



.NET 平台技术

Dot Net Platform Technology

厦门大学 信息学院（国家示范性软件学院） 赵江声

2023年9月





老师组

1. 赵江声 邮件: zjs@xmu.edu.cn
2. 黄玺 邮件: huangxi_011@qq.com

课程说明



学习.NET平台的
意义

现实意义
未来意义

教学计划与课时
安排

教学计划
课时安排

教学大纲

教学大纲

考核方式与成绩
的构成

考核方式
成就构成

本课程的学习方
法

学习方法

学习.NET平台的意义

现在

- ✓ 修学分，为出国和保研加分
- ✓ 掌握目前两大流行的开发平台之一
- ✓ 软件工程专业学生的三大技术方向之一
- ✓ 多学一门新技术，同时巩固面向对象的编程思想



毕业后不久的将来

- ✓ 扎实的编程基本功、正确的方法和技能
- ✓ 为谋取一份好工作需要
- ✓ 发展势头好，将来开发工作中很可能需要




学习.NET平台的意义

送给同学们的两句话：

1. 科班学生不学点.NET是不行的，它是重要的技术方向；
2. 大家一起努力，让.NET平台技术课程成为精英教育的一部分。





.NET平台的优势

◆一流的工具

- VS Code使用人数位居第一，是 IDE 的绝对统治者；
- 处于第二的则是 Visual Studio。

◆随处运行

- 可以运行在 Windows、macOS、iOS、Android、Linux、大型机甚至微控制器上；
- 也可运行在云端，Azure、AWS 和谷歌云都提供了内置的 .NET 应用程序支持。

◆一系列优雅的编程语言

- C#，F#等；
- .NET Core 也带来了更好的性能，突破 .NET Framework 运行时的限制，性能获得极大提升。

◆强大的社区和开发者

◆受企业信任

- 至少95% 的财富 500 强企业都在使用 Azure，
- 至少80% 的财富 500 强企业在使用 .NET。

教学计划与课时安排



本课程是“软件工程”专业的选修课程，总共54(36+18)个课时，2个学分，是课堂面授、实例演示、案例分析和实验（工具 and 语言的使用）相结合的教学方式。



本课程的安排与说明

一. 基础内容

.NET架构、C#基础语法、C#面向对象的编程、Core基础

二. ASP.NET平台开发

1. ASP.NET概述

2. C#11 高阶语法、数据结构

3. C#11 数据集操作 (LINQ)

4. .NET的前端技术

5. Core基础

6. Core开发

7. JSON

8. 测试驱动开发

本课程的安排与说明

三. .NET构架与案例介绍

1. 有关应用软件系统的架构（MVC等）
2. 某软件系统开发（实际案例，学生管理与图书管理系统等等）

本课程的安排与说明(实验课)



1.使用工具:

Visual studio 2022 简体中文版

2. 实验内容:

3. 大作业:



本课程的学习方法

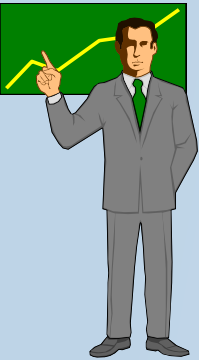


■ 以课件为主进行学习

课件融合多方资料和资源，面向.NET平台的开发，结合基础和实用的案例。

■ 选择合适的参考资料，善用网络资源，提高自学能力

1. 《C#程序设计》 杨律青 编著 上海交通大学出版社
2. 《ASP.NET基础与案例开发详解》 清华大学出版社 李天志、易巍编著
(2014年版本)
3. Core: <https://learn.microsoft.com/zh-cn/aspnet/core/?view=aspnetcore-7.0>
4. C# : <https://learn.microsoft.com/zh-cn/dotnet/csharp/>





21世纪高等学校计算机系列规划教材



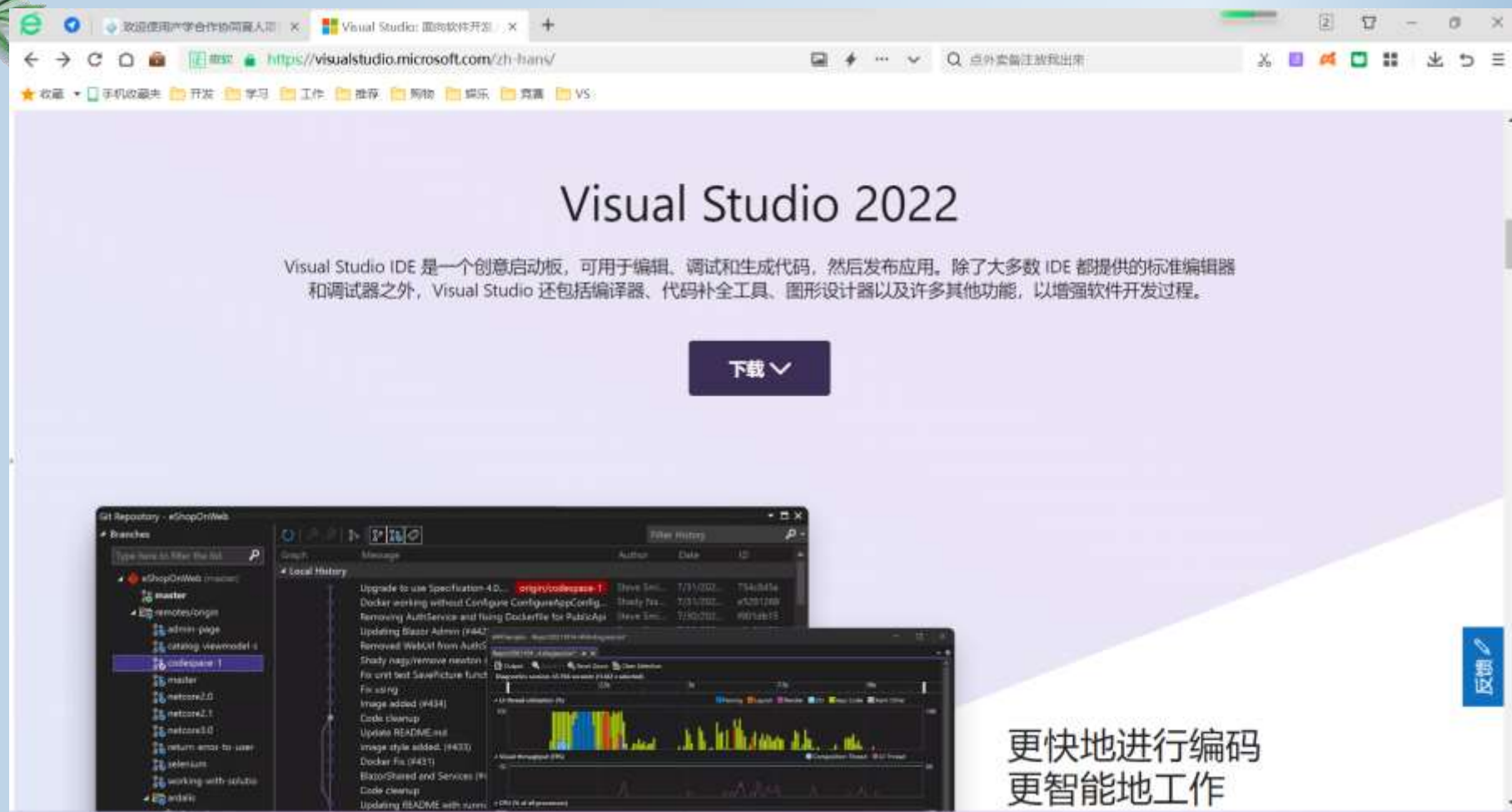
C#程序设计

杨律青 编著



清华大学出版社
Tsinghua University Press

<https://www.visualstudio.com/>



The image shows a screenshot of the Visual Studio 2022 website in a web browser. The browser's address bar displays <https://visualstudio.microsoft.com/zh-hans/>. The website's main heading is "Visual Studio 2022". Below this, a paragraph describes the IDE as a creative launchpad for editing, debugging, and generating code, highlighting features like compilers, code completion, and graphical designers. A prominent "下载" (Download) button with a downward arrow is centered on the page. In the bottom left corner, there is a preview of the Visual Studio IDE interface, showing the "Git Repository" sidebar, the "Local History" panel, and a code editor with a commit message "origin/codebase-1". To the right of the IDE preview, the text "更快地进行编码" and "更智能地工作" is displayed. A small blue button with a white icon is visible on the right edge of the IDE preview.

Visual Studio 2022

Visual Studio IDE 是一个创意启动板，可用于编辑、调试和生成代码，然后发布应用。除了大多数 IDE 都提供的标准编辑器和调试器之外，Visual Studio 还包括编译器、代码补全工具、图形设计器以及许多其他功能，以增强软件开发过程。

下载

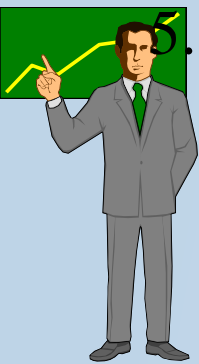
更快地进行编码
更智能地工作

本课程的学习方法



■ 理论与实验相结合

1. 认真听课，积极参加课堂讨论和发言，首先理解和掌握基本概念和理论，将所学的内容进行条理化、层次化。
2. 自己动手，在老师指导下做好每一次实验，通过实验来理解编程的流程、程序的内部结构和体会控件、语言语法的使用。
3. 课上存在的问题，做上标识，可查阅资料或问老师。
4. 学习的内容不要停留在课堂上，通过参与将来的企业实际项目，你的能力将得到极大的提升。
5. 有重点地学习，掌握最基础、最重要和最常用的知识点和技能，学会把有限的精力投入无限的编程中去。

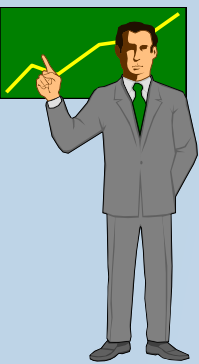


本课程的学习方法

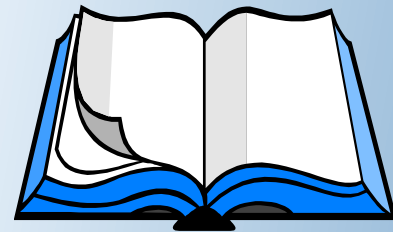


选择了软件工程师（程序员）职业，就选择了充满挑战 and 机会、辛苦又充满乐趣、单调和富有成就感的职业。

首先要学好基础的编程语言。



考核方式与成绩的构成




1. 考勤(含实验出勤)与课堂发言 10%
2. 平时实验作业 15%
3. 小组案例作业 35%
4. 期末考试成绩 (100分试卷) 50%

课件及资料下载:

<ftp://121.192.180.66> /教学课件/赵江声/2023-2024
第1学期DotNet20软工/





第一章 .NET框架与C#基础概述

本章结构

1.1 .NET平台介绍

1.2 C#语言基础

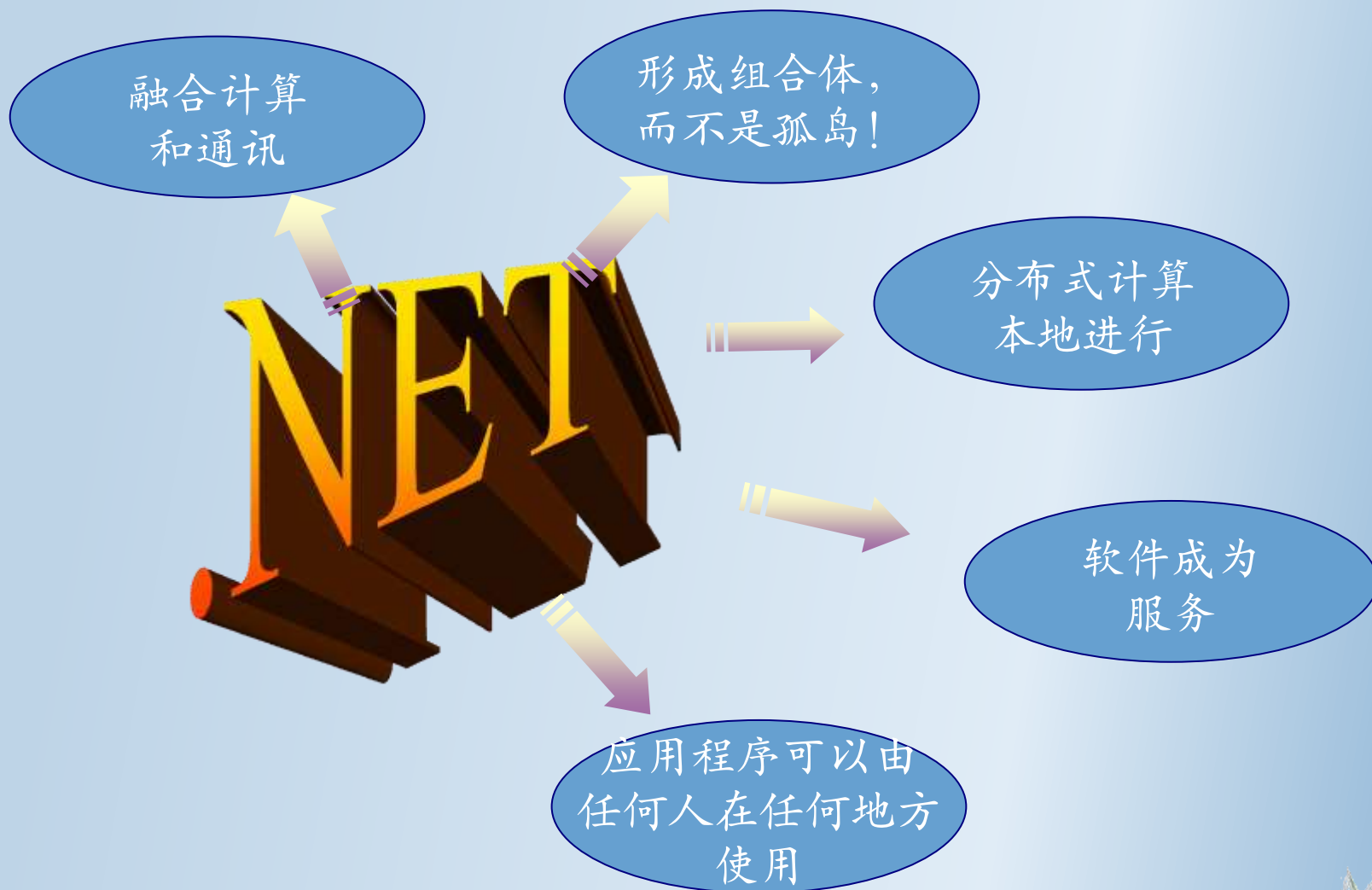
1.3 VS 2022的使用




.NET平台介绍

1. Microsoft .NET 全新框架平台
2. 公共语言运行时 (CLR)
3. 面向 .NET 的全新开发语言—C#

.NET --演变的结果





.NET平台的优势

◆一流的工具

- VS Code使用人数位居第一，是 IDE 的绝对统治者；
- 处于第二的则是 Visual Studio。

◆随处运行

- 可以运行在 Windows、macOS、iOS、Android、Linux、大型机甚至微控制器上；
- 也可运行在云端，Azure、AWS 和谷歌云都提供了内置的 .NET 应用程序支持。

◆一系列优雅的编程语言

- C#，F#等；
- .NET Core 也带来了更好的性能，突破 .NET Framework 运行时的限制，性能获得极大提升。

◆强大的社区和开发者

◆受企业信任

- 至少95% 的财富 500 强企业都在使用 Azure，
- 至少80% 的财富 500 强企业在使用 .NET。

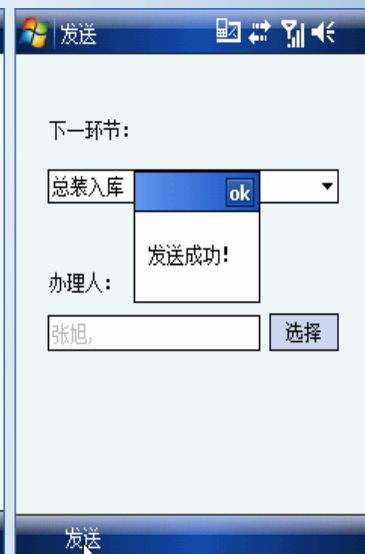
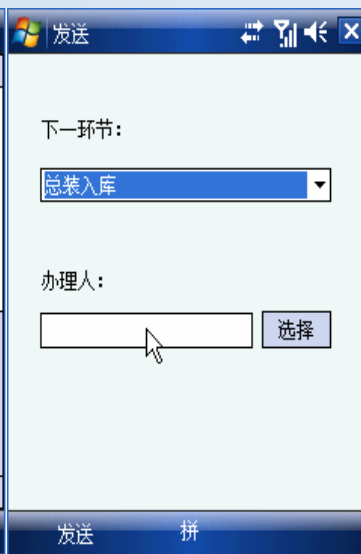
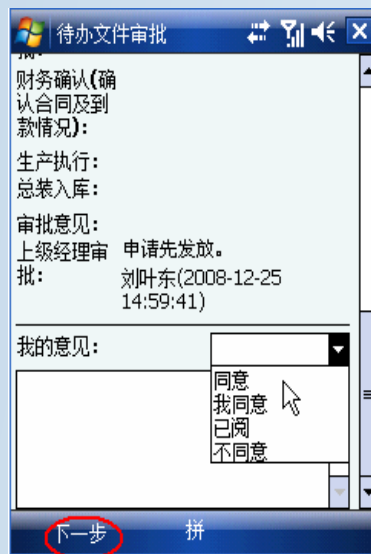


.NET -- 以互联网为核心

- ◆ 用户数据存放在网络上 - 可以随时随地进行访问
- ◆ .NET - 以 Internet 为中心的一种全新的平台
- ◆ 创建可以通过多种浏览器、多种设备访问的应用程序，可以从任何 .NET 兼容设备（PDA、手机、智能终端、嵌入式设备等）访问数据。
- ◆ 可用于创建出色的 Windows、Android 和 iOS 应用程序以及 Web 应用程序和云服务。
- ◆ 支持团队作业（团队资源管理器，多文件的合并）



整合性：移动手机办公



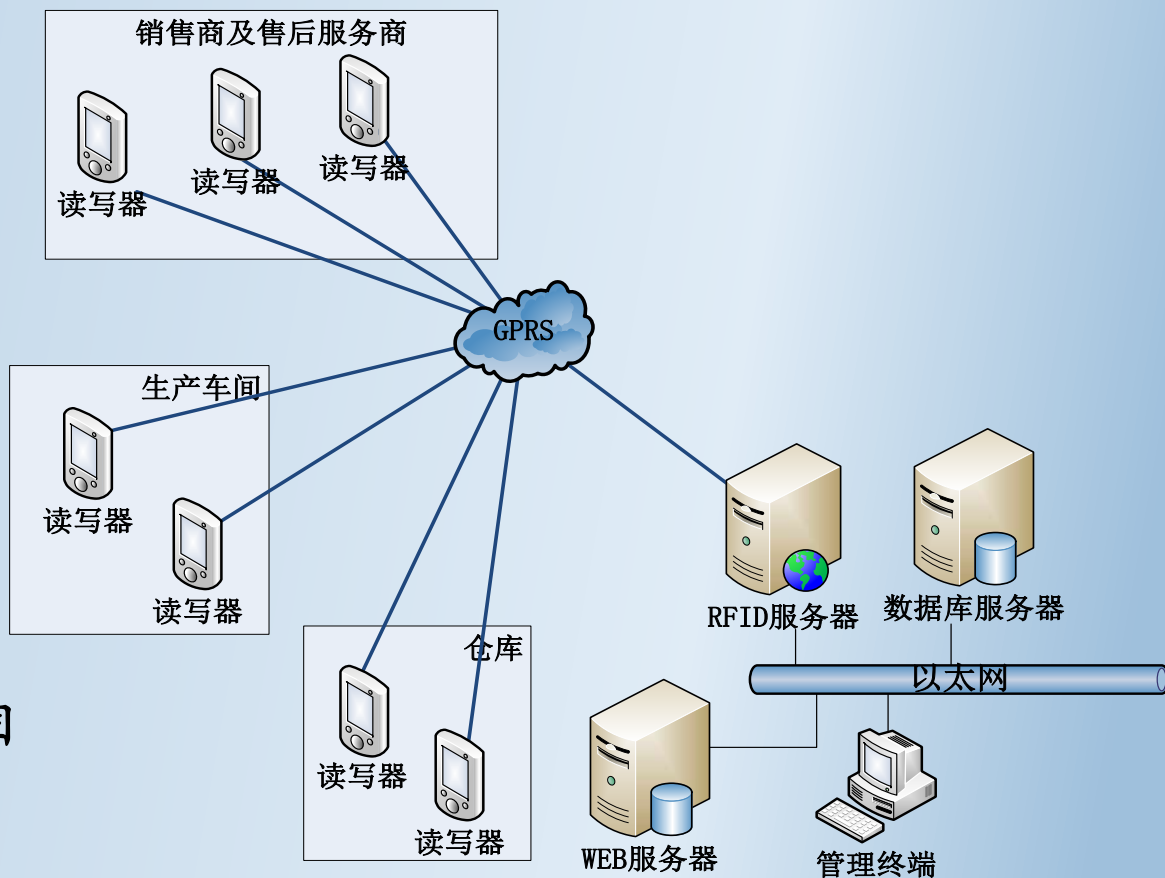
生产过程的阅读器



手持式电子
标签阅读器



应用系统的物理结构图



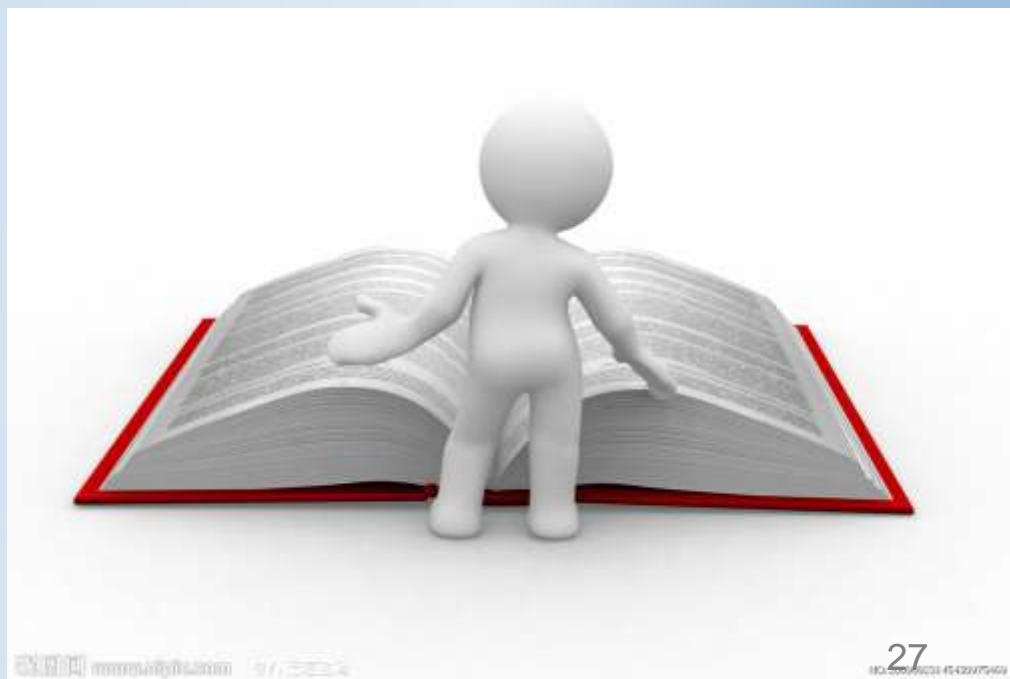
备注：读写器须手持式与固定式的相结合

用.NET平台开发读写器端和WEB服务器应用



.NET上的软件体系结构或框架

- C/S
 - WinForm
 - WPF
- B/S
 - WebForm
 - MVC
- RIA
 - Flex
 - Silverlight
 - Html5





.NET 支持的语言

C#

VB.NET

VC++

JScript.NET

ADA

Python

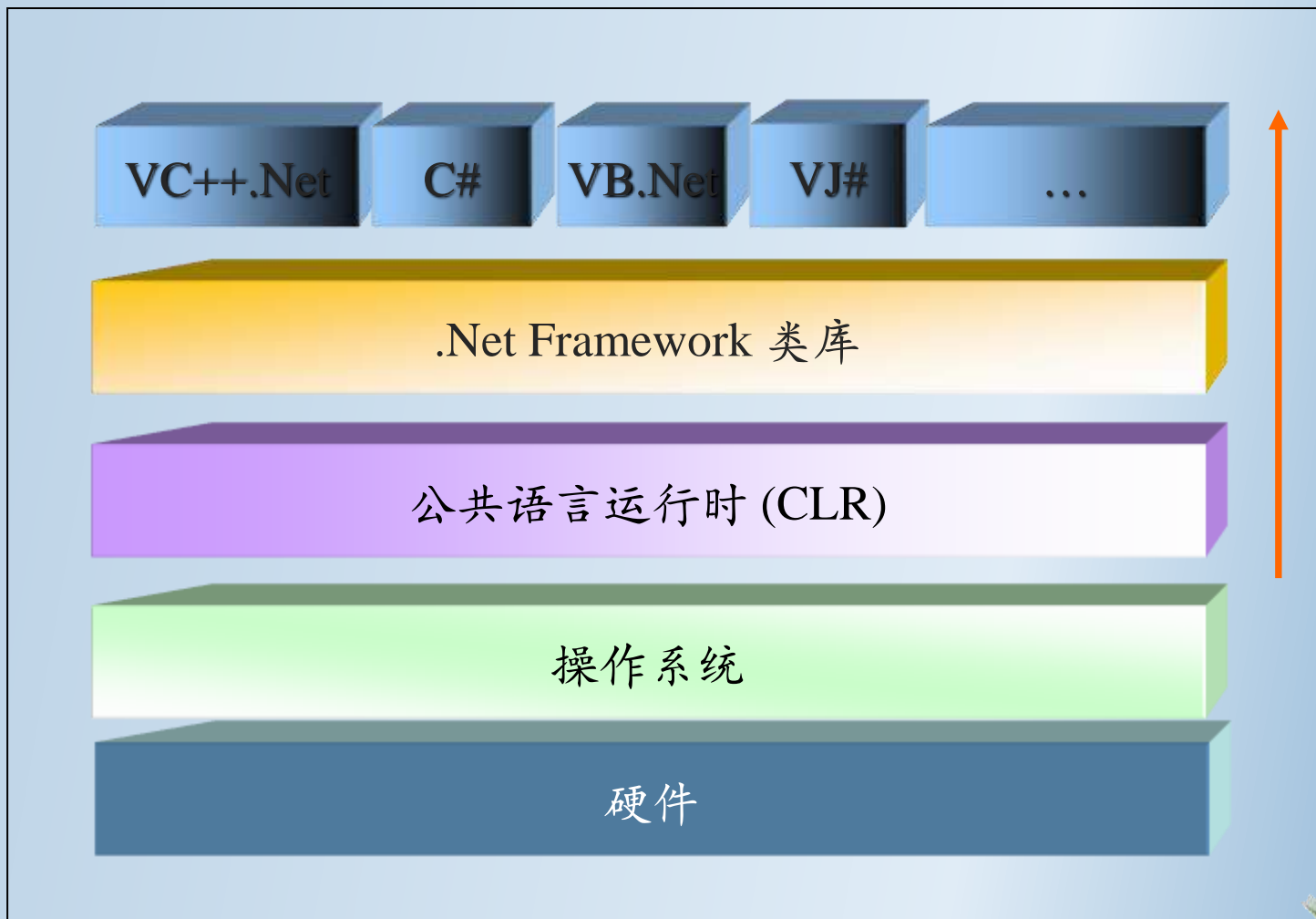
Eiffel

COBOL

SmallTalk

支持当前二十多种主流编程语言

.NET Framework 的体系结构





.NET Framework 体系结构

- ❑ .NET Framework 包含两个组件，公共语言运行时 (CLR) 的核心执行环境和一组类库。
- ❑ 用 C# 编写的源代码被编译成符合 CLI 规范的中间语言 (IL)。
- ❑ 执行 C# 程序时，程序集将加载到 CLR。CLR 会直接执行实时 (JIT) 编译，将 IL 代码转换成本机指令。



.NET Framework 简介

- ❑ .NET Framework 类似于 JVM
- ❑ .NET Framework 两个主要组件：
 - ❑ 公共语言运行时 (CLR, 运行库)
 - ❑ 统一的类库集 (FCL, 即基本框架类库)
- ❑ .NET 的类库的功能：
 - ❑ 线程
 - ❑ 文件输入/输出 (I/O)
 - ❑ 数据库支持
 - ❑ XML 解析
 - ❑ 提供数据结构
 - ❑

.NET Framework 的结构图

.Net Framework

基本框架类库 FCL



CLR

CLS

CTS



基本框架类

基本框架类库（Framework Class Library，FCL）。包括通用基础类、集合类、线程和同步类、XML类。

- 1) 基础类型。【整数、实数、字符串等类型定义】
- 2) 数据结构封装。【集合、链表、队列、堆栈等数据类型】
- 3) Windows和Web等界面要素。【主要是按钮、标签、文本框、菜单等可视化控件】
- 4) Web Service要素。【用于Web服务的定义、描述、配置、解析等】
- 5) XML文档处理。【如XML文件、属性、元素、节点、读写器、解析器等类型】
- 6) 文件的输入输出。【如目录、文件、流、读写器等类型】

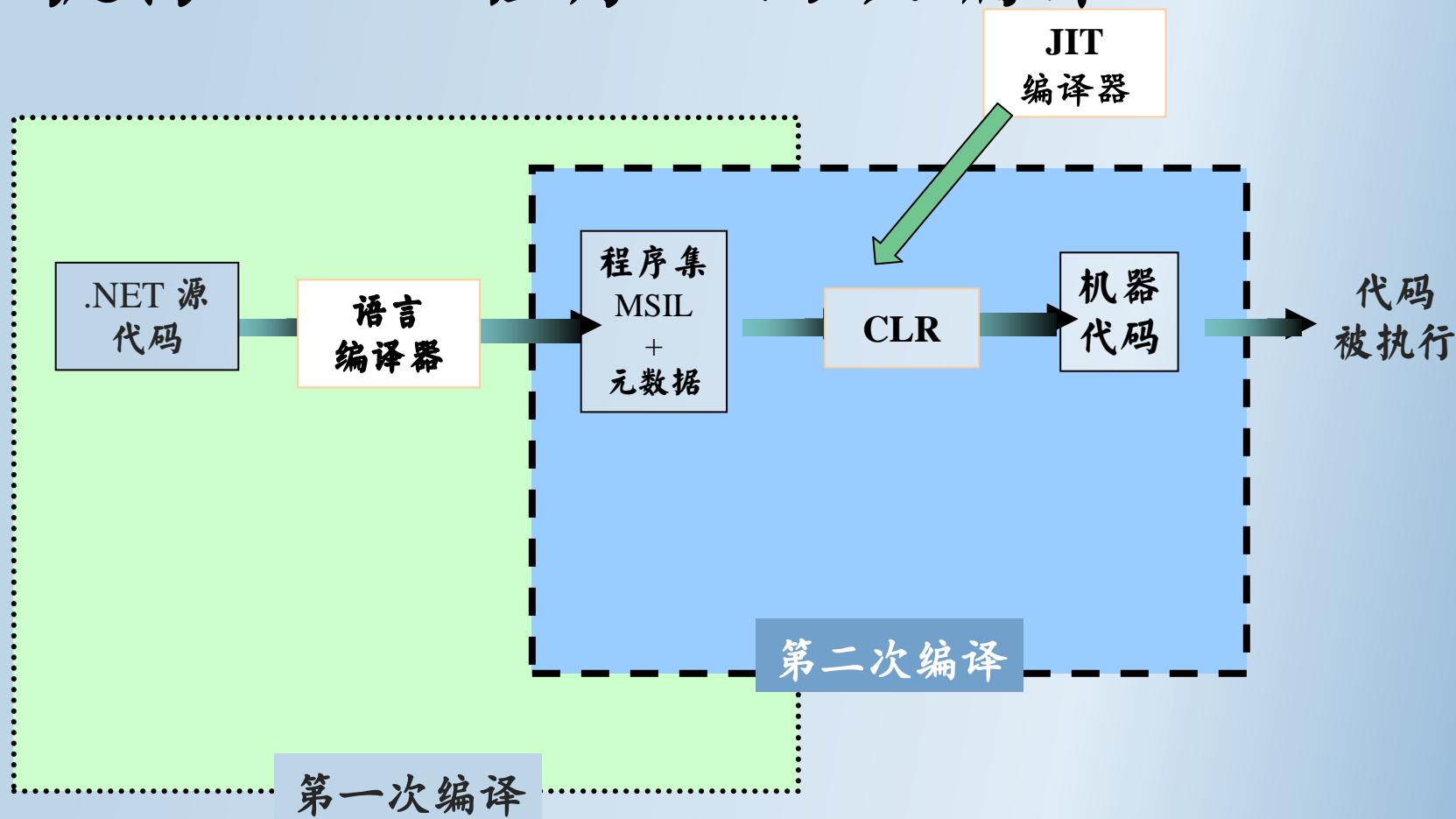


基本框架类

基本框架类库（Framework Class Library，FCL）。包括通用基础类、集合类、线程和同步类、XML类。

- 7) 数据访问。【如数据连接、数据命令、数据集、数据表、记录等类型】
- 8) 网络通信。【如主机、管道（pipeline）、套接字、消息等类型】
- 9) 异常处理。【用于处理系统和应用程序所引发的各种异常】
- 9) 类型反射。【用于获取程序集、对象的方法、属性、字段等目标的元数据信息类型】
- 10) 用于应用程序管理、操作系统功能封装、安全性控制等其他方面的类型。

执行 .NET 程序 - 两次编译



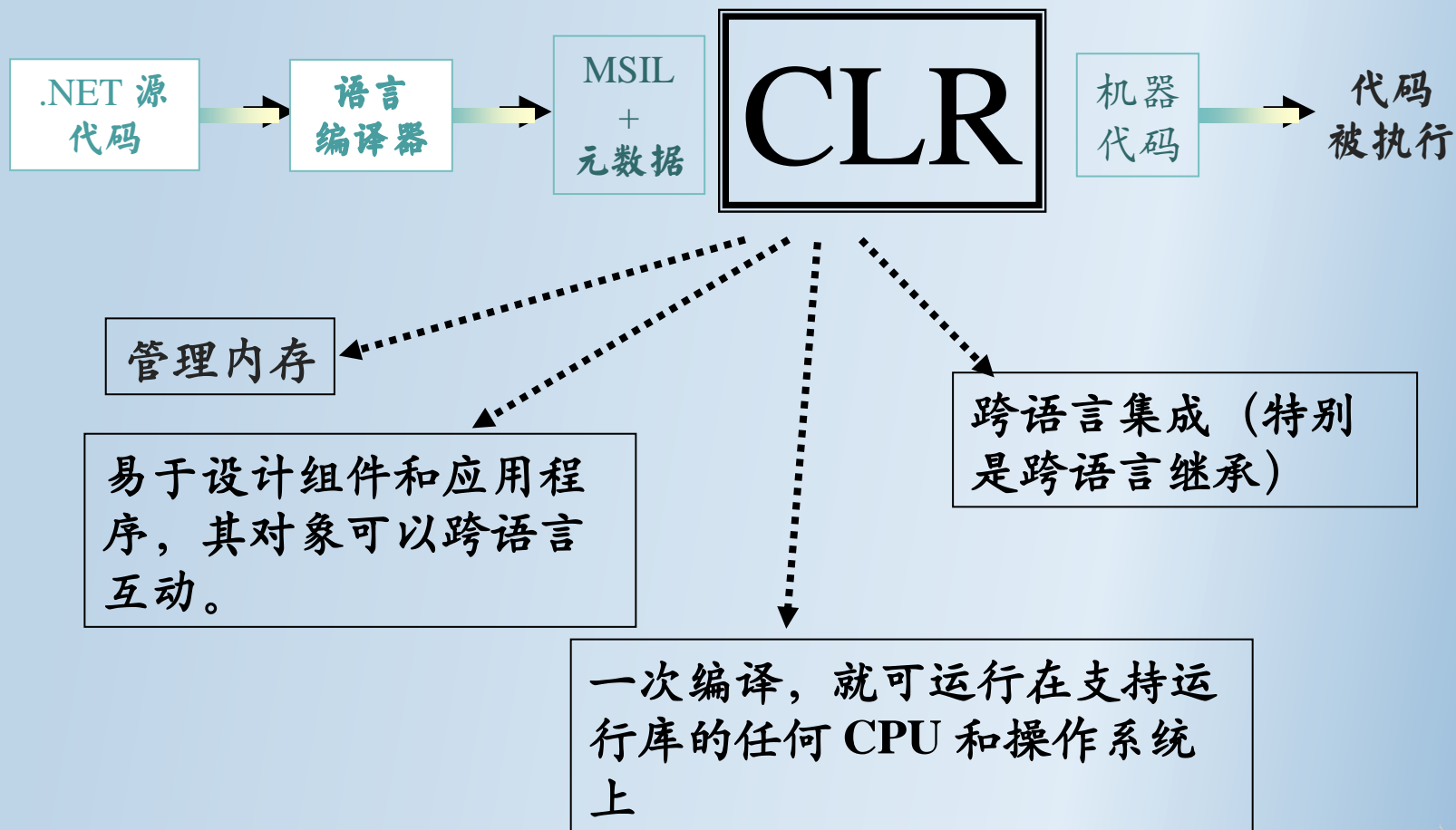
.NET 程序被编译两次，第一次编译很慢，而第二次编译较快！



CLR和MSIL

- ❖ CLR是公共语言运行时(Common Language Runtime)，它负责资源管理(内存分配和垃圾收集)。
- ❖ Microsoft 中间语言 (MSIL)或称CIL(公共中间语言)由一组特定的指令组成，这些指令指明如何执行代码。

公共语言运行时



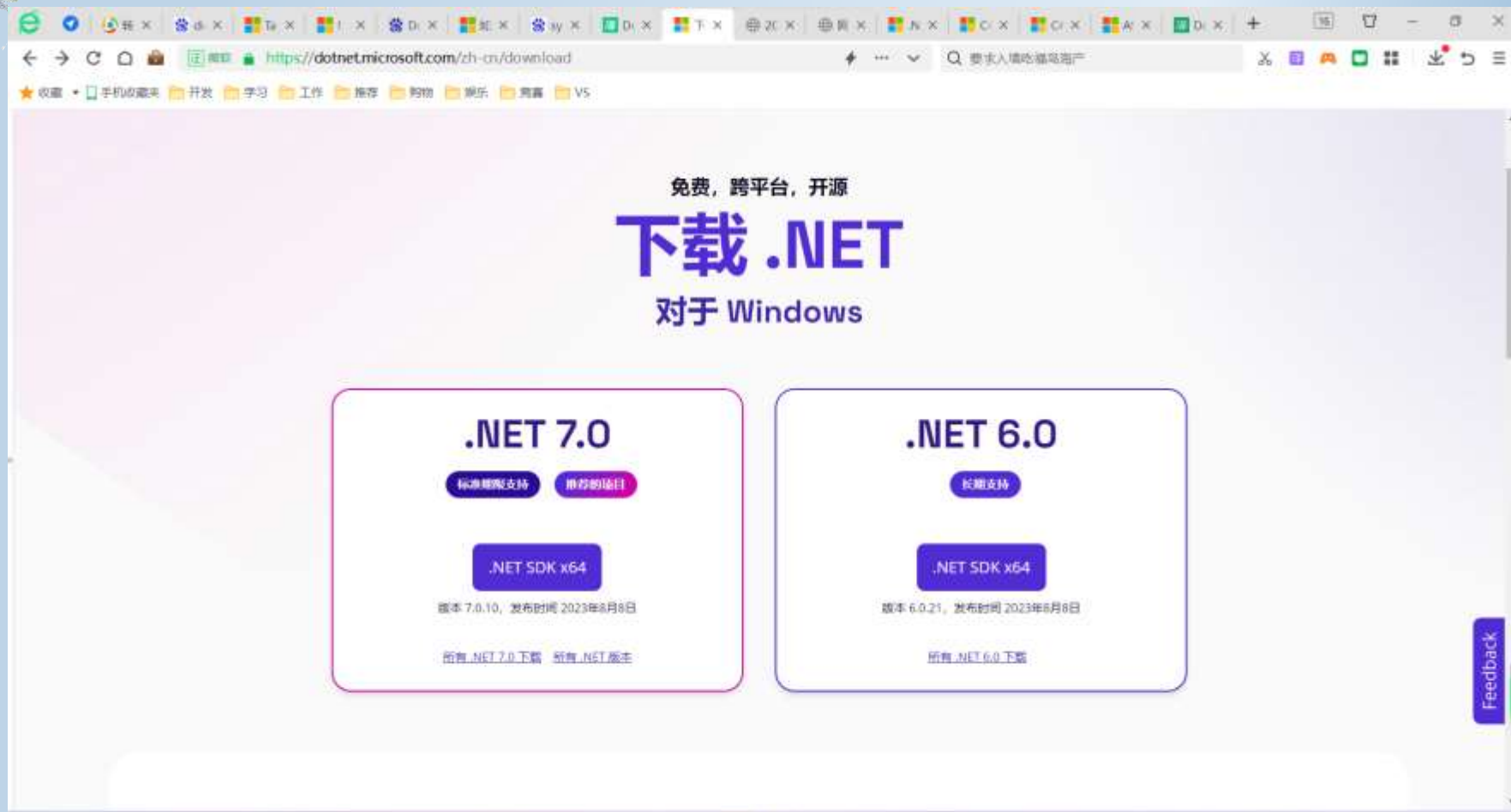


.NET 与开发环境历史发展

- Visual Studio.NET集成开发环境.NET
- Framework 3.5对应VS 2008, C#3.0
- .NET Framework 4.0对应VS 2010, C#4.0
- .NET Framework 4.5对应VS 2012, C#5.0
- .NET Framework 4.6.2 对应VS 2015, C#6.0
- .NET Framework 4.7 对应VS 2017 C#7.0
- C# 8.0 版是专门面向 .NET C# Core 的第一个主要 C# 版本, 2019.9
- C#9 .NET 5 2020.11
- C#10 2021.11
- C#11 2022.11 VS2022

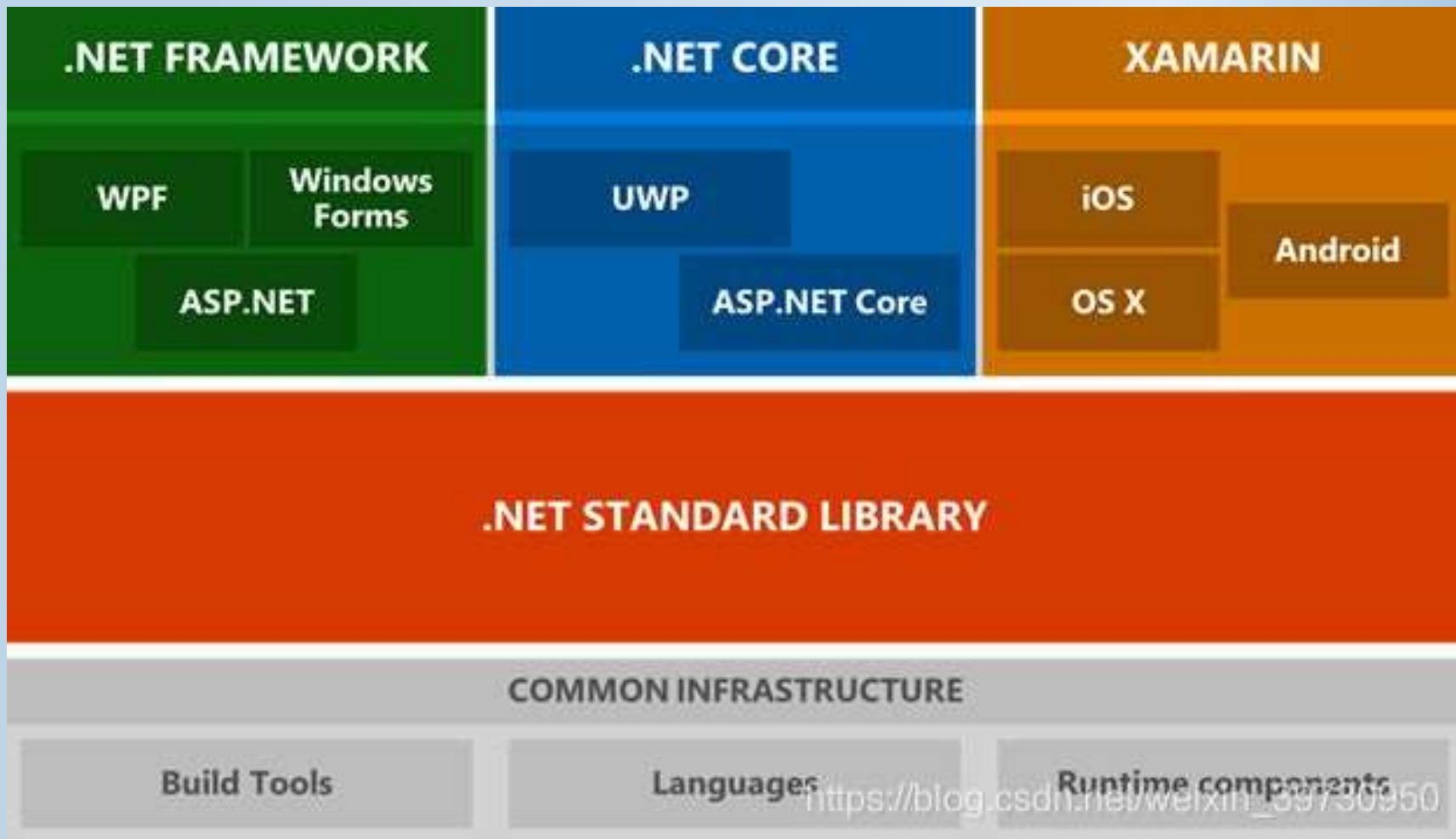


.NET Core 的下载界面



.NET Core 是对 .NET Framework 的重新设计, 是具有跨平台 (Windows、Mac OSX、Linux) 能力的应用程序开发框架。用于生成启用云且连接 Internet 的新式应用。⁴⁰

如今的.NET核心系统架构



.NET CORE VS .NET FRAMEWORK

.NET CORE

.NET FRAMEWORK

https://blog.csdn.net/weixin_39730950



Xamarin 毫无争议，当你想用C#构建一个**移动应用**时，Xamarin 是你唯一的选择。

.NET Framework 支持窗体和网页应用。现在，你可以使用Windows Forms, WPF, and UWP去创建一个Windows应用在.NET Framework平台。**ASP.NET MVC**往往被用来构建网页应用在.NET Framework平台。

.NET Core 是**新的开源和跨平台**框架，用于为所有操作系统（包括Windows、Mac和Linux）构建应用程序。.NET Core仅支持UWP和ASP.NET Core。UWP用于构建Windows10目标窗口和移动应用程序。ASP.NET Core用于构建基于浏览器的Web应用程序。






.NET 的未来

- 微软刚刚发布了 .NET 7，2023 年已经可用。





第一章 .NET框架与C#基础概述

本章结构

1.1 .NET平台介绍

1.2 C#语言基础

1.3 VS 2022的使用



面向.NET的全新开发语言 — C#

- ❖ C和C++一直是最有生命力的编程语言，这两种语言提供了强大的功能、高度的灵活性以及完整的底层控制能力；缺点是开发周期较长。开发效率更高的语言，如Visual Basic，在功能方面又具有局限性(如关机的功能)。于是，在选择开发语言时，许多程序设计人员面临着两难的抉择。
- ❖ 针对这个问题，微软公司发布了称之为C# (C Sharp)的编程语言。C#是为.NET平台量身定做的开发语言，采用面向对象的思想，支持.NET最丰富的基本类库资源。
- ❖ C#提供快捷的开发方式，又没有丢掉C和C++强大的控制能力。C#与C和C++非常相似，熟悉C和C++的程序设计人员能够很快掌握C#。




开发友的体验

如果抛开一切非技术方面的因素，C#无疑是这个星球上有史以来最好的编程语言，它几乎集中了所有关于软件开发和软件 engineering 研究的最新成果：

- 面向对象
- 类型安全
- 组件技术
- 自动内存管理
- 自动垃圾处理
- 跨平台处理

.....





C#的特点

C#是专门为.NET应用而开发的语言。在.NET类库的支持下，C#能够全面地体现.NET Framework和CORE的各种优点。总地说，C#具有以下突出的优点。

1. 语法简洁
2. 彻底的面向对象设计,将面向对象的三个特性发挥得淋漓尽致
3. 与Web应用紧密结合
4. 强大的安全机制
5. 完善的错误、异常处理机制
6. 灵活版本处理技术（如自动升级）
7. 语言兼容性强 #BK



1.2 C#语言基础

- C#的语法设计有很多地方与C/C++相似。本节帮助同学们回顾和介绍C#程序设计基础知识，包括数据类型、常量和变量、运算符和语句结构。



1.2.1 数据类型

- C#数据类型包括基本类型（类型中最基础的部分），如int、char、float等，也包括比较复杂的类型，如string、array等。

作为完全面向对象的语言，C#中的所有数据类型是一个真正的类（从什么类派生？）。根据在内存中存储位置的不同，C#中的数据类型可分为以下两类：

1. 值类型：该类型的数据长度固定，存放于栈内。
2. 引用类型：该类型的数据长度可变，存放于堆内。

指针类型(也可归引用类型)

C# 中的数据类型

值类型

简单类型
(simple)

整数类型 integer

实数类型 real

字符类型 char

布尔类型 bool

枚举类型
(enumeration)

结构类型
(struct)

引用类型

类类型 (class)

代理类型 (delegate)

接口类型 (interface)

数组类型 (array)



1.2.1.1 值类型

C#内置的值类型是最基本的数据类型，例如整数、浮点数、字符、布尔类型等。

1. 整数类型 如Int (32位,4个字节),Long (64位,8个字节),Short(16位,2个字节),Byte(8位,1个字节)等等

2. 浮点数类型

float(占用字节：4) —— (精度：7位)

double(占用字节：8) —— (精度：15位)

3. Decimal (十进制,金融类型)

(占用字节：16) —— (精度：28位)





1.2.1.1 值类型

4. 布尔型

C#的布尔型是bool，其取值包括True和False

5. 字符型

Char (2个字节, C#中统一采用Unicode字符集)

1.2.1.1 值类型

除上面介绍的简单值类型之外，用户还可以定义复合值类型。常用的复合值类型包括结构和枚举。

6. 结构

- 结构属于值类型
- 可以有方法
- 可以拥有构造函数
- 不能实现继承

结构和类的比较

结构存放在栈中并以值传递
类存放在堆中并以引用传递

```
...
struct structEx
{
    public int structDataMember;
    public void structEx()
    {
        //实现构造函数
    }

    public void structMethod1()
    {
        //structMethod1 实现方法
    }
}
...
```



举例

```
struct PhoneBook  
{  
    public string name;  
    public string phone;  
    public string address;  
}  
  
PhoneBook p1;
```

对结构成员的访问通过结构变量名加上访问符
“.”号，再跟成员的名称：

p1.name=“Mike”;



1.2.1.1 值类型

7. 枚举

枚举 (enumeration) 其实是一个整数类型，用于定义一组基本整数数据，并可以给每个整数指定一个便于记忆的名字。

枚举例子

```
public class Holiday
{
    public enum WeekDays
    {
        Monday,
        Tuesday,
        Wednesday,
        Thursday,
        Friday
    }
    public void GetWeekDays(String EmpName, WeekDays DayOff)
    {
        //处理 WeekDays
    }
    ...
}
static void Main()
{
    Holiday myHoliday = new Holiday();
    myHoliday.GetWeekDays("Richie", Holiday.WeekDays.Wednesday);
}
}
```

程序中声明了仅接受5个值的枚举类型 WeekDays



枚举

- C# 中的枚举元素都有与值相关联的数字
- 默认情况下，枚举数的第一个元素被指定为 0 值，后续的各个枚举数元素的值依次递增
- 没有显式声明类型，则使用32位 Int类型
- 默认值可以在初始化阶段中重写

```
public enum WeekDays
{
    Monday=1,
    Tuesday=2,
    Wednesday=3,
    Thursday=4,
    Friday=5
}
```



枚举的好处

- 枚举可以使代码更易于维护，有助于确保给变量指定合法的、期望的值。
- 枚举使代码更清晰，允许用描述性的名称表示整数值，而不是用含义模糊的数来表示。
- Program: 枚举举例



1.2.1.2 引用类型

C#中要处理堆中的数据就需要使用引用数据类型，使用new关键字实例化引用类型的对象，并指向堆中的对象。例如：

```
Obj1 = new Obj();
```

Obj1即指向堆中的Obj对象。



1.2.1.2 引用类型

C#中常用的内置引用数据类型有：

1. 指针（仅用于 unsafe 模式）
2. 数组
3. 类、接口
4. 委托（代理，delegate）



数组

数组是一组相同的有序数据（C#中System.Array是所有数组的基类）

- 声明： `int[] arr`

计算数组的元素的个数： `arr.Length`

下标从0开始；

`DataType[number of elements] ArrayName;`





类

Object是所有类的基类，可以对**object**类的变量赋任何类型的值。

例如，**string**类是专门对字符串操作的类，在**System**命名空间中定义(**System.string**)，可以进行连接和获取字符操作。

```
例： string S1="Welcome!";  
  
      string S2="NET AND C#";  
  
      bool=(S1==S2);
```



1.2.1.3 装箱和拆箱

1. `int i=10;`
2. `string s=i.ToString();`

说明：

`i`是一个值类型数据，存放在栈(Stack)内存中；

`s`是一个引用类型的String对象，存放在堆(Heap)中。

1.2.1.3 装箱和拆箱

在C#类型系统中，任何值类型、引用类型和Object类型之间进行转换：

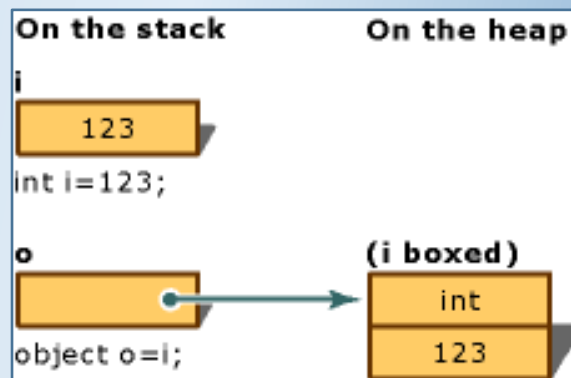
- 装箱：将一个值类型隐含转换成Object类型(根类型)；

例：`int i=123;`

`object o=i;`

注意：装箱的时候，被装箱的值(i)

拷贝一个副本赋给对象。



- 拆箱：将对象类型显式地转换为一个值类型；

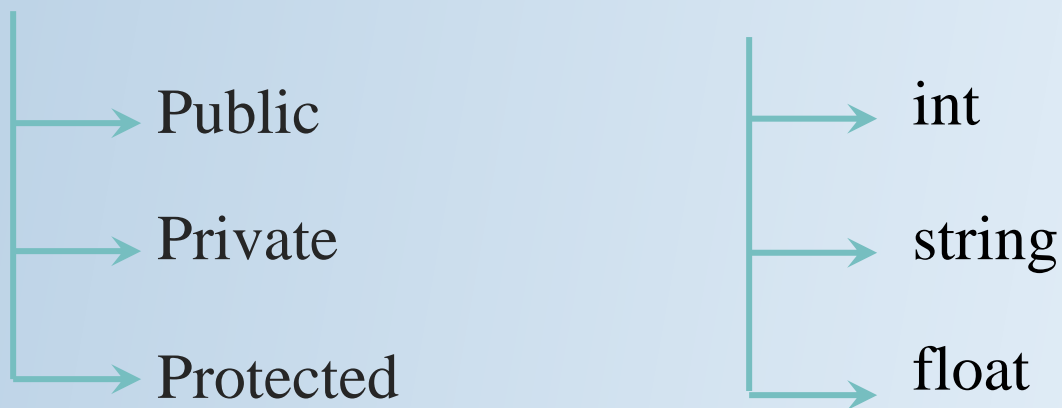
例：`i=(int)o;`



1.2.2 变量和常量

C# 中的变量以下列方式进行声明:

AccessModifier Data Type Variable





1.2.2.1 变量

使用C#变量的基本原则是：先定义，后使用。C#中的变量命名规范如下：

- (1) 必须以字母或下划线开头；
- (2) 只能由字母、数字、下划线组成，不能包含空格、标点符号、运算符，以及其他符号；
- (3) 不能与C#关键字（如class、new等）同名。

注意：C#对大小写是敏感的。



1.2.2.2 常量

- 常量一旦初始化就不再发生变化，可以理解为符号化的常数。使用常量可以使程序变得更加灵活易读。

例如，可以用常量PI来代替3.1415926，一方面程序变得易读，另一方面，需要修改PI精度的时候无需在每一处都修改，只需在代码中改变PI的初始值即可。

- 常量的声明和变量类似，需要指定其数据类型、常量名以及初始值，并需要使用const关键字，例如：`[public] const double PI=3.1415;`



1.2.3 运算符

- C#中的运算符是用来对变量、常量或数据进行计算的符号，指挥计算机进行某种操作。
- 可以将运算符理解为交通警察的命令，用来指挥行人或车辆等不同的运动实体（运算数），最后达到一定的目的。

例如，“+”是运算符，而“2+3”完成两数求和的功能。

1.2.3.1 算数运算符

- 算术运算符 (arithmetic operators) 用来处理四则运算的符号，是最简单、最常用的符号，尤其数字的处理几乎都会使用到算术运算符。

符号	示例	意义
+	$a+b$	加法运算
-	$a-b$	减法/取负运算
*	$a*b$	乘法运算
/	a/b	除法运算
%	$a\%b$	取余数
++	$a++$	累加
--	$a--$	递减



1.2.3.2 字符串运算符

- 字符串运算符 (string operator) 只有一个，就是加号 “+”。它除了作为算术运算符之外，还可以将字符串连接起来，变成合并的新字符串。示例代码如下：

1. `string s="Hello";`
2. `s=s+", World.";`
3. `Console.WriteLine(s);` //输出：Hello, World.



1.2.3.3 赋值运算符

- 赋值运算符 (assignment operator) 把其右边表达式的值赋给左边变量或常量。

符 号	示 例	意 义
=	a=b	将右边的值赋值到左边
+=	a+=b	将右边的值加到左边 (数字或字符串都可)
-=	a-=b	将右边的值减到左边
=	a=b	将左边的值乘以右边
/=	a/=b	将左边的值除以右边
%=	a%=b	将左边的值对右边取余数

1.2.3.4 逻辑运算符

- 逻辑运算符 (logical operators) 通常用来测试真假值。

符 号	示 例	为 真 条 件
<	$a < b$	当 a 的值小于 b 值时
>	$a > b$	当 a 的值大于 b 值时
<=	$a \leq b$	当 a 的值小于或等于 b 值时
>=	$a \geq b$	当 a 的值大于或等于 b 值时
==	$a == b$	当 a 的值等于 b 值时
!=	$a != b$	当 a 的值不等于 b 值时
&&	$a \&\& b$	当 a 为真并且 b 也为真时
	$a b$	当 a 为真或者 b 也为真时
!	!a	当 a 为假时



1.2.3.5 其他运算符

除上面运算符之外，C#还包括一些特殊的运算符。

符 号	示 例	意 义
<code>new</code>	<code>new Class1();</code>	创建一个类的实例
<code>typeof</code>	<code>typeof(int);</code>	获取数据类型说明
<code>.</code>	<code>Obj.method();</code>	获取对象的方法或属性
<code>?:</code>	<code>(expr1)?(expr2):(expr3);</code>	若 <code>expr1</code> 则 <code>expr2</code> ; 否则 <code>expr3</code>

Program: 获取数据类型

Program: 三目运算符





1.2.4 语句结构

同其他高级语言类似，C#的程序结构主要有顺序结构、分支结构和循环结构，另外，C#也支持无条件跳转。



1.2.4.1 条件语句

条件语句主要有两个：if语句和switch语句。

1. if语句

if语句是最常用的条件语句，通过判断布尔表达式的值，选择执行后面的内嵌语句。

2. switch语句

当程序面临多叉路口时，就可以使用switch语句进行分支。



1.2.4.2 循环语句

循环语句主要有两个：while语句和for语句

1. while循环语句

当程序需要重复执行某种功能，直到达到某种条件才停止时，需要采用循环程序结构。





1.2.4.2 循环语句

2. for循环语句

for语句同样用来实现循环结构，与while功能类似，语法如下所示：

1. **for (expression1; expression2; expression3)**
2. **{**
3. **statement**
4. **}**

```
for (int i = 0; i < array.Length; i++)
```





控制循环

在循环中如果想结束循环时，需要使用跳转语句 `break` 或 `continue` 语句。

`break` 语句不仅可以用在 `switch` 中，也可以用在循环语句中，用于中断循环，从循环中跳出。而 `continue` 语句的作用在于，可以提前结束一次循环过程中执行的循环体，直接进入下一次循环。



1.2.4.3 异常处理语句

- 再熟练的程序设计人员也不能保证自己写的代码没有任何问题。可以说，代码中异常陷阱无处不在，如数据库连接失败、I/O错误、数据溢出、数组下标越界等。
- 鉴于此，C#提供了异常处理机制，允许程序设计人员捕捉程序运行时可能的异常。



异常处理语句

- Try..finally 演示

- 无论是否发生异常，都会执行finally块中的代码

```
try
{
    //try 的代码
}

catch
{
    //异常捕获代码
}


finally
{
    //finally 的代码
}
```



高质量编码标准

- 1.好的编码结构
- 2.好的注释风格
- 3.好的命名规范
- 4.避免文件过大(方法<25行)
- 5.使用异常处理





第一章 .NET框架与C#基础概述

本章结构

1.1 .NET平台介绍

1.2 C#语言基础

1.3 VS2022的使用



C#编程神器visual studio (VS)

❖ 代码重构

程序代码写的不严谨,写的乱,代码重构是获得结构良好的方法,通过重构,我们在保持功能不变的情况下,改善代码的质量,提高代码的复用程度。

❖ 断点调试

❖ 智能感知

❖ 单元测试

❖ 扩展管理(如.NET中访问C++组件)

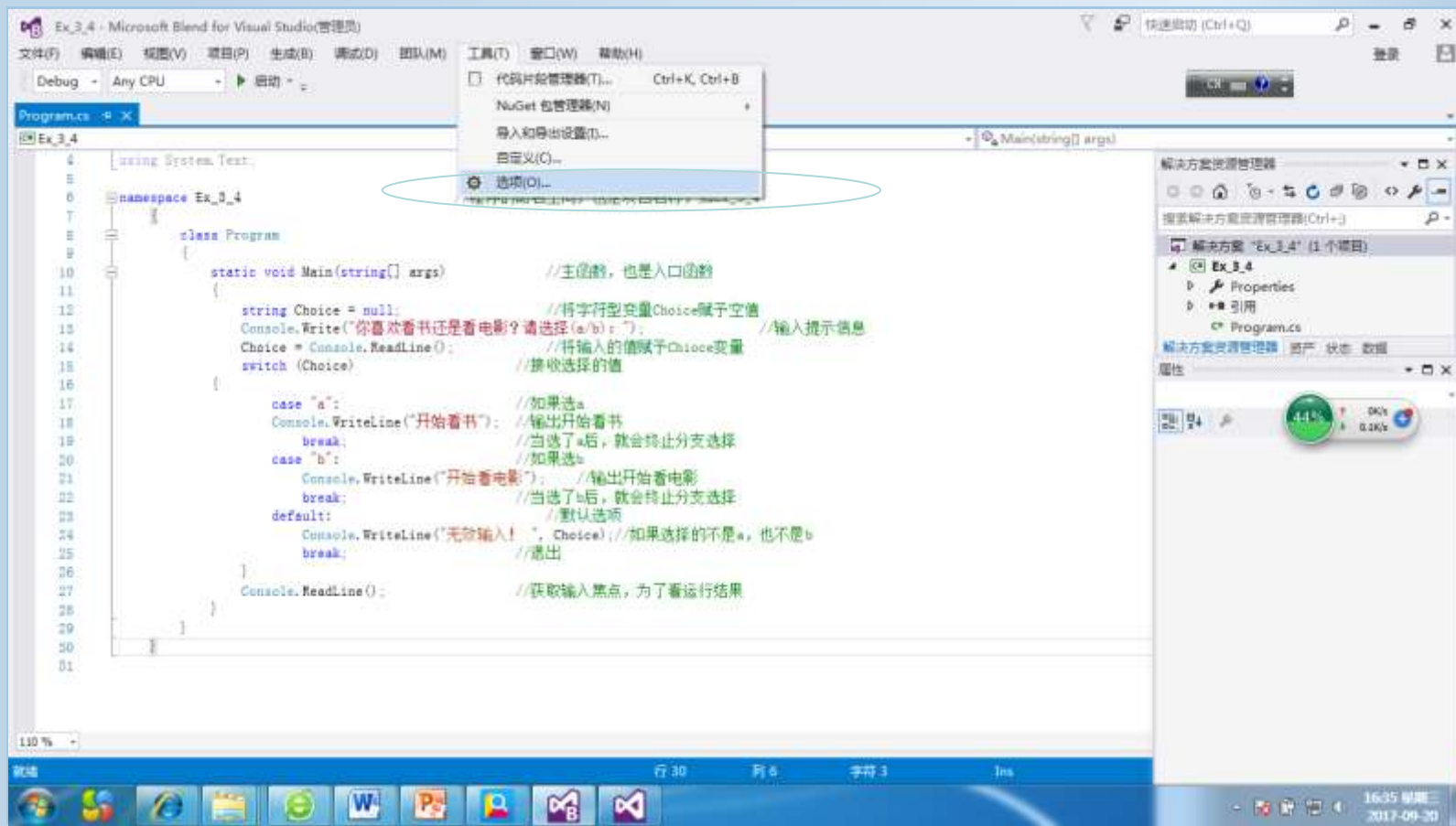
❖



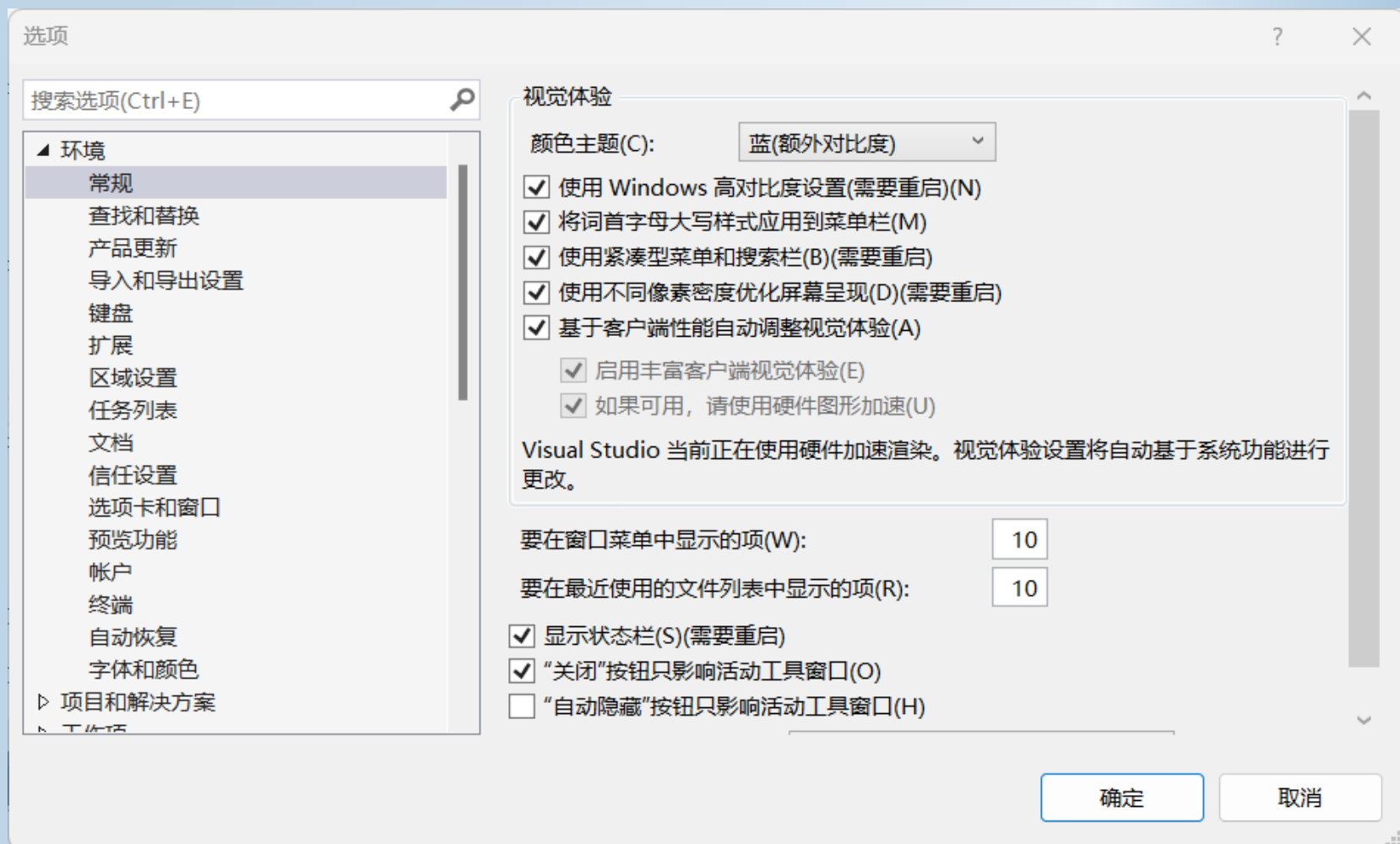
起始页



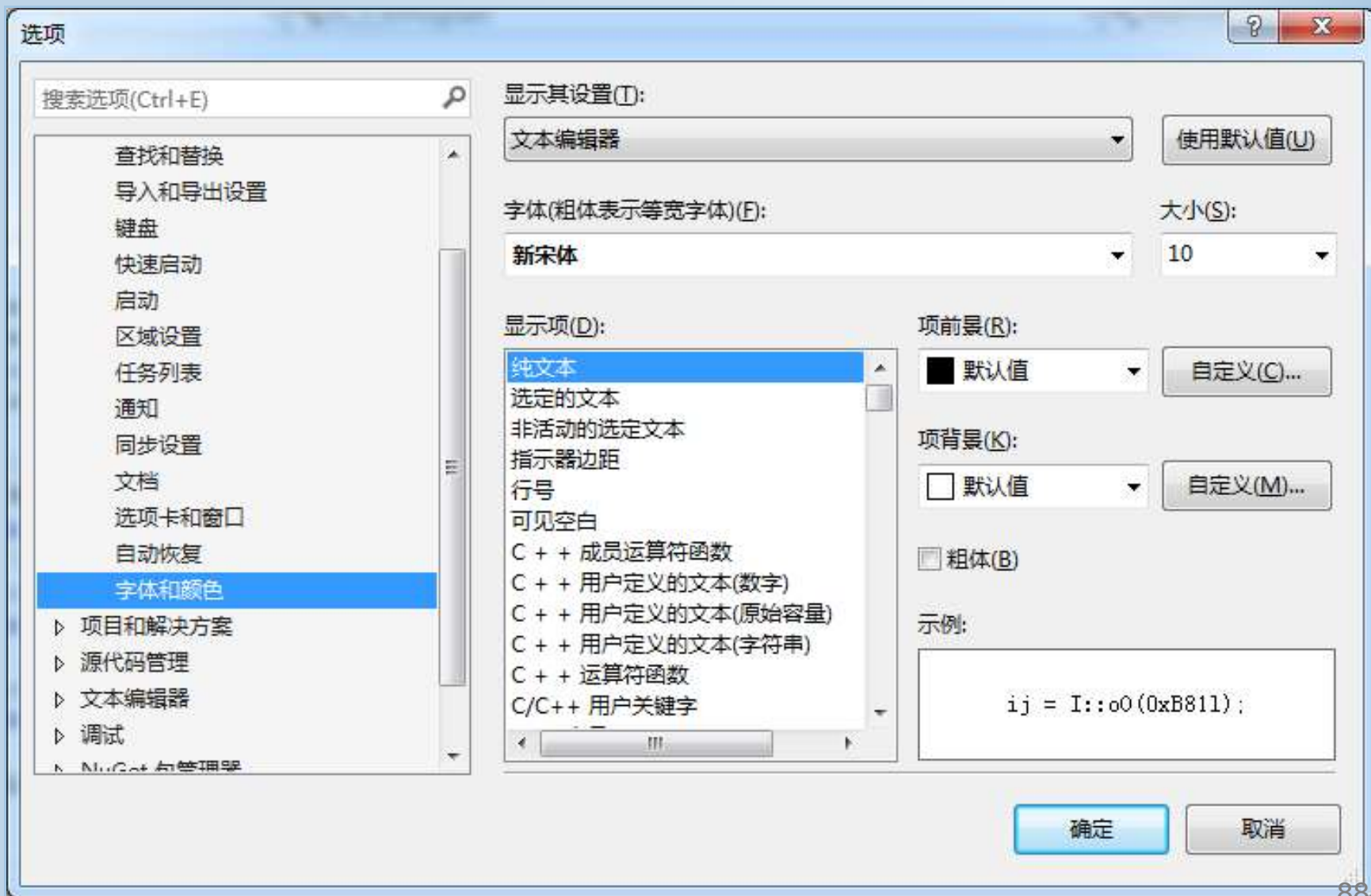
设置选项



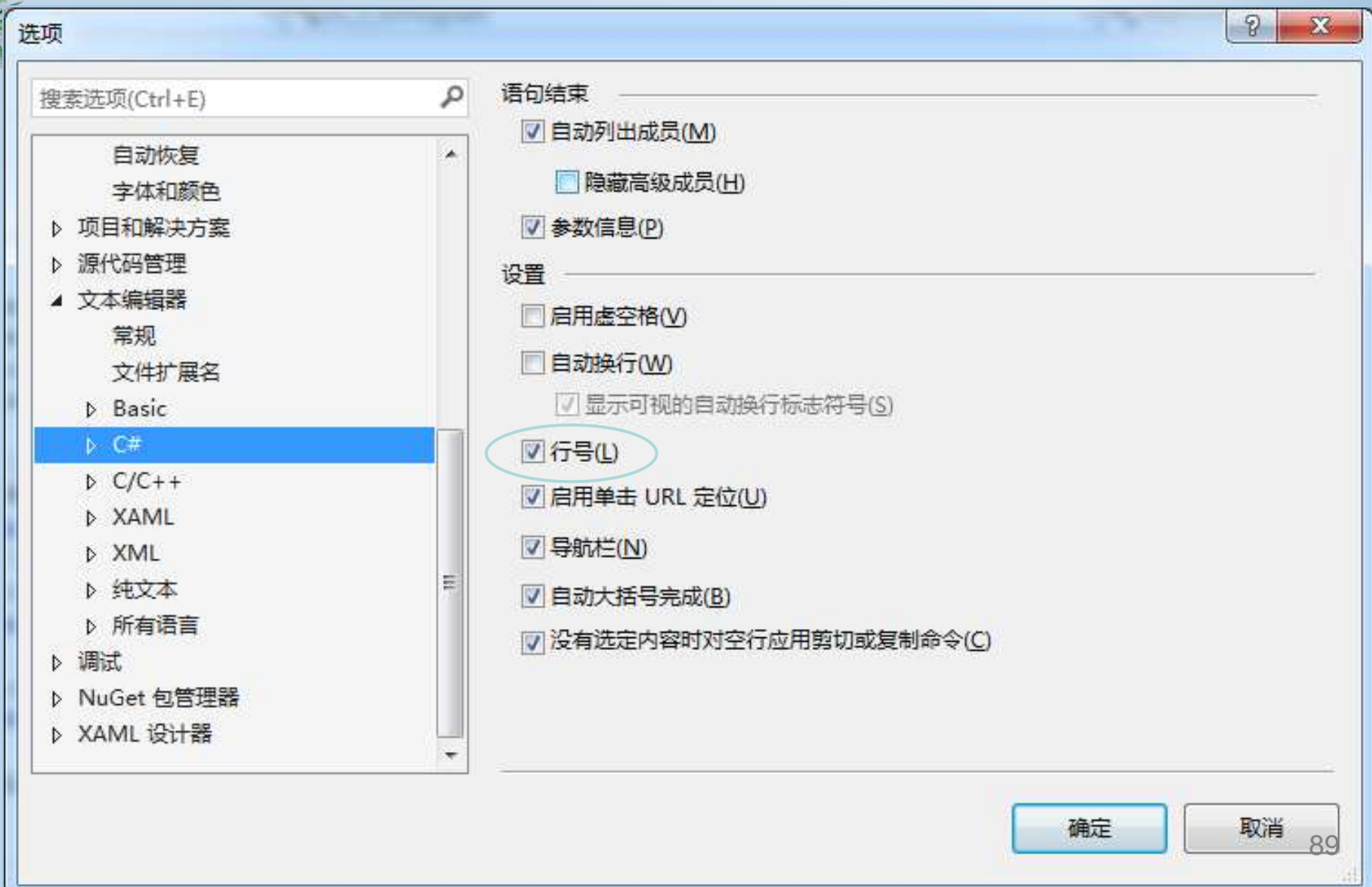
设置主题颜色



设置编辑器的字体和颜色




编辑器加入行号的选项



新建各类项目

创建新项目


最近使用的项目模板(R)

 控制台应用

C#

 ASP.NET Core Web 应用


C#


 MSTest 测试项目

C#

 类库

C#

 空白解决方案

 WPF 应用程序

C#

搜索模板(Alt+S)(S)

全部清除(C)

C#

所有平台(P)

所有项目类型(T)



控制台应用

用于创建可在 Windows、Linux 和 macOS 上 .NET 上运行的命令行应用程序的项目

C#

Linux

macOS

Windows

控制台



ASP.NET Core Web 应用

用于创建包含示例 ASP.NET Core Razor Pages 内容的 ASP.NET Core 应用程序的项目模板

C#

Linux

macOS

Windows

云

服务

Web



Blazor Server 应用

用于创建 Blazor Server 应用的项目模板，该应用会在 ASP.NET Core 应用内运行服务器端并对通过 SignalR 连接进行用户交互进行处理。此模板可用于具有丰富动态用户界面(UI)的 Web 应用。

C#

Linux

macOS

Windows

Blazor

云

Web



ASP.NET Core Web API

用于创建包含 RESTful HTTP 服务示例控制器的 ASP.NET Core 应用程序的项目模板。此模板还可以用于 ASP.NET Core MVC 视图和控制器。

C#

Linux

macOS

Windows

云

服务

Web

WebAPI



类库

用于创建面向 .NET 或 .NET Standard 的类库的项目

C#

Android

Linux

macOS

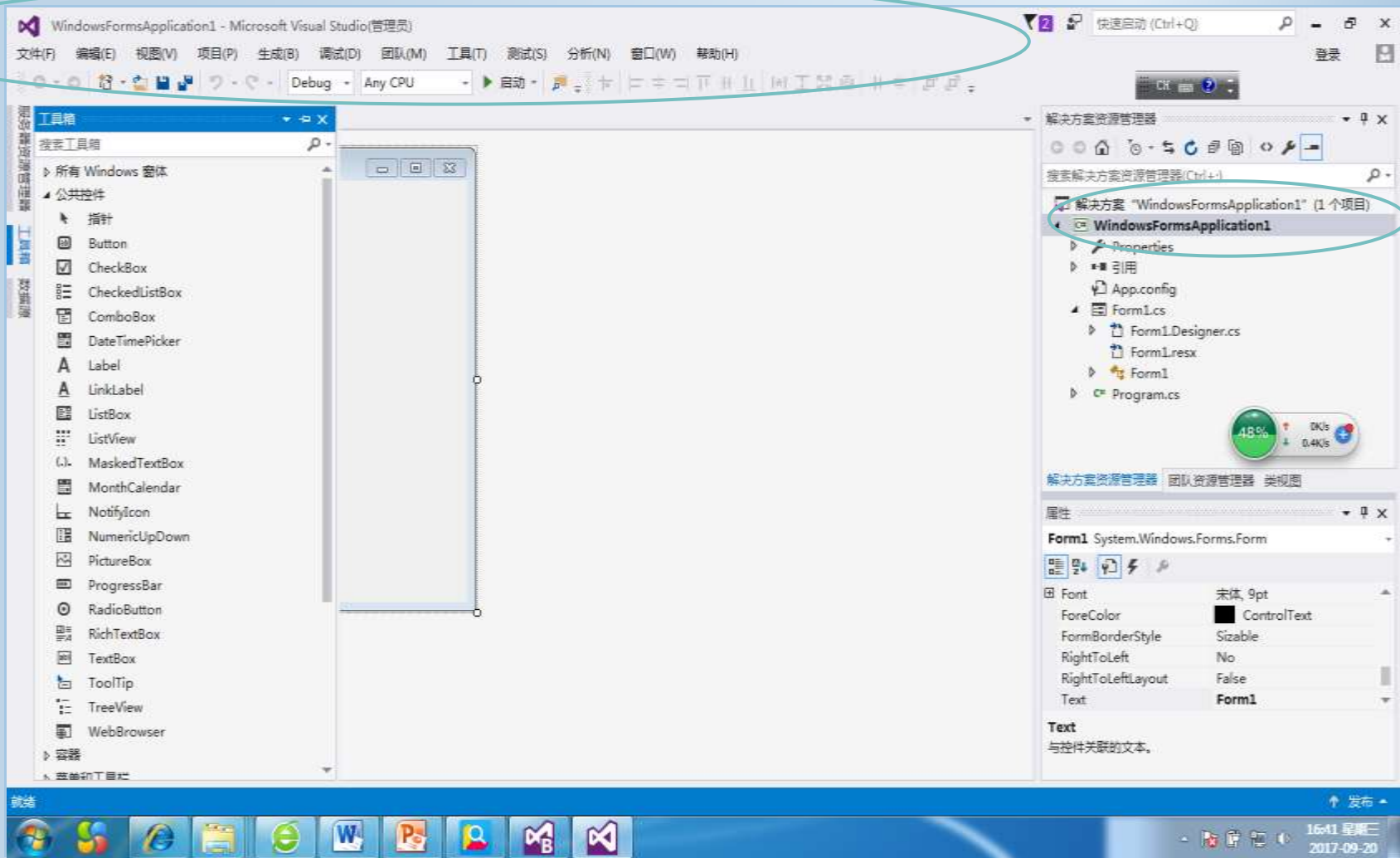
Windows

库

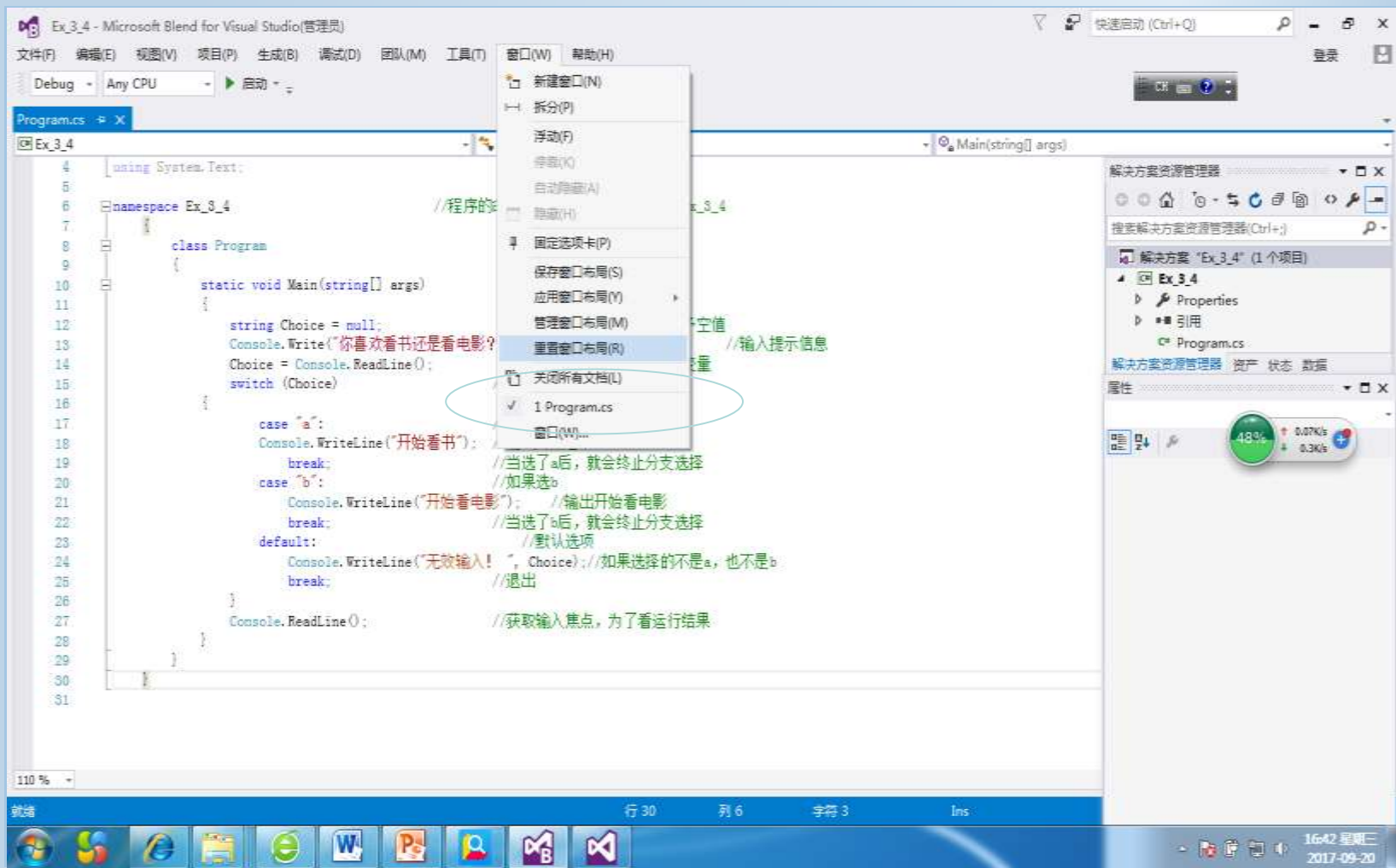
上一步(B)


下一步(N)

开发界面



重置窗口布局





安装 VS 2022 的系统要求



VS 2022安装界面





Thank You !

...