```
using System;
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
namespace BrainfxxkCompiler.Properties {
	[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.Settings
Designer.SettingsSingleFileGenerator", "17.11.0.0")]
	internal sealed partial class Settings : global::System.Configuration.ApplicationSettingsBase {
		private static Settings defaultInstance =
((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new Settings())));
		public static Settings Default {
			get {
				return defaultInstance;
			}
		}
	}
}
namespace BrainfxxkCompiler
{
	partial class Form1
	{
		private System.ComponentModel.IContainer components = null;
		protected override void Dispose(bool disposing)
		{
			if (disposing && (components != null))
			{
				components.Dispose();
			}
			base.Dispose(disposing);
		}
		#region Windows 窗体设计器生成的代码
		private void InitializeComponent()
		{
			this.components = new System.ComponentModel.Container();
			System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Form1));
			this.splitContainer1 = new System.Windows.Forms.SplitContainer();
			this.codeBox = new FastColoredTextBoxNS.FastColoredTextBox();
			this.splitContainer2 = new System.Windows.Forms.SplitContainer();
			this.resultBox = new System.Windows.Forms.RichTextBox();
			this.resultDataBox = new System.Windows.Forms.ListBox();
			this.toolStrip1 = new System.Windows.Forms.ToolStrip();
```

```
            this.runBFButton = new System.Windows.Forms.ToolStripButton();
            this.toolStripSeparator1 = new System.Windows.Forms.ToolStripSeparator();
            this.compileBFButton = new System.Windows.Forms.ToolStripButton();
            this.toolStripLabel2 = new System.Windows.Forms.ToolStripLabel();
            this.compilePath = new System.Windows.Forms.ToolStripTextBox();
            this.toolStripLabel1 = new System.Windows.Forms.ToolStripLabel();
            this.fileName = new System.Windows.Forms.ToolStripTextBox();
            this.toolStripSeparator2 = new System.Windows.Forms.ToolStripSeparator();
            this.timer1 = new System.Windows.Forms.Timer(this.components);
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer1)).BeginInit();
            this.splitContainer1.Panel1.SuspendLayout();
            this.splitContainer1.Panel2.SuspendLayout();
            this.splitContainer1.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.codeBox)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer2)).BeginInit();
            this.splitContainer2.Panel1.SuspendLayout();
            this.splitContainer2.Panel2.SuspendLayout();
            this.splitContainer2.SuspendLayout();
            this.toolStrip1.SuspendLayout();
            this.SuspendLayout();
            this.splitContainer1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.splitContainer1.Location = new System.Drawing.Point(0, 27);
            this.splitContainer1.Name = "splitContainer1";
            this.splitContainer1.Orientation = System.Windows.Forms.Orientation.Horizontal;
            this.splitContainer1.Panel1.Controls.Add(this.codeBox);
            this.splitContainer1.Panel2.Controls.Add(this.splitContainer2);
            this.splitContainer1.Size = new System.Drawing.Size(1029, 536);
            this.splitContainer1.SplitterDistance = 325;
            this.splitContainer1.TabIndex = 0;
            this.codeBox.AutoCompleteBracketsList = new char[] {
        '(',
        ')',
        '{',
        '}',
        '[',
        ']',
        '\"',
        '\"',
        '\'',
        '\''};
            this.codeBox.AutoScrollMinSize = new System.Drawing.Size(33, 20);
            this.codeBox.BackBrush = null;
            this.codeBox.CharHeight = 20;
            this.codeBox.CharWidth = 11;
            this.codeBox.Cursor = System.Windows.Forms.Cursors.IBeam;
            this.codeBox.DisabledColor = System.Drawing.Color.FromArgb(((int)(((byte)(100)))),
((int)(((byte)(180)))), ((int)(((byte)(180)))), ((int)(((byte)(180)))));
            this.codeBox.Dock = System.Windows.Forms.DockStyle.Fill;
            this.codeBox.Font = new System.Drawing.Font("Courier New", 10.8F);
            this.codeBox.ImeMode = System.Windows.Forms.ImeMode.On;
            this.codeBox.IsReplaceMode = false;
            this.codeBox.Location = new System.Drawing.Point(0, 0);
            this.codeBox.Name = "codeBox";
            this.codeBox.Paddings = new System.Windows.Forms.Padding(0);
            this.codeBox.SelectionColor = System.Drawing.Color.FromArgb(((int)(((byte)(60)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))), ((int)(((byte)(255)))));
            this.codeBox.ServiceColors =
((FastColoredTextBoxNS.ServiceColors)(resources.GetObject("codeBox.ServiceColors")));
            this.codeBox.Size = new System.Drawing.Size(1029, 325);
            this.codeBox.TabIndex = 0;
            this.codeBox.Zoom = 100;
            this.codeBox.TextChanged += new
System.EventHandler<FastColoredTextBoxNS.TextChangedEventArgs>(this.codeBox_TextChanged);
            this.splitContainer2.Dock = System.Windows.Forms.DockStyle.Fill;
            this.splitContainer2.Location = new System.Drawing.Point(0, 0);
            this.splitContainer2.Name = "splitContainer2";
            this.splitContainer2.Panel1.Controls.Add(this.resultBox);
            this.splitContainer2.Panel2.Controls.Add(this.resultDataBox);
            this.splitContainer2.Size = new System.Drawing.Size(1029, 207);
            this.splitContainer2.SplitterDistance = 810;
            this.splitContainer2.TabIndex = 0;
            this.resultBox.BackColor = System.Drawing.SystemColors.Window;
            this.resultBox.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.resultBox.Dock = System.Windows.Forms.DockStyle.Fill;
            this.resultBox.Location = new System.Drawing.Point(0, 0);
            this.resultBox.Name = "resultBox";
            this.resultBox.ReadOnly = true;
            this.resultBox.Size = new System.Drawing.Size(810, 207);
            this.resultBox.TabIndex = 0;
            this.resultBox.Text = "";
            this.resultDataBox.Dock = System.Windows.Forms.DockStyle.Fill;
            this.resultDataBox.FormattingEnabled = true;
            this.resultDataBox.ItemHeight = 15;
            this.resultDataBox.Location = new System.Drawing.Point(0, 0);
            this.resultDataBox.Name = "resultDataBox";
            this.resultDataBox.Size = new System.Drawing.Size(215, 207);
            this.resultDataBox.TabIndex = 0;
            this.toolStrip1.ImageScalingSize = new System.Drawing.Size(20, 20);
            this.toolStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.runBFButton,
            this.toolStripSeparator1,
            this.compileBFButton,
            this.toolStripLabel2,
            this.compilePath,
            this.toolStripLabel1,
            this.fileName,
            this.toolStripSeparator2});
            this.toolStrip1.Location = new System.Drawing.Point(0, 0);
            this.toolStrip1.Name = "toolStrip1";
            this.toolStrip1.Size = new System.Drawing.Size(1029, 27);
            this.toolStrip1.TabIndex = 1;
            this.toolStrip1.Text = "toolStrip1";
            this.toolStrip1.ItemClicked                    +=                    new
System.Windows.Forms.ToolStripItemClickedEventHandler(this.toolStrip1_ItemClicked);
            this.runBFButton.DisplayStyle                                         =
System.Windows.Forms.ToolStripItemDisplayStyle.Image;
            this.runBFButton.Image                                                =
((System.Drawing.Image)(resources.GetObject("runBFButton.Image")));
            this.runBFButton.ImageTransparentColor = System.Drawing.Color.Magenta;
            this.runBFButton.Name = "runBFButton";
            this.runBFButton.Size = new System.Drawing.Size(29, 24);
            this.runBFButton.Text = "运行";
            this.runBFButton.Click += new System.EventHandler(this.runBFButton_Click);
            this.toolStripSeparator1.Name = "toolStripSeparator1";
            this.toolStripSeparator1.Size = new System.Drawing.Size(6, 27);
            this.compileBFButton.DisplayStyle                                     =
System.Windows.Forms.ToolStripItemDisplayStyle.Image;
            this.compileBFButton.Image                                            =
((System.Drawing.Image)(resources.GetObject("compileBFButton.Image")));
            this.compileBFButton.ImageTransparentColor = System.Drawing.Color.Magenta;
            this.compileBFButton.Name = "compileBFButton";
            this.compileBFButton.Size = new System.Drawing.Size(29, 24);
            this.compileBFButton.Text = "编译为控制台程序";
            this.compileBFButton.Click += new System.EventHandler(this.compileBFButton_Click);
            this.toolStripLabel2.Name = "toolStripLabel2";
            this.toolStripLabel2.Size = new System.Drawing.Size(103, 24);
            this.toolStripLabel2.Text = "编译输出位置:";
            this.compilePath.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.compilePath.Font = new System.Drawing.Font("Microsoft YaHei UI", 9F);
            this.compilePath.Name = "compilePath";
            this.compilePath.Size = new System.Drawing.Size(100, 27);
            this.toolStripLabel1.Name = "toolStripLabel1";
            this.toolStripLabel1.Size = new System.Drawing.Size(43, 24);
            this.toolStripLabel1.Text = "名称:";
            this.fileName.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.fileName.Font = new System.Drawing.Font("Microsoft YaHei UI", 9F);
            this.fileName.Name = "fileName";
            this.fileName.Size = new System.Drawing.Size(100, 27);
            this.fileName.Text = "BF";
            this.fileName.TextChanged += new System.EventHandler(this.fileName_TextChanged);
            this.toolStripSeparator2.Name = "toolStripSeparator2";
            this.toolStripSeparator2.Size = new System.Drawing.Size(6, 27);
            this.timer1.Interval = 1000;
            this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 15F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(1029, 563);
            this.Controls.Add(this.splitContainer1);
            this.Controls.Add(this.toolStrip1);
            this.Name = "Form1";
            this.ShowIcon = false;
            this.Text = "Brainfxxk 编译器";
            this.splitContainer1.Panel1.ResumeLayout(false);
            this.splitContainer1.Panel2.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer1)).EndInit();
            this.splitContainer1.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)(this.codeBox)).EndInit();
            this.splitContainer2.Panel1.ResumeLayout(false);
            this.splitContainer2.Panel2.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer2)).EndInit();
            this.splitContainer2.ResumeLayout(false);
            this.toolStrip1.ResumeLayout(false);
            this.toolStrip1.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();
        }
        #endregion
        private System.Windows.Forms.SplitContainer splitContainer1;
        private System.Windows.Forms.RichTextBox resultBox;
        private System.Windows.Forms.ToolStrip toolStrip1;
        private System.Windows.Forms.ToolStripButton runBFButton;
        private System.Windows.Forms.ToolStripButton compileBFButton;
        private System.Windows.Forms.ToolStripSeparator toolStripSeparator1;
        private System.Windows.Forms.ToolStripLabel toolStripLabel1;
        private System.Windows.Forms.ToolStripTextBox fileName;
        private System.Windows.Forms.ToolStripSeparator toolStripSeparator2;
        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.ToolStripLabel toolStripLabel2;
        private System.Windows.Forms.ToolStripTextBox compilePath;
        private System.Windows.Forms.SplitContainer splitContainer2;
        private System.Windows.Forms.ListBox resultDataBox;
        private FastColoredTextBoxNS.FastColoredTextBox codeBox;
    }
}
using System;
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
[assembly:
global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETFramework,Version=v4.8",
FrameworkDisplayName = ".NET Framework 4.8")]
namespace BrainfxxkCompiler
{
    partial class InputIntDialog
    {
        private System.ComponentModel.IContainer components = null;
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
            this.numericUpDown1 = new System.Windows.Forms.NumericUpDown();
            this.okButton = new System.Windows.Forms.Button();
            ((System.ComponentModel.ISupportInitialize)(this.numericUpDown1)).BeginInit();
            this.SuspendLayout();
            this.numericUpDown1.Location = new System.Drawing.Point(12, 12);
            this.numericUpDown1.Name = "numericUpDown1";
            this.numericUpDown1.Size = new System.Drawing.Size(218, 25);
            this.numericUpDown1.TabIndex = 0;
            this.okButton.Location = new System.Drawing.Point(136, 43);
            this.okButton.Name = "okButton";
            this.okButton.Size = new System.Drawing.Size(94, 40);
            this.okButton.TabIndex = 1;
            this.okButton.Text = "确定";
            this.okButton.UseVisualStyleBackColor = true;
            this.okButton.Click += new System.EventHandler(this.okButton_Click);
            this.AcceptButton = this.okButton;
            this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 15F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(242, 95);
            this.ControlBox = false;
            this.Controls.Add(this.okButton);
            this.Controls.Add(this.numericUpDown1);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "InputIntDialog";
            this.ShowIcon = false;
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
            this.Text = "输入 ASCII 码";
            ((System.ComponentModel.ISupportInitialize)(this.numericUpDown1)).EndInit();
            this.ResumeLayout(false);
        }
        #endregion
        private System.Windows.Forms.NumericUpDown numericUpDown1;
        private System.Windows.Forms.Button okButton;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace BrainfxxkCompiler
{
	public class BFInterpreter
	{
		private int dataLength;
		public int DataLength
		{
			get => dataLength;
		}
		private int pointer;
		public int Pointer
		{
			get => pointer;
		}
		private string code;
		public BFInterpreter(int dataLength, string code)
		{
			this.dataLength = dataLength;
			this.pointer = 0;
			this.code = code;
		}
	}
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace BrainfxxkCompiler
{
	internal static class Program
	{
		[STAThread]
		static void Main()
		{
			Application.EnableVisualStyles();
			Application.SetCompatibleTextRenderingDefault(false);
			Application.Run(new Form1());
		}
	}
}
using FastColoredTextBoxNS;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
namespace BrainfxxkCompiler
{
	public partial class Form1 : Form
	{
		Thread listUpdater = new Thread(() => { });
		Style GreenStyle = new TextStyle(Brushes.Green, null, FontStyle.Regular);
		Style BlueStyle = new TextStyle(Brushes.Blue, null, FontStyle.Bold);
		Style GrayStyle = new TextStyle(Brushes.Gray, null, FontStyle.Regular);
		Style BlackStyle = new TextStyle(Brushes.Black, null, FontStyle.Bold);
		public Form1()
		{
			InitializeComponent();
			CheckForIllegalCrossThreadCalls = false;
			codeBox.DefaultStyle = (TextStyle)BlackStyle;
		}
		private void compileBFButton_Click(object sender, EventArgs e)
		{
			string path = compilePath.Text + fileName.Text + ".exe";
			BFCompiler.BFToExe(codeBox.Text, path);
			MessageBox.Show("编译完成");
			Process.Start("explorer.exe", "/select, " + path);
		}
		private void runBFButton_Click(object sender, EventArgs e)
		{
			resultBox.Text = BFCompiler.Run(codeBox.Text, out int[] data);
			if (listUpdater.ThreadState == System.Threading.ThreadState.Running)
			{
				listUpdater.Abort();
			}
			listUpdater = new Thread(() =>
			{
				if (resultDataBox.Items.Count != BFCompiler.dataLength)
				{
					resultDataBox.Items.Clear();
					for (int i = 0; i < data.Length; i++)
					{
						resultDataBox.Items.Add($"[{i}]" + "\t" + data[i]);
```

```
                    }
                    return;
                }
                for (int i = 0; i < BFCompiler.dataLength; i++)
                {
                    var o = resultDataBox.Items[i];
                    string oS = o.ToString().Split(new char[] { ']' },
StringSplitOptions.RemoveEmptyEntries)[1];
                    if (int.Parse(oS) != data[i])
                    {
                        resultDataBox.Items[i] = ($"[{i}]" + "\t" + data[i]);
                    }
                }
            });
            listUpdater.Start();
        }
        private void toolStrip1_ItemClicked(object sender, ToolStripItemClickedEventArgs e)
        {
        }
        private void fileName_TextChanged(object sender, EventArgs e)
        {
            if (fileName.TextLength < 1)
            {
                fileName.Text = "BF";
            }
        }
        private void codeBox_TextChanged(object sender, TextChangedEventArgs e)
        {
            e.ChangedRange.ClearStyle(GreenStyle);
            e.ChangedRange.SetStyle(GreenStyle, @"
            e.ChangedRange.SetStyle(BlueStyle, @"[\[\]]", RegexOptions.Multiline);
            e.ChangedRange.SetStyle(GrayStyle, @"[^\>\<\+\-\.\,\[\]]", RegexOptions.Multiline);
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
namespace BrainfxxkCompiler
{
    public partial class InputIntDialog : Form
    {
        public int num;
        public InputIntDialog()
        {
            InitializeComponent();
        }
        private void okButton_Click(object sender, EventArgs e)
        {
            num = (int)numericUpDown1.Value;
            DialogResult = DialogResult.OK;
            this.Hide();
        }
    }
}
using System;
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
namespace BrainfxxkCompiler
{
    public static class BFCompiler
    {
        public static int dataLength = 3000;
        public static string Run(string code, out int[] data)
        {
            int dataPointer = 0;
            int instructionPointer = 0;
            data = new int[dataLength];
            Stack<int> loopStack = new Stack<int>();
            StringBuilder output = new StringBuilder();
            while (instructionPointer < code.Length)
            {
                char currentInstruction = code[instructionPointer];
                switch (currentInstruction)
                {
                    case '>':
                        dataPointer++;
                        break;
                    case '<':
                        dataPointer--;
                        break;
                    case '+':
                        data[dataPointer]++;
                        break;
                    case '-':
                        data[dataPointer]--;
                        break;
                    case '.':
                        output.Append((char)data[dataPointer]);
                        break;
                    case ',':
                        InputIntDialog intDialog = new InputIntDialog();
                        int i = 0;
                        if (intDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                        {
                            i = intDialog.num;
                        }
                        data[dataPointer] = i;
                        break;
                    case '[':
                        if (data[dataPointer] == 0)
                        {
                            int loopCount = 1;
                            while (loopCount > 0)
                            {
                                instructionPointer++;
                                if (code[instructionPointer] == '[')
                                    loopCount++;
                                else if (code[instructionPointer] == ']')
                                    loopCount--;
                            }
                        }
                        else
                        {
                            loopStack.Push(instructionPointer);
                        }
                        break;
                    case ']':
                        if (data[dataPointer] != 0)
                        {
                            instructionPointer = loopStack.Peek();
                        }
                        else
                        {
                            loopStack.Pop();
                        }
                        break;
                    default:
                        break;
                }
                instructionPointer++;
            }
            return output.ToString();
```

```
        }
        public static string CompileToCSharp(string brainfuckCode)
        {
            StringBuilder cSharpCode = new StringBuilder();
            cSharpCode.AppendLine("using System;");
            cSharpCode.AppendLine("using System.Collections.Generic;");
            cSharpCode.AppendLine("using System.Text;");
            cSharpCode.AppendLine("namespace BrainfuckProgram");
            cSharpCode.AppendLine("{");
            cSharpCode.AppendLine("class Program");
            cSharpCode.AppendLine("{");
            cSharpCode.AppendLine("static void Main(string[] args)");
            cSharpCode.AppendLine("{");
            cSharpCode.AppendLine($"byte[] memory = new byte[{dataLength}];");
            cSharpCode.AppendLine("int pointer = 0;");
            cSharpCode.AppendLine("List<char> output = new List<char>();");
            cSharpCode.AppendLine("int inputIndex = 0;");
            cSharpCode.AppendLine("try");
            cSharpCode.AppendLine("{");
            for (int i = 0; i < brainfuckCode.Length; i++)
            {
                char c = brainfuckCode[i];
                switch (c)
                {
                    case '>':
                        cSharpCode.AppendLine("pointer++;");
                        break;
                    case '<':
                        cSharpCode.AppendLine("pointer--;");
                        break;
                    case '+':
                        cSharpCode.AppendLine("memory[pointer]++;");
                        break;
                    case '-':
                        cSharpCode.AppendLine("memory[pointer]--;");
                        break;
                    case '.':
                        cSharpCode.AppendLine("output.Add((char)memory[pointer]);");
                        break;
                    case ',':
                        cSharpCode.AppendLine("Console.WriteLine(\"请输入 ASCII 码\");");
                        cSharpCode.AppendLine("memory[pointer]                    =
byte.Parse(Console.ReadLine());");
                        break;
                    case '[':
                        cSharpCode.AppendLine("while (memory[pointer] != 0)");
                        cSharpCode.AppendLine("{");
                        break;
                    case ']':
                        cSharpCode.AppendLine("}");
                        break;
                }
            }
            cSharpCode.AppendLine("}");
            cSharpCode.AppendLine("catch (Exception ex)");
            cSharpCode.AppendLine("{");
            cSharpCode.AppendLine("Console.WriteLine(ex.Message);");
            cSharpCode.AppendLine("}");
            cSharpCode.AppendLine("Console.WriteLine(new string(output.ToArray()));");
            cSharpCode.AppendLine("Console.ReadKey();");
            cSharpCode.AppendLine("}");
            cSharpCode.AppendLine("}");
            cSharpCode.AppendLine("}");
            return cSharpCode.ToString();
        }
        static void CompileToExe(string cSharpCode, string outputFilePath)
        {
            CodeDomProvider codeProvider = CodeDomProvider.CreateProvider("CSharp");
            CompilerParameters compilerParams = new CompilerParameters();
            compilerParams.GenerateExecutable = true;
            compilerParams.OutputAssembly = outputFilePath;
            CompilerResults                    compilerResults                    =
codeProvider.CompileAssemblyFromSource(compilerParams, cSharpCode);
            if (compilerResults.Errors.Count > 0)
            {
                foreach (CompilerError error in compilerResults.Errors)
                {
                    if (!error.IsWarning)
                        throw new Exception(error.ErrorText);
                }
            }
        }
        public static void BFToExe(string brainfuckCode, string outputFilePath)
        {
            CompileToExe(CompileToCSharp(brainfuckCode), outputFilePath);
        }
    }
}
using System;
using System.Reflection;
[assembly:
global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETFramework,Version=v4.8",
FrameworkDisplayName = ".NET Framework 4.8")]
namespace BrainfxxkCompiler.Properties {
    using System;

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyType
dResourceBuilder", "17.0.0.0")]
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
```

```
	internal class Resources {
		private static global::System.Resources.ResourceManager resourceMan;
		private static global::System.Globalization.CultureInfo resourceCulture;

[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
		internal Resources() {
		}

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Editor
BrowsableState.Advanced)]
		internal static global::System.Resources.ResourceManager ResourceManager {
			get {
				if (object.ReferenceEquals(resourceMan, null)) {
					global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("BrainfxxkCompiler.Properties.Resources",
typeof(Resources).Assembly);
					resourceMan = temp;
				}
				return resourceMan;
			}
		}

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Editor
BrowsableState.Advanced)]
		internal static global::System.Globalization.CultureInfo Culture {
			get {
				return resourceCulture;
			}
			set {
				resourceCulture = value;
			}
		}
	}
}
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
[assembly: AssemblyTitle("BrainfuckCompiler")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("LX")]
[assembly: AssemblyProduct("BrainfuckCompiler")]
[assembly: AssemblyCopyright("Copyright © LX 2023")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
[assembly: ComVisible(false)]
[assembly: Guid("2b056b14-b42d-4aac-b378-56082f2c8f33")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```