Лекция 3

Визуален език C++ Builder и Java Builder.
Визуалните програмни езици C++ Builder и Java Builder са разработени също от софтуерната компания Borland. C++ Builder представлява C++ вариант на Delphi. Програмната среда е същата, както средата на Delphi. Разликата е, че програмният код с който се създава изходния код на приложението се записва с програмен език C++. Това означава, че начинът на създаване на приложения е аналогичен, както при Delphi, но детайлите, особено при програмиране на процедурите обработчици на събития се различава съществено. Това различие се определя от различните стилове на програмиране с Pascal и C++. Преминаването на работа от Delphi на C++ Builder, не представлява проблем за програмисти, познаващи програмните езици Pascal и S++.
Java Builder има доста по-различна структура и организация на работа от Delphi и C++ Bulider, поради различната структура и организация на езика Java. Java е език, който работи с използване на специална среда (‘машина на Java’) и това прави визуалната среда по-различна (параграф 6.3).

Други визуални програмни среди
Наред с Borland, визуални програмни среди се разработват и от други софтуерни компании, лидер сред които безспорно е Microsoft. Тя разработва визуални среди за C++ (Visual C++), Basic (Visual Basic), Dbase (Visual Dbase), Java и други. Най-известният софтуерен продукт, който съдържа няколко компилатора за различни програмни езици е Developer Studio. От няколко години, Microsoft пусна и нова програмна среда, така наречената .NET платформа. Тя също съдържа няколко компилатора, но работи с помощта на специална програмна среда CLR (Common Language Runtime). В нейната комплектация се съдържа и компилатор за най-новият програмен език, създаден за разработване и на програмни среди – езикът C# (Си шарп).

1. Developer Studio и Visual C++
Visual C++ е един от най-масово използваните средства за разработване на софтуерни продукти, при това софтуер от най-различен тип. С него се създават операционни системи, компилатори, приложни програми, игри, средства за обучение. Това е така, защото този език е много популярен, мощен и преносим между различни платформи. Visual C++ се състои от два софтуерни продукта: компютърният език C++ и развойната среда и набор от развойни средства, чрез които се създава програмно приложение. Програмната среда за разработване на визуални програмни приложения разработена от Microsoft се нарича Developer Studio. Това е съвкупност от програмни средства и инструменти в която са интегрирани компилатори за няколко програмни езика. Така например, когато се разглежда визуалният език Visual C++, спецификата на програмиране се съобразява с програмния език C++, но програмната среда е средата на Developer Studio. Така Microsoft, решава лесно въпросът за между езиковата съвместимост, особено използването на създадените вече визуални компоненти (независимо на кой от програмните езици).
Визуалната програмна среда Developer Studio е подобна на програмната среда Delphi, разгледана в предишния параграф. Съществуват обаче някои важни различия, които правят програмирането в двете платформи доста различни. Към най-важните различия между двете платформи трябва да отнесем:

- Програмната среда на Visual C++ е много по близо до обектната структура на езика C++, отколкото Delphi до Object Pascal или C++ Builder до обектният C++. По този начин, програмистите на Delphi или C++ Builder, участват по-малко в генерирането на програмния код на приложението (програмната среда го прави вместо програмистите).

Програмистите на Visual C++ участват доста по-активно в процеса на създаване на програмен код. Те контролират създаването на класове и променливи за отделните контроли и да ги използват при описване на процедурите за обработка на събитията
Този подход има предимства и недостатъци. Към предимствата, можем да отнесем това, че създаваният програмен продукт е в по-голяма степен управляем и достъпът до системните ресурси е по-лесен. Затова при създаване на приложения, използващи системните ресурси на компютърните системи (операционни системи, компилатори) тази програмна среда има предимства. Необходимостта от съобразяване с подробностите на обектното програмиране, от друга страна води до допълнителни трудности при разработване на чисто приложни програмни продукти – за научно-технически задачи, обработка на данни и други.

- Визуалните компоненти във Visual C++ се наричат контроли и са стандартни COM или ActiveX обекти. Всички визуални компоненти се разглеждат (третират) по един и същи начин, което унифицира работата с тях. В Delphi съществуват специално разработени компоненти, които се разполагат във формата много лесно и друга част (COM и ActiveX компоненти), които се вграждат посредством правилата на OLE спецификацията.

- Преди да се пристъпи към писане на програмен код в Developer Studio (Visual C++) е необходимо за всяка контрола (без някои статични контроли) да се дефинира променлива. Тези променливи участват в създаването на кода, като въвежданите данни от потребителя с тези контроли се присвояват на променливите. Съществуват правила за съставяне на наименованията на променливите.

2 .Net платформа на Microsoft.

Всеобщото мнение на разработчиците на програмни приложения е, че в близко бъдеще основни софтуерни платформи, към които ще бъдат ориентирани приложните програмни системи ще бъдат **.NET Framework** (.Net платформа) на Microsoft и **J2EE** (Java 2 Enterprise Edition) на Sun Microsystem. Двете платформи имат аналогична архитектура и организация на работа, но се различават по конкретната реализация. **J2EE** съдържа работната среда за програмния език Java и осигурява възможност за изпълнение на програмите върху различни компютърни системи и платформи. Основно внимание тук се отделя на **.NET Framework,** а за **J2EE** ще бъдат дадени само някои характерни особености**.**

В съвременния свят на компютърните мрежи към Web се свързват хора, използващи най-различни устройства, като персонални компютри, лаптопи, джобни компютри и клетъчни телефони. Това разнообразие на хардуер и софтуер изисква разработване на приложения, работещи на много операционни и програмни платформи. Възниква нуждата от разработване на приложения за разпределени среди във Web. Web услугите са разпределени приложения, които осигуряват достъп до кода от различни платформи и програмни езици.

За първи път на това предизвикателство се опита да отговори Sum Microsystems с разработване на платформено независимия програмен език Java. Той използва така наречената виртуална машина на Java за интерпретиране на програмния код на езика, която присъства във голяма част от използваните компютърни платформи. По-късно беше разработена и визуалната програмна среда J2EE.

Динамичното развитие на Internet технологиите и конкуренцията между софтуерните компании, непрекъснато поставят програмистите пред нови предизвикателства. Неотложна потребност в момента е разработване на еднаква основа, на която да се опират разработчиците на програмни приложения. Наскоро от Microsoft беше обявена новата концепция за начина по който потребители, приложения и устройства ще си

взаимодействат чрез Web. Основата на тази концепция е така наречената **.NET** платформа (‘дот нет платформа’), която все повече набира скорост и голяма част от програмистите се ориентират към разработване на приложения за нея.
Основната идея на тази платформа е, посредством XML Web услуги да се предостави лесен, надежден и ефективен начин за обмен на данни между потребителите и широк кръг от устройства, като персонални компютри, електронни бележници, мобилни телефони и други.
Архитектурните особености на .NET могат да се представят със следните характеристики:
- **Работа в разнородни обкръжения** (среди). С това се дава отговор на едно от най-големите предизвикателства пред съвременните програмни приложение – възможността за лесно интегриране в разнообразни обкръжения. Повечето големи организации и компании притежават голям спектър от терминали, локални мрежи, клиентски системи, Web базирани системи и други. Новите приложения трябва да взаимодействат освен с клиентски системи да обслужват и наследени данни, намиращи се на компютри от висок и среден клас, произведени от различни производители. Sun Microsystems първи отговориха на това изискване с многоплатформената архитектура на Java. В Java тази характеристика е известна като преносимост или платформена независимост и Microsoft трябваше да отговорят на това предизвикателство.
- Мащабируемост (разширяемост). Преди масовото навлизане на Internet, изчислителната среда на програмните приложение беше затворена система, поради ограничените ресурси и изискванията за достъп. Разширяемостта в тези системи лесно се контролираше поради факта, че програмните приложения се разработваха с години и имаше достатъчно време за планиране на възможностите за разширяване на системите. Масовото използване на Internet промени коренно представите за развитие на програмните приложение. Новите технологии изискват системите да се приспособяват към потребителски бази, включващо от по-малко от 100 до повече от 1000000 потребители. Това изисква съвсем нов подход и средства за разработване на програмните приложения.
- **Бързо разработване и внедряване.** Внедряването на Internet постави под съмнение традиционните подходи за разработване на програмни продукти. Компаниите не са готови да чакат с години разработването на дадено приложение, защото концепциите на работа се променят доста бързо и те трябва да реагират адекватно на това. От гледна точка на инвестициите беше поставен въпросът дали е разумно да се влагат инвестиции в приложения, които могат да остареят, преди да бъдат завършени. За да бъдат конкурентни разработчиците трябваше да създават приложенията ‘при поискване’ (just-in-time JIT), готови за незабавна употреба. За да се постигне това трябваше напълно да се преосмисли подхода към разработките на приложения.
- **Мрежови приложения**. Мобилните компютърни технологии (лаптопи, електронни бележници, джобни компютри) се развиха много бързо и поставиха сериозни изисквания пред разработчиците на приложения. Потребителите очакват да имат възможност да използват приложенията и услугите по всяко време без прекъсване. Това повиши многократно изискванията за работа на приложенията в мрежи от най различен тип.
Всички тези предизвикателства поставят проблеми, за които трябва да се намери подходящо решение. Архитектурата на .NET Framework дава решения на тези проблеми и дори осигурява много повече. Тя осигурява основа на разработчиците да създават добре обмислени програмни приложения.

Философията на .NET цели осигуряване на успешни бизнес решения. Ключов принцип за създаване на разпределени приложения с .NET Framework е логическото разделяне на приложението на три фундаментални слоя:

- Слой за представяне

- Слой за бизнес логика
- Слой за достъп и съхранение на данни.

Един прост модел на приложението се състои от *клиент*, или *слой за представяне,* който комуникира със средния слой. Той предоставя интерфейса за комуникация на потребителя с приложението. *Средният слой*, или *слоят за бизнес логиката*, се състои от сървър за приложението и модули съдържащи бизнес логиката (обработка, управление) на приложението. Средният слой от своя страна комуникира с базата данни посредством система от услуги за данни, които осигуряват поддръжката и съхранението на данните. Тази система от услуги образува *слоя за достъп и съхранение на данните*.

Аналогична философия и организация на приложенията има и в работната среда за Java – J2EE.

С навлизане на .NET Framework Microsoft представи набор от продукти и услуги, които образуват Фамилия .NET. Фамилията .NET включват .NET Enterprise Server, .NET server Windows XP и услугата .NET Passport. Освен това Visual Studio .NET е също неделима част от Фамилията .NET позволявайки разработването на приложения за работа в .NET среда.

Visual Studio .NET.

Visual Studio .NET осигурява среда за разработка на програмни приложения за .NET Framework. Visual Studio .NET интегрира най-доброто от програмните езици в един интерфейс, който може да се използва за разработване на корпоративни Web приложения и високопроизводителни настолни приложения. С помщта на Visual Studio .NET могат да създават различни приложения, като: конзолни приложения, приложения за Windows, приложения за ASP.NET (Active Server Pages), Web услуги и други.

Могат да се създават приложения и Web услуги с програмните езици предлагани от Visual Studio Net: Visual Basic.Net, Visual C#, Visual FoxPro, Visual C++.Net. Най-новите версии на визуалните програмни среди на Borland (Delphi, Builder) са разработени в съответствие с технологията .NET Framework и използват същия интерфейс както Visual Studio .NET. Следователно, програмните езици използвани в .NET Framework се допълват от Pascal и Java.

Основните характеристики и възможности на Visual Studio .NET са:

Реализиране на формуляри Web Forms. Visual Studio .NET предлага формуляри, с които могат да се създават Web приложения. Тези приложения могат да се използват с всеки браузер или мобилно устройство. За да се осигури съвместимост между устройствата, Web формулярите включват елементи на управление, които генерират HTML код съвместим с дадения браузер.

Реализиране на Web услуги. С помощта на Visual Studio .NET могат да се разработват и трасират Web услуги (Web Services).

Реализиране на формуляри Windows Forms. Visual Studio .NET поддържа Windows Forms (формуляри за Windows), които могат да се използват за създаване на Windows приложения за .NET Framework. Тези формуляри са обектно-ориентирани и служат за изграждане на слоя за предствяне в приложението (интерфейса за комуникация на потребителя с приложението).

Реализиране на проектно независим обектен модел. Visual Studio .NET като инструмент за разработване на приложения предоставя на разработчиците необходимите компоненти и обекти за комплектоване на формулярите и Web формите. За целта се използва проектно независим достъп до компонентите и събитията от развойната среда. Този модел включва компоненти, инструменти, редактори на код, трасиращи програми, кодови обекти, документи и дуги.

Разширено трасиране. Visual Studio .NET осигурява вградена трасираща програма за откриване на грешки в програми написани на различни езици. Като допълнение, може да се асоциира трасираща програма с вече стартирано приложение. Това позволява да се трасират няколко програми едновременно.
Програмиране на ASP.NET. Visual Studio .NET поддържа програмиране на ASP.NET. Така могат да се създават динамични Web приложения. Също така могат да се използват инструментите на Visual Studio .NET като визуален редактор за Web страници.
Разширена интегрирана среда за разработки. Интегрираната среда за разработки (IDE – Integrated Development Environment) на Visual Studio .NET се използва за всички негови програмни езици. Могат да се създават допълнителни инструменти за разширяване на възможностите на Visual Studio .NET.

Архитектура на .NET Framework
.NET Framework включва необходимите класове и модули за създаване на програмни приложения и Web услуги. .NET Framework съдържа два основни компоненти:

- Обща среда за изпълнение – CLR
- Библиотека с класове на .NET Framework

J2EE има аналогична архитектура, като роля на CLR играе виртуалната машина на Java (JVM), а библиотеката на J2EE съдържа аналогични декларации на класове.

CLR.
Основната част от обкръжението на .NET архитектурата се явява средата CLR (Common Language Runtime). Това е програмна среда работеща в реално време, която не само управлява изпълнението на кода, но и предоставя услуги, които улесняват програмирането. Компилаторите създават управляем код, който е предназначен за работа с тази среда за изпълнение. По този начин се създава междуезикова интеграция, междуезиково управление на изключенията, подобрена сигурност, проследяване на изпълнението на програмите (дебъгинг) и други. CLR осигурява управление на паметта, като специален модул за отстраняване на излишните обекти освобождава памет, премахвайки обекти и променливи, чиито жизнен цикъл е изтекъл. По този начин се отстранява един от най-сериозните недостатъци на програмирането със C++ - контролът за освобождаваната от обектите памет.
За да може CLR средата да осигури тези възможности, компилаторите които обработват първичния код на програмите, трябва да генерира така наречените мета-данни заедно с двоичния управляем код. Метаданните описват типовете, използвани в кода и се съхраняват заедно с компилирания код. Управляемият код, генериран от компилаторите работещи в .NET платформата не е собствен код, а код на междинен език (Intermediate Language – IL). Този IL код се използва като входен поток за изпълнение на процесите от CLR. Най-съществено предимство на IL кода е че той е платформено независим, всички компилатори създават унифициран за работа в CLR среда код. По това си качество, програмните езици подържани от .NET платформата приличат на езика Java, който работи с така наречен бета код. Платформената независимост е относителна, тъй като за да се изпълни дадено приложение е необходимо наличието на CLR среда.
Първичният език на CLR е **C#.** По-голяма част от поддържащата .NET архитектура е написана на този език. Поради тази причина, компилаторът на C# е най-тестваният и оптимизиран от всички компилатори включени в .NET платформата.
Обща езикова спецификация CLS (Common Language Specification). За да може CLR да интерпретира кода написан на различни програмни езици е необходимо да има определен набор от езикови характеристики и правила, които позволяват разработването на контролиран код, достъпен за използване от разработчиците на различни програмни

езици. По този начин, програмните езици, който подържа CLR се различават от малко стандартните езици. Най-често се въвеждат някои допълнения или ограничения в традиционните езици за да се осигури съвместимост при преобразуране на кода. Компонентите, които се придържат към правилата на CLS и използват само съвместими с нея възможности, се наричат съвместими с CLS компоненти.

Обща система от типове – CTS (Common Type System). CTS е система от типове, която подържа обектните класове и операции, които се използават в програмните езици. Тя определя как типовете се декларират, използват и управляват в средата за изпълнение на кода.Общата система от типове подържа използването на много програмни езици и дефинира правила, които програмният език трябва да следва, за да осигури взаимодействие между обектите, написани на различни програмни езици.

Garbage collection - GC (събиране на боклук). Garbage collection е механизъм, който позволява на компютърът да открие обект, който вече не се използва, и да освобождава паметта, заемана от този обект. Едно от предимствата на CLR е автоматичното управление на паметта, което използва този механизъм. Модулът за събиране на боклука управлява заемането и освобождаването на паметта за дадено приложение. Това означава, че няма нужда да се пише програмен код за управление на паметта. По този начин се избягват често срещаните грешки със заемане и освобождаване на памет при разположение на обектите от приложението.

MSIL (Microsoft Intermediate Language – Междинен език на Microsoft), JIT (Just in Time compailer – компилиране при поискване) и метаданни. Когато се създава изходен код в .NET Framework, компилаторът генерира код, който се нарича MSIL. Този код представлява набор от инструкции, до който се компилират програмите в .NET Framework. MSIL не зависи от процесора, който изпълнява този код. Той осигурява абстрактно хардуерно ниво, което позволява на приложението да бъде разработвано така, че да бъде независимо от операционната система.

Инструкциите на MSIL не могат да се изпълняват директно от процесора на компютърната система. Поради това задачата на CLR е да преобразува тези инструкции в инструкции за конкретния процесор. CLR не превръща целия код на приложението от MSLL инструкции в инструкции на микропроцесора в момента на зареждане на приложението. Инструкциите на MSIL се компилира, когато бъде извикана дадената функция от приложението. Този процес се нарича *компилиране при поискване*.

Файловете с изходен код се компилират със съответните компилатори и резултатът е код на MSIL. Този код е съвместим с .NET фаил .exe или .dll. Като допълнение компилаторът добавя метаданни във файловете .exe или .dll. Метаданните са помощна информация за компилирането на файловете и съдържат декларации за типовете, класовете и техните членове.

Библиотека с класове

Библиотеката с класове съдържа набор от обектно-ориентирани класове за многократна употреба. Класовете са съвместими с CLS. Затова тези класове могат да се използват от всеки програмен език, чиито компилатор е съвместим с CLS. Това позволява външни компоненти да бъдат интегрирани с класовете на .NET Framework.

Класовете на .NET Framework осигуряват основата, върху която се изграждат приложения, компоненти и елементи за управление на .NET. По-важните библиотеки с класове в .NET Framework са:

WinForms. Библиотека на .NET за работа с прозорци. Тя съответстав на JFC (Java Foundation Classes)/Swing или AWT (Abstract Windowing Toolkit) в език Java. Тези библиотеки са създадени за разработване на графическия интерфейс на приложенията. Практически всички приложения, използващи WinForms се разработват във Visual Studio

.NET. Главен клас в библиотеката WinForms е класyt Form1. Той е аналог на класа JFrame в средата за разработка J2EE. Във WinForms са включени всички елементи необходими за разработване на графически интерфейс: бутони, текстови кутии, списъци и т.н.
ASP.NET. Подържа функционалността на Web сървърите в .NET. Библиотеката ASP.NET WebForms се явява аналог на JSP (Java Server Pages). В ASP.NET влиза подържка на Web обслужване. Друг важен елемент се явява технологията HttpHandler, който има сходни функции като технологията на сервлетите (servlets).
ADO.NET (ActiveX Data Objects за .NET Framework). Тази библиотека осигурява средствата за работа с бази данни. ADO.NET е аналог на JDBC (Java Database Connectivity).

3 Език C#

Като част от .NET архитектурата на Microsoft, беше представен новият програмен език C# (Си шарп). Той беше анонсиран, като мощно средство за разработване на високопроизводителни Web приложения и компоненти – от XML базирани Web услуги до приложения работещи на системно ниво.
Какви са основните качества на езика C# и защо е необходим още един програмен език? Рекламният отговор на този въпрос е, че той е първичният език на .NET платформата и най-добре може да използва нейните възможности. Но освен чисто пазарните аргументи, за този език трябва да се каже, че той е модерен, добре структуриран и типово безопасен език.
Първоначално, Microsoft създава визуална версия за езика Java – Visual J++ през 1996 година, но след съдебен процес с Sun Microsystem се отказва от нея. През 2000 година е обявено разработването на .NET Framework и нов програмен език C#. С този език е трябвало да се обединят най-добрите страни на Java и C++. Платформата .Net и програмният език C# се появяват на пазара през 2002 година и бързо получават много голяма популярност и разпространение.
В последно време все по-често се чуват изказвания, че C# не е нищо повече от преименувана версия на Java и, че Microsoft се е отказала от наименованието Visual J++ поради загубено съдебно дело със Sun Microsystem. В действителност има много общи неща между C# и Java, но и различията са доста съществени. Те се определят от различията в програмните среди в които работят двата езика. Тъй като между .NET Framework (програмна среда за C#) и J2EE (програмна среда за Java) има много аналогични елементи, то и между двата езика съществуват доста аналогични конструкции. Освен това, и двата езика произлизат от C++ и много от програмните елементи на този език са намерили място в Java и C#. Очевидното сходство между Java и C# позволява на запознатите с Java програмисти лесно да усвоят езика C#.
Поради тази причина тук ще бъдат изложени само общите характеристики на езика C#, а основните програмни конструкции с малки модификации са както при език Java.
От друга страна, един от създателите на C# е Андерс Хейлсберг, водещ проектант на Turbo Pascal и Delphi. Това означава, че са взети пред вид най-добрите елементи от езика Pascal. Същото се отнася и за програмната среда Delphi, която е повлияла в много голяма степен интерфейса на Visual Studio .NET.
Програмният език C# произлиза от C и C++. Програмистите, които познават C++ не биха имали проблеми с програмиране на C#. Голяма част от операторите на C# са директно пренесени от C++. Ако не се обръща внимание на детайлите, една програма написана на C# изглежда много близка да програма от C++. Основните характеристики на език C# могат да се опишат, като се направи сравнение със C++:

- **Опростен**. Нещо, което със сигурност може да се припише на C++ е, че той е труден за изучаване. Това не може да се каже за C#, тъй като основната цел при

създаването му е била да се разработи ефективен и опростен език. Указателите са най-забележителната характеристика, която липсва при C#. По подразбиране се работи с управляем код, в който не дотам безопасните операции за директно адресиране в паметта на са позволени. Тази възможност, не е изключена напълно, но за да се използва, трябва да се превключи за работа в специален (необезопасен) режим на работа.
Премахнати са и някои неефективни и доста объркващи оператори от езика C++, като "::", "." и "->" с които се обозначават глобални променливи и принадлежност към структури. В C# се използва стандартният начин за обозначаване на вложени имена, както е например в Pascal, с dot оператора (".").
За подобряване на сигурността, в езика C#, целочислените и булевите типове данни са напълно различни типове. Това означава, че грешно присвоени данни в един оператор, веднага ще бъде санкционирано от компилатора. Вече не може да има противопоставяне на операторите за сравнение и оператора за присвояване в логическите изрази.

- **Модерен.** В C# са реализирани доста характеристики, които е трябвало допълнително да се програмират в C++ или въобще са липсвали. Например, финансовите типове данни са добре възприемана добавка, особено за разработката на приложни програми. В допълнение, тук могат много лесно да се създават нови типове данни, които са специфични за дадено приложение.

Тъй като указателите не са част от наличните инструменти на езика, управлението на паметта не е задължение на програмиста – интерпретаторът на средата за .NET платформата предоставя така наречения "колектор на остатъци" (garbage collector), който е отговорен за управлението на паметта.

- **Обектно ориентиран**. Като модерен програмен език C# не може да не подържа обектно ориентираните концепции, като капсулиране, наследяване, полиморфизъм. C# е изцяло обектен език и моделът на класовете е изграден върху основата на .NET Virtual Object System (VOS). Обектният модел е част от инфраструктурата и вече не е част от програмният език.

Основавайки се на обектно ориентираната концепция, в C# няма глобални функции, променливи и константи. Всичко трябва да бъде капсулирано в класа. Това прави кода на C# по-лесен за разчитане, както и спомага за намаляване на потенциалните възможности за конфликти при именоването на величините.
В C# не се подържат класове с множествена основа. Тези класове се използват рядко, и в повечето случаи доносят допълнителни проблеми отколкото ползи. Възможностите, които предоставят класовете с множествена основа могат да се реализират с други прийоми

- **Типово безопасен**. Отново можем да се позовем на указателите. Когато се декларира указател в C++, той може да се насочи към всеки тип данни, създавайки по този начин всевъзможни абсурдни ситуации. Доколкото всяка информация в паметта може да се интерпретира по различен начин, програмата работи "някак си" при адресиране на указател към неподходящи данни, но това не е желателно за сигурността на програмата.

Поради тези причини, C# реализира стриктно съблюдаване на типовете на данните. Това се постига чрез въвеждане на някои правила при употребата на променливи и константи, различни от тези в C++. Такива са например невъзможността за използване на неинициализирани променливи, проверката за границите на масиви и други.
Като обобщение, може да се каже, че езикът C# произлиза от C и C++ и е създаден за корпоративните разработчици, които са готови да жертват част от моща на C++ за сметка на по-голямото удобство и продуктивност. В това отношение, той се придвижва в посока към простотата и продуктивността на Pascal и Object Pascal. C# е модерен, опростен, обектно ориентиран и типово безопасен. Той заимства много от C и C++ и същевременно е много различен. По отношение на обектните си характеристики, той се приближава до

Java, но има по-големи възможности при разработване на визуални приложения и използване на богатите възможности на COM и други визуални библиотеки.