים) אפשר תמיד לבחור רק אחד , ומתיבות הסימון אפשר לבחור הרבה . ניתן לאחד מספר כפתורים כאילו בקבוצה על ידי GroupBox לדוגמה וכך הם יעבדו ביחד (אחד בשביל כולם וכולם בשביל אחד)
- Panel – זהו רכיב שתפקידו להכיל רכיבים אחרים – הוא ההורה שלהם. יש לו מאפיין בשם Align שקובע להיכן הוא ידבק בעת עיצוב ושינוי הטופס – האם ימינה שמאלה או היהי רשע וינסה לתפוס את כל החלון לעצמו.
- List – מציג רשימה של מחרוזות , הנשמרות במשתנה Items שהוא רשימה מקושרת של מחרוזות כמו בMemo , יש לה כמה מאפיינים שרצוי לשים לב :
  1. MultiSelect – אם ניתן לבחור יותר משורה אחת .
  2. ExtendedSelect – אם המאפיין הקודם הוא true , אז אפשר לבחור טווח של שורות בעזרת מקשי shift and ctrl .
  3. sorted – אם למיין את הרשימה , המיון הוא מיון מחרוזות אז אם הרשימה שלכם מכילה מספרים יהיה בלגן
  4. ItemIndex – מאפשר לבדוק מי נבחר בזמן ריצה (ולכן לא יופיע בזמן עיצוב התוכנית ב Object inspector) , 0 עבור השורה הראשונה ו -1 עבור כלום. (כלומר לאיזו שורה יש פוקוס)

  הבן דוד של הרשימה נמצא בכרטיסיית נוספים והוא מכיל כפתור תיבת סימון שאפשר להכניס לה V ליד כל פריט.
- BitBtn – כמו כפתור רק יותר מגניב , נשים לב לתכונת kind שאומרת איזה סוג הוא – מוסיף ציור חביב על הכפתור ויוצר ערך מוחזר כשנלחץ.
- StringGrid – מאפשר להציג טבלת מחרוזת , יכול להיות מאוד שימושי אבל תקראו את העזרה טוב טוב לפני שתשתמשו בו – אני זוכר שהוציא לי את הנשמה עד שהבנתי מה צריך לעשות שם (זה היה כל כך מזמן שאני כבר לא זוכר אז אין לי טיפים בקשר לזה)
- PageControl – נמצא בכרטיסיית win32 ודומה ל tabControl , שניהם משמשים כרטיסיות (כמו אילו שאליהן יש את הרכיבים) או דפים כמו שניתן

למצוא ב Editor Options . ב PageControl על מנת להוסיף ולהוריד דפים צריך להוריד את הרכיב לטופס ,ללחוץ ימין עכבר ולבחור בפעולה המתאימה . כל אובייקט שמייצג דף נקרא TabSheet , ותכונה מרכזית של PageControl היא ActivePage שאומרת איזה דף עכשיו בפוקוס.

- ImageList – משמש לאגירת רשימת תמונות, את הרכיב עצמו לא נראה בזמן ריצה, יש לו עורך שנותן להוסיף ולהוריד תמונות.
- ProgressBar – רכיב שבא לציין התקדמות בפעולה... די נחמד רק תיקחו בחשבון שהוא דורש משאבים (עדכון המסך וכו וכו) ולכן יש סיכוי שיאט את התהליך שלגביו הוא מראה אינדיקציה. להפעלתו קבעו את ערכי מינימום ומקסימום , ולקידום השתמשו בפונקציה StepIt ואל תקדמו ידנית את הערך ב position
- כפתורים כתפריט – בסוף בכרטיסיה יש מספר רכיבים שמאפשרים ליצור כפתורים לגישה מהירה לאפשרויות פעולה – אפשר ליצור בעזרתם כפתורי תפריט כמו של סביבת העבודה (כולל גרירה, האלמה וסידור מחדש)
- תיבות דו-שיח - תיבות פתח קובץ , שמור קובץ וכו תפקידם לקבל את שם הקובץ מהמשתמש. כדי להפעיל אותם פשוט יש לשים את הדיאלוג המתאים בטופס ולהשתמש בפונקציה Execute כדי להפעיל ולהציג את התיבה . במידה והמשתמש לוחץ על OK יוחזר הקבוע mrOk והשם המלא של הקובץ הנבחר ימצא במאפיין בשם FileName . כמובן ניתן לקבוע פילטרים ומיקום התחלתי לחיפוש שם הקובץ.

זו הייתה סקירה קצרה על חלק מהרכיבים הבסיסים . הגישה שלי היא שכדי לעבוד צריך לדעת מה הכלים העומדים לרשותנו , ואיך ללמוד על הכלים הנל – להשתמש בעזרה זה קליל פשוט לוחצים F1 . כאשר קוראים את העזרה של הרכיבים , נשים לב לפרוט המאפיינים , אירועים ופונקציות מחלקה של כל רכיב.

נקודה קטנה נוספת בהקשר לרכיבים , לכל רכיב יש הורה ובעל.

1. הורה – או בשמו הלועזי parent תפקידו להציג את הרכיב – הוא אחראי לציור מחדש אם הוסתר . לדוגמה אם אתם מציגים כפתור על הטופס אז הטופס הוא ההורה של הכפתור (פעם היו קוראים לזה אב אבל התנועה לשחרור האישה התחזקה בשנים האחרונות ועברנו להורה)
2. בעל – owner הוא אותו טיפוס מסתורי שאחראי לשחרור הזיכרון של הרכיב.

שימוש ברכיבים זה דבר מדליק וחוסך הרבה עבודה אך תמיד צריך לזכור את החסרונות שבעבודה עם דבר מוכן – תופס יותר זיכרון, יותר נפח ועובד יותר לאט . לרוב לא נרגיש את זה אבל אם אתם כותבים מנוע גרפי או מערכת הנחיה לטיל פופאי תעבדו ב C משולב באסמבלר.

תרגיל

- בחרו את רכיב comboBox וקראו קצת בעזרה אליו.
- התחילו פרויקט חדש ובצעו :

1. הכניסו לתוכו PageControl וצרו לו 4 דפים.
2. בדף 1 – הכניסו רשימה שאפשר לבחור בה הרבה פריטים
3. בדף 2 – הכניסו RichText שיכלול לידו 3 כפתורים : טעינה, שמירה והדפסה.
4. בדף 3 – הכניסו 2 קבוצות של כפתורי רדיו
5. בדף 4 – תתפרעו

אירועי לב ולמה הם טובים

לפני כמה ימים קיבלתי מייל שאומר מה לעשות בזמן של אירוע של התקף לב , נאמר שם שצריך להשתעל בכל הכוח . שמח וטוב לב חשבתי שעכשיו אני יודע מה לעשות באירוע כזה אבל מסתבר שטעיתי והשיטה זהו לא טובה (תודה לענת שאירה את עיני) .
אומנם אני לא יודע מה לעשות באירוע של התקף לב אבל אני כן יודע מה לעשות בזמן אירוע של לחיצת עכבר , הצגת טופס , לחיצה על מקש ועל זה אני הולך לדבר בפרק הזה.

כידוע גברת חלונות היא סביבת עבודה (צולעת פחות או יותר) שמבוססת על אירועים , כלומר הלוגיקה של התוכנית היא שיש להגיב על אירועים של המשתמש ובכך לפעול. בנוסף לכך ,התוכנית עצמה יכולה ליצור ולהפעיל אירועים .

סביבת העבודה נותנת לנו פטור מלהתעסק עם הודעות של מערכת ההפעלה וכל מה שנותר לנו הוא לבחור רכיב , ומ Object Inspector לבחור לאיזה אירוע אנו רוצים להגיב , ולכתוב את הקוד שיתבצע כתוצאה של האירוע.

בלי ללכלך את הידיים לא תלמדו כלום אז קדימה , להעלות את סביבת העבודה , לפתוח פרויקט חדש ולהוסיף לו PageControl שיכלול 3 דפים אשר בכל אחד מהם נכתוב כמה אירועים .

דף ראשון :

בדף זה נבחן אירוע של לחיצת עכבר , נקרא באנגלית OnMouseDown כמובן יש אירוע של שחרור העכבר ותזוזת העכבר , כל האירועים שניתן ליצור להם פונקצית תגובה נמצאים ב Object Inspector .

כאשר אנו יוצרים אירוע תגובה למצב שבו המשתמש החביב לוחץ על העכבר אנו מקבלים מידע על הלחיצה – כמובן אין חובה להשתמש במידע זה ...אבל כמו שתמיד אני אומר "מידע זה כוח" ... , אז מה נקבל :

- TObject *Sender - בד"כ באירועים שאנו נקבל מצביע לאותו רכיב שיצר את האירוע - קל לראות שהוא מסוג TObject ולכן אם אנו רוצים להשתמש במידע זה יש לבצע המרה או Casting לטיפוס המתאים.

- TMouseButton Button - משתנה שיכול להכיל ערך שהוא אחד מהשלושה mbLeft, mbRight, mbMiddle ונותן לנו אינדיקציה איזה כפתור נלחץ.
תרגיל כיתה : נחשו מה מציין ??mb הראשון שיגלה יקבע סוף שבוע זוגי בחול – (עזה) כרטיסים (צו 8) על חשבון צה"ל.

- TShiftState Shift – זהו משתנה מסוג Set , שיתן לנו מידע איזה כפתורים מהמקלדת היו לחוצים הזמן הקליק. די קל לעבוד עם משתנה מסוג Set תכתבו נקודה ותקבלו את הפונקציות שמשתנה מסוג זה תומך בהם , החשובה לדעתי היא

Contains שאומרת אם קבוע מסוים נמצא בקבוצה . כמו מיקדום יש מספר קבועים שמציינים כפתורים מקלדת לדוגמה : ssCtrl , ssAlt לרשימה המלאה גשו לעזרה.

- int X, int Y – הקורדינטות של היכן הייתה הלחיצה.

חמושים במידע זה ניצור אירוע שכל לחיצת עכבר שמאלית תוסיף את הקודרדינטות של הנקודה לרשימה ואם לוחצים ותוך כדי כך ה CTRL לחוץ הרשימה תתנקה .

הקוד די טריוויאלי , נעבור אליו במהירות , קודם אני בודק שאכן הלחיצה היא של הכפתור השמאלי , ואז אני בודק אם ה CTRL לחוץ בעזרת Contains , במידה וכן אני מנקה את הרשימה ואם לא אני יוצר מחרוזת על פי ה X,Y ומכניס אותה לרשימה.

אחרי שעברנו על האירוע הזה אני מצפה ממכם להיות מסוגלים להוסיף אירועים ולהבין כיצד להוציא מידע מאירועים התוכנית שלכם.

דרך אגב אם אתם רוצים לבצע לחיצה ויראטולית כלומר לגרום ללחיצה על כפתור מתוך התוכנית אז יש פונקציה שנקראת OnClick שמדמה לחיצה של המשתמש על כפתור .

ונעבור לדף השני ובו נפגוש עוד אספקטים מגניבים – פחות או יותר של אירועים.

דף שני :

יש רכיבים שאנו נרצה לאתחל אותם במהלך ריצת התוכנית , לדוגמה למלא רשימות במספרים בטווח מסוים , לעדכן מידע על טופס וכו וכו . בצורה סמטרית אנו נרצה לבצע פעולות כאשר הטופס נסגר .

בדף השני יצרתי 2 תיבות קומבו שמילאתי אותן במספרים כאשר הטופס מוצג , כלומר כתגובה לאירוע OnShow , וביציאה מהטופס תוצג הודעה - היא מופעלת על ידי אירוע של OnClose את שתיהן ניתן למצוא באירועים שהטופס מגיב אליהן .

אני מנצל את ההזדמנות גם להדגים כיצד ניתן להגן על התוכנה נגד משתמשים טיפשים ויש די הרבה כאלה – או מצד שני לעשות להם חיים קלים.....
נניח שאני צריך לקלוט 2 מספרים שמייצגים טווח - מינימום ומקסימום ... אני רוצה למנוע מהמשתמש להכניס לתיבת המינימום מספר שגדול יותר מהמספר במקסימום ולהפך ...איך נעשה זאת?? (זו הייתה שאלה רטורית)

ניצור 2 פונקציות תגובה לאירועים...האחד כתגובה לשינוי בתיבת המינימום והשני כתגובה לשינוי בתיבת המקסימום. הרעיון הוא שבמידה והמשתמש מכניס ערך שמפר את האיזון בין מינימום למקסימום אז אנו נעדכן את התיבה השניה בהתאם כך שלא נפר את היחס של מינימום מקסימום .

אם הצלחתי לבלבל אתכם אם הדוגמה ואתם שואלים את עצמכם מה לעזאזל הוא רוצה אז פשוט תפעילו את הדוגמה ,תנסו להכניס ערכים לא נכונים ותראו איך כמו במטה קסם הכל מסתדר – כמובן אל תשכחו לעיין בקוד.

את כל מה שכתבי שם אפשר לצמצם לבערך 2 שורות אך כתבתי את זה בצורה ברורה ככול שאפשר והקטע החשוב הוא משפט ה IF שבו אני בודק אם min>max דבר שאסור שיתקיים (כמו מוסיקת טכנו - אבל זו רק דעה שלי) ובמידה וכן מתקיים אני מתקן על ידי עדכון בערכים מתאימים.

אני לא זוכר אם הזכרתי את ,ToInt אז אזכיר בקצרה , כך הופכים מחרוזת למספר int , ואם יש חשד לכך שההמרה לא תצליח נשתמש ב ToIntDef , שמקבלת פרמטר אותו תחזיר אם ההמרה לא הצליחה.

תעשו לעצמכם אירוע הפסקה שאחריה נעבור לדובדבן שבקצפת , לטופ של ה Top לדבר שלקח לי חודש שלם (עם הפסקות) לגלות.

דף שלישי :

יום אחד היה ערב ואני ישבתי לי וכתבתי פונקציה שחישבה לה משהו לא מסובך אבל עקב הכמות הגדולה של נתונים לקח לה זמן די רב להסתיים ולכן רציתי להוסף כפתור שיאפשר למשתמש לעצור באמצע החישוב – ליגיטמי ואפילו די פשוט אבל לא מצאתי בשום מקום איך לעשות זאת....

בדף השלישי הכניסו ProgressBar ושני כפתורים , על אחד ייכתב התחל ועל השני עצור. כתגובה לאירוע לחיצה על התחיל צרו לולאה אין סופית שבכל איטרציה תקדם את ה ProgressBar ב מיקום אחד . הריצו את התוכנית ... מי שעשה מה שנאמר לו ולא קפץ ישר לקוד המוכן נתקל בבעיה שאף כפתור לא מגיב ...אי אפשר לסיים את התוכנית בדרך רגילה אלא רק על ידי סביבת העבודה (לחיצה על PAUSE ואז ctrl f2) .

הבעיה היא שהמחשב "מחשב" את חישוביו ולא מתפנה לטפל באירועים שנוצרים על ידי התוכנית ... כדי לפתור את הבעיה נבצע מספר פעולות:

1. בלולאה בנוסף לקידום ה ProgressBar , נומר לתוכנית לבצע את האירועים שממתינים לה בתור האירועים...זאת על ידי השורה () Application->ProcessMessages שבה אנו פונים לאובייקט גלובלי שמייצג את בתוכנית ואומרים לו במפורש לעבד את ההודאות.
2. ניצור תנאי עצירה ללולאה , כל עוד משתנה פרטי בשם Stop הוא שקר הלולאה תמשיך לרוץ (לא נשכח גם לאתחל אותו לפני הכניסה ללולאה) . הגדרת משתנה פרטי תתבצע בקובץ ה H של הטופס להגיע לשם על ידי CTRL F6 כשנמצאים בקוד של ה CPP.
3. כאשר המשתמש לוחץ עצור נעדכן את המשתנה Stop להיות אמת דבר שיעצור את הלולאה .. אם לא ההינו מבצעים את שלב אחד אז שינוי ערכו של Stop אף פעם לא היה מתרחש....

שחקו עם זה ... תראו שעובד ....

עוד נקודה למחשבה ... נניח שהוספתם כפתור ולחצתם אליו פעמים - התוצאה היא שנוצרה פונקציה שבה ניתן לכתוב קוד. נניח שלא בא לכם לממש את הכפתור מסיבה כלשהי....בשום אופן אל תמחקו את הקוד שנוצר אוטומטי ...תם לא כתבתם אותו אתם לא צריכים למחוק אותו – סביבת העבודה מוחקת אוטומטית פונקציות ריקות שהיא יצרה. אם אתם תמחקו אותו זה סתם יבלבל את סביבת העבודה , אפשר להתגבר ולא יקרה שום נזק אבל למה לעבוד קשה.....

מקווה שעכשיו אם תרצו להגיב לאירוע כלשהו יש בידכם את הכלים והמידע הבסיסי לעשות זאת...גישת תכנות מונחה אירועים שונה מזו של ימי הדוס העליזים ודי מוזרה בהתחלה אבל מתרגלים....

מי רוצה טופס 102 - יצרה ועיצוב של יותר מטופס אחד

לומר את האמת , אני לא צריך טופס 102 - דחו לי את המילואים אבל זה סיפור אחר...
לעומת זאת אני לרוב רוצה לחלק את התוכנית שלי לכמה טפסים שכל טופס אחראי לפעולה אחרת כגון : קלט לנתונים , פלט , הודעות שגיאה שמורכבות קצת יותר ממחרוזת וסימן X .

ברמת העיקרון ישנם שלושה סוגי תוכניות נפוצות בסיבת חלונות :

- MDI – בסוג זה יש חלון ראשי וכל החלונות האחרים הם חלונות בנים שלו ויכולים להתקיים רק בתוך המסגרת של החלון הראשי – כמו סביבת העבודה של VISUAL C++ . ליצור פרויקט כזה לכו לNEW ואז ל PROJECTS ובחרו באופציה זו . אני מאמין שאחרי קראית הפרק הזה ועם קצת עזרה מהעזרה תוכלו לכתוב פרויקטים מסוג זה.
- SDI – האפליקציה בנויה מחלון ראשי אחד , ליצור אפליקציה כזו יש לבחור כמו מקודם ב NEW וכו כו ... די דומה לסוג השלישי אז קדימה.
- DIALOG – אפליקציות מסוג זה הם ברירת המחדל של סביבת העבודה.
כשיוצרים אפליקציה חדשה היא מסוג זה. אפשר לפתוח כמה מספר חלונות ולעבוד איתם במקביל או להמתין עד שחלון נסגר ...אבל שוב אני מקדים את המאוחר.

נתחיל בסקירה מהירה של מספר דברים שחשוב לדעת:

- אף אחד לא יודע אם P == NP (לא קשור לחומר הזה....)
- טופס הוא אובייקט לכל דבר , אפשר להוסיף לו משתנים פרטיים , פונקציות מסוג PUBLIC וכמובן להקצות אותו דינאמית.
- כל טופס שומר את צורת סידור הרכיבים ומיקומם בקובץ עם סיומת DFM שלצערי זה קיצור של DELPHI FORM...ניתן לראות את התוכן בצורת טקסט עם לוחצים ימין עכבר על הטופס ואז "צפייה כטקסט" כמובן באנגלית.
- כל טופס בפרויקט יכול להיות "זמין" או נוצר אוטומטית ורק טופס אחד יכול להיות הטופס הראשי – זה שיעלה ראשון בפרויקט. אני ירחיב על נקודה זו בהמשך....

בעצם ההמשך הגיע....נתחיל לכתוב קוד אחרת אנשים יעזבו אותי למען באפי ציידת הערפדים (אני היחיד בעולם שלא ראה אף פרק של זה)...

1.צרו פרויקט חדש.
2. בחרו ב : FILE->NEW->FORM
3. תעשו SAVE להכל את הטופס הראשון תשמרו בשם ufrmMain ואת השני תשמרו בשם ufrmInput את שם הפרויקט תבחרו לבד ...אתם ילדים גדולים.
4. את שמות הטפסים (NAME) תתנו אותו דבר רק בלי U לפני.
5.בטופס הראשון הכניסו כפתור והפונקציה שתופעל על ידי לחיצה אליו תציג את הטופס השני על ידי ;()ufrmInput->Show .

6. לא נשכח את ההגדרה משפט "#include "ufrmInput.h בקובץ של הטופס הראשון
7. להריץ ולגלות שכל פעם שלוחצים על הכפתור מתגלה הטופס השני שכעת הוא ריק.

כמה נקודות למחשבה:

- הוסיפו פקודת ShowMessage אחרי פקודת הצגת הטופס והריצו....מה קורה כאן? התוכנית ממשיכה לבצע את הפונקציה של הכפתור לאחר הצגת הטופס...לא טוב אם רוצים להמשיך לפי קלט של משתמש...נפתור זאת בקרוב
- לכו ל Project->Options->Forms שם תגלו את מה שאמרתי על טפסים זמינים ואילו שנוצרו אוטומטית.

טופס זמין – יש ליצור אובייקט שלו לפני השימוש כלומר להשתמש ב NEW וכמובן אחר כך ב DELETE כדי לגשת לטופס – נשתמש בזה כאשר התוכנית מלאה בטפסים אשר זוללים זיכרון ואנו מרחמים על המחשב המסכן של המשתמש
טופס שנוצר אוטומטי – נוצר בתחילת התוכנית ואנו יכולים לגשת אליו ולהשתמש בו כלומר להציג אותו לגשת למשתניו וכו וכו... באפליקציות פשוטות תשתמשו רק באופציה זו.

טוב... נניח שאנו רוצים לקבל קלט מהמשתמש דרך הטופס השני , ואז להציג אותו בתוך תווית מהטופס הראשון.... וכמובן נשתולל ונעשה זאת בעזרת טופס שהוא רק זמין ולא נוצר אטוטמטית ... כמובן אני רוצה שהתוכנית לא תמשיך עד שהמשתמש סוגר את חלון הקלט ...

1. הוסיפו כפתור חדש ותנו לו כותרת ושם ....
2. העבירו את טופס הקלט להיות זמין ...(בעזרת > בטופס המתאים שמגיעים אליו דרך Project->Options->Forms
3. בפונקציה נכתוב..... :

```
TfrmInput *frm;
frm = new TfrmInput(this);
frm->ShowModal ();

delete frm ;
```

אני מגדיר משתנה מסוג הטופס , מקצה לו זיכרון , ומציג אותו.... די טריוויאלי

4. נשים את הקוד הקודם של הכפתור הראשון בהערה כי הוא יצור כעת EXAPTION כי הטופס לא קיים .... אלא רק זמין....
5. שחקו אם זה לראות שהכל עובד.

עכשיו לחלק המעניין פחות או יותר , בטופס השני הכניסו תיבת טקסט וכפתור מסוג BITBTN ושנו את ה KIND שלו ל BKOK. שם תיבת הטקסט יהיה edInput הטופס הראשון הכניסו תווית בשם... : labShow

כל מה שנשאר לעשות הוא להוסיף את השורה :

```
labShow->Caption = frm->edInput->Text;
```

ברווח של הקוד הקודם... השורה הנל מבצעת השמה מתיבת טקסט הטופס השני לתווית של הטופס הראשון...פשוט וקל ... נסו לשחק ולרשום רק Show במקום ShowModal ותראו מה קורה (אם אתם רוצים לראות את הטופס הורידו את השורה של שחרור הזיכרון)

אנו ממש קרובים לסיום רק עוד הסבר קצר . טופס שמוצג בעזרת ShowModal ייסגר על ידי השמת ערך למשתנה בשם ModalResult הערכים שהוא מקבל מתחילים ב mr ואז הפעולה כמו OK , בעזרה תמצאו את הרשימה המלאה.. כאשר נתנו KIND לכפתור יצרנו השמה לModalResult כך שהמשתמש לחץ על הכפתור התבצע הכנסת ערך ל ModalResult והחלון נסגר.

הפונקציה ShowModal מחזירה את הערך ModalResult שבעזרתו נסגר הטופס , כך אפשר להבחין בין אם המשתמש בחר ביטול , OK או כל פעולה אחרת ולפעול בהתאם.

עוד נושא קטן שאם לא אכתוב אליו אני לא ישן בלילה...ואנו לא רוצים שזה יקרה....

הטופס כאמור הוא אובייקט , הוא מכיל נתונים שחלקים אפילו נמצא ברכיבים עצמם כמו למשל כותרות של תוויות , השאלה היא האם יש לשמור את המידע הנ"ל בתוך אובייקט נפרד מהממשק הגרפי או להשאיר אותו בממשק... ההיתרון ביצירת אובייקט שמייצג את המידע הוא שקל להעביר אותו בין טפסים שונים – בעזרת העברת מצביע ואנו נשמור יפה יפה את כל הכללים שלמדנו בתכנון C++ אבל לפעמים אם יש רק פריט מידע אחד או שנים למה לטרוח ולעבוד קשה... עצתי מניסיון של אלפי שנים .... עטפו הכל יפה יפה באובייקט והרחיקו אותו מהממשק שמחר יכול להשתנות ... אבל אם אתם מתכנתים בלילה ואף אחד לא רואה אתכם ויש רק קצת מידע להעביר בין הטפסים אפשר ורצוי להשתמש בממשק (בתנאי שלא מספרים לאף אחד)

וממש לסיום יש כמה תכונות שמיוחדות לאובייקט טופס :

- FormStyle- מציין את סוג הטופס בהתאם לסוגי התוכניות שיש בחלונות... החלק שהסברתי בהתחלה.
- BorderStyle – קובע כיצד תראה המסגרת של החלון... קשור לאפשרויות אם ניתן לשנות את גודלו של החלון... יש לשים לב שבאפשרויות מסוימות יעלם לכל התפריט – אני משאיר לכם לגלות שאיזה אופציה זה קורה.
- Position – מיקום החלון בזמן ריצה .. היכן הוא יוצג ...אני אוהב להציג את החלון במרכז שולחן העבודה
- WindowState – אומר מה מצב החלון כשמוצג – אם הוא יהיה מקסימלי מינימלי או נורמלי....לפחות זה מתחרז.

תצרו הרבה טפסים בחלומות (חלונות) שלכם....
דרך אגב מי שהגיע עד לפה ולא נשבר בדרך כמו איזה רכיכה כבר יכול לעשות דברים די מגניבים

בואו נעשה את זה בקבוצה (כולם בשביל אחד ואחד בשבילי)

בפרק זה אני הולך ללמד אתכם מספר קיצורים שמי שיאמץ אותם בחום אל ליבו ימנע ממנו הצורך לכתיבת קוד מיותר ומכוער אך הזהרה לי מראש לכל הביישנים ...אני הולך להשתמש ב dynamic_cast אז מי שלא זוכר זה הזמן לפתוח ספר (אני הולך להסביר בכלליות נראה אם יהיה לי כוח)

בחלק הראשון של הפרק אלמד אתכם ידידי היקרים (אנו מכירים כבר 5 פרקים) איך לדוגמה לשנות לכל התוויות בטופס את הצבע נניח .....לכחול , או אפשר לעבור על כל תיבות הסימון (check box) ולבטל את כל מי שנבחר או שאפשר ממש להתפרע ולספור כמה רכיבים יש בטופס....

למה בכלל אנו צריכים את כל זה?? התשובה פשוטה **עיצוב ממשק משתמש** , תוכנה ידידותית (שבה המשתמש יכול לשנות את מראה הטופס לדוגמה) יכולה לעשות לו את החיים יפים. כמובן, זה לא מחפר על אלגוריתמים רעים שמאחורי העטיפה הנוצצת (אל תביט בקנקן....כן בטח)

למה אני מקדיש לזה פרק שלם ? אני לא רוצה לראות בוגרי מדריך זה כותבים קוד מהצורה :

```
CheckBox1->Checked = false;
CheckBox2->Checked = false;
CheckBox3->Checked = false;
CheckBox4->Checked = false;
...
CheckBox10000->Checked = false;
```

מה יקרה אם מחר נוסיף או נוריד כפתור ?
אפשר לפתור את הכל בלולאה אחת פשוטה...אז לאחר ההסברים הארוכים ומיגעים פתחו פרויקט חדש ו.....

1. הוסיפו תווית , 3 תיבות סימון , ותיבת טקסט , תנו להם שמות וכו (לא ממש חייבים)
2. הוסיפו כפתור שתפקידו לנקות את הטופס.

נגדיר כמה כללים כבר כאן כדי שהקוד יכנס במכה אחת לדף הבא.
ComponentCount – זהו תכונה שאומרת כמה רכיבים חיים בתוך הרכיב שעליו אנו שואלים. כמובן הטופס הוא רכיב ולכן אם נרצה לעבור בלולאה על כל הרכיבים שיש לו נשתמש בתכונה זו.
dynamic_cast – יחזיר מצביע שהוא לא NULL אם ההמרה של סוגי מצבעים הצליחה כך אפשר לבדוק אם המצביע של הרכיב הנוכחי מצביע לרכיב שעליו אני רוצה לפעול וכמובן לבצע פעולות שלא היו אפשריות כאשר החזקתי מצביע לאב.
Components[] – זהו מערך של רכיבים שמחזיק הטופס .

עכשיו לקוד :

```
int i;                          //for index
TComponent *Com;        //pointer to a current component

for(i=0;i<ComponentCount;i ++)
{
Com = this->Components[i];  //get the current component

//if the component is Check Box
if (dynamic_cast<TCheckBox *>(Com)) //if not null do
dynamic_cast<TCheckBox *>(Com)->Checked = false; //uncheck it

//if the component is Edit
if (dynamic_cast<TEdit *>(Com))  //if not null do
dynamic_cast<TEdit *>(Com)->Text = "";     //clear it

}
```

ממש סיוט להעתיק קוד לכאן...
הלולאה עוברת על כל הרכיבים בטופס , אם הרכיב הוא מסוג טקסט או תיבת סימון אז ננקה אותו ואחרת לא נעשה כלום , כמובן בתוכנית רגילה ההיתי כותב את הקוד אחרת כמו לדוגמה שומר את תשובת ההמרה במשתנה מתאים וכו וכו – כאן רציתי לקצר שתבינו את הרעיון.

3 . עוד דוגמית קטנה על אותו נושא.. תנסו לבד ואז תבדקו איך אני עשיתי זאת....
צרו כפתור שבלחיצה עליו יפתח DIALOG שיתן לי לבחור כתב , צבע וכו ואז כל הרכיבים בטופס ישתנו לכתב הנ"ל....

```
if (DlgFont->Execute())      //show the dialog
{
TComponent *Com; //pointer to a current component
int i; //for index
for(i=0;i<ComponentCount;i++)  //loop
 {
  Com = this->Components[i];  //get the current component
  if (dynamic_cast<TEdit *>(Com ))
   dynamic_cast<TEdit *>(Com)->Font =DlgFont->Font ;
else if (dynamic_cast<TCheckBox *>(Com ))
  dynamic_cast<TCheckBox *>(Com)->Font =DlgFont->Font;
else if (dynamic_cast<TBitBtn *>(Com  ))
 dynamic_cast<TBitBtn *>(Com)->Font =DlgFont->Font      ;
else if (dynamic_cast<TLabel *>(Com   ))
  dynamic_cast<TLabel *>(Com)->Font =DlgFont->Font   ;  }}
```

בקוד אני מעלה DIALOG ששואל את המשתמש איזה כתב הוא רוצה ואז עובר על הרכיבים הטופס ומשנה את אילו שיש להם כתב , לומר את האמת חשבתי שיש לרכיבים הנ"ל אב משותף שנותן להם את האפשרות לקביעת הכתב בהורשה ...טעיתי אין להם לכן אני משנה אחד אחד .....

טוב עכשיו הפסקה - שאחריה ניצור מערכים של רכיבים ונקשר את כולם לעבוד בשבילינו.

המטרה של חלק זה היא ליצור בצורה דינאמית רכיבים בזמן ריצה , למה ?? טוב ששאלתם , לדוגמה אם אנו רוצים ליצור מטריצה 20 על 20 של "נקודות" שכל נקודה יכולה להיות מסומנת או לא.... לא הגיוני שנשב ונילחץ 400 פעם בזמן עיצוב התוכנית כדי לקבל את המטריצה המבוקשת....המחשב יעשה את זה בשבילינו לפני הקוד כמה נקודות :

- Tag – לכל רכיב יש "תג" שבו אפשר לשמור מספר - עוזר בזיהוי הרכיב הספציפי כך לדוגמה אפשר לומר לתוכנה לסמן את הנקודה ה 33 שאותה נזהה על פי אותו תג.
- אני אשתמש ב Panel על מנת לייצג נקודות , די נוח כי אפשר להחליף לו את הצבע ויש לו מסגרת כך שאני יכול להבדיל Panel אחד מהשני.
- נא להיזכר בנושא של Parent – אחראי על תצוגה לעומת OWNER שאחראי על שחרור זיכרון – מוזכר בפרק 2.

הדוגמה תהיהי קצת יותר מורכבת ממה שראינו עד עכשיו אז אחלק את הקוד והפעולות לשלושה חלקים , וודאו שכל חלק עובד לפני שעוברים לחלק הבא .

בשלב הראשון אני רוצה ליצור מערך דו ממדי של Panel בגודל 40 עמודות ו10 שורות . מה מבנה הנתונים שיחזיק את המערך הנ"ל ? התשובה בשאלה כמובן מערך (בשלב זה לא נשתמש ב STL כדי שתראו בדיוק איך כל דבר עובד) .

1. צרו פרויקט חדש .
2. הגדירו 2 קבועים שיחזיקו את השורות ועמודות.
3. הגדירו משתנה פרטי של אובייקט הטופס על מנת שיחזיק את המערך.

```
TPanel *board[ROW][COL ];
```

כמובן לא נשכח שתמיד מחזיקים מצביע לרכיב ולא את הרכיב עצמו ולכן יש כוכבית לפני שם המשתנה.

4. צרו Panel בטופס עצמו – על ה Panel הזה אנו ניצור את המטריצה.
5. כעת נשאר ליצור את המטריצה זה יקרה בבנאי של הטופס.

כשיוצרים רכיב בזמן ריצה אילו מאפיינים אנו צריכים ורוצים לשנות לו:

- מי האב שלו – כלומר מי אחראי לצייר אותו על הטופס

- מיקומו – יקבע על ידי הפינה השמאלית עליונה.
- גודלו יקבע על ידי רוחב ואורך
- אם יש לו כותרת אז נקבע אותה יחד עם קביעת מאפייני הכתב .
- אם יש פונקציות שהכפתור צריך להפעיל כתגובה לאירועים זה המקום לבצע את הקישור .

אז בלי יותר מדי מילים הביטו בקוד הבא שיוצר את המטריצה :

```
int i,j;    //index
int cell_height,cell_width;  //keep the cell height and wdth

cell_height = panMartix->Height / ROW; //get proportional height
cell_width =  panMartix->Width / COL;  //get proportional width

for(i = 0 ; i < ROW ; i++)          //loop row
for(j = 0 ; j < COL ; j++)         //loop col
{
board[i][j] = new TPanel(this); //create the cell

board[i][j]->Parent = panMartix; //set the parent

set the size//
board[i][j]->Height = cell_height; //set height
board[i][j]->Width = cell_width;   //set width

set position//
board[i][j]->Top = i * cell_height ;  //set top
board[i][j]->Left = j * cell_width;   //set left

set color and tag//
board[i][j]->Color = clGreen;      //set color to green
board[i][j]->Tag  = I * ROW + j;   //set the tag
}
```

בקוד אני מקצה מקום דינאמית עבור Panel ואז בעזרת 2 לולאות אני עובר מנותן ערכים למאפיינים שלהם. שימו לב שגודל "תא" במטריצה הוא פרופורציונלי למספר שורות , מספר עמודות ולגודל הכללי של ה Panel שעליו כל המטריצה יושבת.

שימו לב שאם מריצים את התוכנית ונעזרים ב Code gurad לא נקבל הזהרה על אי שחרור משאבים אותם תפסנו על ידי הקצאה דינאמית – וזאת בגלל שה OWNER אחראי לשחרור הזיכרון – במקרה זה ה OWNER הוא הטופס וקבענו שהוא האחראי בבנאי של ה Panel .

עכשיו לאחר יצירת המטריצה , נעבור לשלב שני ובו אני רוצה שכל לחיצה על תא במערך תגרום לביצוע 2 פעולות :
1 . תדליק את התא להיות אדום.
2 . תדפיס את ה TAG של התא ככותרת שלו.

קוד די פשוט שורות אבל מכוון שזו פונקציה שהולכת להיות מבוצעת כתגובה של אירוע אזי יש לשים לב להגדרה. הקוד :

ב Class של הטופס בחלק המוגדר כ __published יש לרשום את ההצהרה על הפונקציה :

```
void __fastcall Cell_Click(TObject *Sender);
```

ובקובץ ה CPP נרשום את גוף הפונקציה :

```
void __fastcall TfrmMain::Cell_Click(TObject *Sender)
{
TPanel *pan;       //pointer to current panel
pan = dynamic_cast<TPanel *>(Sender); //cast from TObject


if (pan == NULL)  //if the click wasnt from a panel
 return;          //dont do a thing

pan->Caption = pan->Tag;  //set the caption
pan->Color = clRed;       //set the tag
}
```

תשנו את שם הטופס להיות השם שאותו בחרתם....ואם אני צודק לא שיניתם בכלל את שם הטופס...ילדים רעים!

מי שגורם לאירוע מוצבע על ידי Sender , אנו צריכים להמיר אותו להיות מסוג TPanel כדי שנוכל לעבוד איתו – לשנות את הצבע ואת הכותרת.

יופי ,יש לנו פונקציה מדהימה שמשנה דברים אבל מי שיריץ את הפרויקט ברגע זה לא יראה שום שינוי רק משבצות ירוקות מגעילות ממלאות לו את המסך.... לא קישרנו את הפונקציה לתאים במטריצה .
חזרו לבנאי של הטופס והוסיפו את השורה הבאה :

```
board[i][j]->OnClick =  Cell_Click;  //set the onClick
```

השורה הנ"ל קובעת איזה פונקציה תתבצע בזמן לחיצה של העכבר. שימו לב שבעצם ביצענו כאן השמת מחרוזות פשוטה.

בדקו שהפרויקט עובד ....ונעבור לחלק השלישי שיכניס קצת (ודגש על קצת) חיים לטופס שלנו.

בחלק זה אני רוצה שכל שניה שעוברת כל משבצת שנמצאת מימין למשבצת אדומה גם תידלק , כמובן נוסיף כפתור שידאג לניקוי כללי של הטופס.

נתחיל מהכפתור ניקוי – יותר קל...

הוסיפו כפתור שהתגובה ללחיצה עליו היא מעבר על כל המטריצה וקביעת הכותרת להיות מחרוזת ריקה והצבע לירוק. הקוד ממש דומה לקוד שמופיע בבנאי ולכן אני לא טורח להעתיק אותו מהדוגמה...

עכשיו , כדי לגרום לפעולה להתבצע כל שניה , נוסיף שעון שמיצר לנו אירוע כל זמן קבוע ... רכיב השעון נמצא ב SYSTEM אז זה **הזמן** והוסיף אותו לטופס וליצור את פונקצית התגובה שלו .

אז... כל שניה אנו נעבור על כל המטריצה(ללא הטור הכי ימיני...תחשבו לבד למה) ואם התא הנוכחי אדום אז נדליק את התא הימיני שלו על ידי לחיצה "ויראטולית" שתגרום לו להיות אדום ובכך נקבל את מה שרציתי. גבירותיי ורבותיי **הקוד** :

```
int i,j; //index loop

for(i = ROW-1 ; i >= 0 ; i--)          //loop row
for(j = COL-2 ; j >=0 ; j--)          //loop col
if (board[i][j]->Color == clRed)    //if col is red
  Cell_Click (board[i][j+1]); //set the next to be red to
```

הסיבה שאני מבצע לולאה הפוכה היא בגלל שאם אדליק את התא הימיני ואז אבדוק אם הוא אדום (והוא יהיה אדום וכתוצאה מכך ידליק את התא הימיני שלו) אז נקבל מעיין אפקט דומינו שבכל שניה כל השורה הופכת להיות אדומה (תנסו לבד אם אתם לא מאמינים לי) .

להפעיל את הפונקציה פשוט קוראים לה ומעבירים כפרמטר את מי שקרא לפונקציה , יתבצע אולי קצת עבודה מיותרת כי אפשר היה במקום לשנות את הצבע של התא הימיני אבל הקוד כך יותר ברור ונקי ואם מחר נוסיף עוד פעולות ללחיצה הם יתווספו אוטומטית גם לאירוע של הזמן.

זהו זה , אנשים מקווה שהפרק הזה תרם לכם ושהיה כיף לעבור אליו , ברמת העיקרון אנו בהתחלה של הסוף.... לא נשאר עוד הרבה.. עדי

שמור את חוכמת החיים בקובץ

כפי שאפשר בקלות להבין מהשם של הפרק – אני הולך לדבר על קבצים , איך ליצור אותם , איך לעבוד איתם וכו וכו. מי שמכיר את ofstream וחבריו וחושב לעצמו לדלג על הפרק – שישכח מזה אני הולך לדבר כאן על דברים אחרים.

איפה אני ?

לפעמים רצוי וכדאי לדעת מאיזו ספריה הופעלה התוכנית , כלומר מהו ה PATH של התוכנית , שימושי לדוגמה עם רוצים ליצור תת מחיצה . הקוד :

```
AnisString temp(ParamStr(0));
temp = ExtractFilePath(temp);
```

התוצאה של שתי השורות הנל היא שבמשתנה temp יש את ה path לתוכנית.
אפשר ללמוד עוד מספר דברים מהדוגמה :

- ParamStr - הוא פונקציה שבעזרתה אפשר לקבל את הפרמטרים שנשלחו לתוכנית , אינדקס 0 הוא הדרך לתוכנית כולל שמה כמו ב C .
- בשורה ראשונה אנו מפעילים בנאי של מחרוזת עם מחרוזת.
- ExtractFilePath – זוהי פונקציה שמחזירה את PATH לתוכנית כאשר הפרמטר שהיא מקבלת הוא מחרוזת שמתארת דרך לתוכנית. יש לה הרבה בני משפחה והם :
  1. ExtractFileDir
  2. ExtractFileDrive
  3. ExtractFileExt
  4. ExtractFileName

אפשר בקלות להבין מה תפקידם על פי שמם. בזמנכם הפנוי כיתבו תוכנית שמקבלת שם מלא של קובץ ומציגה בתוויות שונות את התוצאות של הפעלת הפונקציות הנ"ל על שם הקובץ.

מחיצות

על מנת לעבוד עם מחיצות דבר ראשון יש לרשום את השורה :

```
#include <FileCtrl.hpp>
```

יש מספר פונקציות שקשורות לטיפול במחיצות :

- bool DirectoryExists(AnsiString path); - מחזיר אמת אם המחיצה המתקבלת כפרמטר קיימת.
- bool CreateDir(AnsiString path); - יוצר את הספריה המבוקשת ומחזיר אמת אם הצליח.

יש עוד הרבה פונקציות הקשורות לטיפול בקבצים ומחיצות , השנים שהזכרתי בהן אני משתמש הכי הרבה , במידה ותרצו לעשות משהו שקשור לקבצים תגשו לעזרה נניח

של DirectoryExists ומשם אל קטגורית הפונקציות הקשורות לקבצים – מאוד טכני אין טעם לפרט עוד.

דרך אגב לא לשכוח לרשום \\ במקום \ כשעובדים עם ספריות !!!!

לסיום הקבצים עצמם

מי שרגיל לעבוד עם stream , יכול לוותר על הקטע הבא שבעזרתו ניתן לרשום בצורה בינארית נתונים לקובץ.

בניגוד ל C שם אנו יוצרים משנתה שהוא מצביע לקובץ , כאן אנו נפנה לקובץ בעזרת פונקציות שנעביר להן int ,כמזהה לקובץ.

הפונקציות הן :

- bool FileExists(AnsiString filename);
- int FileCreate(AnsiString filename);
- int FileOpen(AnsiString filename,int mode);
- int FileRead(int handel,void *buffer,int count);
- int FileWrite(int handel,coid *buffer,int count);
- bool DeleteFile(AnsiString filename);
- void FileClose(int handel);

המצבים שאפשר לפתוח את הקובץ איתם הם משתנים שמתחילים ב fm כלומר File mode ואז נכתוב Open כך נקבל לדוגמה fmOpenRead יפתח את הקובץ לקריאה , את הרשימה השלמה תקבלו מהעזרה על FileOpen .

שימו לב למספר הנקודות הבאות :

- פתיחה או יצירה של קובץ מחזירים int , אותו int ישלח ל FileRead לדוגמה על מנת שנדע מהיכן לקרוא. מספר שלילי בפתיחת קובץ אומר שלא הצלחנו לפתוח אותו – לא להתעצל ולבדוק שאכן מצליחים.
- אם הקובץ לא קיים לא תצליחו לפתוח אותו בעזרת FileOpen ויש ליצור אותו בעזרת FileCreate לכן אני תמיד בודק אם הקובץ קיים ואז פותח או יוצר אותו.
- FileRead ו FileWrite גם מחזירים מספר שלם שאומר כמה בתים שמרנו או קראנו מהקובץ – אם נקבל תשובה שהיא מספר שלילי סימן שלא הצלחנו.

אין יותר מדי מה לדבר על קבצים , פשוט תתחילו לעבוד איתם וכל עוד שמבצעים בדיקות תקינות אין שום סיבה להסתבך.

דיאלוגים

יש Dialogs סטנדרטים של פתיחת וסגירת קובץ , כמה מפתיע אבל הם נמצאים ב TAB של ה Dialogs . אם לאחר הצגתם המשתמש בחר בקובץ ואז OK או לחץ לחיצה כפולה על קובץ אזי במאפיין FileName של הרכיב שמייצג את הדיאלוג יופיע השם המלא של הקובץ .

שווה לשחק אם הפילטרים , כך שיוצגו רק קבצים בעלי סיומת שקשורה לתוכנית שלכם בפתיחת הדיאלוג_במאפיינים של רכיב הדיאלוג לחיצה כפולה על מאפיין ה FILTER יציג את עורך הפילטרים שם תנו תיאור לכל פילטר ומהו הסיומת שקשורה אליו.

מילת הזהרה בקשר לשימוש בפילטרים , כבר קיבלתי מקרים בהם התוכנה הוסיפה אוטומטית את הפילטר לשם הקובץ כך לדוגמה רציתי לפתוח את הקובץ בשם : Adi.txt ול FileName נכנס Adi.txt.txt. לקח לי די הרבה זמן להבין מה קורה . בזמן כתיבת הקוד , תבדקו שאכן אתם מקבלים מה שאתם מצפים לקבל .

יש עוד מלא מלא דברים שאפשר לעשות עם קבצים ,נגעתי רק במקומות החשובים וגם זה היה ממש על קצה המזלג , במידה ואתם רוצים לעשות משהו שלא מוסבר כאן לכו לעזרה ומשם תבוא לכם הישועה.

מקווה שהיה לכם לפחות משעמם לקורא את הפרק כמו שלי היה לכתוב אותו – אבל בתיקווה שכן עזר לכם להבין מי נגד מי למה וכמה.

עוד דברים כללים

לסיום דברים חשובים , שלא יתפסו פרק שלם אבל צריך להקדיש להם מספר מילים :

- הפרויקט עצמו .
- מציאת BUGS.
- חריגים ומיקומם בחברה.
- רכיבים.

הפרויקט עצמו

שווה להקדיש כמה דקות להבין את כל האפשרויות שמקבלים כאשר הולכים למאפייני הפרויקט דרך התפריט : project-->options שם יש מספר TAB :

- Application- שם אפשר לקבוע את שם התוכנית ולשנות את ה ICON ברירת המחדל מבניין רב קומות למשהו יותר מתאים.

- Compiler – שם אפשר לבקש אופטימזציה של הקוד, אילו הזהרות יש להציג ולקבול דברים כללים בקשר לקומפילציה – שתנו רק אם אתם יודעים מה אתם עושים.

- מחיצות – שם אפשר לקבוע היכן ישמרו הקבצים שהמחשב יוצר אם לא משנים כל הקבצים נוצרים במקום אחד , דבר היוצר "בלגן" אדיף לחלק את הקבצים למחיצות , מחיצה לקבצי ה OBJ ומחיצה לקובץ EXE , נקבע אותם על ידי נתינת ערך ל intermediate output , final output . כך שבספריה הראשית ישמר רק קוד – אותו אני לגבות לאחר השלמת פעולה מסוימת.

- Packages – אילו חבילות רכיבים סביבת העבודה תכיר. כל הכפתורים תוויות תיבות טקסט וכו נמצאים בחבילות הסטנדרטיות שמופיעות שם עם V ליד. בדף זה יש אפשרות לבקש RUNTIME PACKAGES דבר האומר שהתוכנה תחפש את הPackages בזמן ריצה , במילים אחרות שאתם משוויצים בתוכנה שלכם לא יספיק ה EXE ויש להוסיף עוד כמה קבצי עזר שהם החבילות , לכן אני ממליץ בחום להוריד את ה V באפשרות זו כך שה EXE יעמוד בפני עצמו.

יש עוד הרבה אפשרויות אותן תלמדו עם הזמן , מניסיון רצוי לקבוע את כל המאפיינים לפני כתיבת התוכנית ואז להתחיל לעבוד - כמובן שאפשרי לשנות לאחר סיום כתיבת התוכנית אבל לי באופן אישי זה עשה בעיות יותר מפעם אחת.

מציאת BUGS

את הטעויות של חסר נקודה פסיק וכו די קל למצוא כל מה שיש לעשות זה לקרוא טוב את הודעת השגיאה , הבעיה היא למצוא את הטעויות הלוגיות , בשביל זה יש אפשרות לשים "נקודות עצירה" , על ידי לחיצה משמאל לקוד – דבר שיצבע את השורה באדום , וכאשר התוכנית תגיע למקום הנ"ל היא תעצור. כאן אין הבדל בין הרצת התוכנית במצב DEBUG או במצב רגיל , כלומר אם יש נקודת עצירה התוכנית תעצור . יש אפשרות לקבוע מאפיינים לנקודת העצירה על ידי לחיצה בימים עכבר על הנקודה האדומה שמייצגת את נקודת עצירה ובחירת מאפיינים מהתפריט.
F7 – אומר מעקב אחרי ביצוע תוך כדי כניסה לפונקציות.
F8 – מעקב שכולל אי היכנסות לתוך פונקציות.
F9 – המשך לרוץ.
Ctrl + f2 – אם נלחץ באשר יש פוקוס על הקוד, יגרום להפסקת ריצת התוכנית (במידה ויש תוכנית שרצה כמובן)

אם עומדים מעל משתנה מסוים בזמן ריצת התוכנית , יוצג הערך שיש בתוכו ואם עומדים אחרי ביטוי חישובי – יוצג הערך שהביטוי נותן.

כמובן ניתן להגדיר WATCH LIST שדרכו ניתן לראות את הערך של מספר משתנים בו זמנית .

בזמן הרצת התוכנית , ניתן ללחוץ בעזרת כפתור ימין של העכבר ,ולבחור בDEBUG ומשם להגיע לאפשרויות של שינוי וצפייה במשתנים , צפייה ב CPU ועוד דברים שיעזרו לכם למצוא היכן כתבתם שטויות בקוד.

לא יאופשר לכם לבצע בDEBUG אם במאפיין הפרויקט ב TAB של ה LINKER הורדתם את ה V של יצירת מידע בDEBUG (Create debug information) , זהו קובץ די גדול שישמר ליד הקובץ ה EXE .

ישנם מקרים שתקבלו שגיאות LINKER אז דבר ראשון יש להתפלל ולהבטיח להיות טובים אם מצליחים לפתור את ה BUG הנ"ל , ואז להתחיל לנסות לפתור . אחת השגיאות הנפוצות מסוג זה שקרו לי היא שעבדתי עם מספר קבצים , עשיתי include כמו שצריך אבל לא הוספתי את הקבצים לפרויקט בעזרת Shift-f11 או בעזרת התפריט , דבר זה נותן UNRESOLVED EXTERNAL ... ברגע זה חסכתי לכם הרבה עבודה וייסורים. במידה וזו לא הבעיה , נסו ללמוד אליה כמה שיותר מהעזרה ולכו למאפייני הפרויקט כדי לשנות שם פרמטרים כך שהכל יעבוד.

בדיחה לסיום : מתכנת יושב עם חברה שלו ומעשן סיגריות , היא אומרת לא שזה לא טוב ואפילו כתוב על הקופסה "אזהרה : משרד הבריאות קובע...." , אז הוא אומר לה שמתכנתים לא מעניין אותם אזהרות רק שגיאות (באנגלית נשמע יותר טוב)

חריגים ומיקומם בחברה

בדיוק 2 מילים על זה. Borland הוסיפה מעיין מחלקה בסיסית שממנה נגזרים כל החריגים , וכך אפשר לתפוס אותם ולהציג את הודעת השגיאה וכו. התוכנית עצמה כולה עטופה ב TRY…CATCH גדול כך שגם אם אתם לא תופסים את החריג תוצג הודעת שגיאה מתאימה. מי שאוהב (או צריך) לעבוד אם חריגים שיקרא קצת בעזרה , המהדר תומך בכל האפשרויות המתקדמות של חריגים כמו FINALY , זריקת חריגים (throw)

רכיבים

כמו שהבנתם כל הממשק בנוי על רכיבים (components) אותם אתם מוסיפים מהחבילות הסטנדרטיות שמגיעות עם סביבת העבודה. יש אפשרות לכתוב רכיבים באופן עצמאי ולהוסיפם לסביבת העבודה כך שבכל תוכנית מאותו רגע ניתן יהיה להשתמש בהם.

- בד"כ רכיבים שאותם נוסיף יתנו לנו פונקיונליות שלו קיימת ברכיבים הסטנדרטים לדוגמה רכיב שיודע לעשות ZIP או כל מני רכיבים שמציגים כפתורים בצורה אחרת .

יום אחד אני אכתוב מדריך איך לכתוב רכיבים , עד אז אתם יכולים לחפש בקישורים שאתן רכיבים מוכנים שאחרים כתבו , הטובים באמת עולים כסף אבל מצד שני יש הרבה הרבה רכיבים בחינם שאפשר להוסיף (לרוב יש קובץ הסברים על איך להוסיף ומה הפונקציונליות של הרכיבים)

סיכום

נתחיל במספר קישורים שמצאתי בעבר לגבי סביבת העבודה , באתרים אלו חפשו מדריכים על נושאים ספציפים , רכיבים שאפשר להוסיף לסביבת העבודה ואם יש פורום אז מדליק יעזרו לכם להתגבר על בעיות. השלושה הראשונים ממש טובים האחרים פחות .

http://www.temporaldoorway.com/programming/cbuilder
http://www.bytamin-c.com
http://www.bcbdev.com
http://www.lohninger.com/cppcomp.html
http://visualcomponentlibrary.com/bcb
http://www.gexperts.org
http://www.buddydog.org/C++Builder/c++builder.html

נמשיך בכך שכדי לכסות את כל הנושאים שאפשר לדבר אליהם בסביבת העבודה הייתי צריך לכתוב ספר שיש בו 1700 עמודים (יש אחד כזה) . מטרתי הייתה לתת למי שלא יודע כלום קצת דחיפה על איך להסתדר עם סביבת העבודה ומי שאכן עקב והפנים את מה שניסיתי להסביר כבר יכול לכתוב תוכניות די מגניבות לסביבת חלונות . כל מי שרוצה ללמוד עוד על הסביבה יש לו את המדריכים באתרים הנ"ל ושימצא את הספר שמלמד (רק שלא יפול לכם על הרגל – לא תוכלו ללכת שבוע) . אם אתם רוצים ללמוד עוד על הסביבה יש אפשרות לעבור על הדוגמאות שמגיעות יחד עם הסביבה שווה להעיף שם מבט.

הנושאים שלא דיברתי אליהם:

- גרור ועזוב.
- כתיבת רכיבים
- בססי נתונים
- יצירת דפי HTML דינאמים
- קישור לאובייקטים של OFFICE
- תקשורת דרך PORT
- THREADS

אם למישהו יש זמן וכוח שיתפוס לעצמו נושא ויכתוב אליו מספר מילים שיסבירו איך להשתמש בו וכמובן שישחרר את המידע לרשת שעוד אנשים יוכלו ללמוד.

אני מרשה להפיץ את המדריך הזה ברשת ,לחברים ולכל מי שמתעניין בתנאי שכל שינוי במה שכתבתי יופיע כהערה הסוגרים שמציין שהתבצע שינוי .

כמובן אני לא אחראי למה שאנשים לתוכניות שאנשים יכתבו בעזרת המדריך ,ואם בטעות כתבתי דבר לא נכון אז אני לא אחראי לנזקים שיקרו ....

מי שממש רוצה לכתוב לי הערות אז הדואר שלי (דרך אגב התחלתי לחפש עבודה אז אם מישהו רוצה להציע לי עבודה....)
la_adi@hotmail.com

זהו , היה נחמד אולי בהמשך אוסיף פרקים
מקווה שעזרתי לכם .

עדי