



دانشگاه بوعلی سینا

نام و نام خانوادگی دانشجو: محمد امین احمدی رشته: کامپیوتر شماره دانشجویی: 9912358001

نام استاد: خانم بطحائیان

موضوع: فاز چهارم پروژه کشاورز

سورة الفاتحة

فهرست مطالب

4	1. شرح پروژه
5	2. مقدمه
6	3. معرفی دستورات مهم زبان سی شارپ
10	4. فایل Program.cs
11	5. User Interface
12	6. فایل ها و عملکرد آیتم ها
12	6-1. فایل MainForm.cs
13	6-2. فایل StartForm.cs
14	6-3. فایل NewFarmerForm.cs
15	6-4. فایل ToolsForm.cs
17	6-5. فایل TractorForm.cs
18	6-6. صفحه ی نتیجه
20	7. نمای کلی برنامه

شرح پروژه

در این فاز سعی کرده ام ایده های خود را که به صورت کد و صرفا نوشته بود را تا حد امکان به صورت گرافیکی در بیاورم.

در این راه از زبان سی شارپ و رابط گرافیکی ویژوال استودیو استفاده کرده و از آموزش های دوست خوبم میلاد بهره گرفته ام.

فاز چهارم پروژه ی کشاورز کاملاً مرتبط با سه فاز قبلی بوده و دقیقاً همان اهداف را دنبال میکند.

مشخصات کشاورز، نوع بذر او و وسیله ی مورد استفاده ، قدرت و وزن تراکتور پیشرفته مواردی است که در این پروژه حائز اهمیت بوده و در روند طی شدن فرایند برنامه تاثیر گذار است.

مقدمه

فاز چهار از پروژه ی کشاورز که به کمک زبان سی شارپ و رابط گرافیکی ویژوال استودیو پیاده سازی شده است، سعی شده که بستر مناسبی برای کاربر این برنامه جهت وارد کردن اطلاعات فراهم گردد.

از این جهت که برنامه به شکلی است که با استفاده از فیلد ها و نوشته های درون فیلد ها و بعضا عنوان صفحه، کاربر را به سوی ارائه ی اطلاعات به صحیح ترین روش ممکن راهنمایی می کند.

همانطور که گفته شد نوع انتخاب گزینه ها نقش اصلی در روند طی شدن فرایند برنامه را تعیین می کند: به طور مثال اگر کاربر مایل به استفاده از تراکتور جهت آسودگی بیشتر در کاشت بذر ها باشد به صفحه ای هدایت میشود که در آن درمورد وزن و قدرت تراکتور اطلاعاتی گرفته میشود اما اگر کاربر بخواهد که عملیات کاشت بذر توسط شخص کشاورز انجام شود برنامه به شکل دیگری پیش می رود.

برنامه ی کشاورز شامل فیلد ها و دکمه ها و متن هایی است که هر کدام به نحوی به سهولت برنامه برای کاربر و نزدیک شدن به هدف برنامه که همان کاشت بذر است کمک میکند.

در این پروژه از دستوراتی استفاده شده که شاید برای کسانی با زبان سی شارپ کار نکرده اند نامفهوم به نظر بیاید.

پس تصمیم گرفتم که در ادامه برخی از مهمترین این دستورات که کمی متفاوت تر از زبان های معمول برنامه نویسی هستند را معرفی کنم.

معرفی دستورات مهم زبان سی شارپ

اغلب برنامه های سی شارپ با بخش دستورات **using** آغاز می شوند. این بخش فضاهای نامی که در طول برنامه مکرراً از آنها استفاده می شود را لیست می کند. استفاده از فضاهای نام در ابتدای برنامه در نوشتن کدها صرفه جویی می کند و در این صورت نیاز نیست برای دسترسی به یک متد از نام کامل آن استفاده کرد.

با استفاده از **Generic** ها می توانید مشخصات نوع داده ای عناصر برنامه نویسی را در یک کلاس یا یک متد تعریف کنید. به شرط آن که این نوع واقعاً در برنامه استفاده شده باشد. به بیان دیگر با استفاده از **Generic** ها می توانید کلاس یا متدی را بنویسید که بتواند در کنار هر نوع داده ای کار کند.

System.ComponentModel : کلاس هایی را ارائه می دهد که برای پیاده سازی رفتار زمان اجرا و زمان طراحی اجزا و کنترل ها استفاده می شود. این فضای نام شامل کلاس های پایه و رابط ها برای پیاده سازی ویژگی ها و مبدل های نوع، اتصال به منابع داده و مؤلفه های صدور مجوز است.

System.Data : حاوی کلاسهایی برای دستیابی به داده های بانک اطلاعاتی است.

System.Drawing : حاوی کلاسهایی برای ترسیم و گرافیک است.

System.Linq : کلاس ها و رابط هایی را ارائه می کند که از جستارهایی پشتیبانی می کنند که از Query یکپارچه با زبان (LINQ) استفاده می کنند.

System.Text : شامل کلاس هایی است که رمزگذاری کاراکترهای ASCII و Unicode را نشان می دهد. کلاس های پایه انتزاعی برای تبدیل بلوک های کاراکتر به بلوک های بایت. و یک کلاس کمکی که اشیاء String را بدون ایجاد نمونه های میانی از String دستکاری و قالب بندی می کند.

System.Windows.Forms : شامل کلاس هایی برای ایجاد برنامه های کاربردی مبتنی بر ویندوز است که از ویژگی های رابط کاربری غنی موجود در سیستم عامل مایکروسافت ویندوز بهره کامل می برد.

چیسیت؟ **public partial class**

هر کلاس در C# در یک فایل فیزیکی جداگانه با پسوند cs قرار می گیرد. C# توانایی اجرای یک کلاس را در چند فایل cs با استفاده از کلمه کلیدی **partial** فراهم می کند. **partial** می تواند به یک کلاس، متد، رابط یا ساختار اعمال شود. کامپایلر این کلاس های جزئی (Partial) را به یک کلاس ترکیب می کند.

گاریرد متد initializecomponent() به چه صورت است؟

ویژوال استودیو از یک متد به نام initializecomponent() برای ساختن یک فرم در زمان اجرا (Run Time) استفاده می کند. تمام کنترل ها و خصوصیات (properties) آنها که در زمان طراحی تنظیم شده اند در این متد قرار می گیرد. این متد به صورت یک دستور در Construtor یک فرم صدا زده می شود.

به عنوان مثال هنگامی که بر روی فرم یک کنترل TextBox می سازیم ، خود ویژوال استودیو تمام خصوصیت ها (Properties) این کنترل را ایجاد کرده و توسط متد initializecomponent تنظیم می کند. و تا زمانی که این متد در Constructor کلاس فرم صدا زده نشود ، در زمان اجرا ، هیچ کنترلی را بر روی فرم نمایش نمی دهد. نکته: هر تعداد Constructor که برای کلاس فرم تعریف می شود باید متد initializecomponent هم تعریف شود .

object sender و EventArgs در رویدادها :

این دو آرگومان، برای تمام رویدادهای مربوط به اشیاء مختلف وجود دارند و هر رویدادی (event) که تعریف شود متناسب با آن رویداد و آن شی این مقادیر متفاوت خواهند بود.

در حالت کلی object تعیین کننده کلاس آن شی هست که قرار است برای آن رویداد مورد نظر تعریف شود و e هم یکسری خصوصیات از رویداد مورد نظر برای همان شی در اختیارمان قرار میدهد.

مثلا فرض کنید روی یکی از فرم ها دابل کلیک کنید و (برای مثال) رویداد Form1_Load رو فعال کنید. در این مورد object sender به اینصورت خواهد بود که object مشخص میکند این رویداد (رویداد Load) برای یک فرم (از جنس کلاس Form) هست و sender هم مشخص میکند که این ارسال کننده Form1 هست.

در مورد EventArgs که با e مشخص میشود؛ برای بعضی از رویدادهای خاص در یکسری از اشیاء، اطلاعات ارزشمندی رو در اختیارما می گذارد.

مثلا برای یک Textbox رویدادی وجود دارد به نام KeyDown که زمانی فعال میشود که یک دکمه پایین نگه داشته شود. در اینجا نام آرگومان به KeyEventArg تغییر پیدا می کند و آرگومان e به ما کمک می کند تا متوجه شویم کاربر چه دکمه ای را فشار داده است.

:this.Close()

وقتی از این متد در جایی مثلا در رویداد مربوط به یک دکمه استفاده می کنیم باعث خروج از فرم جاری می شود. به کد زیر توجه کنید کلمه ی کلیدی this به فرمی که درون آن در حال نوشتن کد هستید اشاره می کند. و برای بستن فرمی که درون آن هستیم از این دستور استفاده می کنیم.

```
private void button1_Click(object sender, EventArgs e)
{
    this.Close();
}
```

گامپرونت **PrintForm** : به گونه ای طراحی شده است که توانایی چاپ آسان فرم ویندوز را فراهم میکند.

new کردن در سی شارپ:

هر زمان یک متغیر (از هر جنسی : int, textbox,...) تعریف شود یک مقدار فضایی در ram اشغال میشود , حال بنا بر نیازی که پیش می آید میتوان این متغیر را تعریف کرد و هر جا که لازم شد آن را new کرد. یعنی تا زمانی که new نشود حافظه ای را اشغال نمیکند.

ShowDialog() : فرم را به عنوان یک کادر محاوره ای معین نشان می دهد.

چرا متد **Application.EnableVisualStyle** مورد نیاز است؟

اگر این خط **Application.EnableVisualStyles()** را کامنت کرده و برنامه را اجرا کنید، خواهید دید که تمام کنترل های موجود در فرم شما فقط با ظاهر کلاسیک ویندوز رندر شده اند.

اگر کامنت **Application.EnableVisualStyles()** را حذف کنید و برنامه را اجرا کنید، می توانید تمام کنترل های خود را با تنظیمات تم فعلی سیستم عامل مشاهده کنید. به این معنی، اگر سیستم عامل شما ویندوز است، از تم سازی داخلی ویندوز برای استایل دادن به کنترل ها و ظاهر کلاسیک ویندوز استفاده می کند. بنابراین دسترسی نیست که اگر کامنت شود باعث از کار افتادن برنامه یا توقف کار شود. اعمال سبک های بصری مانند رنگ ها، فونت ها و سایر عناصر بصری به کنترل های فرم از موضوع فعلی سیستم عامل ضروری است. اگر کنترل و سیستم عامل از این پشتیبانی کنند، در صورت استفاده از این روش، کنترل ها با سبک های بصری ترسیم می شوند. همچنین متد **Application.EnableVisualStyles()** باید به عنوان اولین خط در متد **Main** قبل از فراخوانی هر کنترلی در برنامه برای ایجاد اثر فراخوانی شود.

متد **Application.SetCompatibleTextRenderingDefault**:

SetCompatibleTextRenderingDefault(false) یک روش خودکار است که توسط ویژوال استودیو در متد اصلی فایل **Program.cs** ایجاد می شود. از آنجایی که به طور خودکار تولید می شود، ما معمولاً هیچ توجهی به این روش نمی کنیم.

برخی از کنترل ها در فرم های ویندوز دارای خاصیتی به نام ویژگی **UseCompatibleTextRendering** هستند. این کنترل ها در فرم های ویندوز می توانند متن خود را با استفاده از کلاس **TextRenderer** یا کلاس **Graphics** ارائه دهند.

کلاس **TextRenderer** مبتنی بر کتابخانه گرافیکی **GDI** و کلاس **Graphics** بر اساس کتابخانه گرافیکی **+GDI** است.

روش **SetCompatibleTextRenderingDefault** برای تنظیم پیش فرض استفاده می شود

ویژگی **UseCompatibleTextRendering** که در سراسر برنامه کنترل می کند.

اگر مقدار پارامتر متد `SetCompatibleTextRenderingDefault`، `true` باشد، کنترل‌های جدیدی که از `UseCompatibleTextRendering` پشتیبانی می‌کنند از کلاس `Graphics` مبتنی بر `GDI+` برای رندر متن استفاده می‌کنند. اگر مقدار پارامتر `false` باشد، کنترل‌های جدید از کلاس `TextRenderer` مبتنی بر `GDI` استفاده می‌کنند.

`Application.Run()`:

فرم را برای کاربر قابل مشاهده می‌کند. این اولین فرمی است که در حافظه بارگذاری می‌شود. و این فرم را در یک حلقه پیام اجرا می‌کند، به طوری که شما همه رویدادهای کاربر را دریافت می‌کنید.

فایل Program.cs

نقطه ی شروع پروژه فایل Program.cs است. در این فایل ما یکسری متغیر ها که برایمان مهم است را تعریف میکنیم و در فایل های دیگر برنامه می توانیم از آن استفاده کنیم و مقدار آن ها را تغییر دهیم.

این متغیر ها عبارتند از:

```
public static string Name = "امین احمدی";  
public static string Age = "32";  
public static string seed = "";  
public static string tools = "";  
public static string Tracktor = "استفاده نشده است";  
public static string weight = "-";  
public static string Power = "-";  
public static bool UseTracktor = false ;
```

همانطور که می بینید برخی از متغیر ها مقداری از پیش تعیین شده دارند که ممکن است توسط کاربر تغییر نکنند و به عنوان مقادیر پیش فرض به نمایش دربیایند.

احتمالا توجه شما به عبارت static موجود در تعرف متغیر ها و همچنین تعریف کلاس Program جلب شده باشد. علت Static تعریف شدن این کلاس و متغیر های درونش این است که پس از خارج شدن برنامه از scope کلاس Program مقادیر آن از بین نرود و تا انتهای برنامه هر تغییری را حفظ کند.

User Interface

خب از فایل Program که فراتر برویم ما به ابزار و محلی برای دریافت اطلاعات از سوی کاربر احتیاج داریم. پس به سراغ بخش Design می رویم. در بخش UI رابط گرافیکی و ویژوال استودیو به صورت آماده کامپوننت ها و اکستنشن ها و کنترلر ها و آیتم های مورد نیاز مانند دکمه ها و فیلد ها و کادر و پس زمینه موجود است و ما با Drag & Drop کردن آیتم ها و تنظیم آن ها به شکل دلخواه می توانیم صفحات برنامه را شکل دهیم. این آیتم ها را با استفاده از کد زنی نیز می توان ایجاد کرد ولی برای صرفه جویی در زمان و افزایش سرعت این امکان به ما داده شده که از آیتم ها به شکل آماده استفاده کنیم. اما همانطور که مستحضرید این آیتم ها و نحوه ی چینش آنها در صفحه و محل قرار گیری و اندازه و رنگ و... آنها شامل کد هایبست که در فایل های مربوط به هر صفحه ی برنامه ذخیره می شود. این فایل ها که با پسوند Designer ایجاد می شوند در واقع توسط خود برنامه تولید شده و ذخیره می شوند. در این پروژه نام آنها به ترتیب بدین شکل است که هر کدام مربوط به یک صفحه ی به خصوص می باشند:

1. MainForm.Designer.cs
2. StartForm.Designer.cs
3. NewFarmerForm.Designer.cs
4. ToolsForm.Designer.cs
5. TractorForm.Designer.cs
6. FinishedFormORD.Designer.cs
7. FinishedFormTrack.Designer.cs

بخش UI برنامه همانند بدن بی جانی است که تنها ظاهر دارد اما اینکه هر کدام از این اندام و آیتم ها به چه شکل کار کنند و چگونه عملکردی داشته باشند همانند روحی است که برنامه نویس به بدن بی جان این برنامه می دمد و به آن جان می دهد.

شاید این سوال پیش می آید که برنامه نویس چگونه این کار را انجام میدهد.

رابط کاربری این امکان را برای ما ایجاد کرده که وقتی که برای مثال ما دکمه ی close را Drag & Drop می کنیم، با دبل کلیک کردن روی آن، رویداد کلیک خود به خود برای ما ایجاد می شود. در واقع یک بلاک برای ما در محیط کد زنی ایجاد می کند، حال بسته به کدی که ما در آن بلاک میزنیم دکمه ی close در زمان Run Time رفتار مورد نظر را از خود بروز می دهد و کد های ما اجرا می شود.


در همین مثال دکمه ی close، رخ داد های فراوانی را می توان برای این دکمه ست کرد. اولین و معمول ترین رخ داد می تواند این باشد که به محض کلیک کاربر روی دکمه ی close صفحه بسته شود. یا اینکه می توان رخ داد آن را طوری تنظیم نمود که زمانی که کاربر کلیک کرد و برای چند ثانیه نگه داشت صفحه بسته شود یا اتفاق دیگری بیافتد و...

لازم به ذکر است که در فایل هایی که رابط کاربری برای کد زنی ما ایجاد نموده تمامی کلاس های موجود در فایل ها از کلاس Form ارث می برند که این کلاس مربوط به UI برنامه می شود.


فایل ها و عملکرد آیتم ها

اولین فایل **MainForm.cs** نام دارد که مربوط به صفحه ی نخست یعنی صفحه ی زیر می شود:



با فشردن دکمه  دستور درون بلاک `private void lblClose_Click(object sender, EventArgs e)` اجرا میشود:
`this.Close();`


و پنجره بلافاصله بسته می شود.

با فشردن دکمه  دستور درون بلاک `private void btnStart_Click(object sender, EventArgs e)` اجرا
می شود:
`new StartForm().ShowDialog();`

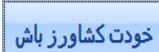
و یک استارت فرم جدید ایجاد میشود.

در صفحه ی بعد ما با فایل **StartForm.cs** کار داریم که مربوط به صفحه ی زیر می شود:




با فشردن دکمه  دستور درون بلاک `private void lblClose_Click(object sender, EventArgs e)` اجرا میشود:
`this.Close();`

و پنجره بلافاصله بسته می شود.

با فشردن دکمه  دستور درون بلاک `private void btnNewFarmer_Click(object sender, EventArgs e)` اجرا می شود:

```
new NewFarmerForm().ShowDialog();
```

و یک فارمر فرم جدید ایجاد میشود که کاربر نام خود یا نام کشاورز مورد علاقه ی خود را برای ادامه ی کار و انجام عملیات وارد می کند.


با فشردن دکمه  دستور های درون بلاک `private void btnInitialFarmer_Click(object sender, EventArgs e)` اجرا می شود:

```
Program.Name = "امین احمدی";
```


```
Program.Age = "32";
```

```
new ToolsForm().ShowDialog();
```

کشاورز پیش فرض برنامه عملیات را پیش میبرد و مستقیماً وارد فرم ابزار برای تعیین بذر و ابزار برای این کشاورز می شویم.

با فشردن دکمه  دستور درون بلاک `private void lblClose_Click(object sender, EventArgs e)` اجرا میشود:
`this.Close();`

و پنجره بلافاصله بسته می شود.

بعد از وارد کردن اطلاعات در فیلد های این پنجره با توجه به عنوان صفحه فشردن دکمه  دستور های درون بلاک `private void btnSave_Click (object sender, EventArgs e)` اجرا می شود:

```
Program.Name = txtFName.Text + " " + txtLName.Text;
```

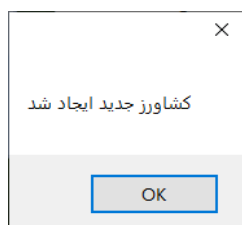
```
Program.Age = txtAge.Text;
```

```
MessageBox.Show("کشاورز جدید ایجاد شد");
```

```
this.Close();
```


```
new ToolsForm().ShowDialog();
```

پس از اجرای دستور `MessageBox.Show` یک کادر به نمایش در می آید و این اطلاع را به کاربر می دهد که کشاورز جدید ایجاد شده است:



سپس پنجره ی مربوط به اطلاعات کشاورز جدید با دستور `this.Close` بسته می شود و با دستور `new ToolsForm.ShowDialog` وارد فرم ابزار برای تعیین بذر و ابزار برای این کشاورز می شویم.



با فشردن دکمه  دستور درون بلاک `private void lblClose_Click(object sender, EventArgs e)` اجرا می شود:

و پنجره بلافاصله بسته می شود.

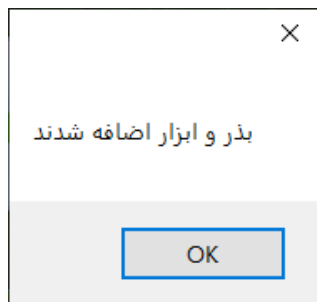


بعد از وارد کردن اطلاعات در فیلد های این پنجره با توجه به عنوان صفحه، با فشردن دکمه دستور های درون بلاک `private void btnSave_Click (object sender, EventArgs e)` اجرا می شود:

```
Program.seed = txtSeed.Text;
Program.tools = txtTools.Text;
MessageBox.Show("بذر و ابزار اضافه شدند");
this.Close();

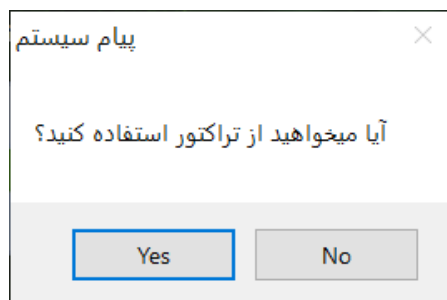
DialogResult dr = MessageBox.Show("آیا میخواهید از تراکتور استفاده کنید؟", "پیام سیستم", MessageBoxButtons.YesNo);
if (dr == DialogResult.Yes)
{
    new TractorForm().ShowDialog();
}
else if (dr == DialogResult.No)
{
    new FinishedFormORD().ShowDialog();
}
{
```

پس از اجرای دستور `MessageBox.Show` یک کادر به نمایش در می آید و این اطلاع را به کاربر می دهد که بذر و ابزار اضافه شده است:



سپس پنجره ی مربوط به اطلاعات ابزار و بذر با دستور `this.Close` بسته می شود.

بعد از این موارد از کاربر در قالب یک کادر `Yes_No question` در مورد استفاده یا عدم استفاده از تراکتور، جهت سهولت کار و سپردن وظایف به این وسیله، سوال می شود:



در واقع این اتفاق ها پس از اجرای دستور زیر رخ می دهد:

```
DialogResult dr = MessageBox.Show("آیا میخواهید از تراکتور استفاده کنید؟", "پیام سیستم", MessageBoxButtons.YesNo);
```

پس از انتخاب دکمه `Yes` یا `No` توسط کاربر شرط زیر به اجرا درمی آید:

```
if (dr == DialogResult.Yes)
{
    new TractorFrom().ShowDialog();
}
else if (dr == DialogResult.No)
{
    new FinishedFormORD().ShowDialog();
}
```

با فشردن دکمه ی `No` توسط کاربر، کار توسط کشاورز صورت میگیرد و با دستور:

```
new FinishedFormORD.ShowDialog();
```


وارد فرم نتیجه ی عملیات توسط کشاورز می شود.


با فشردن دکمه ی `Yes` توسط کاربر کار توسط تراکتور پیشرفته صورت میگیرد و با دستور:

```
new TractorFrom().ShowDialog();
```

وارد فرم تراکتور جهت وارد کردن اطلاعات، می شود.

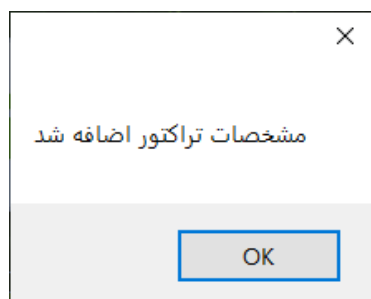
تراکتور فرم در [فایل TracktorFrom.cs](http://TracktorFrom.cs) کد زنی شده و مربوط به صفحه ی زیر می باشد:

با فشردن دکمه  دستور درون بلاک `private void lblClose_Click(object sender, EventArgs e)` اجرا میشود:
`this.Close();`
و پنجره بلافاصله بسته می شود.

بعد از وارد کردن اطلاعات در فیلد های این پنجره و با توجه به عنوان صفحه ،با فشردن دکمه  دستور های درون بلاک `private void btnSave_Click (object sender, EventArgs e)` اجرا می شود:

```
Program.Power = txtPower.Text + "";  
Program.weight = txtWeight.Text + "";  
Program.Tracktor = "استفاده شده است";  
Program.UseTracktor = true;  
MessageBox.Show("مشخصات تراکتور اضافه شد");  
this.Close();  
new FinishedFormTrack().ShowDialog();
```

پس اجرای دستور `MessageBox.Show` یک کادر به نمایش در می آید و این اطلاع را به کاربر می دهد که مشخصات تراکتور اضافه شده است:



سپس پنجره ی مربوط به اطلاعات قدرت و وزن بار تراکتور با دستور `this.Close` بسته می شود و با دستور زیر:

```
new FinishedFormTrack ().ShowDialog();
```

وارد فرم نتیجه ی عملیات توسط تراکتور می شود.


صفحه ی نتیجه

فایل FinishedFormORD.cs زمانی فراخوانی می شود که کاربر تمایلی به استفاده از تراکتور برای کاشت بذر ها نداشته باشد و بخواهد این کار مستقیماً توسط کشاورز صورت گیرد. این فایل مربوط به پنجره ی زیر میباشد:



با فراخوانی این فایل، دستور درون بلاک private void FinishedFormORD_Load(object sender, EventArgs e) به اجرا در می آید:

```
txtResult.Text = "انجام شد" + " " + Program.tools + " " + "با استفاده از ابزار" + " " + Program.Name + " " + "توسط کشاورز" + " " + Program.seed + " " + "عملیات کاشت بذر";
```

با فشردن دکمه  دستور درون بلاک private void lblClose_Click(object sender, EventArgs e) اجرا میشود:

```
this.Close();
```


و پنجره بلافاصله بسته می شود.

اما فایل FinishedFormTrack.cs زمانی فراخوانی می شود که کاربر از تراکتور برای کاشت بذر ها استفاده کند و این فایل مربوط به پنجره ی زیر میباشد:



با فراخوانی این فایل دستور درون بلاک
private void FinishedFormORD_Load(object sender, EventArgs e)
به اجرا در می آید:

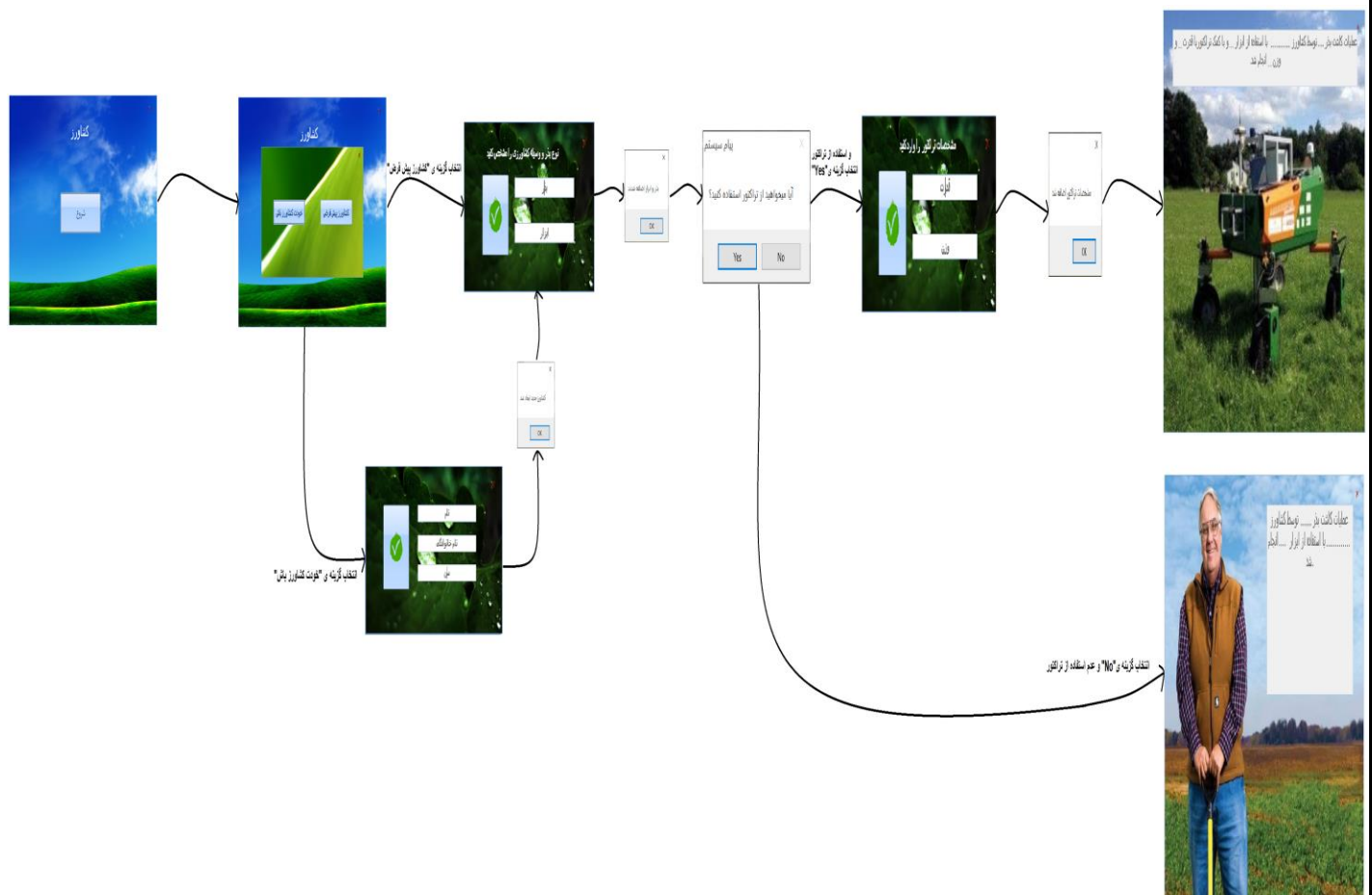
```
txtResult.Text = " عملیات کاشت پنبه " + " " + Program.seed + " " + " توسط کشاورز " + " " + Program.Name + " " + " استفاده از ابزار " + " " + Program.tools + " " + " " + " با کمک تراکتورهای قدرت " + " " + Program.Power + " " + " و وزن " + " " + Program.weight + " " + " انجام شد " ;
```

با فشردن دکمه  دستور درون بلاک private void lblClose_Click(object sender, EventArgs e) اجرا میشود:

```
this.Close();
```

و پنجره بلافاصله بسته می شود.

نمای کلی برنامه



در پناه خدا

شاد و پیروز باشید