

ЕЛЕМЕНТИ НА ИНТЕГРИРАНИТЕ СРЕДИ

Гл. ас. д-р Елица Ибрямова

Elbryamova@ecs.uni-ruse.bg

ИНТЕГРИРАНА СРЕДА

Основното предназначение на всяка интегрирана среда е:

- да улесни работата на програмиста;
- да увеличи продуктивността на неговия труд в процеса на изграждане на съвременните сложни и мултифункционални приложения.



ОСНОВНИ ХАРАКТЕРИСТИКИ

Основните характеристики, които определят качествата на една интегрирана среда са три:

- **вътрешна архитектура на ИС;**
- **програмните парадигми, характерни за езиците й за програмиране;**
- **типът и качествата на инструментите, интегрирани в средата.**

ВЪВЕДЕНИЕ

ИС включват следните **основни** компонента:

- 1) Текстови редактори на програмен код;**
- 2) Езикови транслатори (компилатори или интерпретатори);**
- 3) Свързващи редактори (linker);**
- 4) Библиотеки с програмен код;**
- 5) Инструменти за проверка за грешки (дебъгери);**
- 6) Среда за изпълнение на програмите с цел тяхното тестване (run-time environment).**
- 7) Език за програмиране**

ВЪВЕДЕНИЕ

Съвременните ИС често включват следните **допълнителни** инструменти:

- Множество инструменти за графичен интерфейс за обработка на специализирани видове обекти (мултимедия, графика, данни и др.);
- Инструмент за контрол на версиите;
- Инструмент за управление на проекти;
- Библиотеки с дизайнерски модел;
- Инструменти за интегриране на нови компоненти в средата;
- Генератор на документация;
- Инсталатори и други инструменти, необходими за разработването на съвременни софтуерни приложения.

ВЪВЕДЕНИЕ

Някои ИС също могат да включват:

- браузър за класовете (*class browser*);
- браузър на обектите (*object browser*), и/или
- диаграма на йерархията на класа (*class hierarchy diagram*):

инструменти, които помагат на разработчиците да визуализират структурата на обектно-ориентирания програмен код.

ИС могат да се използват за всяка методология за разработване, включително *agile* и *waterfall software development*.

ОСНОВНИ ИНСТРУМЕНТИ НА ИНТЕГРИРАНИТЕ СРЕДИ



ТЕКСТОВ (КОДОВ) РЕДАКТОР *CODE EDITOR*

- Всяка ИС има **редактор на кода** (текста), предназначен за писане и манипулиране на изходния код.
- Някои ИС могат да имат визуални компоненти за drag-and-drop контроли, но повечето имат прост интерфейс с подчертаване на синтаксиса, специфичен за езика.

ТЕКСТОВ (КОДОВ) РЕДАКТОР

Характеристики на текстовия редактор (ТР):

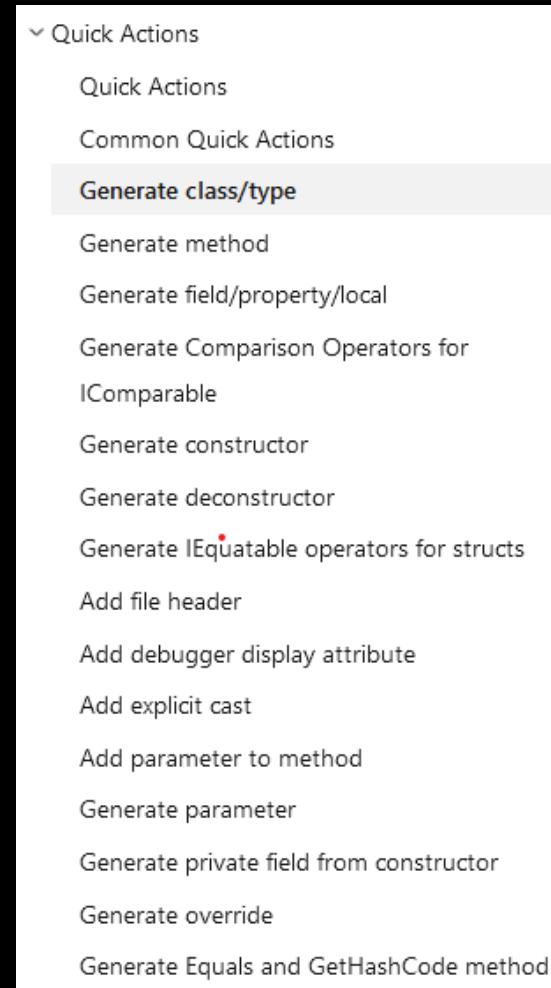
- ТР поддържа функцията **IntelliSense (Content Assist)**, която включва така нареченото подчертаване на синтаксиса и попълване на код;
- ТР може да се използва за всички програмни езици, поддържани от Visual Studio плюс XML, CSS и JavaScript, когато създавате уеб сайтове и уеб-базирани приложения;
- ТР също поддържа отметки в кода за по-бърза навигация (Visual Studio);
- Свиване на сегмента на кода;
- Функция за динамично търсене в кода.

ТЕКСТОВ (КОДОВ) РЕДАКТОР

- Чрез текстовия редактор могат да бъдат създавани шаблони от определени фрагменти на кода, които се ползват по-често (**snippets**).
- Тези шаблони могат да бъдат модифицирани и използвани във всеки следващ проект.
- За управление на създадените шаблони има добавен специален инструмент.
- Тези инструменти се визуализират като плаващи прозорци, които може да се настройт, да се скриват автоматично, когато не се ползват или да се прибират в страни от работния екран.

ТЕКСТОВ (КОДОВ) РЕДАКТОР

- ТР на Visual Studio включва и компилиране на заден фон (**background compiler**) на кода.
- Функция на компилирането на заден фон е да довършва кода още докато потребителят го създава, за да може да предостави обратна връзка за възможни грешки в синтаксиса и компилацията.



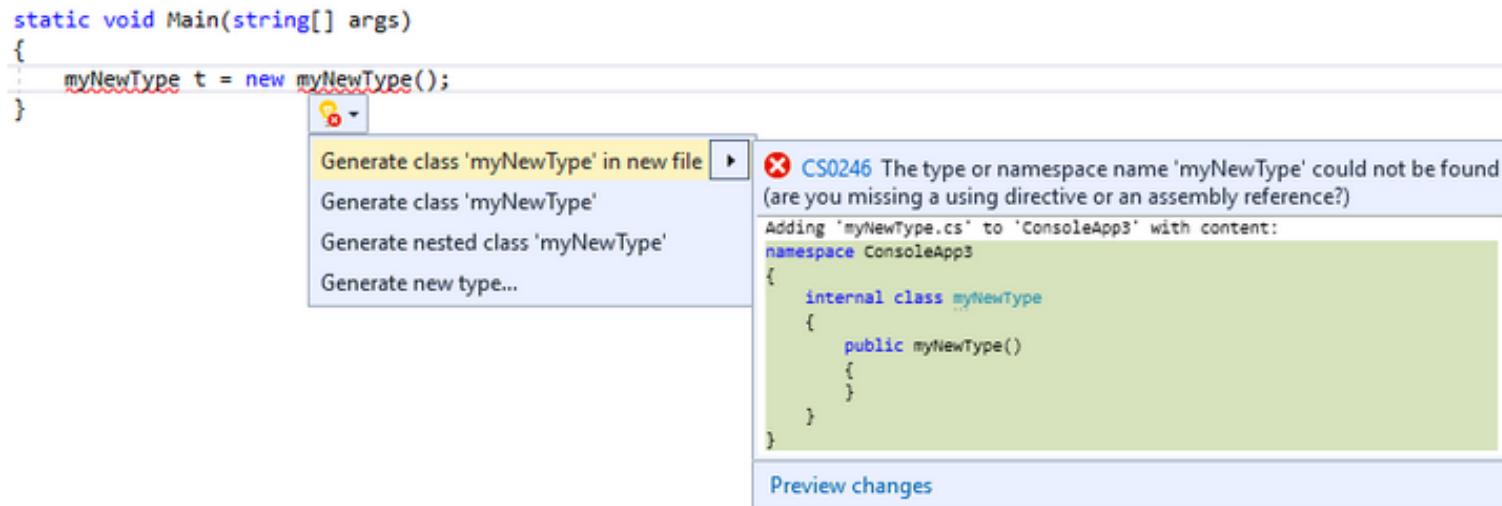
ТЕКСТОВ (КОДОВ) РЕДАКТОР

- Keyboard

- Press **Ctrl+.** to trigger the **Quick Actions and Refactorings** menu.

- Mouse

- Right-click and select the **Quick Actions and Refactorings** menu.
 - Hover over the red squiggle and click the  icon that appears.
 - Click the  icon that appears in the left margin if the text cursor is already on the line with the red squiggle.



```
static void Main(string[] args)
{
    myNewType t = new myNewType();
}
```

The screenshot shows a code editor with the following code:

```
static void Main(string[] args)
{
    myNewType t = new myNewType();
}
```

A red squiggle is underlined under the word `myNewType`. A context menu is open at the cursor position, with the "Generate class..." option highlighted. To the right of the menu, a tooltip displays error information and a preview of generated code:

CS0246 The type or namespace name 'myNewType' could not be found
(are you missing a using directive or an assembly reference?)

Adding 'myNewType.cs' to 'ConsoleApp3' with content:

```
namespace ConsoleApp3
{
    internal class myNewType
    {
        public myNewType()
        {
        }
    }
}
```

Preview changes

ТРАНСЛАТОРИ

- Транслаторът е програмна система, която превежда програми от един език във функционално еквивалентни програми на друг език.
- Когато входният език е от високо ниво пълната трансляция може да се състои от няколко етапа, с няколко междинни езика.
- В зависимост от начина на работа транслаторите са два вида:
 - 1) Компилиращи (компилятори)
 - 2) Интерпретиращи (интерпретатори)

КОМПИЛАТОР (COMPLIER)

- Компилаторите са програми, които превеждат програмния език във форма, която машините могат да обработват, например двоичен код.
- При компилиацията се извършва пълен превод на входната програма в програма на машинен език или машинно ориентиран език.
- След това обектната програма може да бъде изпълнена произволен брой пъти без помощта на **транслатора**, т.е. фазите на трансляция и изпълнение са напълно разграничени.

КОМПИЛАТОР

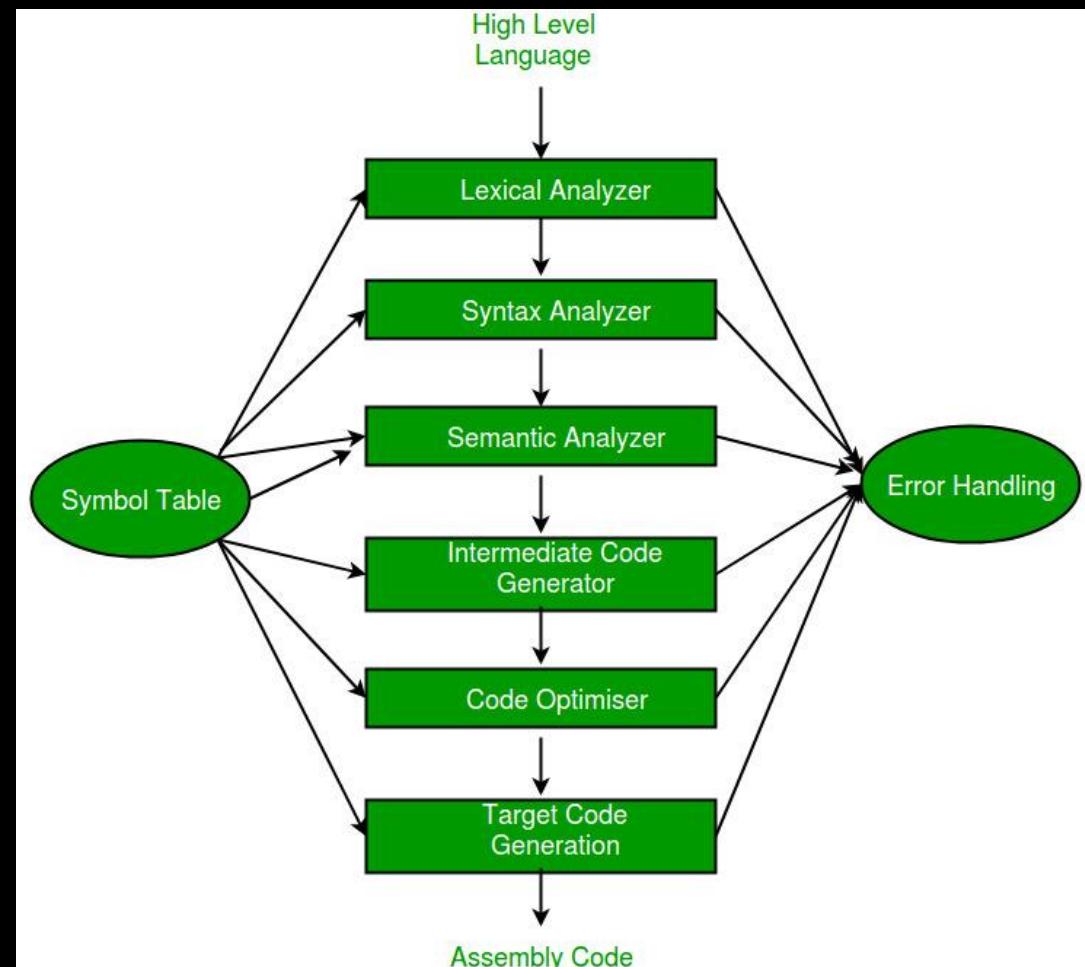
Компилаторът извършва следните операции:

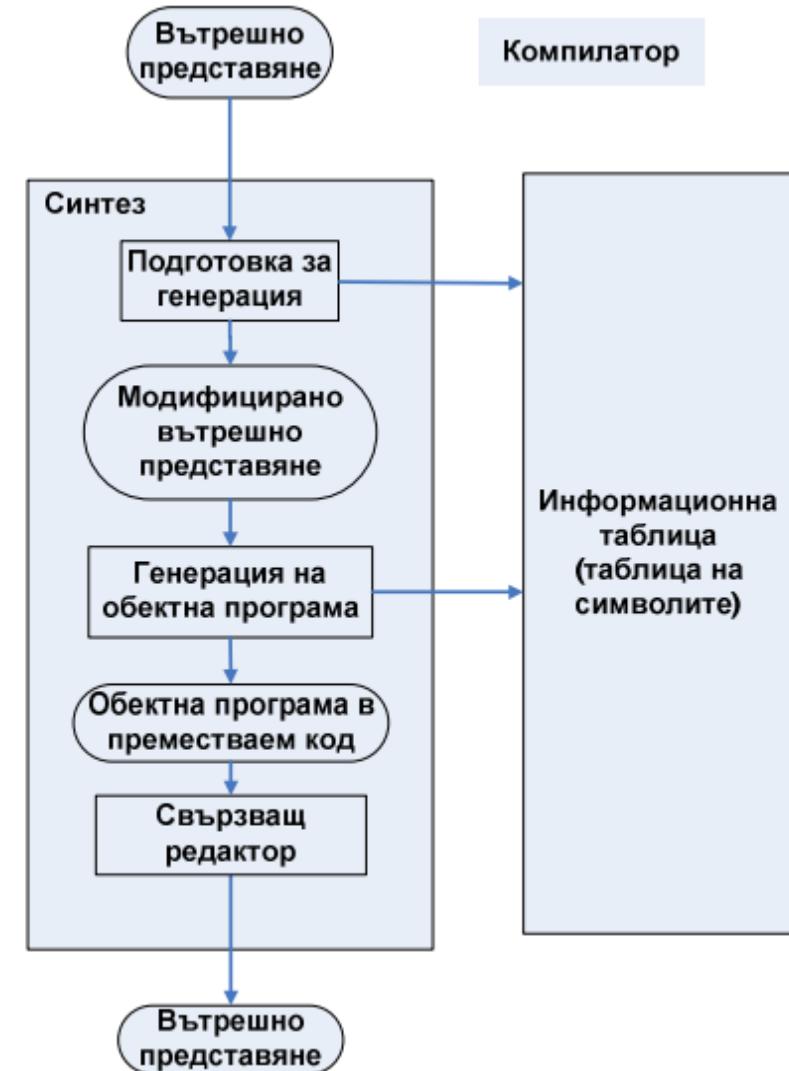
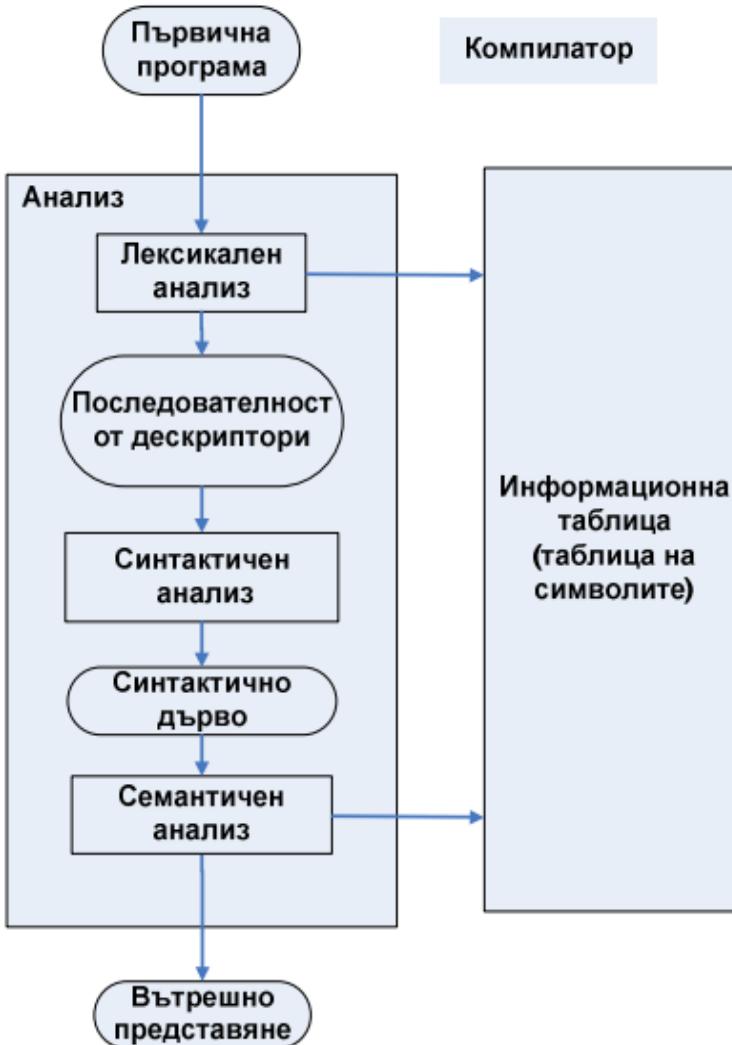
- предварителна обработка;
- лексикален анализ;
- разбор;
- семантичен анализ (превод, насочен към синтаксис);
- преобразуване на входни програми в междинно представяне;
- оптимизация на кода и
- генериране на код.

Грешките в програмата, причинени от неправилно поведение на компилатора, могат да бъдат много трудни за проследяване и заобикаляне; следователно, изпълнителите на компилатора инвестират значителни усилия, за да гарантират коректността на компилатора.

КОМПИЛАТОР

Таблицата със символи е важна структура от данни, създадена и поддържана от компилатора с цел съхраняване на информация за появата на различни данни като **имена на променливи, имена на функции, обекти, класове, интерфейси и т.н.** Таблицата със символи се използва както за анализ, така и за синтез от компилатора.





ДЕКОМПИЛАТОР (DECOMPILER)

- Програма, която превежда от език на ниско ниво на по-високо ниво, е декомпилатор.
- Може да се използва за възстановяване на изгубен изходен код, а също така е полезен в някои случаи за компютърна сигурност, оперативна съвместимост и коригиране на грешки .
- Декомпилаторите не са в състояние да възстановят перфектно оригиналния изходен код, поради което често създават **obfuscation code**. Въпреки това декомпилаторите остават важен инструмент в **обратното инженерство** на компютърен софтуер .
- Успехът на декомпилиацията зависи от количеството информация, налична в кода, който се декомпилира, и сложността на анализа, извършен върху него.
- **.NET Reflector** е клас браузър , декомпилатор и статичен анализатор за софтуер, създаден с .NET Framework.

ИНТЕРЕЧО

- **Obfuscation:** замъгляване, прикриване
- **Obfuscation software development:** is the act of creating source or machine code that is difficult for humans or computers to understand.
- Програмистите могат умишлено да объркат кода, за да прикрият неговата цел (сигурност чрез неизвестност) или неговата логика, основно, за да предотвратят подправяне, да възпрат обратното инженерство или дори да създадат пъзели или - развлекателно предизвикателство за някой, който чете изходния код.

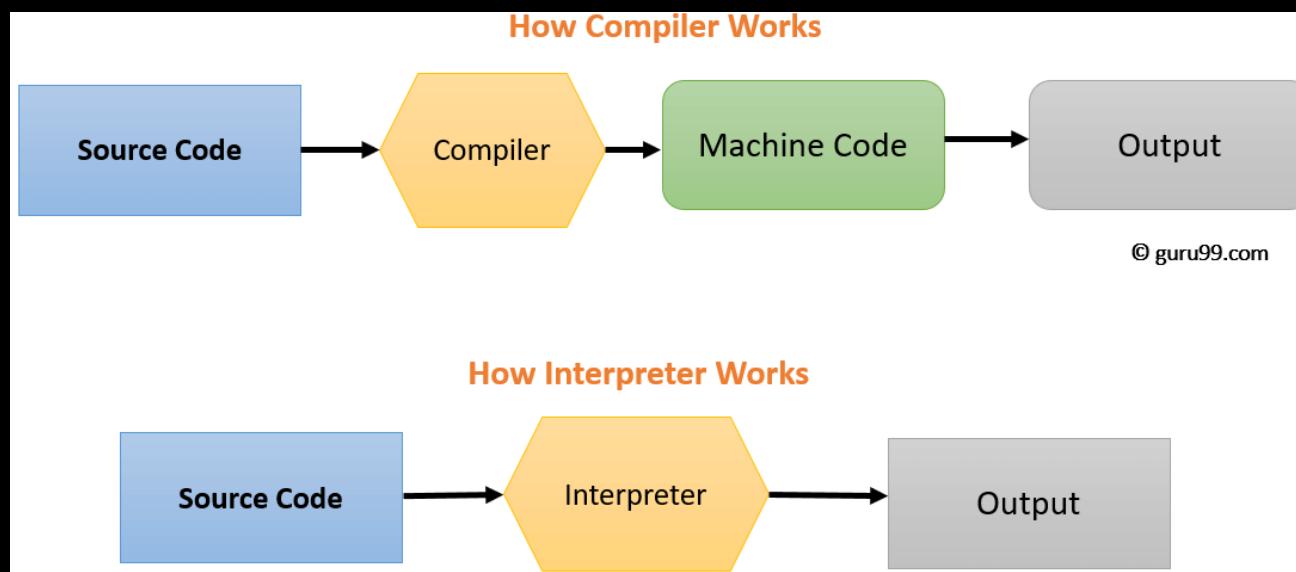
ИНТЕРЕСНО

Според Ник Монтфорт техниките могат да включват:

1. объркане на наименованията, което включва **именуване на променливи по безсмислен или измамен начин**;
2. **объркане на данни/код/коментар**, което включва реализиране на действителния код, но той да изглежда като коментари или объркане на синтаксиса с данни;
3. **двойно кодиране**, което може да бъде показване на код в поетична форма или интересни форми.

ИНТЕРПРЕТАТОР (INTERPRETER)

Интерпретаторът е компютърна програма, която директно изпълнява инструкции, написани на език за програмиране или скриптове, без да изисква преди това да са били компилирани в програма за машинен език.



При интерпретацията всяка прочетена инструкция се изпълнява веднага след трансляцията си, т.е. преди да се транслира следващата инструкция.

ПРЕДИМСТВА И НЕДОСТАЪЦИ

Компилатори

- 1.) По-голямо бързодействие при изпълнение, защото вече има готова обектна програма (липсва транслацията) и тя е обикновено оптимизирана;
- 2.) По-малки изисквания за памет при изпълнение – при изпълнение транслаторът не е зареден в оперативната памет.

На практика по-често се използват компилатори.

Интерпретатори

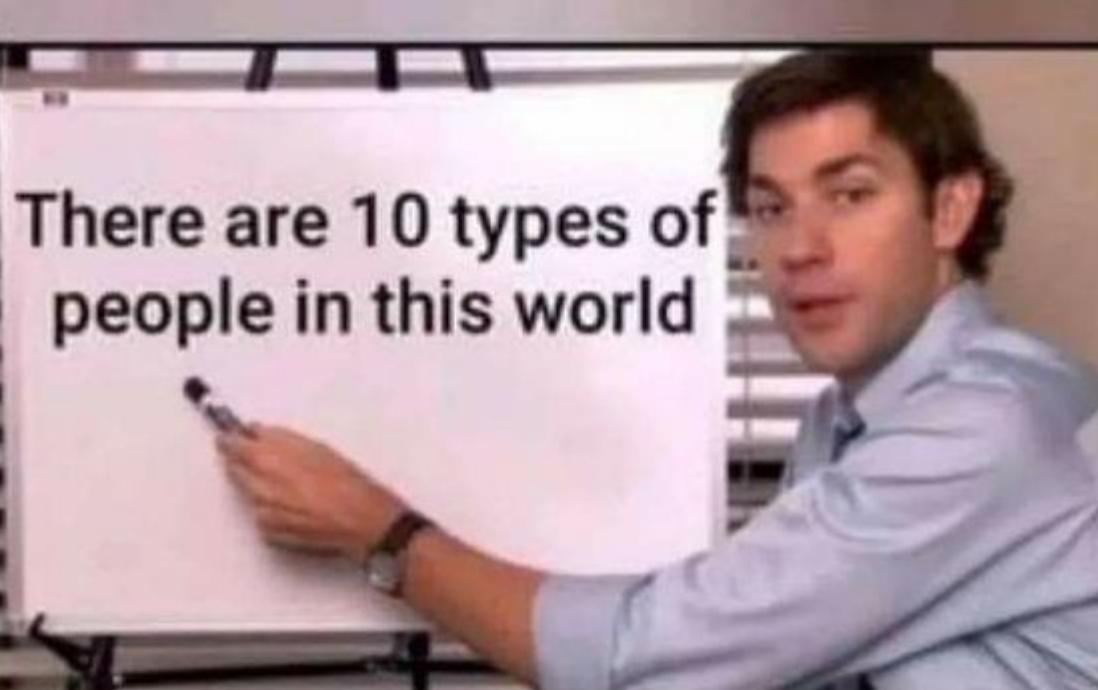
- 1.) Значително по-прости от компилаторите (не създава цяла обектна програма, няма оптимизация и т.н.) Самите те заемат малко памет, използват по-малко памет.
- 2.) Самата транслация е по-бърза, но не и изпълнението.
- 3.) Не се транслират части от входната програма, през която не се минава по време на изпълнение (но при цикли и многократно изпълнение се транслира всеки път).
- 4.) Много удобни са за настройка на програмите – лесно се предават съобщения за грешки директно в терминологията на първичната програма, изпълнението на програмата не изиска претранслиране на цялата програма, може да се следят стойностите на програмата във всеки момент.

КАК РАБОТЯТ КОМПИЛАТОРИТЕ И ИНТЕРПРЕТАТОРИТЕ

Light

KNOWLEDGE IS LIFE AND CURE





There are 10 types of people in this world



Those who understand
binary and
those who not

BASIS FOR COMPARISON	COMPILER	INTERPRETER
Input	It takes an entire program at a time.	It takes a single line of code or instruction at a time.
Output	It generates intermediate object code.	It does not produce any intermediate object code.
Working mechanism	The compilation is done before execution.	Compilation and execution take place simultaneously.
Speed	Comparatively faster	Slower
Memory	Memory requirement is more due to the creation of object code.	It requires less memory as it does not create intermediate object code.
Errors	Display all errors after compilation, all at the same time.	Displays error of each line one by one.
Error detection	Difficult	Easier comparatively
Pertaining Programming languages	C, C++, C#, Scala, typescript uses compiler.	PHP, Perl, Python, Ruby uses an interpreter.

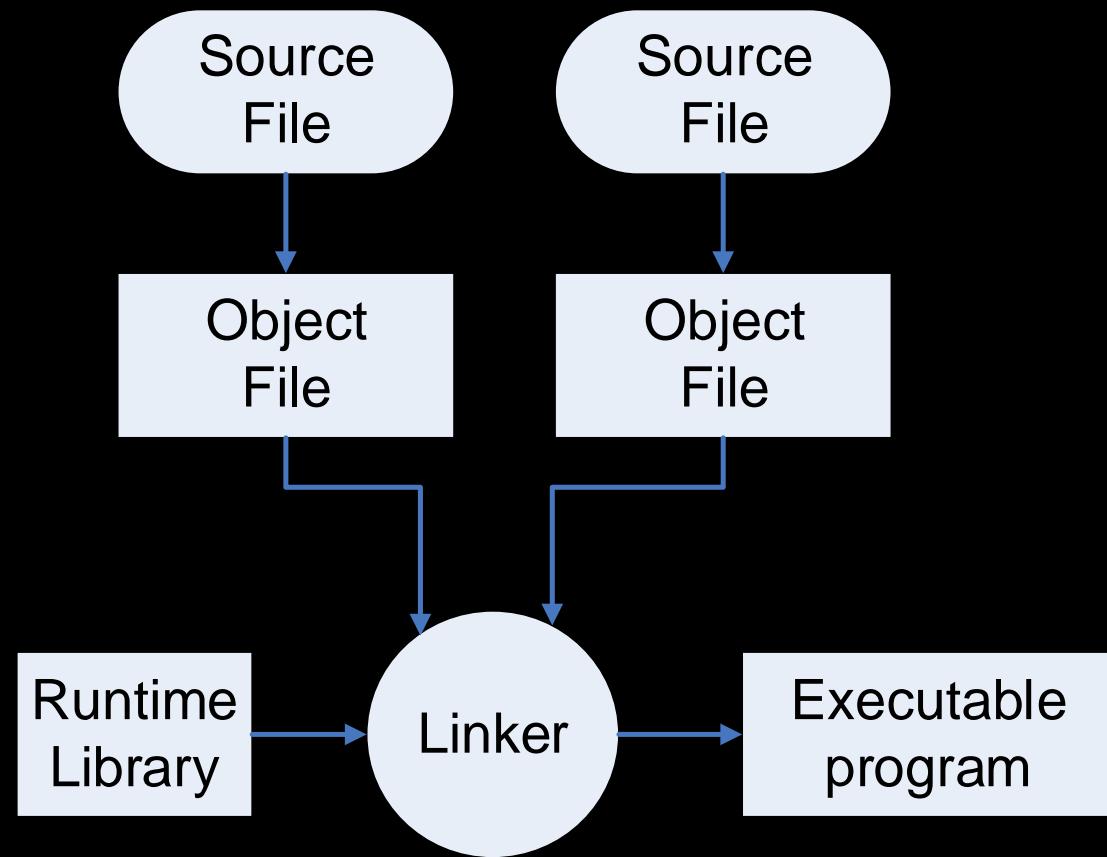
INTERPRETER

ЛИНКЕР (LINKER OR LINK EDITOR)

- Много големите програми се състоят от отделни обособени части, които се пишат от различни колективи от програмисти.
- **След компилиране на отделните части**, те трябва да се свържат в едно цяло, при това включвайки доста сложни връзки между тях. **В една обща изпълнима програма се обединяват няколко обектни модула и библиотеки.**

ЛИНКЕР (LINKER OR LINK EDITOR)

Линкер (свързващ редактор) е програма, която взема един или повече обектни файлове, генериирани от компилатора и ги комбинира в един изпълним файл, библиотечен файл или друг обектен файл.



БИБЛИОТЕКИ (LIBRARIES)

- Библиотеката е колекция от предварително компилирани подпрограми, които една програма може да използва.
- В тях най-често се включват библиотеките от програми, в които се съхраняват обектни модули за някои често използвани процедури, които автоматично се извличат оттам чрез свързващи редактори.
- Библиотеките съдържат подпрограми, които се използват най-често, например, подпрограми за вход/изход, тригонометрични функции, подпрограми за изчертаване на графични примитиви - линия, дъга, правоъгълник и т. н.

ДЕБЪГЕР (DEBUGGER)

- Инструментите за отстраняване на грешки помагат на потребителите при идентифициране и отстраняване на грешки в изходния код. Те често симулират сценарии от реалния свят, за да тестват функционалността.
- Програмистите и софтуерните инженери обикновено могат да тестват различните сегменти на кода и да идентифицират грешки, преди приложението да бъде пуснато.
- Точката на прекъсване (**breakpoint**) е умишлено спиране или поставяне на пауза в програма, поставена на място за отстраняване на грешки. Също така понякога се нарича просто пауза.

ДЕБЪГЕР

Следните видове грешки могат да възникнат по време на разработването на програмата:

- **Грешка при писане на програмата** – отстраняват се с текстов редактор;
- **Грешки по време на компиляция** (синтактични грешки) – откриват се от транслатора;
- **Грешки по време на изпълнение** (Run-time error) – откриват се с дебъгер;
- **Грешки в алгоритъма** (логически грешки) – най-трудни за откриване, откриват се по време на тестване на програмата и в най-лошия случай – при експлоатация.

ДЕБЪГЕР

- За да нямате „Грешки по време на изпълнение“:
- Избягвайте да използвате променливи, които не са инициализирани.
- Проверете всяко едно появяване на елемент от масива и се уверете, че не е извън границите.
- Избягвайте да декларирате (отделяте) твърде много памет. Проверявайте за ограничението на паметта.
- Избягвайте да декларирате твърде много стекова памет. Големите масиви трябва да се декларират глобално извън функциите.
- Използвайте `return` като крайно състояние.
- Избягвайте препращане към свободна памет или нулеви указатели.

ГРЕШКИ В АЛГОРИТМЕ



RUN-TIME ENVIRONMENT

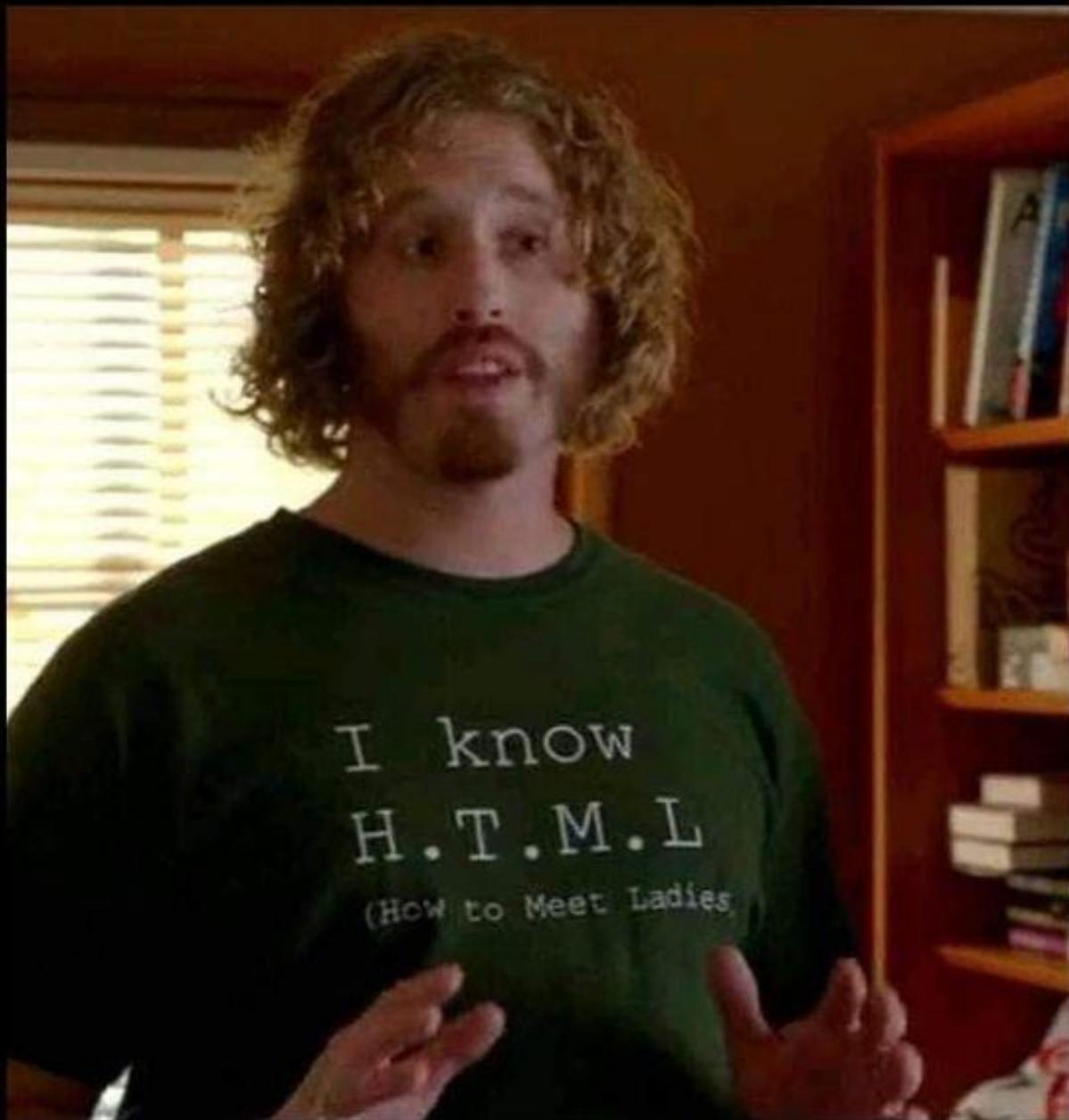
- Среда за изпълнение е приложение или софтуер, предоставена от операционната система.
- В среда на изпълнение приложението може да изпраща инструкции или команди до процесора и да осъществява достъп до други системни ресурси като RAM, което иначе не е възможно, тъй като повечето използвани езици за програмиране са езици на високо ниво.

БИБЛИОТЕКИ

- Библиотеките съхраняват обектни модули за някои често използвани процедури, които автоматично се извличат оттам чрез свързващи редактори.
- Библиотеките съдържат подпрограми, които се използват най-често, например, подпрограми за вход/изход, тригонометрични функции, подпрограми за изчертаване на графични примитиви - линия, дъга, правоъгълник и т. н.

ПРОГРАМЕН ЕЗИК (ПЕ)

- Езикът за програмиране е официален език, който включва набор от инструкции, които произвеждат различни видове продукция.
- Езиците за програмиране се използват за реализиране на алгоритми.
- Повечето езици за програмиране се състоят от инструкции за компютри. Има програмируеми машини, които използват набор от конкретни инструкции, а не общи езици за програмиране.



17:06

I started to develop a website
with HTML, and this is very
good

Do you want to see?



Yes, send me the link.

<http://localhost/index.html>

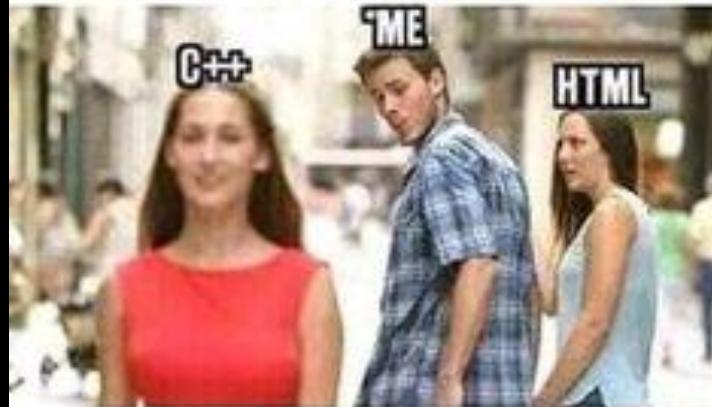
17:28



Your website looks like mine.

DevVerse English

Programmers be like..



ДОПЪЛНИТЕЛНИ ИНСТРУМЕНТИ НА ИТЕГРИРАНИТЕ СРЕДИ

Microsoft Visual Studio



ДИЗАЙНЕРИ

- Windows Forms Designer
- WPF Designer
- Web designer/development
- Class designer
- Data designer
- Mapping designer

WINDOWS FORMS DESIGNER

- Дизайнерът Windows Forms се използва за създаването на приложения с графичен потребителски интерфейс (на английски: Graphical User Interface, GUI).
- Позволява да се добавят и оформлят различни менюта и бутони.
- Полетата, показващи някакви данни, могат да бъдат свързвани с различни източници на данни, като бази от данни или заявки. Тези полета се добавят чрез изтегляне от прозореца **Data Sources** върху създадения формуляр.
- Потребителският интерфейс задължително се свързва с програмен код, за създаването на приложение.

WINDOWS PRESENTATION FOUNDATION DESIGNER

- WPF дизайнърът, наричан още Cider, базиран на Windows Presentation Foundation, за първи път се появява във Visual Studio 2008.
- Използва се за създаване на потребителски интерфейс.
- Поддържа всички WPF функционалности, включително възможността за свързване на данни (на английски: *data-binding*) и автоматичното управление на изгледа.
- WPF дизайнърът генерира XAML код за потребителския интерфейс. Генерираният файл е съвместим с Microsoft Expression Design, продукт ориентиран към дизайнерите.
- XAML кода се свързва с кода на потребителя чрез модела *code-behind*.

CLASS DESIGNER

- Class дизайнерът се използва за създаването и редактирането на класовете (това включва техните елементи и достъпа до тях) чрез UML моделиране.
- Class дизайнерът може да генерира C# и VB.NET очертания за класовете и методите.
- Също така може да генерира класови диаграми от ръчно написани класове.

MAPPING DESIGNER

- След Visual Studio 2008, mapping дизайнерът се използва от LINQ to SQL за създаването на връзки между схеми на базите данни и класовете, които съдържат данни.
- В настоящето съществува ново решение, създадено чрез ORM подход — ADO.NET Entity Framework, което подобрява остарялата технология.

First Law of Software Quality

errors = (more code)²

$$e = mc^2$$

ORACLE

redhat

ING

Google

ДРУГИ ИНСТРУМЕНТИ

- Open Tabs Browser
- Properties Editor
- Object Browser
- Solution Explorer
- Team Explorer
- Data Explorer
- Server Explorer
- Dotfuscator Software Services Community Edition
- Text Generation Framework

OPEN TABS BROWSER

- Open tabs браузърът се използва за извеждане на всички отворени табове и за улесняване на превключването между тях, чрез клавишната комбинация CTRL+TAB.

PROPERTIES EDITOR

- Properties редакторът се използва за редактирането на настройките в GUI панела, вложен във Visual Studio.
- Той съдържа списък на всички функционалности, за всички обекти, включително класове, форми, уеб страници и др.

OBJECT BROWSER

- Object браузърт е търсачка на именно пространство (от английски: namespace) и обекти в библиотеката на Microsoft.NET.
- Може да се използва за търсене на namespace (които са подредени йерархично) в управляемите асемблита.
- Йерархията може да се отрази на организацията на файловата система.

SOLUTION EXPLORER

- На езика на Visual Studio, Solution-а представлява набор от програми файлове и други ресурси, които се ползват за създаването на приложение.
- Файловете в solution-а са подредени йерархично. Solution експлорерът се използва за управление и търсене на файловете в solution.

TEAM EXPLORER

- Team Explorer се използва за интегриране на възможностите на Team Foundation Server.
- Той дава възможност да преглеждате и управлявате отделните работни елементи (включително грешки, задачи и други документи), както и за разглеждане на TFS статистика.
- Тя е включена като част от TFS системата и също е на разположение за изтегляне за Visual Studio отделно.
- Team Explorer се предлага и като самостоятелен среда само за достъп до TFS услуги.

DATA EXPLORER

- Data Explorer се използва за управление на бази данни инстанцирани на Microsoft SQL Server.
- Той позволява създаване и променяне на таблици от бази данни, чрез използването на Data дизайнърът или чрез издаването на T-SQL команди. Може да се използва също така за създаване на заявки и процедури, чрез T-SQL или чрез код управляван чрез SQL CLR.
- Експлорерът поддържа функциите дебъгване и IntelliSense.

SERVER EXPLORER

- Инструментът Server Explorer се използва за управление връзките на базата данни с достъпния компютър.

DOTFUSCATOR SOFTWARE SERVICES COMMUNITY EDITION

- Visual Studio включва бесплатна „лека“ версия на продукта на PreEmptive Solutions' - Dotfuscator.
- Целта на приложението е да анализира програмите, за да ги направи по-малки по-бързи и по трудни за декомпилация. Dotfuscator кодира кода, като преименува променливи, методи и криптира стринговите литерали. Това са само някои от използваните техники.
- След Visual Studio 2010, версииите на Dotfuscator включват функционалностите на Runtime Intelligence, които позволяват на програмиста да събира информация за представянето на техните приложения, след пускането им на пазара.

TEXT GENERATION FRAMEWORK

- Visual Studio включва софтуерна рамка, генерираща текст, наречена T4.
- Тя позволява на Visual Studio да създава текстови файлове от шаблони на средата за разработка или чрез създаване на код.

РАЗШИРЕНИЯ

- Макроси:

Представляват повтарящи се задачи и действия, които разработчиците могат да записват програмно за съхраняване, отговаряне и дистрибуция.

Макросите обаче, не могат да имплементират нови команди или да създават прозорци с инструменти.

РАЗШИРЕНИЯ

Add-ins:

- Дават достъп до обектния модел на Visual Studio и могат да си взаимодействат с инструментите на средата за разработка.
- Те могат да се използват за имплементирането на нови функционалности и добавянето на нови прозорци с инструменти.
- Add-In-ите се свързват със средата за разработка чрез компонентно обектен модел (на английски: Component Object Model, COM) и могат да бъдат създавани чрез всеки COM – съвместим език.

РАЗШИРЕНИЯ

Пакети:

- Създават чрез набора от инструменти за разработка (на английски Software Development Kit, SDK) на Visual Studio.
- Те могат както да създават дизайнери и други инструменти, така и да интегрират нови програмни езици.

DATA DESIGNER

- Data дизайнърът се използва за графични редакции на схеми от бази данни.
- Това включва готови таблици, първични и вторични ключове и ограничители.
- Може да се ползва и за създаването на заявки от графичния изглед.

WEB DESIGNER/ DEVELOPMENT

- Позволява на потребителите да създават уеб страници като добавят различни изпълними модули (от английски: widget).
- Този редактор се използва за създаването на ASP.NET приложения и поддръжа HTML, CSS и JavaScript.
- За всички версии на програмата след Visual Studio 2008, използваният от уеб дизайнера layout engine, е съвместен с Microsoft Expression Web.
- Има също така ASP.NET MVC поддръжка за MVC технологиите и ASP.NET Dynamic Data проектиране, които могат да бъдат допълнително инсталирани.

ПОЛЗИ ЗА ПРОГРАМИСТА?

- Служи като единна среда за повечето, ако не и всички нужди на разработчика, като системи за контрол на версии, инструменти за отстраняване на грешки и Platform-as-a-Service.
- Възможностите за завършване на кода подобряват работния процес на програмиране.
- Автоматично проверява за грешки, за да осигури код с най-високо качество.
- Възможностите за рефакторинг позволяват на разработчиците да правят цялостни и без грешки промени в преименуването.
- Поддържа плавен цикъл на развитие.
- Увеличава ефективността и удовлетвореността на разработчиците.
- Доставяйте най-висококачествен софтуер по график.

ПОЛЗИ ЗА ПРОГРАМИСТА?

- Общата цел и основната полза от интегрирана среда за разработка е **подобрената производителност на разработчиците**. IDE **повишават производителността**, като намаляват времето за настройка, увеличават скоростта на задачите за разработка, поддържат разработчиците актуални и стандартизират процеса на разработка.
- **По-бърза настройка:** Без IDE интерфейс разработчиците ще трябва да отделят време за конфигуриране на множество инструменти за разработка. С интеграцията на приложения на IDE, разработчиците имат един и същ набор от възможности на едно място, без необходимост от постоянно превключване на инструменти.

ПОЛЗИ ЗА ПРОГРАМИСТА?

- **По-бързи задачи за разработка:** По-тясното интегриране на всички задачи за разработка подобрява производителността на разработчиците. Например, кодът може да се анализира и да се провери синтаксисът, докато се редактира, осигурявайки незабавна обратна връзка при въвеждане на синтаксисни грешки. Разработчиците не трябва да превключват между приложения, за да изпълняват задачи.
- В допълнение, инструментите и функциите на IDE помагат на разработчиците да **организират ресурси, да предотвратяват грешки и да вземат преки пътища**.

Un Informático con insomnio





- Студент се явява на изпит видимо притеснен.
Професорът решава да го успокои:
 - Не се вълнувайте колега. На изпит е като в театъра. Вие сте актьорът, а аз - зрителят!
 - Благодаря Ви, г-н професор, но тогава нали мога да си повикам супълъор.