# **Титулна страница**

**Име на училището:** Професионална Техническа Гимназия „Цар Симеон Велики“

**Населено място:** град Търговище

**Тема на дипломния проект:** Графично приложение за складиране на градивни елементи

**Трите имена на ученика:** Стелиан Росенов Христов

**Професия | специалност:** Компютърен техник | Компютърна техника и технологии

**Име и фамилия на ръководителя:** Нина Михайлова

# **Съдържание**

1. Описание на средата за разработка на Microsoft за програмиране на приложения с графичен интерфейс Visual Studio.
2. Описание на програмния език за дипломния проект ( C# ).
3. Описание за създаването на приложения с графичен интерфейс на C#.
4. Описание на елементите използвани в графичното приложение.
5. Посочване и обяснение на условни оператори, включително вложени условни оператори.
6. Описание как се работи с приложението.
7. Списък с градивни елементи и тяхното действие.

# **Увод**

Графично приложение проектирано на езика C# (CSharp), за складиране на градивни елементи. Приложението помага на потребителя да може да си складира елементи като всеки елемент съдържа (име, количество и сегашна дата, която може да се променя при актулизиране на елемент).

# **Основна част**

**1. Описание на средата за разработка на Microsoft за програмиране на приложения с графичен интерфейс Visual Studio.**

Средата за разработка, която се използва е създадена от Microsoft. Microsoft Visual Studio е IDE(интегрирана среда за разработка). Тя се използва за разработка на компютърни програми, също така уебсайтове, уеб приложения, уеб услуги, мобилни приложения и игри. Може да произвежда както собствен код, така и управляван код.

Visual Studio включва редактор на код, поддържащ IntelliSense(помага за завършването на код и става по-бързо писането), също и рефакторинг на код.

Интегрираният дебъгер работи както като дебъгер на ниво източник така и на ниво източник, така и като дебъгер на ниво машина. Други вградени инструменти са инструмент за профилиране на код, дизайнер за изграждане на графични приложения(GUI), уеб дизайнер, дизайнер на класове и дизайнер на схеми на бази данни.

Той също така може да приема всякакви добавки, които разширяват функционалноста на почти всяко ниво – включително и добавяне на инсрументи като редактори и виуални дизайнери за специфични за домейна езици или набори от инструменти за други аспекти от жизнения цикъл на разработка на софтуер.

Microsoft Visual Studio поддържа 36 различни езика за програмиране и позволява на редактора на код и инструмента за остраняване на грешки да поддържат в различна степен почти всеки език за програмиране, при условие че съществува специфична за езика услуга. Вградените езици включват C, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML и CSS. Поддръжка на други езици като Python, Ruby, Node.js и M, наред с други е достъпен чрез плъгини. Java се е поддържала в миналото.

Най-основното издание на Visual Studio, изданието на Общността, е достъпно безплатно. Безплатното издание се казва Visual Studio Community, което е предназначено за студенти, отворен код и индивидуални разработчици.

**Архиктектура**

Visual Studio не поддържа нито един език за програмиране, решение или инструмент по същество, вместо това позволява включването на функционалност, кодирана като VSPackage. Когато тя е инсталирана, функционалността е достъпна като услуга. Интегрираната среда за разработка (IDE) представя три услуги: SVsSolution, която предоставя възможност за изброяване на проекти и решения. SVsUIShell, който предоставя функционалност за прозорци и потребителски интерфейс (включително раздели, ленти с инструменти и прозорци с инструменти). И последно SVsShell , който се занимава с регистрация на VSPackages. В допълнение, IDE също така отваря за координирането и позволяването на комуникация между услугите. Всички редактори, дизайнери, типове проекти и други инструменти се изпълняват като VSPackages. Visual Studio използва COM за достъп до VSPackages. SDK на Visual Studio включва също така рамката за управлявани пакети (MPF), която е набор от управлявани обвивки около COM-интерфейсите, които позволяват на пакетите да бъдат написани на всеки съвместим с CLI език. Въпреки това, MPF не предоставя цялата функционалност , предоставяна от COM интерфейсите на Visual Studio. След това услугите могат да се използват за създаване на други пакети, които добавят функционалност към IDE.

Поддръжката на езици за програмиране се добавя чрез използване на специфичен VSP пакет, наречен езикова услуга. Езиковата услуга дефинира различни интерфейси, които реализацията на VSPackage може да приложи, за да добави поддръжка за различни функционалности. Функционалностите, които могат да бъдат добавен по този начин, включват оцветяване на синтаксиса, завършване на изрази,съвпадание на скоби, подсказки за информация за параметри, списъци с членове и маркери за грешки за фонова компилация. Ако интерфейсът е внедрен, функционалността ще бъде налична за езика. Езиковите услуги могат да бъдат внедрени или в собствен код или управляван код. За собствен код могат да се използват или собствените COM интерфейси, или Babel Framework (част от Visual Studio SDK). За управляван код MPF включва обвивки за писане на управлявани езикови услуги.

Visual Studio не включва вградена поддръжка за контрол на източника, но дефинира два алтернативни начина за интегриране на системите за контрол на източника с IDE. Source Control VSPackage може да предостави свой собствен персонализиран потребителски интерфейс. За разлика от тях, плугин за контрол на източника, използващ MSSCCI(интерфейс за управление на изходния код на Microsoft) представя набор от функции, които се използват за първи път за интегриране на Visual SourceSafe с Visual Studio 6.0, но по-късно беше отворен чрез Visual Studio SDK. Visual Studio .NET 2002 използва MSSCCI 1.1, а Visual Studio .NET 2003 използва MSSCCI 1.2. Visual Studio 2005, 2008 и 2010 използват MSSCCI версия 1.3, която добавя поддръжка за преименуване и разпространение за изтриването, както и асинхронно отваряне.

Visual Studio поддържа изпълнение на множество екземпляри на средата (всяка със собствен набор от VSPackages). Екземплярите използват различни кошери на системния регистър (вижте дефиницията на MSDN за термина " кошер на регистъра " в смисъла, използван тук), за да съхраняват състоянието си на конфигурация и се различават по своя AppId (идентификатор на приложение). Инстанциите се стартират от специфичен за AppId .exe, който избира AppId, задава основния кошер и стартира IDE. VSPackages, регистрирани за един AppId, се интегрират с други VSPackages за този AppId. Различните продуктови издания на Visual Studio са създадени с помощта на различните AppIds. Visual Studio ExpressПродуктите на изданието са инсталирани със собствени AppId, но продуктите Standard, Professional и Team Suite споделят един и същ AppId. Следователно, можете да инсталирате Express изданията рамо до рамо с други издания, за разлика от другите издания, които актуализират същата инсталация. Професионалното издание включва супернабор от VSPackages в стандартното издание, а екипният пакет включва супернабор от VSPackages и в двете други издания. Системата AppId се използва от Visual Studio Shell във Visual Studio 2008.

**Характеристика**

***Редактор на кодове***

Visual Studio (както всяка друго IDE ) включва редактор на код, който поддържа подчертаване на синтаксиса и завършване на код с помощта на IntelliSense за променливи, функции, методи, цикли и LINQ заявки. IntelliSense се поддържа за включените езици, както и за XML , каскадни стилови таблици и JavaScript при разработване на уеб сайтове и уеб приложения. Предложенията за автоматично довършване се появяват в aбезмоделно списъчно поле над прозореца на редактора на код, в близост до курсора за редактиране. Във Visual Studio 2008 нататък може да се направи временно полупрозрачен, за да се види препятстваният от него код. Редакторът на код се използва за всички поддържани езици.

Visual Studio също така поддържа настройка на отметки в кода за бърза навигация. Други помощни средства за навигация включват свиващи кодови блокове и инкрементално търсене, в допълнение към нормалното търсене на текст и търсенето на регулярни изрази. Редакторът на код също включва клипборд с множество елементи и списък със задачи. Редакторът на код поддържа фрагменти от код, които са запазени шаблони за повтарящ се код и могат да бъдат вмъкнати в кода и персонализирани за проекта, върху който се работи. Вграден е и инструмент за управление на кодови фрагменти. Тези инструменти се показват като плаващи прозорци, които могат да бъдат настроени да се скриват автоматично, когато не се използват или закачени отстрани на екрана. Редакторът на код на Visual Studio също поддържа рефакторинг на код, включително пренареждане на параметри, преименуване на променливи и методи, извличане на интерфейс и капсулиране на членове на класа в свойствата, наред с други. Visual Studio разполага с фонова компилация (наричана още инкрементална компилация). Докато кодът се пише, Visual Studio го компилира във фонов режим, за да предостави обратна връзка за грешки в синтаксиса и компилацията, които са маркирани с червено вълнообразно подчертаване. Предупрежденията са маркирани със зелено подчертаване. Фоновата компилация не генерира изпълним код, тъй като изисква различен компилатор от този, използван за генериране на изпълним код. Фоновата компилация първоначално беше въведена с Microsoft Visual Basic, но сега е разширена за всички включени езици.

***Програма за отстраняване на грешки***

Visual Studio включва средство за отстраняване на грешки, което работи както като средство за отстраняване на грешки на ниво източник, така и като средство за отстраняване на грешки на ниво машина. Той работи както с управляван код, така и с естествен код и може да се използва за отстраняване на грешки в приложения, написани на всеки език, поддържан от Visual Studio. В допълнение, той може също да се свързва към работещи процеси, да наблюдава и да отстранява грешки в тези процеси. Ако изходният код за изпълнявания процес е наличен, той показва кода, докато се изпълнява. Ако изходният код не е наличен, той може да покаже разглобяването. Дебъгерът на Visual Studio може също да създава дъмпове на паметта, както и да ги зарежда по-късно за отстраняване на грешки. Поддържат се и многонишкови програми. Програмата за отстраняване на грешки може да бъде конфигурирана да се стартира, когато приложение, работещо извън средата на Visual Studio, се срине. Visual Studio Debugger позволява задаване на точки на прекъсване (които позволяват изпълнението да бъде спряно временно на определена позиция) и наблюдава (които следят стойностите на променливите с напредване на изпълнението). Точките на прекъсване могат да бъдат условни, което означава, че се задействат, когато условието е изпълнено. Кодът може да се премине през, т.е. да се изпълнява един ред (от изходния код) наведнъж. Може или да влезе във функции за отстраняване на грешки в него, или да премине през него, т.е. изпълнението на тялото на функцията не е достъпно за ръчна проверка. Инструментът за отстраняване на грешки поддържа Edit and Continue, т.е. позволява кодът да бъде редактиран, докато се отстранява грешки. При отстраняване на грешки, ако показалецът на мишката се задържи над която и да е променлива, текущата й стойност се показва в подсказка („подсказки за данни“), където също може да бъде променена, ако желаете. По време на кодирането дебъгерът на Visual Studio позволява ръчно извикване на определени функции от Immediate прозореца с инструменти. Параметрите на метода се предоставят в прозореца Immediate.

***Дизайнер***

Visual Studio включва множество визуални дизайнери, които да помогнат при разработването на приложения. Тези инструменти включват:

***Windows Forms Designer***

Дизайнерът на Windows Forms се използва за изграждане на GUI приложения с помощта на Windows Forms. Оформлението може да се контролира чрез поставяне на контролите в други контейнери или заключване отстрани на формуляра. Контролите, които показват данни (като текстово поле, списъчно поле и изглед на мрежа), могат да бъдат обвързани с източници на данни като бази данни или заявки. Обвързаните с данни контроли могат да бъдат създадени чрез плъзгане на елементи от прозореца Източници на данни върху повърхност за проектиране. Потребителският интерфейс е свързан с код, използвайки програмен модел, управляван от събития. Дизайнерът генерира C# или VB.NET код за приложението.

***WPF дизайнер***

WPF дизайнерът с кодово име Cider беше представен с Visual Studio 2008. Подобно на дизайнера на Windows Forms, той поддържа метафората за плъзгане и пускане. Използва се за създаване на потребителски интерфейси, насочени към Windows Presentation Foundation. Той поддържа всички функции на WPF, включително обвързване на данни и автоматично управление на оформлението. Той генерира XAML код за потребителския интерфейс. Генерираният XAML файл е съвместим с Microsoft Expression Design, ориентираният към дизайнера продукт. XAML кодът е свързан с код, като се използва код зад модел.

***Уеб дизайнер/разработка***

Visual Studio също така включва редактор и дизайнер на уеб сайтове, които позволяват създаване на уеб страници чрез плъзгане и пускане на джаджи. Използва се за разработване на ASP.NET приложения и поддържа HTML, CSS и JavaScript. Той използва модел зад код за свързване с ASP.NET код. От Visual Studio 2008 нататък, машината за оформление, използвана от уеб дизайнера, се споделя с прекратения Expression Web. Има също така поддръжка на ASP.NET MVC за технологията MVC като отделно изтегляне и проект ASP.NET Dynamic Data , наличен от Microsoft.

***Дизайнер на класа***

Дизайнерът на класове се използва за създаване и редактиране на класовете (включително неговите членове и техния достъп) с помощта на UML моделиране. Class Designer може да генерира C# и VB.NET кодови очертания за класовете и методите. Може също така да генерира диаграми на класове от ръчно написани класове.

***Дизайнер на данни***

Дизайнерът на данни може да се използва за графично редактиране на схеми на база данни , включително въведени таблици, първични и външни ключове и ограничения. Може да се използва и за проектиране на заявки от графичния изглед.

***Картографски дизайнер***

От Visual Studio 2008 нататък дизайнерът на картографиране се използва от LINQ to SQL за проектиране на съпоставянето между схемите на базата данни и класовете , които капсулират данните. Новото решение от подхода на ORM, ADO.NET Entity Framework , замества и подобрява старата технология.

**Други инструменти:**

***Отворете браузъра на раздели***

Браузърът на отворени раздели се използва за изброяване на всички отворени раздели и за превключване между тях. Извиква се с помощта на ***CTRL + TAB.***

***Редактор на свойства***

Инструментът Properties Editor се използва за редактиране на свойства в GUI панел във Visual Studio. Той изброява всички налични свойства (както само за четене, така и тези, които могат да бъдат зададени) за всички обекти, включително класове , формуляри, уеб страници и други елементи.

***Браузър на обекти***

Браузърът на обекти е браузър на пространство от имена и библиотека на класове за Microsoft .NET . Може да се използва за разглеждане на пространствата от имена (които са подредени йерархично) в управлявани сборки . Йерархията може или не може да отразява организацията във файловата система.

***Изследовател на решения***

На езика на Visual Studio решението е набор от кодови файлове и други ресурси, които се използват за изграждане на приложение. Файловете в решението са подредени йерархично, което може или не може да отразява организацията във файловата система. Solution Explorer се използва за управление и преглеждане на файловете в решение.

***Team Explorer***

Team Explorer се използва за интегриране на възможностите на Azure DevOps (или Azure DevOps Services или Azure DevOps Server ) в IDE. В допълнение към интеграцията за контрол на версиите, той предоставя възможност за преглед и управление на отделни работни елементи (включително потребителски истории, грешки, задачи и други документи). Той е включен като част от инсталация на Visual Studio и се предлага и като самостоятелно изтегляне.

***Data Explorer***

Data Explorer се използва за управление на бази данни на екземпляр и на Microsoft SQL Server .Позволява създаване и промяна на таблици на база данни (или чрез издаване на T-SQL команди или чрез използване на дизайнера на данни). Може също да се използва за създаване на заявки и съхранени процедури ,като последните в T-SQL или в управляван код чрез SQL CLR . Налична е и поддръжка за отстраняване на грешки и IntelliSense .

***Server Explorer***

Инструментът Server Explorer се използва за управление на връзки към база данни на достъпен компютър. Използва се също за разглеждане на работещи услуги на Windows ,броячи на производителност, журнална събития на Windows и опашки за съобщения и ги използва като източник на данни.

***Dotfuscator Community Edition***

Visual Studio включва безплатна „лека“ версия на Dotfuscator.

***Рамка за генериране на текст***

Visual Studio включва рамка за генериране на пълен текст, наречена T4 ,която позволявана Visual Studio да генерира текстови файлове от шаблони или в IDE, или чрез код.

***Инструмент за администриране на уебсайт ASP.NET***

Инструментът за администриране на уебсайт ASP.NET позволяваконфигуриранетонауебсайтовенаASP.NET .

***Инструменти на Visual Studio за Office***

Visual Studio Tools за Office е комплект за разработване на софтуер (SDK) и добавка за Visual Studio, която включва инструменти за разработване на пакета Microsoft Office .Преди това (за Visual Studio .NET 2003 и Visual Studio 2005) беше отделен SKU, който поддържаш е само Visual C# и Visual Basic езици или беше включен в Team Suite. С Visual Studio 2008 той вече не е отделен SKU, а е включен в Professional и по-високи издания. При внедряване на VSTO решения е необходимо отделно време за изпълнение.

**Разширяемост**

Visual Studio позволява на разработчиците да пишат разширения за Visual Studio, за да разширят неговите възможности. Тези разширения се "включват" в Visual Studio и разширяват неговата функционалност. Разширенията се предлагат под формата на макроси , добавки и пакети . Макросите представляват повтарящи се задачи и действия, които разработчиците могат да записват програмно за запазване, възпроизвеждане и разпространение. Макросите обаче не могат да прилагат нови команди или да създават прозорци с инструменти. Те са написани с помощта на Visual Basic и не са компилирани. Добавките осигуряват достъп до обектния модел на Visual Studio и могат да взаимодействат с инструментите на IDE. Добавките могат да се използват за внедряване на нова функционалност и могат да добавят нови прозорци с инструменти. Добавките се включват в IDE чрезCOM и може да бъде създаден на всякакви COM-съвместими езици. Пакетите се създават с помощта на Visual Studio SDK и осигуряват най-високо ниво на разширяемост. Те могат да създават дизайнери и други инструменти, както и да интегрират други езици за програмиране. SDK на Visual Studio предоставя не управлявани API , както и управляван API за изпълнение на тези задачи. Въпреки това, управляваният API не е толкова изчерпателен, колкото не управлявания. Разширенията се поддържат в стандартните (и по-високи) версии на Visual Studio 2005. Експресните издания не поддържат разширения за хостинг. Visual Studio 2008 представи Visual Studio Shell , който позволява разработването на персонализирана версия на IDE. Visual Studio Shell дефинира набор от VSPackages, които осигуряват функционалността, необходима във всяка IDE. Освен това могат да се добавят и други пакети за персонализиране на инсталацията. Изолираният режим на обвивката създава нов AppId, където са инсталирани пакетите. Те трябва да се стартират с различен изпълним файл. Той е насочен към разработване на персонализирани среди за разработка, за конкретен език или за конкретен сценарий. Интегрираният режим инсталира пакетите в AppId на изданията Professional/Standard/Eeam System, така че инструментите да се интегрират в тези издания. Visual Studio Shell се предлага като безплатно изтегляне. След пускането на Visual Studio 2008, Microsoft създаде Visual Studio Gallery. Той служи като централно място за публикуване на информация за разширения във Visual Studio. Разработчиците от общността, както и търговските разработчици, могат да качват информация за своите разширения във Visual Studio .NET 2002 чрез Visual Studio 2010. Потребителите на сайта могат да оценяват и преглеждат разширенията, за да помогнат при оценката на качеството на публикуваните разширения. Разширението се съхранява във файл VSIX. Вътрешно VSIX файлът е ZIP файл, който съдържа някои XML файлове и вероятно един или повече DLL. Едно от основните предимства на тези разширения е, че не изискват инсталиране на администраторски права. Планирани са и RSS емисии за уведомяване на потребителите за актуализации на сайта и функции за маркиране.

**Поддържани продукти**

**Microsoft Visual C++**

Microsoft Visual C++ е частична реализация на Microsoft на C и пълна реализация на C++ компилатора и свързаните езици-услуги и специфични инструменти за интеграция с Visual Studio IDE. Може да компилира в режим C или C++. За C++, от версия 15.7 той отговаря на C++17 . C реализацията на Visual Studio 2015 все още не поддържа пълния стандарт; по-специално, комплексното число заглавка complex.h , въведено в C99, не се поддържа. Visual C++ поддържа спецификацията C++/CLI за писане на управляван код , както и код в смесен режим (комбинация от собствен и управляван код ). Microsoft позиционира Visual C++ за разработка в собствен код или в код, който съдържа както собствени, така и управлявани компоненти. Visual C++ поддържа COM , както и библиотеката MFC . За разработката на MFC той предоставя набор от съветници за създаване и персонализиране на стандартен код на MFC и създаване на GUI приложения с помощта на MFC. Visual C++ може също да използва дизайнера на форми на Visual Studio за графично проектиране на потребителския интерфейс. Visual C++ може да се използва и с Windows API . Той също така поддържа използването на вътрешни функции ,които са функции, разпознати от самия компилатор и не имплементирани като библиотека. Вътрешните функции се използват за излагане на набора от инструкции SSE на съвременните процесори. Visual C++ също така включва спецификацията OpenMP (версия 2.0).

**Microsoft Visual C#**

Microsoft Visual C#, реализацията на Microsoft на езика C# , е насочена към .NET Framework, заедно с езиковите услуги, които позволяват на Visual Studio IDE да поддържа C# проекти. Докато езиковите услуги са част от Visual Studio, компилаторът се предлага отделно като част от .NET Framework. Компилаторите на Visual C# 2008, 2010 и 2012 поддържат съответно версии 3.0, 4.0 и 5.0 от спецификациите на езика C#. Visual C# поддържа дизайнера на клас Visual Studio, дизайнера на формуляри и дизайнера на данни, наред с други.

**Microsoft Visual Basic**

Microsoft Visual Basic е реализация от Microsoft на езика VB.NET и свързаните с него инструменти и езикови услуги. Той беше представен с Visual Studio .NET (2002). Microsoft позиционира Visual Basic за бърза разработка на приложения . Visual Basic може да се използва за създаване както на конзолни приложения, така и на GUI приложения. Подобно на Visual C#, Visual Basic също поддържа дизайнера на Visual Studio Class, дизайнера на формуляри и дизайнера на данни, наред с други. Подобно на C#, VB.NET компилаторът също е наличен като част от .NET Framework, но езиковите услуги, които позволяват на VB.NET проекти да се разработват с Visual Studio, са достъпни като част от последното.

**Microsoft Visual Web Developer**

Microsoft Visual Web Developer се използва за създаване на уеб сайтове, уеб приложения и уеб услуги с помощта на ASP.NET. Могат да се използват езици C# или VB.NET . Visual Web Developer може да използва Visual Studio Web Designer за графично проектиране на оформления на уеб страници.

**Azure DevOps**

Azure DevOps е предназначен за проекти за съвместна разработка на софтуер и осигурява контрол на версиите , планиране и проследяване на работата, събиране на данни и отчитане . Той също така включва Team Explorer , който е интегриран във Visual Studio. На 10 септември 2018 г. Microsoft обяви ребрандиране на Visual Studio Team Services (VSTS) в Azure DevOps Services и Team Foundation Server (TFS) в Azure DevOps Server.

**Предишни продукти**

**Visual FoxPro**

Visual FoxPro е ориентиран към данни обектно-ориентиран и процедурен език за програмиране, произведен от Microsoft . Той произлиза от FoxPro (първоначално известен като FoxBASE ), който е разработен от Fox Software в началото на 1984 г. Visual FoxPro е тясно интегриран със собствен двигател за релационна база данни, който разширява възможностите на FoxPro xBase, за да поддържа SQL заявки и манипулиране на данни. Visual FoxPro е пълнофункционален динамичен език за програмиране , който не изисква използването на допълнителна среда за програмиране с общо предназначение. През 2007 г. Visual FoxPro беше прекратен след версия 9 Service Pack 2. Поддържаше се до 2015 г.

**Visual SourceSafe**

Microsoft Visual SourceSafe е софтуерен пакет за контрол на източника , ориентиран към малки проекти за разработка на софтуер. Базата данни SourceSafe е многопотребителска, многопроцесова база данни с файлова система, използваща примитиви на базата данни на файловата система на Windows, за да осигури поддръжка за заключване и споделяне. Всички версии са многопотребителски, като използват SMB (файлов сървър) мрежа. Въпреки това, с Visual SourceSafe 2005 бяха добавени други режими клиент-сървър , Lan Booster и VSS Internet (които използваха HTTP / HTTPS ). Visual SourceSafe 6.0 беше наличен като самостоятелен продукт [56]и беше включен в Visual Studio 6.0 и други продукти като Office Developer Edition. Visual SourceSafe 2005 беше наличен като самостоятелен продукт и беше включен в Team Suite 2005. Azure DevOps замени VSS като препоръчана от Microsoft платформа за контрол на източника.

**Microsoft Visual J++ / Microsoft Visual J#**

Microsoft Visual J++ беше реализацията на Microsoft на езика Java (със специфични за Microsoft разширения) и свързаните езикови услуги. Тя беше прекратена в резултат на съдебни спорове от Sun Microsystems и технологията беше рециклирана във Visual J#, Java компилатор на Microsoft за .NET Framework. J# беше наличен с Visual Studio 2005 (поддържа се до 2015 г.), но беше преустановен във Visual Studio 2008.

**Visual InterDev**

Visual InterDev беше използван за създаване на уеб приложения с помощта на технологиите на Microsoft Active Server Pages (ASP). Той поддържа завършване на код и включва инструменти за управление на сървър на база данни . Той е заменен с Microsoft Visual Web Developer.

***Издания***

**Общност**

Изданието на Общността беше обявено на 12 ноември 2014 г. като нова безплатна версия, с подобна функционалност на Visual Studio Professional. Преди тази дата единствените безплатни издания на Visual Studio бяха вариантите Express с ограничени функции. За разлика от експресните варианти, Visual Studio Community поддържа множество езици и осигурява поддръжка за разширения. Индивидуалните разработчици нямат ограничения за използването на изданието на Общността. Следните употреби също позволяват неограничено използване: принос към проекти с отворен код, академични изследвания, в учебна среда в класната стая и за разработване и тестване на драйвери на устройства за операционната система Windows. Цялото друго използване от организацията зависи от това дали сте класифицирани като Предприятие (повече от 250 служители или повече от 1 милион USDв годишни приходи, според Microsoft). Непредприятия могат да използват до 5 копия без ограничение, потребител номер 6 и по-висок изисква търговски лиценз; Корпоративните организации изискват търговски лиценз за използване извън отбелязаните изключения. Visual Studio Community е ориентиран към индивидуални разработчици и малки екипи.

**Професионален**

От Visual Studio 2010, професионалното издание е комерсиалното издание на Visual Studio от начално ниво. (Преди това беше налично стандартно издание с по-ограничени функции.) Той предоставя IDE за всички поддържани езици за разработка. Поддръжката на MSDN е налична като MSDN Essentials или пълната библиотека на MSDN в зависимост от лиценза. Той поддържа XML и XSLT редактиране и може да създава пакети за внедряване, които използват само ClickOnce и MSI . Той включва инструменти като Server Explorer и интеграция с Microsoft SQL Serverсъщо. Поддръжката за разработка на Windows Mobile беше включена в Visual Studio 2005 Standard, но с Visual Studio 2008 тя е достъпна само в Professional и по-високи издания. Поддръжката за разработка на Windows Phone 7 беше добавена към всички издания на Visual Studio 2010. Разработката за Windows Mobile вече не се поддържа във Visual Studio 2010. Тя е заменена от Windows Phone 7 .

**Enterprise**

В допълнение към функциите, предоставени от професионалното издание, изданието Enterprise предоставя нов набор от инструменти за разработка на софтуер, разработка на бази данни, сътрудничество, показатели, архитектура, тестване и отчитане.

**История**

IDE-то има доста версии, като първата е Visual Studio 97.

* Visual Studio 97
* Visual Studio 6.0 (1998)
* Visual Studio .NET 2002 (Microsoft пускат .NET)
* Visual Studio .NET 2003
* Visual Studio 2005
* Visual Studio 2008
* Visual Studio 2010
* Visual Studio 2012
* Visual Studio 2013
* Visual Studio 2015
* Visual Studio 2017
* Visual Studio 2019
* Visual Studio 2022

**Свързани продукти**

**Azure DevOps Services**

На 13 ноември 2013 г. Microsoft обяви пускането на софтуер като услуга за предлагане на Visual Studio на платформата Microsoft Azure ; по това време Microsoft го нарича Visual Studio Online. По-рано обявен като Team Foundation Services, той се разшири върху локалния Team Foundation Server (TFS; сега известен като Azure DevOps Server ), като го направи достъпен в Интернет и внедри модел на непрекъснато издание . Клиентите могат да използват портала на Azure, за да се абонират за Visual Studio Online. Абонатите получават хоствана Git -съвместима система за контрол на версиите , услуга за тестване на натоварване, телеметрияуслуга и редактор на код в браузъра с кодово име "Монако". По време на Connect(); Събитие за разработчици 2015 на 18 ноември 2015 г., Microsoft обяви, че услугата е ребрандирана като „Visual Studio Team Services (VSTS)“. На 10 септември 2018 г. Microsoft обяви друго ребрандиране на услугата, този път на „Azure DevOps Services“. Microsoft предлага нива на достъп на заинтересовани страни, основни и Visual Studio на абонати за услугите Azure DevOps. Основният план е безплатен за до пет потребители. Потребители с абонамент за Visual Studio могат да бъдат добавени към план без допълнителна такса.

**Управление на жизнения цикъл на приложението Visual Studio**

Visual Studio Application Lifecycle Management (ALM) е колекция от интегрирани инструменти за разработка на софтуер, разработени от Microsoft. Тези инструменти в момента се състоят от IDE (Visual Studio 2015 Community и по-големи издания), сървър (Team Foundation Server) и облачни услуги (Visual Studio Team Services). Visual Studio ALM поддържа екипно базирано развитие и сътрудничество, Agile управление на проекти, DevOps , контрол на източниците, пакетиране, непрекъснато развитие, автоматизирано тестване, управление на версиите, непрекъсната доставка и инструменти за отчитане за приложения и услуги. Във Visual Studio 2005 и Visual Studio 2008 марката беше известна като Microsoft Visual Studio Team System (VSTS). През октомври 2009 г. марката Team System беше преименувана на Visual Studio ALM с изданието на Visual Studio 2010 (кодово име „Rosario“). Visual Studio Team Services дебютира като Visual Studio Online през 2013 г. и беше преименуван през 2015 г.

**Управление на лабораторията на Visual Studio**

Visual Studio Lab Management е инструмент за разработка на софтуер, разработен от Microsoft за софтуерни тестери за създаване и управление на виртуални среди. Lab Management разширява съществуващата платформа за управление на жизнения цикъл на приложенията на Visual Studio , за да даде възможност за интегрирана тестова лаборатория, базирана на Hyper-V . От Visual Studio 2012 той вече се доставя като част от него; и може да се настрои след интегриране на Azure DevOps и SCVMM.

**Visual Studio LightSwitch**

Microsoft Visual Studio LightSwitch е разширение и рамка, специално пригодени за създаване на бизнес приложения, изградени върху съществуващи .NET технологии и платформи на Microsoft. Създадените приложения са архитектурно 3-степенни : потребителският интерфейс работи на Microsoft Silverlight или HTML 5 клиент или като приложение на SharePoint 2013; нивото на логиката и достъпа до данни е изградено на WCF Data Services и е изложено като OData канал, хостван в ASP.NET; и основното съхранение на данни поддържа Microsoft SQL Server Express , Microsoft SQL Server иMicrosoft SQL Azure . LightSwitch също така поддържа други източници на данни, включително Microsoft SharePoint , OData и WCF RIA Services.LightSwitch включва графични дизайнери за проектиране на обекти и връзки на обекти, заявки за обекти и екрани на потребителския интерфейс. Бизнес логиката може да бъде написана на Visual Basic или Visual C# . LightSwitch е включен в Visual Studio 2012 Professional и по-нова версия. Visual Studio 2015 е последната версия на Visual Studio, която включва инструментите LightSwitch. Слоят на потребителския интерфейс вече е незадължителен компонент при внедряване на решение на LightSwitch, което позволява разгръщане само за услуга.

Първата версия на Visual Studio LightSwitch, издадена на 26 юли 2011 г., имаше много разлики от текущата пускане на LightSwitch. По-специално инструментът е закупен и инсталиран като самостоятелен продукт. Ако Visual Studio 2010 Professional или по-нова версия вече е инсталирана на машината, LightSwitch ще се интегрира в това. Втората основна разлика е, че средното ниво е изградено и изложено с помощта на WCF RIA Services.От 14 октомври 2016 г. Microsoft вече не препоръчва LightSwitch за разработка на нови приложения.

**Код на Visual Studio**

Visual Studio Code е безплатен редактор на изходен код , заедно с други функции, за Linux, Mac OS и Windows. Той също така включва поддръжка за отстраняване на грешки и вграден Git Control . Той е изграден върху отворен код , и на 14 април 2016 г. е пуснат като версия 1.0.

**Visual Studio Team System Profiler**

Visual Studio Team System Profiler (VSTS Profiler) е инструмент за анализиране на производителността на .NET проекти, който анализира пространствената и времева сложност на програмата. Той анализира кода и подготвя доклад, който включва вземане на проби от процесора, инструментариум, разпределение на .NET памет и спорове за ресурси .

**C# (CSharp)**

C#e програмен език с общо предназначение, мултипарадигмен. Той обхваща статично писане, строго въвеждане, лексикално обхватно, императивно, деклариративно, функционално, генерично, обектно-ориентирано (базирано на клас) и компонентно-ориентирано програмиране.

Програмният език C# е проектиран от Андреан Хейлсберг от Microsoft през 2000г. И по-късно одобрен като международен стандарт от Ecma през 2002г. и ISO през 2003г. Microsoft въвежда C# заедно с .NET Framework и Visual Studio, и двете са със затворен код. По това време Microsoft нямаше продукти с отворен код. Четири години по-късно, през 2004г. стартира безплатен проект с отворен код, наречен Mono, осигуряващ междуплатформен компилатор и среда за изпълнение на езика за програмиране C#. Деситилетие по-късно Microsoft пусна Visual Studio Code (редактор на код), Roslyn (компилатор) и унифицираната платформа .NET (софтуерна рамка), всички от които поддържат C# и са безплатни, с отворен код и кросплатформени. Mono също се присъедини към Microsoft, но не беше обединено с .NET.

От 2021г. най-новата версия на езика е C# 10.0, която беше пусната през 2021г. в .NET 6.0.

**Цели на дизайна**

Стандартът Ecma изброява тези цели на дизайна за C#

* Езикът да бъде прост, модерен, с общо предназначение, обектно-ориентиран език за програмиране.
* Езикът е предназначен за разработване на софтуерни компоненти, подходящи за внедряване в разпределени среди.
* Езикът е предназначен да бъде подходящ за писане на приложения, така и за вградени системи, вариращи от много големи, които използват сложни операционни системи, до много малки със специални функции.
* Въпреки, че C# приложенията са предназначени да бъдат икономични по отношение на изискванията за памет и процесорна мощност, езикът не е предназначен да се конкурира директно по производителност и размер с C или асемблер.

**История**

Microsoft за първи път използва името C# през 1988г. вариант на езика C, предназначен за инкремантална компилация. Този проетк не е завършен, но името живее.

Името ,,Си диез“ е вдъхновено от музикалната нотация, при която остър символ показва, че написанта нота трябва да бъде направена с полутон по-висока по височина. Това е подобно на името на езика на C++, където ‘‘++‘‘ показва, че променливата трябва да бъде увеличена с 1, след като бъде оценена. Острият символ също наподобява лигатура от четири символа ‘‘+‘‘ (в решетка две по две), което допълнително предполага, че езикът е инкремент на C++.

Поради технически ограничения на дисплея (стандартни шрифтове, браузъри и т.н.) и факта, че острия символ # не присъства в повечето клавиатурни подредби, знакът за цифри # е избран, за да се доближи до острия символ в писменото име на езика за програмиране. Тази конвенция е отразена в езикова спецификация на C#.

Изразът ‘‘остър‘‘ се използва от редица други .NET езици, които са вариант на съществуващи езици, включително J# (език .NET, също проектиран от Microsoft, който е извлечен от Java 1.1), А# (от Ada) и функционален език за програмиране F#. Оригиналната реализация на Eiffel за .NET беше наречена Eiffel#, има оттеглено, тък като вече се поддържа пълният език на Eiffel. Символът е използван също и за библиотеки, като Gtk# (обвивка .NET и за GTK и други GNOME библиотеки) и Cocoa# (обвивка за Cocoa).

**Версии**

* C# 1.0 – януари 2002г.
* C# 1.1 и C# 1.2 - април 2003г.
* C# 2.0 – ноември 2005г
* C# 3.0 – ноември 2007г.
* C# 4.0 – април 2010г
* C# 5.0 – август 2012г.
* C# 6.0 – юли 2015г
* C# 7.0 – март 2017г.
* C# 7.1 – август 2017г.
* C# 7.2 – ноември 2017г
* C# 7.3- май 2018г.
* C# 8.0 – септември 2019г.
* C# 9.0 – ноември 2020г.
* C# 10.0 – ноември 2021г.

**Синтаксис**

Оснвовния синтаксис на на езикa C# е подобен на този на други езици в стил като С, С++ и Java, особено:

* Точката и запетаята се използват за обозначаване на края на изпълнение.

Console.WriteLine(“Hello World”);

* Къдравите скоби се използват за групиране на изрази. Изявленията обикновенно се групират в методи (функции), методи в класове и класове в пространства от имена.

Method{

// алгоритъм

}

* Променливите се присвояват с помощта на знак за равенство, но се сравняват с два последователни знака за равенство

const id1 = 1; const id2 = 2;id1 == id2

* Квадратните скоби се използват с масиви, както за декларирането им, така и за получаване на стойност при даден индекс в един от тях

**Преносимост**

По дизайн С# е езикът за програмиране, който най-пряко отразява основната обща езикова инфраструктура (CLI). Повечето от неговите вътрешни типове съотвестват на типове, внедрени от рамката на CLI. Въпреки това, езиковата, спецификация не посочва изискванията на генериране на код на компилатора: тоест не посочва, че компилаторът на С# трябва да е насочен към Common Language Runtime или да генерира Common Intermediate Language (CIL) или да генерира друг специфичен формат. Теоретично, С# компилатор може да генерира машинен код като традиционните компилатори на C++ или Fortran.

**Писане в C#**

C# поддържа силно, имплицитно въведени декларации на променливи с ключовата дума var и неявно въведени масиви с ключовата дума, new[] последвана от нициализатор на колекция.

C# поддържа строг булев тип данни, bool. Изявленията, които приемат условия, while и if, изискват израз от тип, който имплементира true оператора, като булевия тип. Докато C++ също има булев тип, той може свободно да се преобразува към и от цели числа и изрази като if (a) изискват само това, което а е конвертируемо в bool, което позволява а да бъде int или указател. C# забранява този подход ,,цяло число, което означава вярно или невярно“, на основание, че принуждаването на програмистите да използват изрази, които точно връщат, bool може да предотврати определени типове програмни грешки като if (a = b) (използване на присвояване = вместо равенство =).

C# e по-безопасен за типа от С++. Единствените имплицитни преобразувания по подразбиране са тези, които се считат за безопасни, като например резширяване на цели числа. Това се прилага по време на компилиране, по време на JIT и, в някои случай, по време на изпълнение. Не се извършват неявни преобразувания между булеви и цели числа, нито между членове на изброяване и цели числа (с изключение на литерал 0, който може да бъде имплицитно преобразуван във всеки изброен тип). Всяко дефинирано от потребителя преобразуване трябва да бъде изрично маркирано като явно или неявно, за разлика от конструкторите за копиране на С++ и операторите за преобразуване, които и двете са имплицитно по подразбиране.

С# има изрична поддръжа за ковариация на контравариантност в общи типове, за разлика от С++, който има известна степен на подкрепа за контравариантност просто чрез семантиката на типове връщане на виртуални методи.

Членовете на изброяването се поставят в собствен обхват.

Езикът C# не позволява глобални променливи или функции. Всички методи и членове трябва да бъдат декларирани в рамките на класовете. Статичните членове на публичните класове могат да заменят глобалните променливи и функции.

Локалните променливи не могат да засенчват променливите на затварящия блок, за разлика от C и C++.

**Метапрограмиране**

Метапрограмирането може да се постигне по няколко начина:

* Отражението се поддържа чрез .NET API, които позволяват сценарии като проверка на метаданни на типа и извикване на динамичен метод.
* Дърветата на изрази представляват код като абстрактно синтактично дърво , където всеки възел е израз, който може да бъде проверен или изпълнен. Това позволява динамична модификация на изпълнимия код по време на изпълнение. Дървото на изразите въвежда известна хомойконост в езика.
* Атрибутите са метаданни , които могат да бъдат прикачени към типове, членове или цели сглобки , еквивалентни на пояснения в Java . Атрибутите са достъпни както за компилатора, така и за кода чрез отражение. Много от тези атрибути дублират функционалността на зависимите от платформата предпроцесорни директиви на GCC и VisualC++.
* Платформата за компилиране на .NET (Roslyn) предоставя API достъп до услугите за компилиране на език, позволявайки компилирането на C# код от .NET приложения. Той разкрива API за синтактичен ( лексикален ) анализ на кода, семантичен анализ, динамична компилация към CIL и излъчване на код.
* Генераторите на източник, характеристика на компилатора Roslyn C#, позволяват метапрограмиране по време на компилиране. По време на процеса на компилация разработчиците могат да проверят компилирания код с API на компилатора и да предадат допълнителен генериран C# изходен код, който да бъде компилиран.

**Методи и функции**

Методът в C# е член на клас, който може да бъде извикан като функция (последователност от инструкции), а не просто способност за задържане на стойност на свойство на клас. Както в други синтактично сходни езици, като C++ и ANSI C , подписът на метода е декларация, включваща по ред: всякакви незадължителни ключови думи за достъпност (като private), изричната спецификация на неговия тип връщане (като int, или ключовата дума void ако не е върната стойност), името на метода и накрая, последователност в скоби от спецификации на параметри, разделени със запетая, всяка от които се състои от типа на параметъра, неговото официално име и по избор стойност по подразбиране, която да се използва, когато не е предоставена такава. Някои специфични видове методи, като тези, които просто получават или задават свойство на клас чрез връщана стойност или присвояване, не изискват пълен подпис, но в общия случай дефиницията на клас включва пълната декларация на подписа на неговите методи.

Подобно на C++ и за разлика от Java, C# програмистите трябва да използват ключовата дума модификатор на обхват, virtualза да позволят методите да бъдат отменени от подкласове. Методите за разширение в C# позволяват на програмистите да използват статични методи, сякаш са методи от таблицата с методи на клас, което позволява на програмистите да добавят методи към обект, който смятат, че трябва да съществува в този обект и неговите производни.

Типът dynamic позволява обвързване на метод по време на изпълнение, позволявайки подобни на JavaScript извиквания на методи и композиция на обекти по време на изпълнение. C# поддържа силно въведени указатели на функции чрез ключовата дума delegate. Подобно на псевдо-C++ сигнала и слота на Qt рамката , C# има семантика, специфична около събитията в стил публикуване-абониране, въпреки че C# използва делегати за това. C# предлага synchronized извиквания на методи, подобни на Java, чрез атрибута [MethodImpl(MethodImplOptions.Synchronized)] и има поддръжка за взаимно изключващи се заключвания чрез ключовата дума lock.

Пространство от имена

AC# namespace осигурява същото ниво на изолация на кода като Java package или C++ , с много подобни правила и функции като . Пространствата от имена могат да се импортират със синтаксиса "използване".

**Достъп до паметта**

В C# указателите на адрес на паметта могат да се използват само в блокове, специално маркирани като unsafe , и програмите с опасен код се нуждаят от подходящи разрешения за изпълнение. Повечето достъп до обекти се осъществява чрез препратки към безопасни обекти, които винаги или сочат към "жив" обект, или имат добре дефинираната нулева стойност; невъзможно е да се получи препратка към "мъртъв" обект (този, който е бил събран за боклук) или към произволен блок памет. Несигурен указател може да сочи към екземпляр от тип стойност „неуправляван“, който не съдържа никакви препратки към събрани от боклук обекти, масив, низ или блок памет, разпределена от стека. Код, който не е маркиран като опасен, все още може да съхранява и манипулира указатели чрез System.IntPtrтипа,Управляваната памет не може да бъде изрично освободена; вместо това автоматично се събира боклук. Събирането на боклука решава проблема с изтичането на памет , като освобождава програмиста от отговорност за освобождаването на памет, която вече не е необходима в повечето случаи. Кодът, който запазва препратки към обекти по-дълго от необходимото, все още може да изпита по-високо използване на паметта, отколкото е необходимо, но след като окончателната препратка към обект бъде освободена, паметта е налична за събиране на боклука.

**Изключение**

Редица стандартни изключения са достъпни за програмистите. Методите в стандартните библиотеки редовно хвърлят системни изключения при някои обстоятелства и обхватът от изхвърлени изключения обикновено се документира. Персонализирани класове на изключения могат да бъдат дефинирани за класове, позволяващи специфична обработка за определени обстоятелства, ако е необходимо. Отметнатите изключения не присъстват в C# (за разлика от Java). Това беше съзнателно решение, основано на проблемите с мащабируемостта и версията.

# **Заключение**

Проектът „Графично приложение за складиране на градивни елементи“ е използван езикът C# .NET Framework 4.7.2. Приложението позволява на потребителя да си води записки в приложението, колко на брой и какви елементи има налични. Потребителя както може да добавя елементи и количеството на елемента, също така може да редактира количеството, името или дата. Защо датата? Ами защото с датата потребителя да се ориентира, кога е заредил. Също така потребителя има възможност да изтрива елемент, когато е допуснал грешка. За да се подобри приложението съм добавил опция да може да се запазва като файл и след това да се зарежда като файл всичките елементи.

Списък с функционалността на приложението:

* Добавяне на елемент
* Редактиране на елемент
* Премахване на елемент
* Записване като файл
* Зареждане като файл
* Започване на нов файл

**Списък с използвана литература**

https://en.wikipedia.org/wiki/Microsoft\_Visual\_Studio

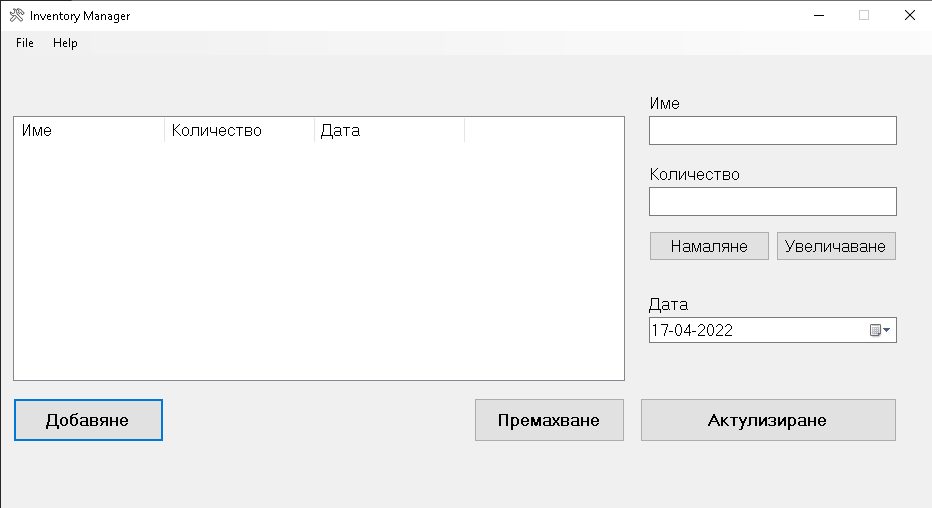
https://en.wikipedia.org/wiki/C\_Sharp\_(programming\_language)

https://www.youtube.com

https://stackoverflow.com

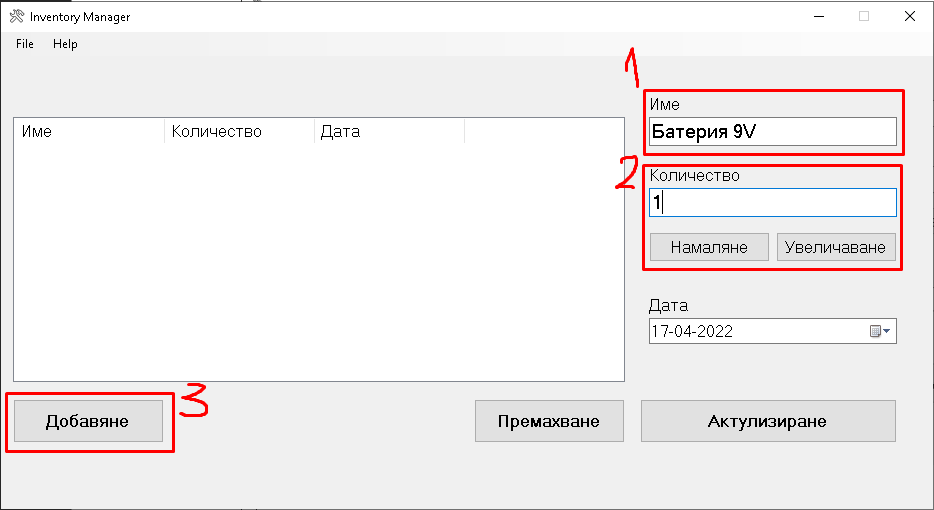
**Приложения**

След като отворим приложението ще ни се покаже този прозорец:



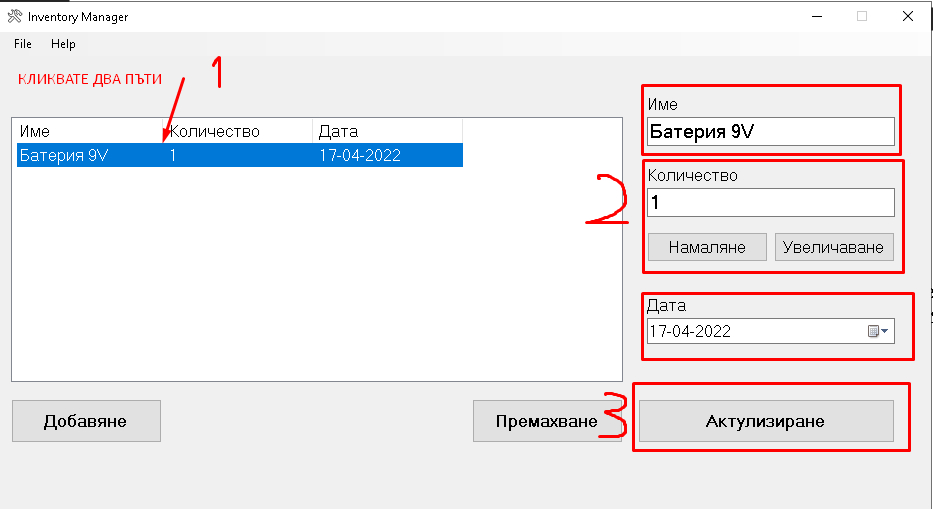
**Добавяне на елемент**

Въвеждаме в полето (Име) името на елемента, който искаме да добавим, след това записваме количеството в полето (Количество). След като сме попълнили можем да натиснем на бутона [Добавяне]. Елементът ни ще бъде показан в таблицата, която се намира в лявата и централна част на приложението.



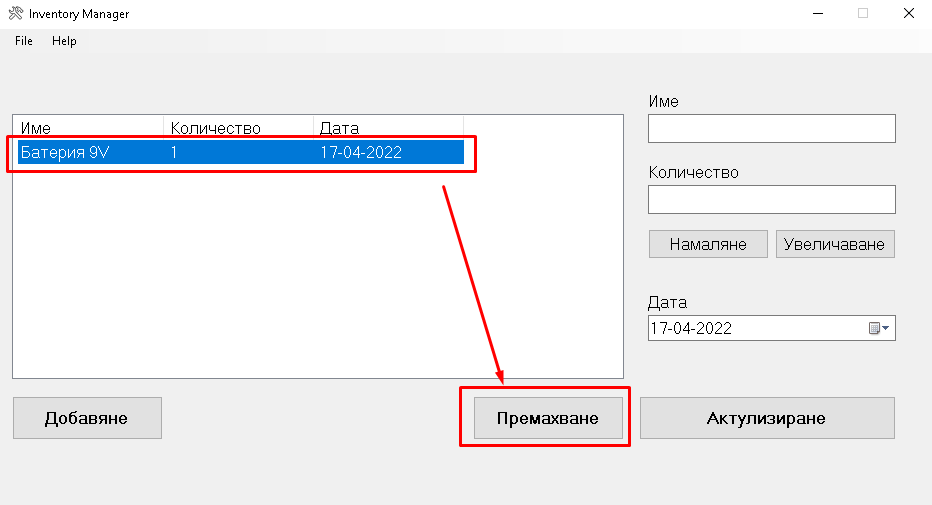
**Редактиране на елемент**

Вече след като имате добавен елемент и желаете да го редактиране (име, количество или дата). Кликвате два пъти върху него полетата ще се попълнят сами и можем да редактираме. След като сме готови натискаме бутона [Актулизиране].



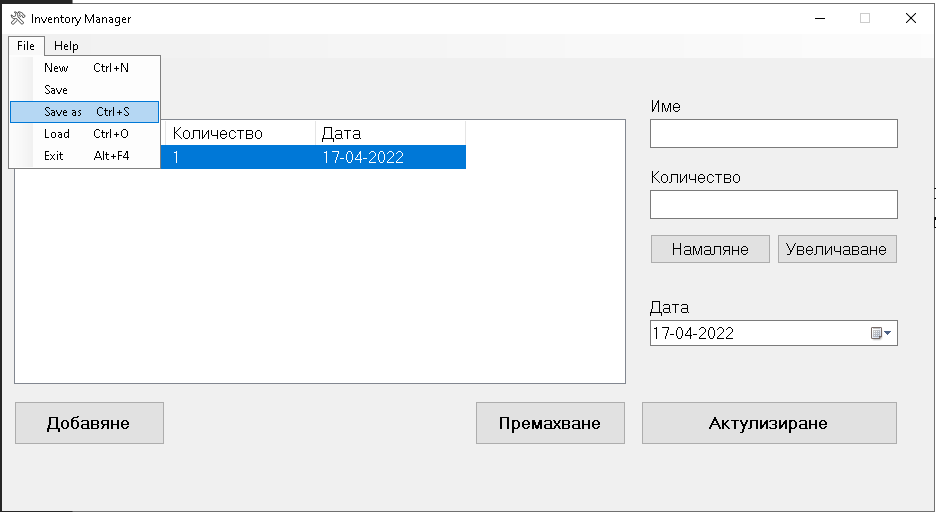
**Премахване на елемент**

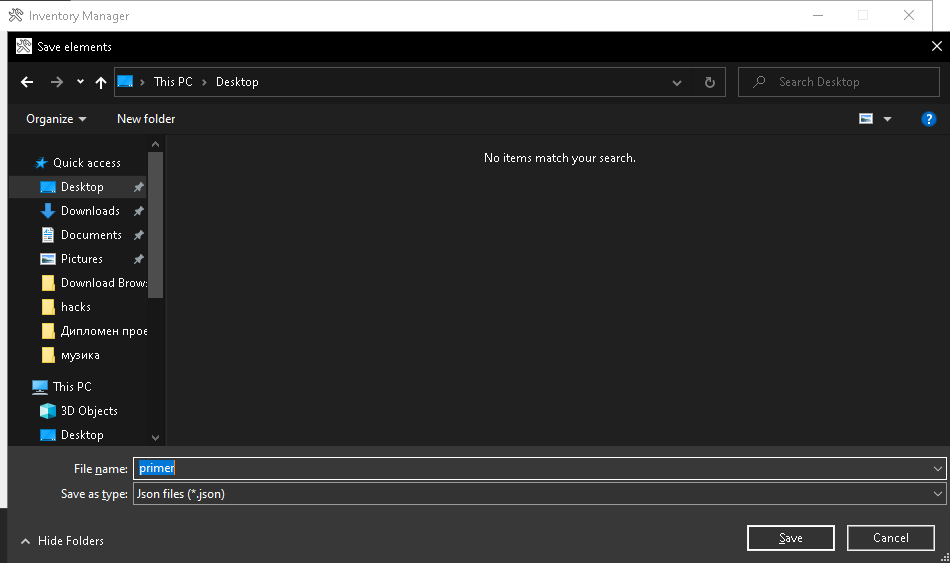
Искаме да изтрием някой елемент защото вече го нямаме в наличност или сме направили грешка. Кликваме един път върху елемента, който желаем да изтрием и натискаме бутона [Премахване].



**Записване като файл**

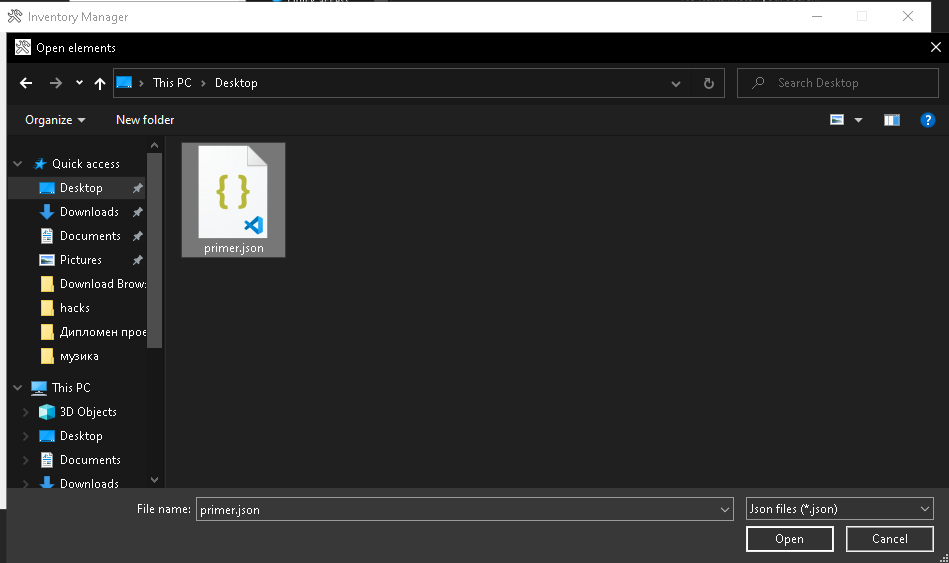
Вече сме вкарали в приложението елементи и искаме да ги запазим. Горе на менюто има бутон [Save As] натискаме го и избираме директория както и име на файла.





**Зареждане като файл**

В горното меню избираме File след това Load избираме файла и го отваряме.



**Започване на нов файл**

Искате да започнете нов файл, но не искате да затваряте приложението.

