C# CALCULATOR APPLICATION AS A TOOL TO LEARN OBJECT 
ORIENTED PROGRAMMING

E. Dumić1, A. Bjelopera2, S. Ružić1

1University North (CROATIA)  
2University of Dubrovnik (CROATIA)

Abstract

This paper presents some of the basic principles of object oriented programming (OOP), using one of the common programming language, C sharp - C#. OOP is nowadays usually used programming paradigm, so it should be used to give students different project assignments. Elementary difference between procedural and object oriented paradigm will be discussed. Comparison between C#, C++ and Java programming languages will be also briefly noted. Afterwards, Windows application using C# will be briefly explained, by programming a simple calculator with basic mathematical functions (add, subtract, multiply, divide), as well as basic trigonometric functions (sine, cosine, power, exponential, factorial, etc). Limitations that pose current standard variables regarding its size will be briefly discussed. Finally, it will be explained that, by using already implemented forms and classes in C# language, students can learn basics about OOP, but also about mathematical principles that can make application run faster.

Keywords: C# calculator, OOP, programming.

1 INTRODUCTION

High-level programming languages have been used because of the need to replace complex and hard understandable machine language with more acceptable programming language easier and more convenient for usage [1]. Development of high-level programming languages has made it easier for a person to write a program. In such programming or program development using high-level programming languages, one command replaces a complete set of machine language commands. The programming logic using such a language is similar to mathematical logic and the logic of human thinking, and the used commands are actually abbreviations of English words.

High-level programming languages enabled writing programs to people who knew very little about the structure of the computer. The program written in one of the high-level programming languages can be applied to different models of processors and computers. Such application could not be used in machine languages, and is considered to be one of the most important features of high-level programming languages. If the program is written in a higher programming language, the computer must have a program translator and its task is to convert the program to a machine language that the computer understands and can execute a particular program. Nowadays, programming is increasingly being used and different programming languages are developed, enabling the writing of programs to different professions experts. The languages are adjusted to different areas and have their own specific tasks. C# (C sharp) programming language is a part of the group of leading programming languages used today. It is object oriented so writing a program using OOP (object oriented programming) is relatively simple. In addition, it is accompanied by a powerful MS (Microsoft) Visual Studio with a large number of features that maximally speed up program development [2]. C# is a programming language designed for the Microsoft .NET platform. The basic part of .NET is certainly the .NET Framework. It is a special infrastructure that offers programmers complete solutions and functionalities in order to speed up and simplify the development of applications of all types and structures.

The most important component of .NET Framework is called the Common Language Runtime (CLR), while the instructions in the source code are translated using the JIT (Just in Time) compiler which analyzes the intermediate language code and provides a very efficient machine code that is quickly executed. When executing the program, the JIT translator only translates what is needed and the translated code is stored in the memory so that it can be reused. C# is intended for developers working on a Windows platform and its syntax is similar to C or C++ languages or other object-oriented languages such as Java and JavaScript. It is often used for creating Web applications and
Windows applications. C# simplifies the creation of standard user interfaces for Windows desktop applications using several template types (Windows Forms, WPF, etc.).

The purpose of this paper was to create a calculator application. The application was developed in the MS Visual Studio 2010 programming environment using the Windows Forms graphical interface. Windows Forms is used to create applications using .NET Framework. The application calculates arithmetic operations, sinus trigonometric and hyperbolic functions, roots, potencies, etc. It also contains the keys which are used, when calculating the sinus functions, to select whether the input is in degrees or radians. The application is simple and useful for calculation of different mathematical problems.

Chapter 2 describes properties of procedural and object oriented programming and also compares C#, C++ and Java programming languages. Chapter 3 explains the C# calculator application and its functions. Chapter 4 brings the conclusion.

2 PROCEDURAL VERSUS OBJECT ORIENTED PROGRAMMING

In procedural programming, the program consists of a series of commands that the computer needs to execute i.e. the computer process in terms of program status and state-changing commands is described [3]. High-level procedural programming languages use variables and more complex commands, and enable the computation of complex mixed expressions that are consisted of relational, arithmetic and logical operators. The expression value is assigned to the memory location after calculation. Procedural programming is mostly based on orders of conditional and unconditional branching and by using loops. When conditional branching is used, in order to execute a block of commands, the set up condition must be satisfied. The unconditional branching commands do not have any conditions, and they are most often used for unconditional leap or for calling of subprogram or the procedure. Loops are used to execute orders a number of times or to change the conditions. Using of procedural programming enables combination of the reversal parts of commands in one place [4]. The name of subprogram is used in order to call a procedure or a subprogram, and after its calling, the commands contained in this subprogram are executed.

Fig. 1 shows the execution of the procedure: the first part of the code that is in the main program is executed. The first procedure is called and it is jumped from the main program back to the subprogram to execute part of its code. After executing these instructions, the program returns back to the main one and resumes execution where it was previously interrupted. Once a part of the code in the main program is executed, a new procedure is called again and the procedure is repeated. The program written in such manner allows the developer a better viewing and less possibility of error, i.e. if the subprogram is correctly written, each time it is called it gives the exact result.

In procedural programming the main program has control over the procedures and controls the transmission of the arguments to each procedure. Thus, one program is divided into several smaller parts called procedures. Along with procedural programming, nowadays the most prominent programming is object-oriented programming [5]. Procedural programming is based on calling a
number of functions and their task while OOP is based on objects and their mutual communication. OOP considers data as critical element in program and ties data more closely to the functions of objects that operate on this data. It allows decomposition of problems into objects which contain data and functions. Data of an object can be accessed only by object’s function, but functions of different objects can communicate (Fig. 2) [6].

OOP has introduced new features to the world of programming, which are at the very top of today’s programming. The object oriented model enables a more logical design and implementation of the program, and in a more comprehensible way depicts the real world since every real system can be represented as a set of objects that are in some kind of relation or interconnected.

The OOP started its development twenty years ago in response to the so-called software crisis. This is the concept that emerged when the demands for more complex software increased because of the faster development of computers and their capabilities. The organization of large software projects could no longer withstand the huge demands of users while designing, manufacturing and maintaining software became too expensive and complex work. Huge amount of effort yielded too little results. Also, the bugs in the software have become hard to notice and even harder to remove. The developers have shown the users what are the capabilities of computers so in response they began to look for much more than the developers could expect. To meet customers’ demands, there was a need to expand developer teams that would be able to work on more complex projects and the key was division of work into parts that have clearly defined tasks and do not enter each other into the “interior” and latter to assemble all the parts.

As already mentioned, the OOP is based on the use of objects constructed of variables called attributes, the functions called methods and identifier that helps in distinguishing objects of the same type [7]. Attributes are variables that describe the current properties of a particular object. Methods are actually functions that can change or read the object properties. Processing in OOP is associated with methods and they are just another expression for procedure. OOP requires description of relationships between procedures and shared data [8]. With grouping of objects with the same attributes and methods, it is possible to create classes and subclasses where these objects belong. Classes can be compared to types in a traditional programming languages but the difference is that users can define new types in an OOP language [8]. The objects in this type of programming can be considered objects and occurrences that we encounter on a daily basis in today's world. Objects can be compared to variables in traditional programming. The creation of a program solution based on customer needs is much easier task for the people who develop the application.

The OOP usually consists of three basic concepts: encapsulation, inheritance and polymorphism. Using these concepts, it is possible to design computer systems that can be used in many different applications. Encapsulation is a procedure used to disable direct access to attribute values of certain objects from some other parts of the code. This means that during the performance on objects, any possibility of the object to end up in unpredictable state is eliminated. This can happen if the object is accessed from several different parts of the code and lead the object into unpredictable state if security mechanisms for such cases are not embedded in the methods for accessing the values of its variables. Inheritance means that the class or object is based on an existing class or object, using the existing attributes and methods. These classes then inherit all or only some parts of their superclass and become its subclass. The subclass can be supplemented with new attributes and methods, and
can modify or supplement the existing methods of its superclass, but not its attributes. Polymorphism can be divided into two different concepts. The first concept allows the object to have multiple methods of the same name which receive different data through the arguments. A particular method is executed depending on which data is sent through those arguments. The second concept is related to inheritance and allows the storage of a subclass type object in the variable defined for the type of data of its superclass. This approach allows the preservation of multiple variables of different subclasses in a single list declared for their joint superclass. OOP gives answers to many problems. It offers mechanisms that automatically control interactions between software parts. Object oriented programming tools help manage complexity and the goal is to provide a way to structure and manage relationships between large number of system components [8]. OOP was conceived by developers as a way to make their life easier to handle. Its structuring reduces number of connections among system components. It is also easier to isolate bugs and determine which methods are responsible for the bugs. Some of OOP’s programming languages are: Java, C ++, C #, Python, Delphi and PHP.

2.1 Comparison of C#, Java and C++

Today, programming languages are so powerful that the quality of the software mostly depends on the capabilities of programmers [9]. However, differences and similarities between languages exist and it is important to be acquainted with them. C#, C++ and Java are successors of C language. They belong to new generation languages and include modern features. C# and Java use the so-called "garbage collection" that frees the developer of thinking about low-level programming parts i.e. frees the memory and "recycles" objects that are not in use for a longer period of time. In C++ this property is not built into the system. Syntax of these programming languages is very similar. The basic common feature for C# and Java is compiling to intermediate language. For C#, that language is called MSIL (Microsoft Intermediate Language) or CIL (Common Intermediate Language), while in Java it is called Java bytecode. Java does not translate source code into machine code yet it is already compiled to bytecode that requires a special environment to be executed. This environment is called JVM (Java Virtual Machine). The idea is that if a code is written and compiled on one platform (e.g. Mac OS), the same bytecode can run on all other platforms that have JVM (Windows, Linux) without the need for recompiling on that platform [10]. C# has obvious similarities with the Java, but both languages have emerged as an upgrade to C++ language. C++, Java and C # are among the most popular programming languages and they are mostly used nowadays. First of all, C# and Java are the choice for those who like C or C++, but also for those who have learned to program in C language and C++, and would like to learn more about new technologies. The future definitely belongs to C# and Java languages.

3 RESULTS

3.1 C# calculator graphical user interface (GUI)

Important feature of the Windows application is a rich graphical user interface (GUI). Windows applications can consist of one or more forms that contain different controls (keys, labels (labels), combobox, listbox, text input field, etc.). Each form and control has the ability to accept the corresponding events produced by the user (e.g., left mouse click or just a click, double left click, right click, etc.). All actions are called events, so this type of programming can be also called program-driven events. Because of this characteristic, this type of programming is also indicated by a different approach in the development of the application in relation to, for example, procedural programming. Firstly, the appearance of the user interface is generated, and then the events for the particular forms and controls to which the system will react, are determined. Finally, for each event, an appropriate function is written that defines the application response to the user action. Since the interface is put into the main focus of the application development, modern Windows development environments offer predefined classes that describe commonly used forms and controls. Often, this kind of Windows application development is also called visual programming. Similar behavior can be also found within Visual Studio environments, offering a rich set of predefined class libraries. The application, as mentioned above, was created using the Microsoft Visual Studio 2010 Integrated development environment (IDE) using Windows Form, Fig. 3.
Fig. 3 shows calculator interface with all functions. The calculator has twenty-four keys with different mathematical functions which will be explained in the next section and two "Radio" keys that are used to calculate the trigonometric functions (sine, cosine, tangent functions) depending on input (in degrees or radians).

Creating the layout of the application's interface is the first step in Windows application form. Afterwards comes the second stage where, for each event that is supposed to control the interface formed in the first step (e.g. by left-clicking, double-clicking the left mouse button, right-clicking the mouse, etc.), associated function is generated and connected when the event is registered. So, programming is actually reduced to writing code functions that need to be activated when an event occurs. That is why this approach to programming is called program-driven or event-driven. This mode of programming is considered natural because it is considered that using an application is nothing more than launching an application and a series of user actions with a "response" program, until the application finishes. Therefore, application creation is reduced to create the visual appearance of the application, i.e. the graphical user interface, using Windows control and writing the appropriate code which describes the desired behavior, i.e. the response of the application to individual user actions. These steps are extremely important and very often, during the realization of the application: i.e. some forms of some controls are created, and then the appropriate code for control events is printed and then the same procedure is repeated for some new forms and controls. From the basic form there is always a version of the application with the appropriate user interface. Particular care needs to be given for the design or appearance of the interface because it determines behavior of the whole application. It can be said that developing a Windows application takes place by firstly creating an interface that includes a list of all events and actions that need to follow these events. Then, in the second stage, the code is written for the functions that are called upon to activate different events.
3.2 Calculator functions

Integrated development environments (IDE) are designed to enable the simplest application development. In this code, in Tab. 1, commands starting with "using" can be noticed, and in this way, the code includes appropriate libraries with the classes that they execute (i.e. corresponding namespaces). In addition to the already defined namespaces, a System.Globalization namespace is inserted that contains classes which define computer-related information including language, state/region, calendar, date formats, sort order, etc. The main form name can be changed arbitrarily, as well as the name of certain functions. The dec_point character variable is defined as a character (char) taken from the system, used to write a decimal comma or decimal point - depending on computer settings. In Croatian language, a comma is used, while for example decimal point is used in English language, so that is why System.Globalization was used. For computer settings, the Croatian decimal point can be used to separate thousands. The dec_point variable is associated with the button19 variable that is defined as a decimal separator and is switched from the character (char) variable to the string.

```
namespace Kalkulator_Zavrśni_rad
{
    public partial class Form1 : Form
    {
        Double rezultat = 0;
        String operacija = "";
        bool operacija_završena = false;
        double kut;
        double d;
        char dec_point = Convert.ToChar(CultureInfo.CurrentCulture.NumberFormat.NumberDecimalSeparator);
        public Form1()
        {
            InitializeComponent();
            button19.Text=dec_point.ToString();
        }
    }
}
```

Table 1. Main program.
Numbers 0 to 9 including the decimal separator are assigned to the same event called **button_Click**, which is described by code in Tab. 2. Sender command is used to assign all buttons to the **button_Click** event. All numbers are stored in **textBox1** string variable. At the beginning of the code a variable **operacija_izvrsena** is defined, which is boolean type set to false, and serves to write multi-digit numbers. When running the application, the calculator screen displays zero. To have this zero removed when a number is written, or when pushing certain keys, the first if command is used. The code within the other if command is used to write decimal numbers, using a **dec_point** variable earlier explained. If the number in the calculator is entered using the computer keyboard and the decimal separator is written several times, the application will recognize that it is not the number and will return the display to 0.

**Table 2. Event example: left button click for numbers 0-9 and decimal separator.**

```
private void button_Click(object sender, EventArgs e)
{
    Button button = (Button)sender;
    if ((textBox1.Text == "0") || (operacija_izvrsena))
        textBox1.Text = ";
    operacija_izvrsena = false;
    if (button.Text == dec_point.ToString())
    {
        if(!textBox1.Text.Contains(dec_point.ToString()))
        {
            if (textBox1.Text=="")
                textBox1.Text = "0" + textBox1.Text + button.Text;
            else
                textBox1.Text=textBox1.Text + button.Text;
        }
    }
    else
        textBox1.Text = textBox1.Text + button.Text;
}
```

All arithmetic operations have assigned an event **button_aritmeticki** described in code in Tab. 3. At the beginning, as in the previous function, the sender joins the function with all the keys that will be used for arithmetic operations (addition, subtraction, multiplication and division). The variable **operacija** (string) is defined at the beginning of the form and serves to print the character of a particular arithmetic operation. Here a **TryParse** method is used with two conditions. The first condition is to enter a string that can contain numbers, letters and other characters, and the other condition is to check whether the string number is written in the correct format. If the number is entered incorrectly (entering letters or other characters) or beyond the boundaries that are defined for the variable type **double**, there will be an error or no "communication" between the first and second conditions and the desired number will have to be re-entered. If the number is entered in the correct format, the variable **d** that is defined at the beginning will recognize that the entry is correct and that number will be saved in the variable **rezultat** that is also defined at the beginning of the form (Tab. 1). This method will be used in most of the functions that the calculator contains and in each of these functions its purpose is the same. When the next number is written, because variable **operacija_izvrsena** was set to true, **textBox1** string will be cleared, and the next number will be written, Tab. 2.

**Table 3. Event example: left button click for all arithmetic operations.**

```
private void button_aritmeticki(object sender, EventArgs e)
{
    Button button = (Button)sender;
    operacija = button.Text;
    if (double.TryParse(textBox1.Text, out d))
    {
        rezultat = d;
        operacija_izvrsena = true;
        label1.Text = rezultat + " " + operacija;
    }
    else
        textBox1.Text = "0";
}
```
With the code in Tab. 4, result for specific arithmetic operation is being performed. At the beginning, TryParse is used to enter the number in the correct format. To perform arithmetic operations, the switch-case command condition was used. The commands of the switch are seen as a condition used by the variable operacija, which is a string type and serves to select the arithmetic operation to be calculate. The first entered number is saved in the variable rezultat and after the selection of a particular operation, another number is stored in the variable d. In this block of commands it is apparent that some mathematical functions are not included, but will be explained later. After executing the switch-case command, the variable operacija_izvrsena is set to false for the same reason that has already been explained.

```
private void button_jednako_Click(object sender, EventArgs e)
{
    if(double.TryParse(textBox1.Text, out d))
    {
        switch (operacija)
        {
            case "+":
                textBox1.Text = (rezultat + d) + "";
                break;
            case "/":
                textBox1.Text = (rezultat - d) + "";
                break;
            case ".":
                textBox1.Text = (rezultat * d) + "";
                break;
            case "/":
                textBox1.Text = (rezultat / d) + "";
                break;
            case "+":
                textBox1.Text = (rezultat % d) + "";
                break;
            case "+":
                textBox1.Text = Math.Pow(rezultat, d) + "";
                break;
            case "+":
                textBox1.Text = Math.Pow(d, 1/rezultat) + "";
                break;
            default:
                break;
        }
        operacija_izvrsena = false;
    }
    else
    {
        textBox1.Text="0";
    }
}
```


Here, we will also explain only calculation for factorial function, of double type, normally (Tab. 5) or using logarithmic approximation (Tab. 6). To normally calculate factorial for some number, Tab. 5, the function is described as a simple for loop that counts product of numbers from 1 to the number that is entered in the calculator. At the beginning, two variables are defined. Variable a of the double type is assigned starting value 1 for the factorial calculation, and the second variable cijeli of the type int is used in a for loop. This variable must be set to integer because the factorial calculation is only performed with integers, so it serves to enter the final number using the TryParse method. The calculator can calculate factorial up to 170, because when entering larger number, the calculator freezes because the factorial is greater than the range that the double type variable can calculate. The double data type uses 8 bytes for data processing. Real numbers are generated in the computer by bits where, for example, in a double type, 1 bit is used to store sign of the number, 11 bits are used to store the exponent and the remaining 52 bits to store mantissa. The 11-bit feature has therefore boundaries: $2^{\pm 2^{10}} = 1.7 \times 10^{\pm 308}$. For this reason, the condition is set with the if command, that if a number greater than 170 is entered, calculator displays "Infinity".
The second function (ln !) was used to calculate the factorial in the logarithmic scale by Ramanujan's approximation formula, Tab. 6. The used formula was:

\[
\ln(n!) \approx n \cdot \ln(n) - n + \frac{1}{6} \ln\left(n\left(1 + 4n(1 + 2n)\right)\right) + \frac{1}{2} \ln(\pi)
\]

At the beginning, a variable logfact was defined and a result of Ramanujan's approximation formula was stored in it. TryParse method inserts the number to be calculated in variable d. The formula used the required mathematical functions Math.Log and constant Math.PI.

### 4 CONCLUSIONS

Integrated development environments (IDE) are designed to make application development simple and interesting. The Calculator (Kalkulator) application was created in the Microsoft Visual Studio 2010 IDE as a Windows form in C#. C# is a very powerful tool that is, alongside Java and C++, nowadays increasingly used and is becoming one of the leading programming language. By opening a new project, the IDE automatically adds the framework of the application form. In the project, class libraries have already been defined, which makes software development simpler to write. The presented application is easy to use, so it can be used by students in different teaching activities, but also to solve some of the simpler mathematical problems.

### REFERENCES


