MINISTRY OF SCIENCE AND EDUCATION

NATIONAL TECHNICAL UNIVERSITY

"KHARKIV POLYTECHNIC INSTITUTE"

DEPARTMENT OF SOFTWARE ENGINEERING AND MANAGEMENT

INFORMATION TECHNOLOGIES

METHODICAL RECOMMENDATION TO

SOFTWARE ENGINEERING

LABORATORY PRACTICE

Kharkiv

2018

CONTENT

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

Lab № 1

Modelling of Domain

Goal: **Learning Use Case and Activity Diagrams by using Visual paradigm**

Tasks:

1. Get the task and analyze the computational algorithms.
2. Develop the Activity diagram.
3. Develop the Use Case diagram.
4. Prepare the report of the work

Progress of the lab.

1. Get the task and analyze the computational algorithms.

According to the task in Table 1 student should analyze computational algorithms.

Table 1 – Task for lab№1

1	$$y = \begin{cases} \sum_{i=1}^{n}(i+x)^2, & x < 0 \\ \sum_{i=0}^{n-1}\prod_{j=1}^{n}\frac{x+i}{i-j}, & x \geq 0 \end{cases}$$	5	$$y = \begin{cases} \sum_{i=2}^{n-1}\frac{x}{i}, & x \leq 0 \\ \sum_{i=0}^{n-1}\sum_{j=0}^{i}\frac{i}{j+x}, & x > 0 \end{cases}$$
2	$$y = \begin{cases} \sum_{i=1}^{n-1}\sum_{j=1}^{n}(x-i+j), & x < 0 \\ \sum_{i=0}^{n-1}\frac{x}{i}, & x \geq 0 \end{cases}$$	6	$$y = \begin{cases} \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{1}{x-i-j}, & x < 0 \\ \prod_{i=0}^{n-3}(x-i), & x \geq 0 \end{cases}$$
3	$$y = \begin{cases} \prod_{j=2}^{n-2}(j+1), & x < 0 \\ \sum_{i=0}^{n-1}\prod_{j=0}^{n-1}(x+i+j^2), & x \geq 0 \end{cases}$$	7	$$y = \begin{cases} \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\frac{1}{x-i+j}, & x \leq 0 \\ \prod_{i=1}^{n}(\frac{1}{x}-\frac{1}{i}), & x > 0 \end{cases}$$
4	$$y = \begin{cases} \prod_{i=0}^{n-1}(i^2+i), & x \leq 0 \\ \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\frac{x}{i+j}, & x > 0 \end{cases}$$	8	$$y = \begin{cases} \sum_{i=0}^{n}\prod_{j=i}^{n-1}(i^2+j), & x < 0 \\ \sum_{i=1}^{n-2}(i-x), & x \geq 0 \end{cases}$$

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

9	$y = \begin{cases} \sum\limits_{i=1}^{n-1}\sum\limits_{j=1}^{n} \dfrac{j}{j^2+j}, x<0 \\ x-\sum\limits_{i=1}^{n-1} i, x \geq 0 \end{cases}$	11	$y = \begin{cases} \prod\limits_{i=0}^{n-1}(i^3+i), x \leq 0 \\ \sum\limits_{i=0}^{n-1}\sum\limits_{j=0}^{n-1} \dfrac{x}{i+j}, x>0 \end{cases}$
10	$y = \begin{cases} \sum\limits_{i=0}^{n}(x-i)^2, x \leq 0 \\ \prod\limits_{i=1}^{n}\prod\limits_{j=0}^{n-1}(x-i-j), x>0 \end{cases}$	12	$y = \begin{cases} \prod\limits_{j=2}^{n-2} j^2, x<0 \\ \sum\limits_{i=0}^{n-1}\prod\limits_{j=0}^{n-1}(x+i^2+j), x \geq 0 \end{cases}$

For example, we obtained the following task:

The application has to calculate the following formula: $S = \sum\limits_{i=1}^{n} a \cdot i^2$. *The application should also provide the data input from keyboard and file and output to screen and file.*

As we see in expression, we have unknown variables a and n. To calculate the formula student should find the values of a and n first.

Let $a = 5$; $n = 4$. Than we obtain:

$$S = \sum_{i=1}^{n} a \cdot i^2 = \sum_{i=1}^{4} 5 \cdot i^2 = 5 \cdot 1^2 + 5 \cdot 2^2 + 5 \cdot 3^2 + 5 \cdot 4^2$$

2. Develop the Activity diagram.

The free specialized software **Visual Paradigm Community Edition** [https://www.visual-paradigm.com/] is recommended to draw the Activity diagram (Figure 1.1 - 1.2).

Figure 1.1 – Visual Paradigm main menu

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018
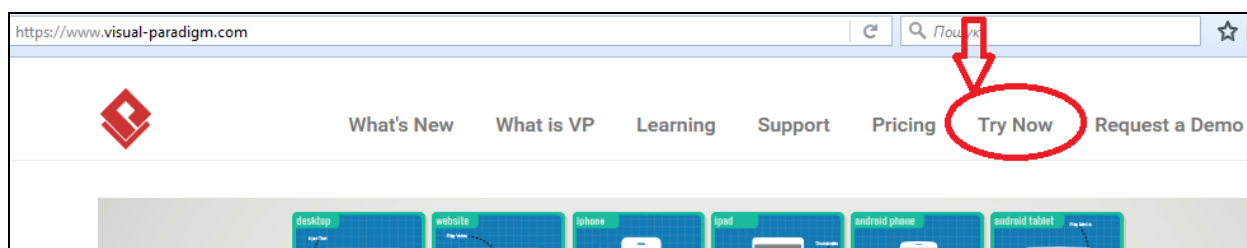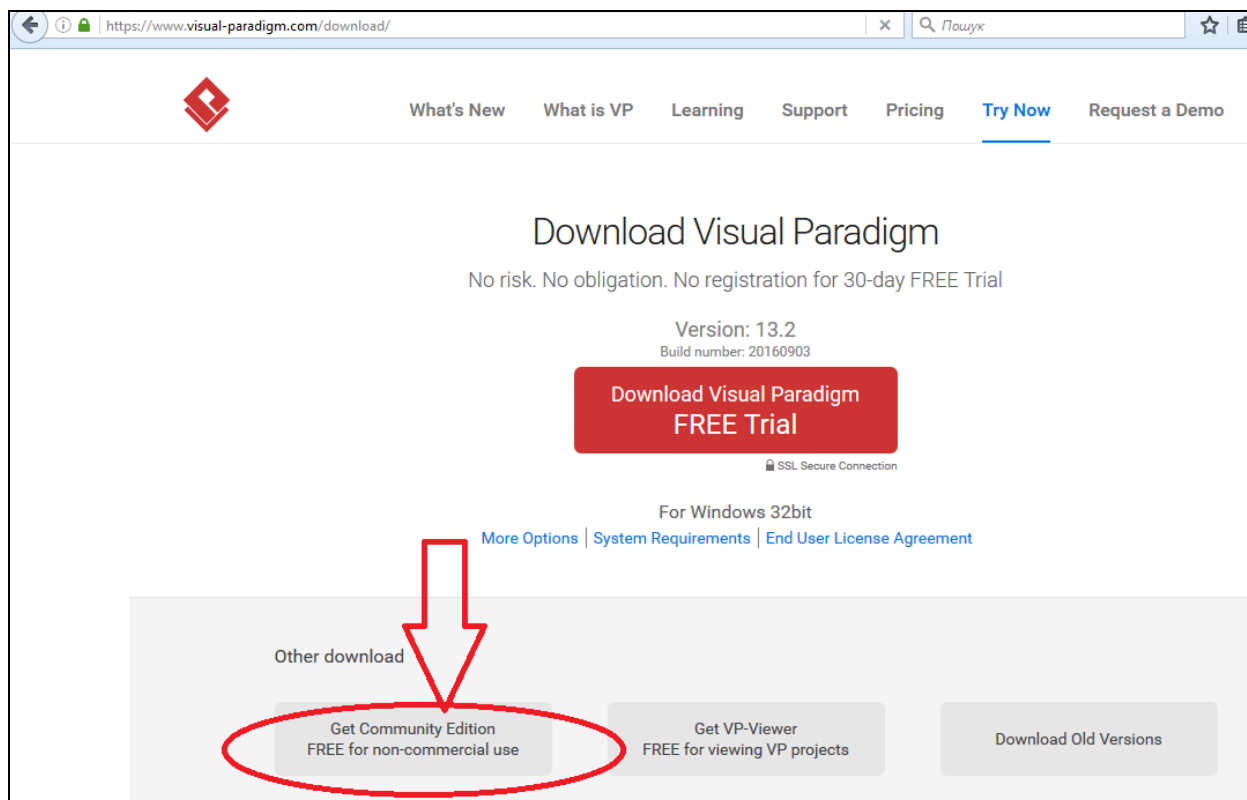
Figure 1.2 – Page with different versions of Visual Paradigm

Start Visual Paradigm. Create new project using menu: "File → New Project". Define the name of project. Then create activity diagram and set the name using menu: "File → New Diagram → New Activity Diagram"; or click the "Activity Diagram" menu in the Diagram Navigator and then select the "New Activity Diagram".

Activity diagram is used to describe dynamical properties of the system. For example, it is possible to use it to describe different algorithms that are necessary to implement in the system. The following nodes and edges are typically drawn on UML activity diagrams (Figure 1.3): activity, partition, action, object, control, activity edge.

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018
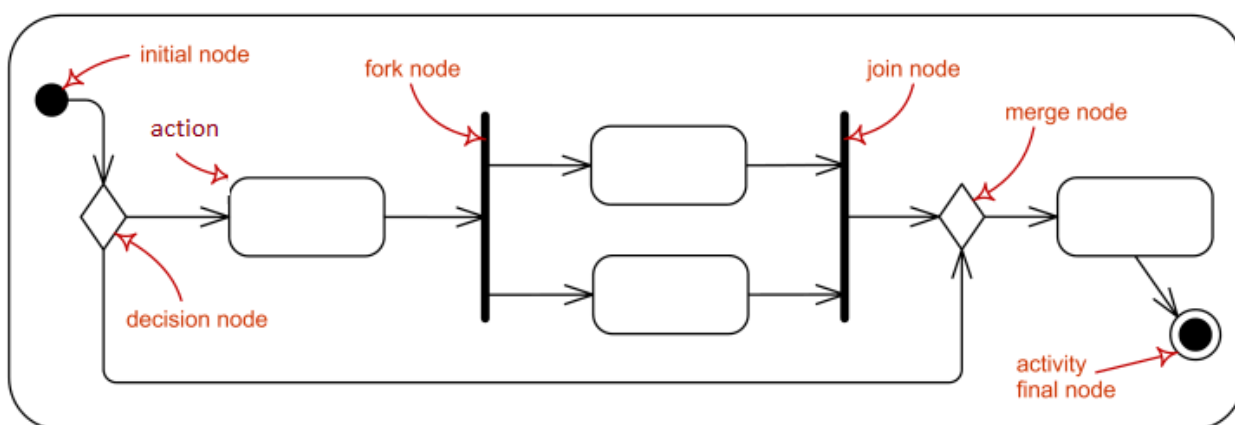
Figure 1.3 – Basic Activity Diagram elements

In our case the computational algorithm should look like shown on Figure 1.4.

There is possibility of changing font value in an activity diagram: right click an element or a diagram, then in "Styles and Formatting" menu student should choose "Formats" menu. User can set the diagram transparency the same way: "Styles and Formatting → Transparent".
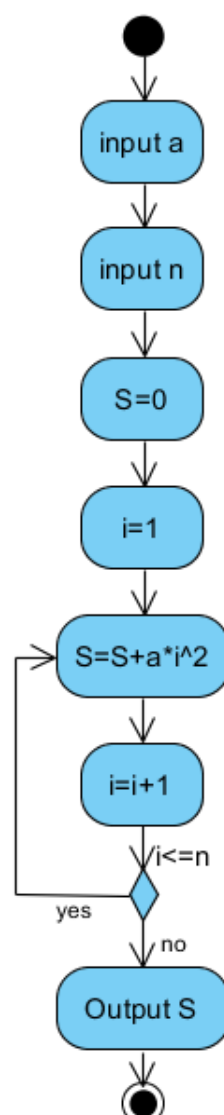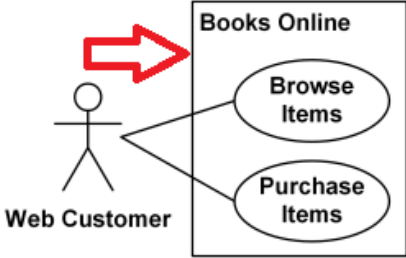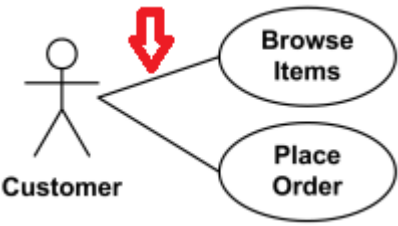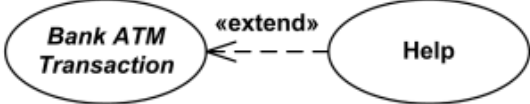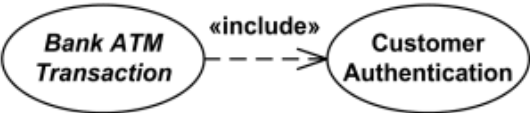

Figure 1.4 –Activity Diagram for chosen task

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018
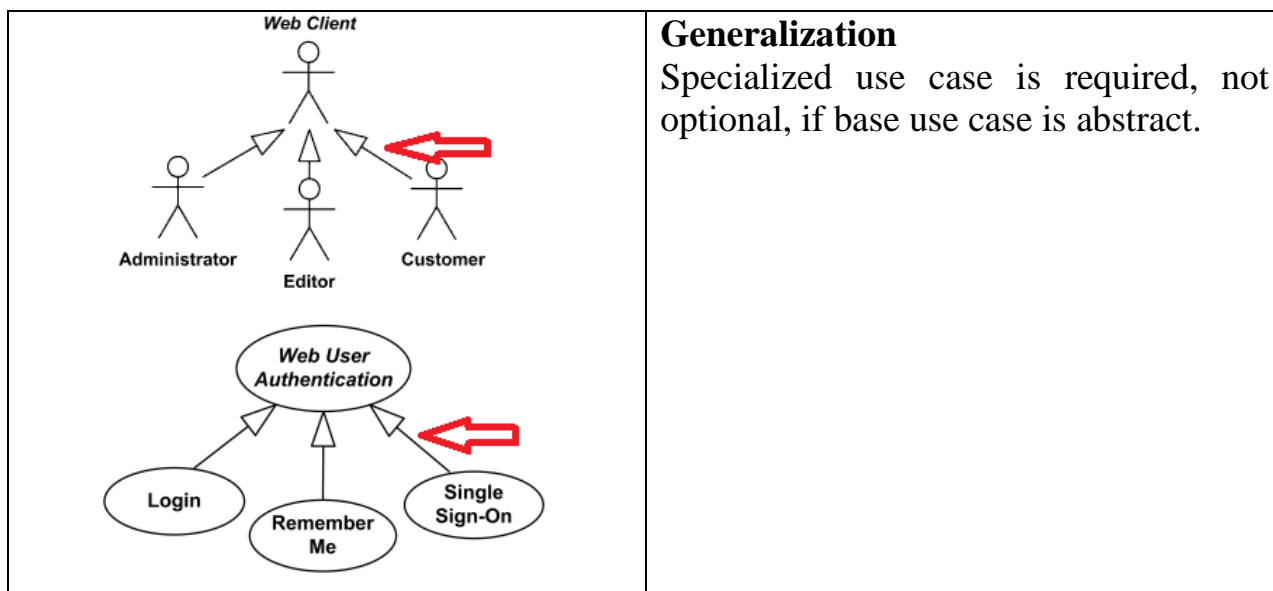
3. Develop the Use Case diagram.

Create Use Case diagram and set the name using: "File → New Diagram → New Use Case Diagram" menu; or click the "Use Case Diagram" menu in the Diagram Navigator and then select the "New Use Case Diagram".

Use case diagrams are used to describe a set of actions (use cases) that some system should or can perform in collaboration with one or more external users of the system (actors). The Use Case diagram reflects functional requirements of software. All main elements of Use Case diagram are described in table 2.

Table 2 – Main elements of Use Case Diagram

	Use cases describe functionality provided by systems, and determine the requirements the systems pose on their environment.
	An **actor** is a person, organization, or external system that plays a role in one or more interactions with your system.
	System boundary boxes (optional) indicates the scope of system.
	An **association** between an actor and a use case indicates that the actor and the use case somehow interact or communicate with each other. An actor could be associated to one or several use cases.
	Extend Extending use case is optional, supplementary.
	Include Included use case required, not optional.

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

| | **Generalization**
Specialized use case is required, not optional, if base use case is abstract. |

The free specialized software Visual Paradigm Community Edition is recommended to draw the Use Case diagram. The Use Case diagram for the given task is shown on Figure 1.5.
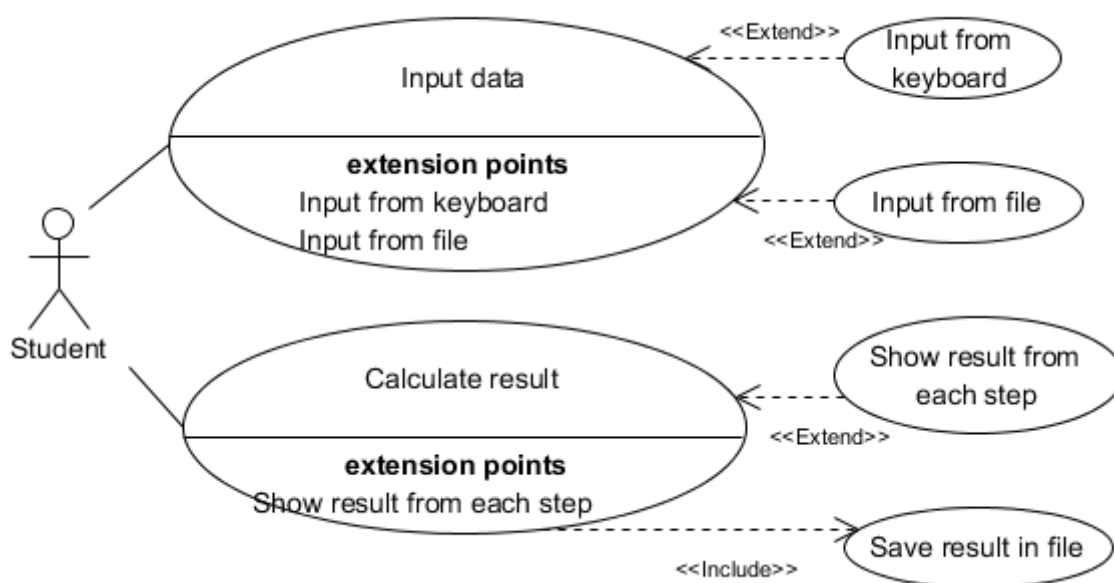

Figure 1.5 – Use Case Diagram for chosen task

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

4. Prepare the report of the work

Requirements for the report:

1. The report shall be drawn up according to the department standards. Template can be found by the following path:

   \\SELENA\Method\Paper work\STVUZ.dot

2. The report shall contain the results of implementation of all the tasks of the lab in accordance with the individual task.

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

Lab № 2

Coding of Domain

Goal: **Learning basic principles of C++**

Tasks:

1. Develop the code for the task from previous lab.

2. Prepare the report of the work

Progress of the lab.

1. Develop the code for the task from previous lab.

According to the task from the previous lab you should develop computational algorithm in Visual Studio.

For example, you obtained the following task:

The application has to calculate the values by the following formula for the range with a predefined step:

$$y = \begin{cases} \sum_{i=2}^{n-2} (x-i)^2, \ x < 7 \\ x+7, \qquad x \geq 7 \end{cases} \tag{2.1}$$

Solution.

First, to calculate the formula you should input the values of x and n. The following constraint: value of n should be greater than 4:

$$\sum_{i=2}^{n-2} (x-i)^2 = \sum_{i=2}^{4-2} (x-i)^2 = \sum_{i=2}^{2} (x-i)^2 = (x-i)^2$$

The following are some examples:

$$n = 5; x = 6 \Rightarrow y = \sum_{i=2}^{5-2} (x-i)^2 = \sum_{i=2}^{3} (x-i)^2 = (6-2)^2 + (6-3)^2 = 25$$

$$n = 5; x = 8 \Rightarrow y = x + 7 = 8 + 7 = 15$$

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

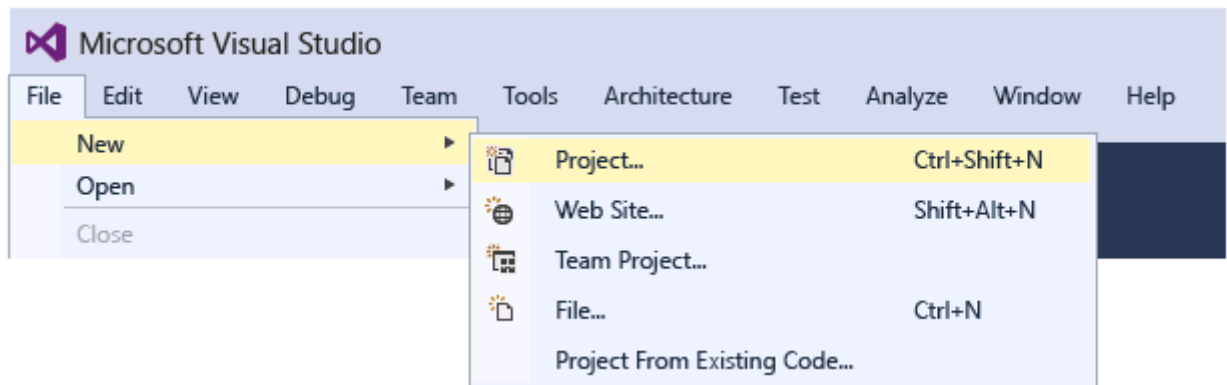Create a new Visual Studio C++ Project. On the menu bar, choose File→New→Project (Figure 2.1.).

Figure 2.1 – Process of creating a blank project

In the Visual C++ category, choose the Win32 Console Application template, and then name the project (Figure 2.2).
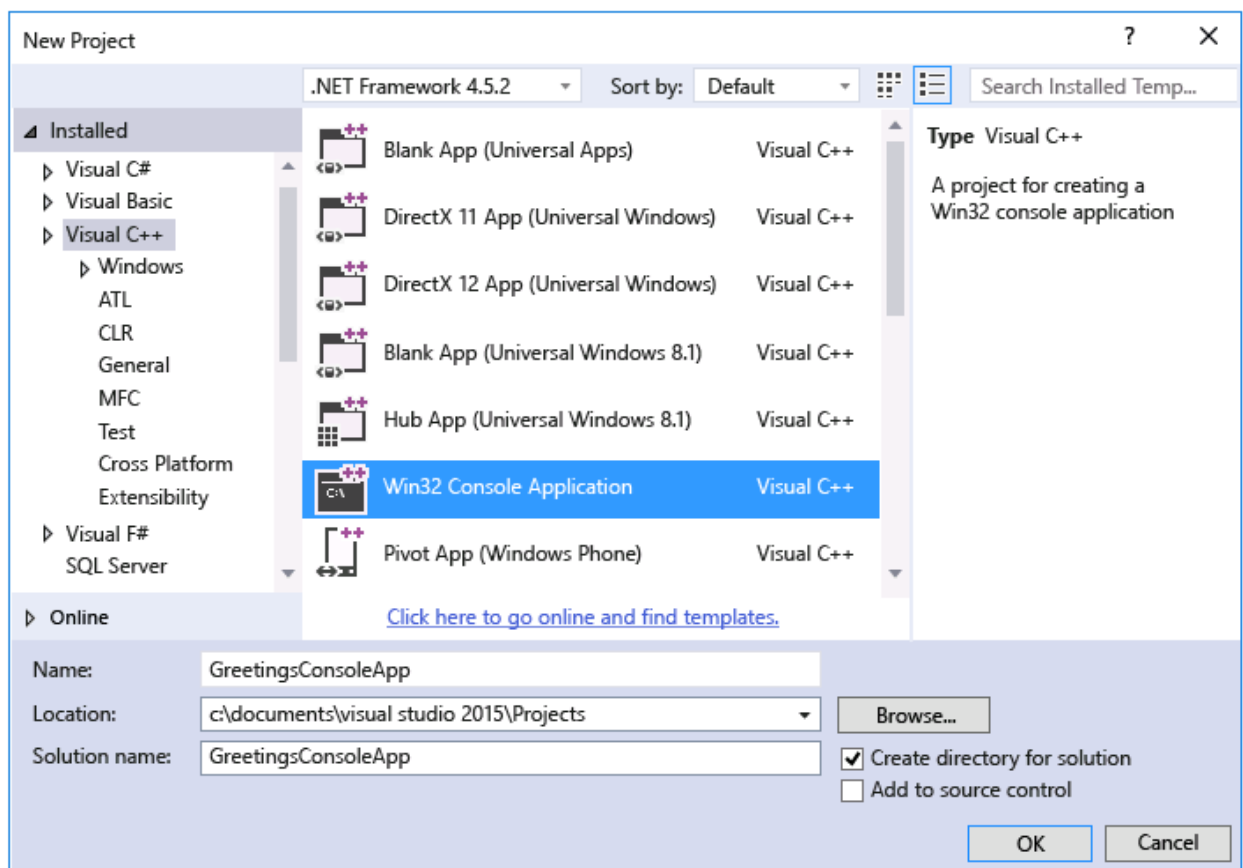

Figure 2.2 – Process of creating a blank project

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

When the Win32 Application Wizard appears, choose the Finish button. Implement the task from the previous lab.

The program for calculation of the function y is given on Figure 2.3. Samples calculated by the program can be found on Figure 2.4.

```cpp
1  #include <iostream>
2  #include <math.h>
3  using namespace std;
4
5  int main()
6  {
7
8  int x,n;
9  double y=0;
10
11     cout << "Input n>=4, n=";
12     cin >> n;
13     cout << "Input x= " ;
14     cin >> x;
15
16     if (x<7)
17     {
18         for (int i=2; i<=n-2; i++)
19         {
20             y+=pow((x-i),2);
21         }
22     }
23     else
24  {
25         y=x+7;
26  }
27     cout << "x= "<< x <<";  "<< "y= " << y <<endl;
28     return 0;
29  }
```

Figure 2.3 – Program for chosen task

```
Input n>=4, n=5
Input x= 6
x= 6;  y= 25
```
```
Input n>=4, n=5
Input x= 8
x= 8;  y= 15
```

Figure 2.4 – Result of execution the program

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

The next step is to modify the program. It should have the following possibilities:

- you should input predefined range (a, b) and step;

- program should calculate function y for each value x from range (a, b) (Figure 2.5);

- the results of each step should be displayed on the screen.
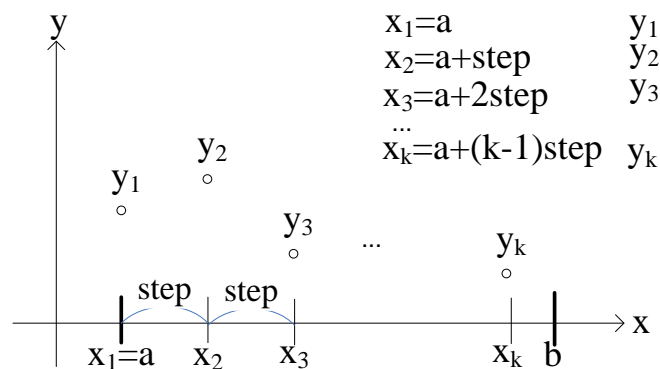
Figure 2.5 – Graphical representation of calculation

2. Prepare the report of the work

Requirements for the report:

1. The report shall be drawn up according to the department standards. Template can be found by the following path:

   \\SELENA\Method\Paper work\STVUZ.dot

2. The report shall contain the results of implementation of all the tasks of the lab in accordance with the individual task.

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

Lab № 3

Planning of system activities

Goal: **To obtain basic experience and knowledge to work with project tracking system**

Tasks:

1. Register on the free xp-dev.com account.

2. Create new project.

3. Add new user to project.

4. Create Iterations and user stories according to previous labs. Add task to implementation of user story. Assign tasks to users of you team.

5. Work with svn repository.

6. Prepare the report of the work.

Progress of the lab.

1. Register on the free xp-dev.com account

This task should execute only one person from team.

In order to register you should have an email account (it is also possible to provide a free email account). In order to register go to https://xp-dev.com/register and create new account (figure 3.1). The account provides you with 2Gb of storage for the repository and 30 days of trial use.

Figure 3.1 – Registering on xp-dev.com

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

2. Create new project

This task should execute only one person from team.

Choose "Create new project" and give a name, abbreviation and description for the project Issue, tracking Type should be XPDev (figure 3.2). It is necessary to create 1 xpdev account for 1 team. Create new project Inside XpDev.

Figure 3.2 – Creating of new project

3. Add new user to project.

Choose the username and password to send invitation (figure 3.3). New user can log in with the credentials sent to his/her email account. Choose the role of user (examine all roles first). The administrator or writer are acceptable.

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

Figure 3.3 – Adding second person to the team

4. Create Iterations and user stories according to previous labs. Add task to implementation of user story. Assign tasks to users of you team.

The modern software development approaches plan the project activities by stages (iteration). Every iteration should implement several software or system functionalities (user stories). Therefore, every laboratory start with new iteration that can have several user stories.

Go to Project tracking and choose "Create new iteration". Name: any of one that describe setup process. Choose dates for iteration (figure 3.4). Create a story for each iteration (figure 3.5). Create a set of tasks for each story (figure 3.6).

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

Figure 3.4 – Creating a new iteration

Figure 3.5 – Creating a new story

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

Figure 3.6 – Creating a new task

Choose any software development lifecycle model for your software. For example: you have chosen waterfall model, than iterations in your projects are stages from waterfall model (table 3.1).

Table 3.1 – Example of iterations

Iterations	User story	Tasks
1. Requirement Gathering	We should analyze given expressions in 1 lab. We should create requirements for software...	1. Analyze domain area
		2. Write software requirement specification
		3. Draw Use-case diagram
2. System Analysis	…	…

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

The result table with all iterations should be in report, example is shown on figure 3.7. There are many tables with set of tasks for each iteration in report (figure 3.8).

Figure 3.7 – Table with all iterations

Figure 3.8 – Result table for the first iteration

5. Work with svn repository.

Create a Repository at xp-dev.com : Choose Tab "Repository", then create a new repository. Set the name of repository. Set Create Initial Directories switch (it will creates additional directories to handle multiple version of project to handle customer requirements changes). Result of these actions is shown on figure 3.9.

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018
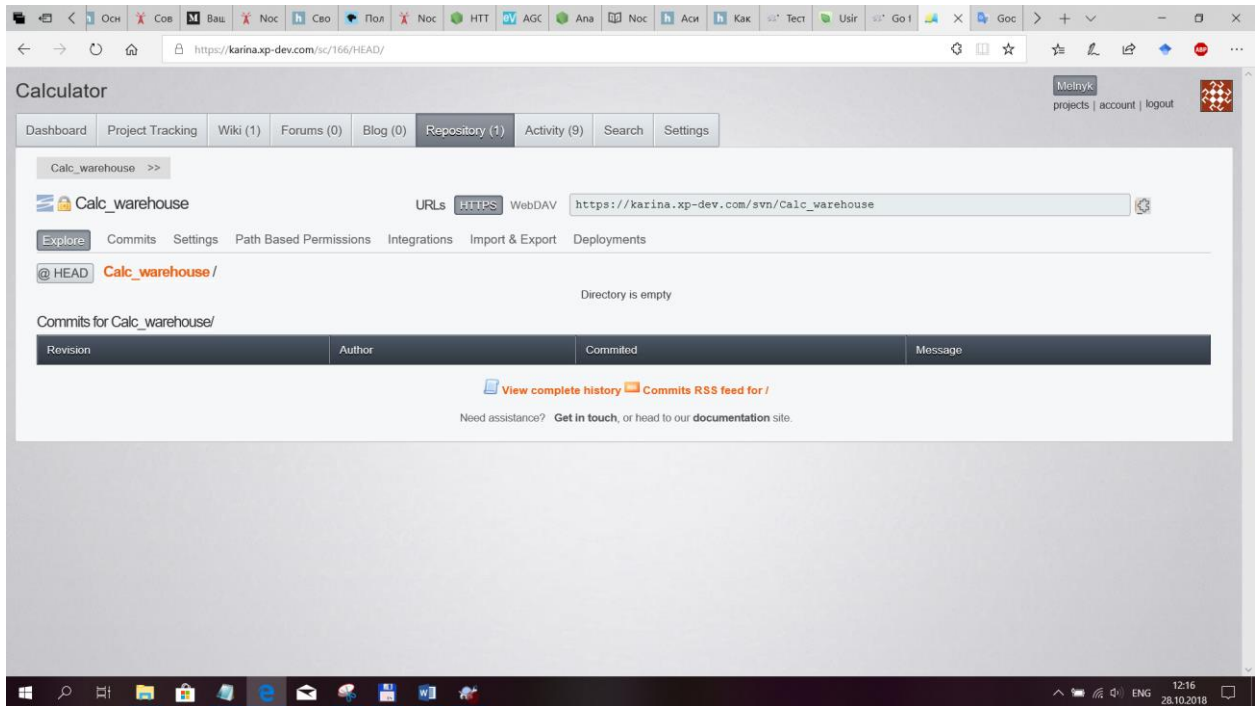
Figure 3.9 – A new repository on xp-dev.com

Open work directory on PC with *.cpp file from previous lab. Now right click on the file and from the context menu choose TortoiseSVN → Repo-browser (figure 3.10). After that type in link with your URL of repository from xp-dev.com (figure 3.11).
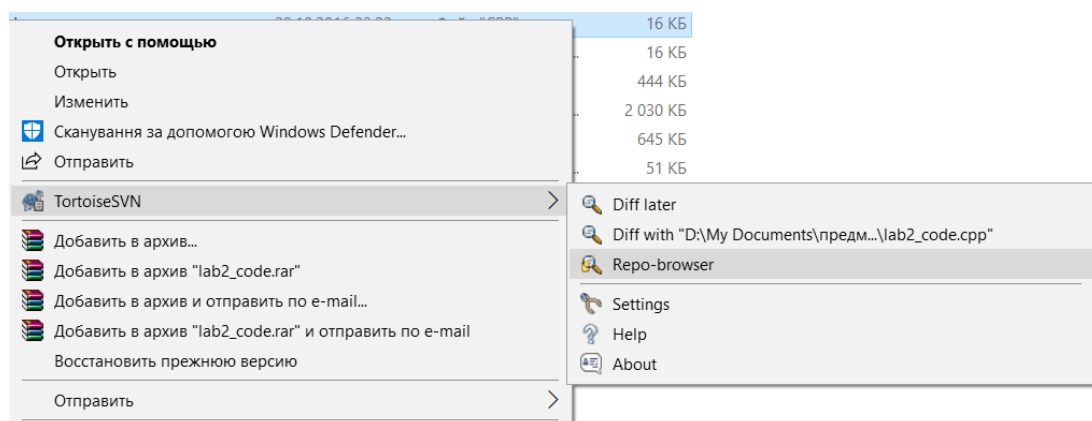
Figure 3.10 – A popup menu of TortoiseSVN

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018
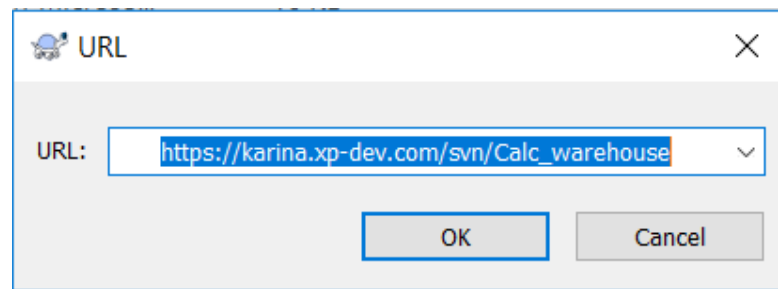
Figure 3.11 – URL of repository from xp-dev.com

You must add file to repository: right click on Tortoise window, then choose "Add file" (figure 3.12). Result of these actions is shown on figure 3.13.
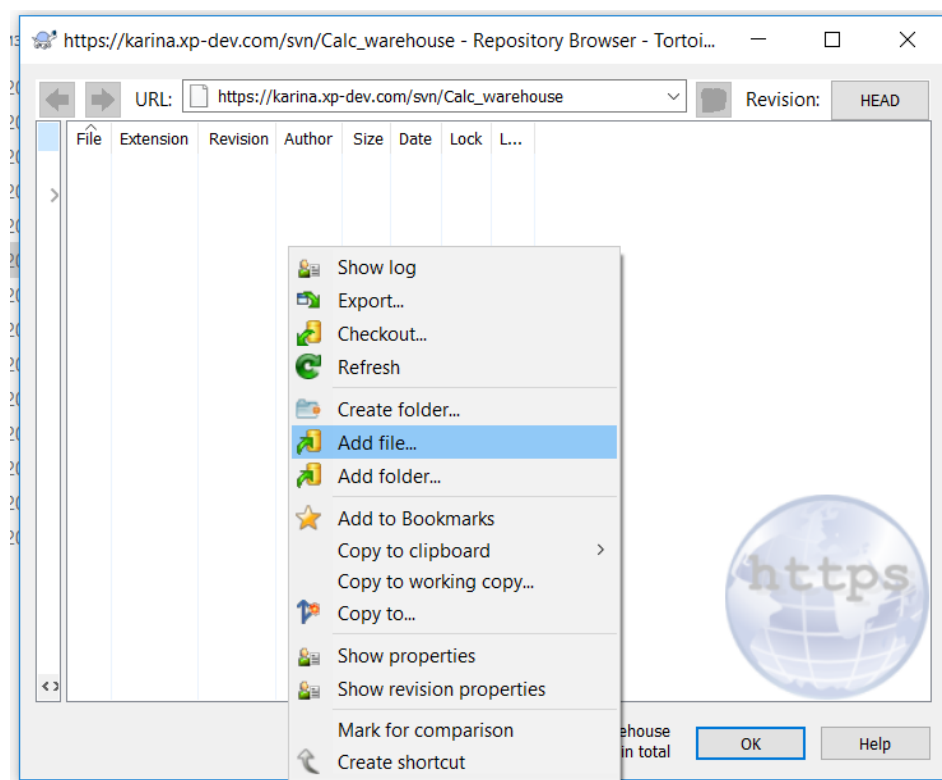
Figure 3.12 – Adding a new file to xp-dev.com repository

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018
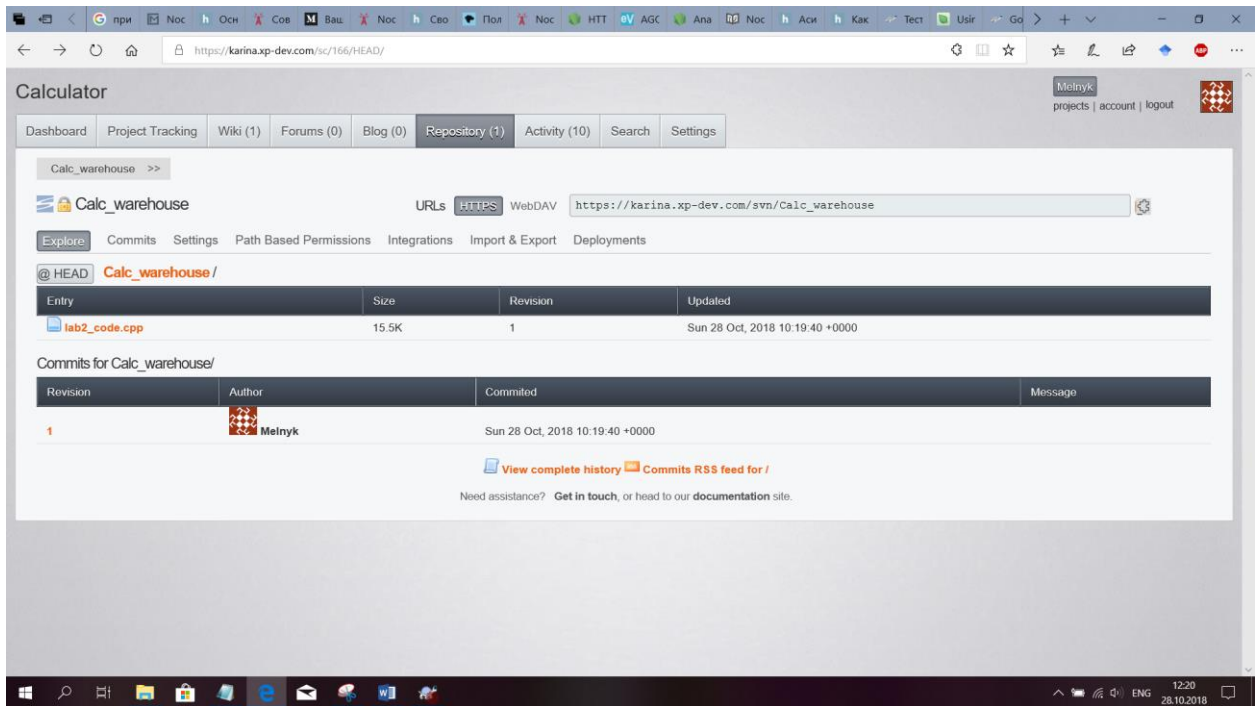
Figure 3.13 – File in repository

If the team member want to make some changes in some files, he/she should make "checkout" for necessary files (figure 3.14) and set the appropriate working directory (figure 3.15).

Figure 3.14 – Checkouting of file

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018
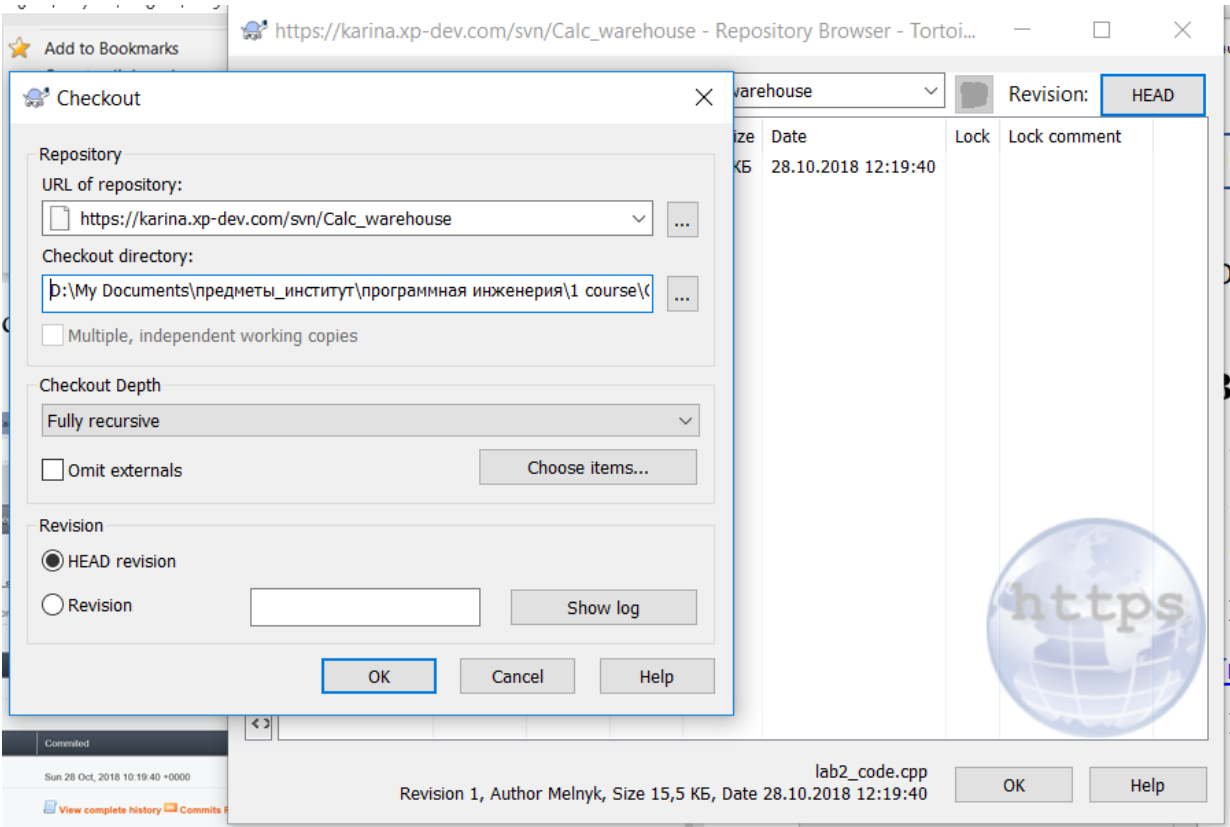
Figure 3.15 – Working directory for file

Result of these actions is shown on figure 3.16.
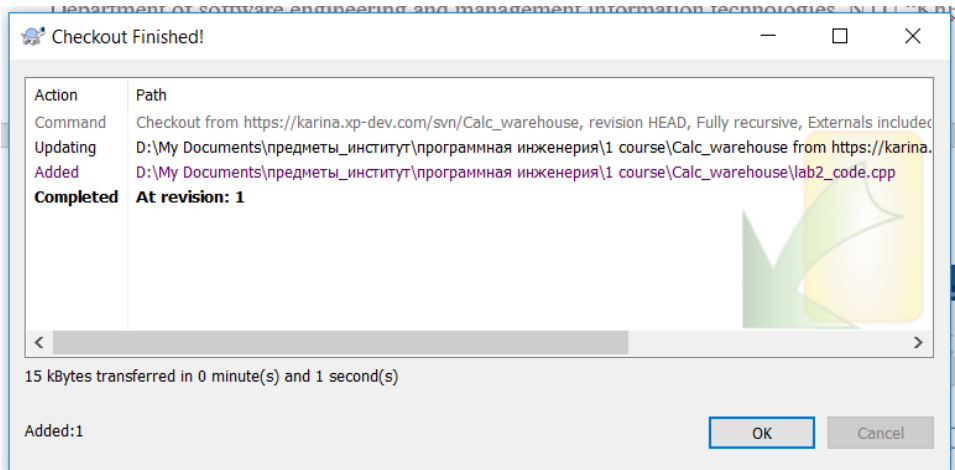
Figure 3.16 – Result of checkout process

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

When the file has been modified, the file should be committed to <u>xp-dev.com</u> repository. Press right click on file, choose "SVN Commit", check file, press Ok (figure 3.17).
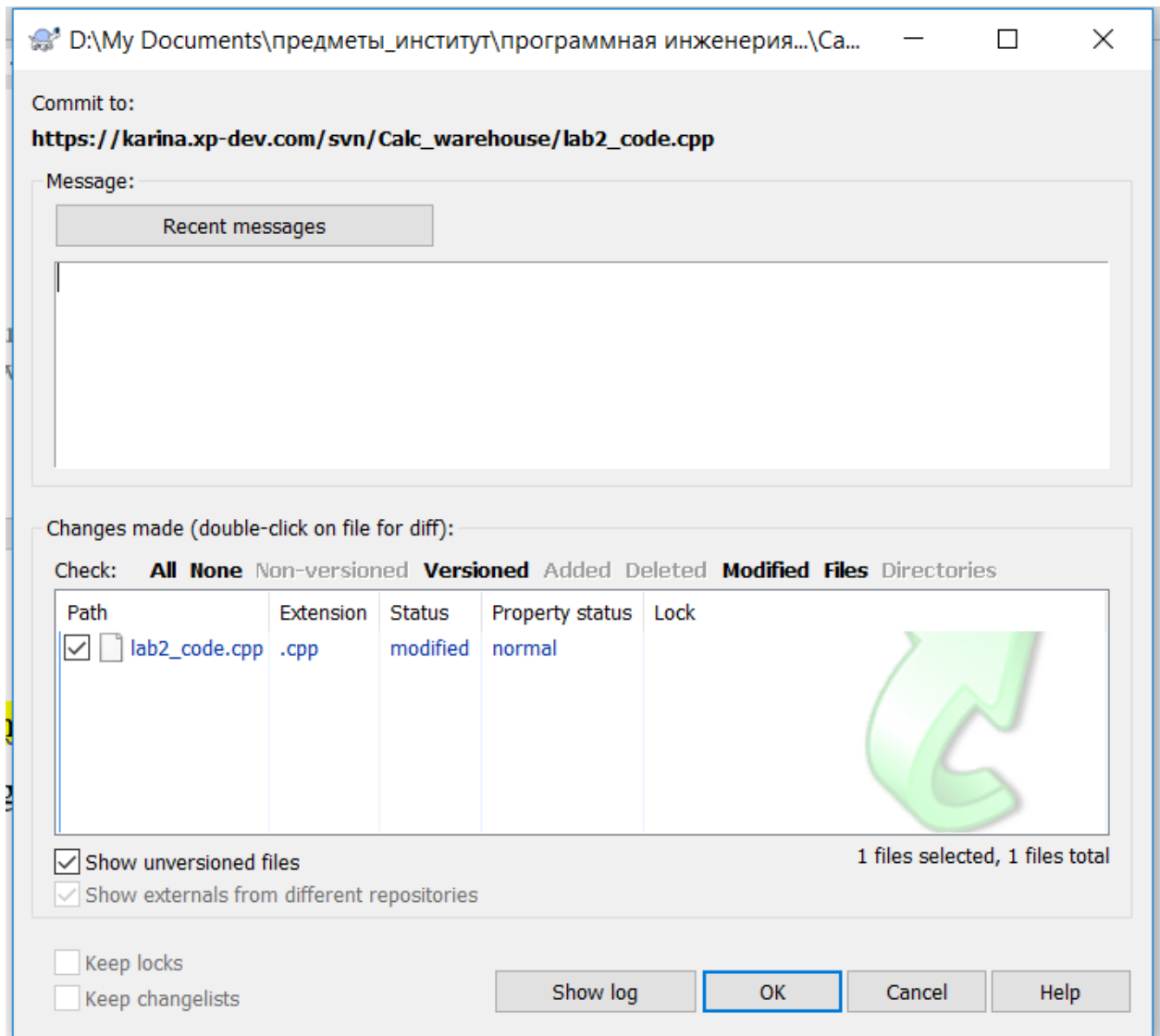
Figure 3.17 – Process of committed file

6. Prepare the report of the work

The report shall contain the results of implementation of all the tasks of the lab in accordance with the individual task.

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

Lab № 4

Creating of Software Requirement Specification

Goal: **Learning basic principles of creating Software Requirement Specification**

Tasks:

1. Choose some pattern for creating SRS.

2. Fill in the SRS according previous labs.

3. Make all necessary actions on xp-dev.com. Show the iteration where you made updating information.

4. Prepare the report of the work.

Progress of the lab.

1. Choose some pattern for creating SRS.

A Software Requirements Specification (SRS) is a document that describes software or application. This document is also known by the names SRS report, software document. A software document is primarily prepared for a software or any kind of application. There are a set of guidelines to be followed while preparing the software requirement specification document. This includes the purpose, scope, functional and nonfunctional requirements, software and hardware requirements of the project. In addition to this, it also contains the information about environmental conditions required, safety and security requirements, software quality attributes of the project etc.

There are many patterns for Software Requirement Specifications. You may use the following pattern of SRS:

1. Purpose
   a. Definitions
   b. System overview
   c. References
2. Overall description
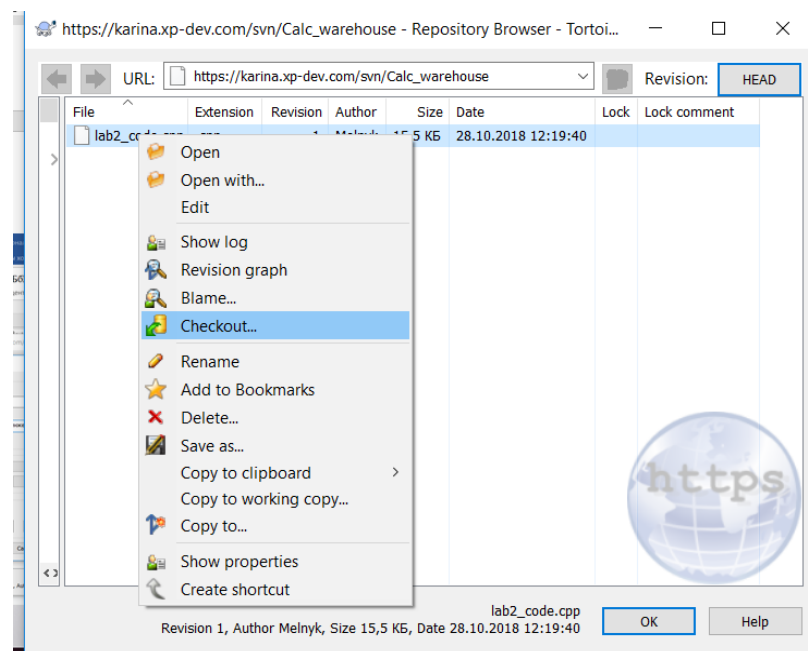
Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

    a. Product perspective

        i. System Interfaces

        ii. User interfaces

        iii. Hardware interfaces

        iv. Software interfaces

        v. Communication Interfaces

        vi. Memory Constraints

    b. Design constraints

        i. Operations

        ii. Site Adaptation Requirements

    c. Product functions

    d. User characteristics

    e. Constraints, assumptions and dependencies

3. Specific requirements

    a. External interface requirements

    b. Functional requirements

    c. Performance requirements

    d. Logical database requirement

    e. Software System attributes

        i. Reliability

        ii. Availability

        iii. Security

        iv. Maintainability

        v. Portability.

    f. Environment characteristics

        i. Hardware

        ii. peripherals

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

Another example of SRS pattern is shown on figure 4.1.

1. Introduction

1.1 Purpose

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

1.4 Project Scope

1.5 References

2. Overall Description

2.1 Product Perspective

2.2 Product Features

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

2.6 Assumptions and Dependencies

3. System Features

3.1 Functional Requirements

4. External Interface Requirements

4.1 User Interfaces

4.2 Hardware Interfaces

4.3 Software Interfaces

4.4 Communications Interfaces

5. Nonfunctional Requirements

5.1 Performance Requirements

5.2 Safety Requirements

5.3 Security Requirements

5.4 Software Quality Attributes

Figure 4.1 – SRS pattern

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

2. Fill in the SRS according previous labs.

Your task is creating software for calculating some system of expressions. Therefore, you should fill in chosen pattern according to previous labs.

3. Make all necessary actions on xp-dev.com. Show the iteration where you made updating information

In this lab we have created SRS, so we can set task "Write Software Requirements Specification" as completed. This action is shown on figure 4.2.

Tasks						
Id	Description	Hours			Hours Added	Assigned To	Completed
		Estimated	Added	Remaining			
#2	Analyze domain area	0.5	0	0	Add hours	Melnyk	100 %
#3	Write software requirement specification	2	0	0	Add hours	Melnyk	100 %
#4	Draw Use-case diagram	1	0	0	Add hours	Melnyk	100 %

Figure 4.2 – Completed task

4. Prepare the report of the work

Make a report with all actions according to the tasks from this lab.

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

Lab № 5

Testing of code

Goal: **Learning basic principles of testing C++ code**

Tasks:

1. Study principles of using functions in C++.
2. Study Exception Handling in C++.
3. Modify the code from lab 2 according to 1 and 2 tasks.
4. Implement unit testing for developed program.
5. Make all necessary actions on xp-dev.com. Show the iteration where you made updating information.
6. Prepare the report of the work

Progress of the lab.

1. Study principles of using functions in C++.

You should modify the code from the previous lab by using functions in C++.

A function is a group of statements that is given a name, and which can be called from some point of the program. Functions allow to structure programs in segments of code to perform individual tasks. The main advantage is a function can actually be called multiple times within a program.

Depending on whether a function is predefined or created by programmer; there are two types of function:

1. **Library Function**. Programmer can use library function by invoking function directly; they don't need to write it themselves.
2. **User-defined Function.** A user-defined function groups code to perform a specific task and that group of code is given a name (identifier).

The most common syntax to define a function is:

```
type name ( parameter1, parameter2, ...) { statements }
```

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

where:

- `type` is the type of the value returned by the function; if no value is returned to the calling function then, `void` should be used;

- `name` is the identifier by which the function can be called;

- `parameters` (as many as needed): The purpose of parameters is to allow passing arguments to the function from the location where it is called from;

- `statements` is the function's body.

2. Study Exception Handling in C++.

An exception is an error condition that prevents the program from continuing along its regular execution path. Certain operations, including object creation, file input/output, and function calls made from other modules, are all potential sources of exceptions even when your program is running correctly. Robust code anticipates and handles exceptions. The C++ language provides built-in support for throwing and catching exceptions. To implement exception handling in C++, you use `try`, `throw`, and `catch` expressions.

```
try
{
    // ...

    throw 1;

}
catch( ... )
{
    // Handle exception here.
}
```

Use a `try` block to enclose one or more statements that might throw an exception. A `throw` expression signals that an exceptional condition – often, an error – has occurred in a `try` block. The `throw` statement behaves much like `return`. You can use an object of any type as the operand of a `throw` expression. Typically, this object is used to communicate information about the error. To handle exceptions that may be thrown, implement one or more `catch` blocks immediately

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

following a `try` block. Each `catch` block specifies the type of exception it can handle.

3. Modify the code from lab 2 according to 1 and 2 tasks.

The function "Calculate" shows how to calculate value on definite range of expression (2.1) from lab 2 (Figure 5.1). This expression has one specific condition: the value of n should be greater than 4. This constraint was developed in function "checkValidParams" (Figure 5.1).

If user enter incorrect value for n and x, for example, character or double value, then program should generate exception. These cases are reflected in the function "checkValidInput".

```cpp
#include "stdafx.h"
#include <iostream>
#include <math.h>
using namespace std;

void checkValidInput()
{
    if (cin.fail())
    {
        throw "Incorrect input";
    }
}

void checkValidParams(int n)
{
    if (n < 4)
    {
        throw "Input correct data";
    }
}

int calculate(int n, int x)
{
    int y = 0;

    if (x < 7)
    {
        for (int i = 2; i <= n - 2; i++)
        {
            y += pow((x - i), 2);
        }
    }
    else
    {
        y = x + 7;
    }

    return y;
}
```

Figure 5.1 – Part of the code

The main function of developed code is on the Figure 5.2.

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

```
41    □int main()
42     {
43         int x, n;
44
45    □    try
46         {
47             cout << "Input n>=4, n=";
48             cin >> n;
49             checkValidInput();
50             checkValidParams(n);
51
52             cout << "Input x=";
53             cin >> x;
54             checkValidInput();
55
56             cout << "x= " << x << ";   " << "y= " << calculate(n, x) << endl;
57         }
58    □    catch (const char* ex)
59         {
60             cout << ex << endl;
61             return -1;
62         }
63    □    catch (...)
64         {
65             cout << "Unknown error" << endl;
66             return -2;
67         }
68     }
```

Figure 5.2 – The "Main" function

4. Implement unit testing for developed program

By this moment you should have been developed a Win32 Console Application with all necessary functions for solving the equation. To test these functions, you have to create unit tests for each function separately. Sometimes it is necessary to create several unit tests for one function. It is done with the aim to fully test the behavior of each function with all possible input parameters.

To create unit tests for the existing project, select File → Add → New Project (Figure 5.3). Select Native Unit Test Project (Figure 5.4).
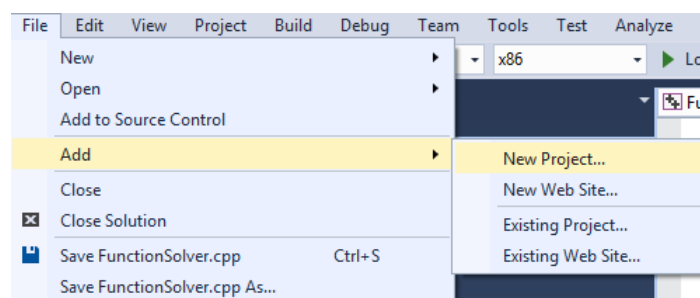
Figure 5.3 – The process of creating of unit test project

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
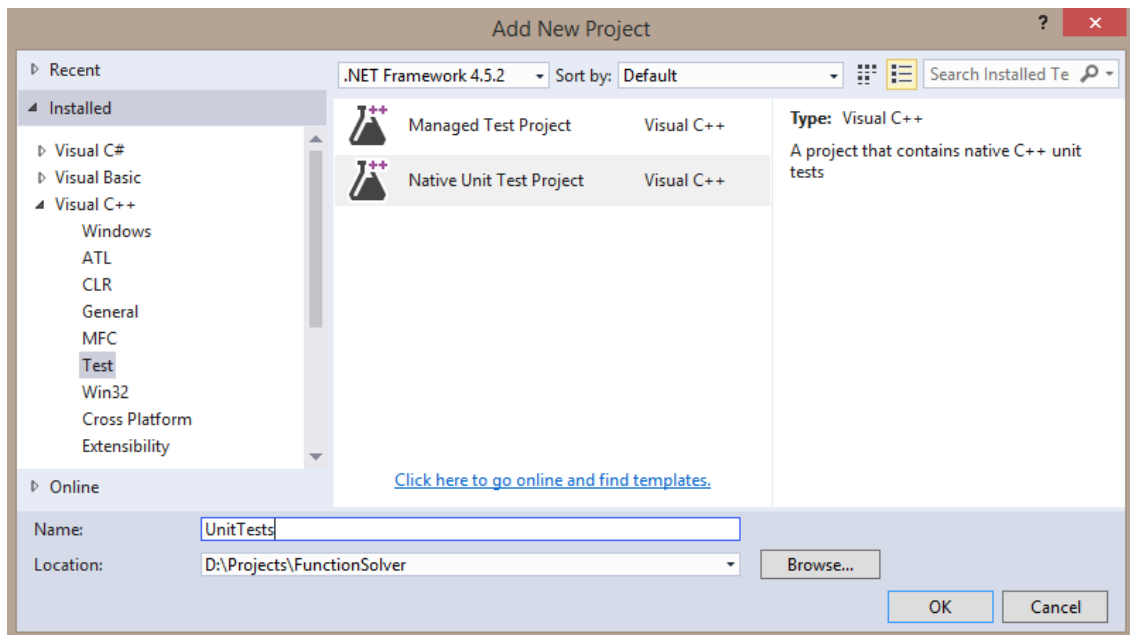Kharkiv, 2018

Figure 5.4 – The process of creating of unit test project

You can see unit test project in the Solution Explorer (Figure 5.5).
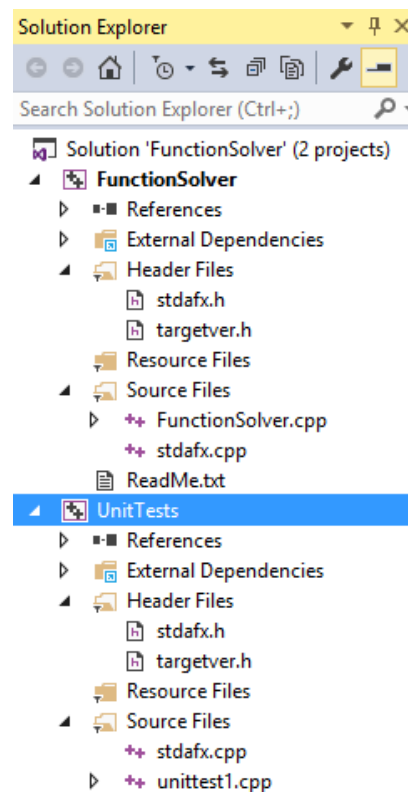
Figure 5.5 – Solution Explorer

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

Open the file unittest1.cpp. Unit tested will be placed in it. Right click on your source cpp file and choose "Copy Path" menu item (Figure 5.6). Paste this path to the file that will contain your unit tests (Figure 5.7).
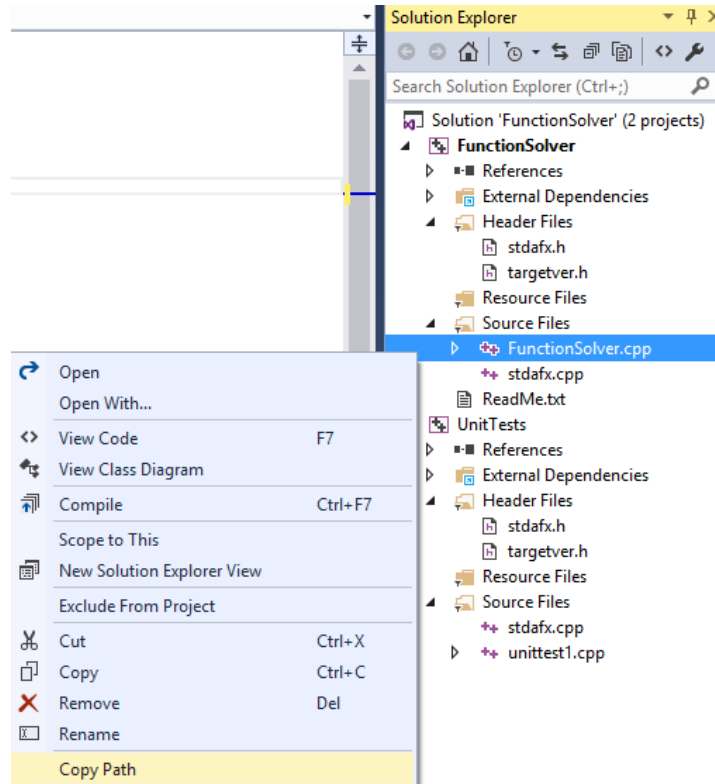
Figure 5.6 – Using source cpp file in test project (1)

Figure 5.7 – Using source cpp file in test project (2)

Create tests for your code. Each test is defined by using `TEST_METHOD(YourTestName){...}`.

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

You do not have to write a conventional function signature. The signature is created by the macro TEST_METHOD. The macro generates an instance function that returns void. It also generates a static function that returns information about the test method. This information allows the test explorer to find the method.

Test methods are grouped into classes by using `TEST_CLASS(YourClassName){...}.`

When the tests are run, an instance of each test class is created. Create the test methods names in the following manner:

`testedFunction_functionArguments_expectedResult.`

For testing purposes use several different methods on your choice from the Assert class. You can read about this class and its methods on MSDN https://msdn.microsoft.com/en-us/library/microsoft.visualstudio.testtools.unittesting.assert.aspx.

The example of tests is shown on Figures 5.8-5.9.

```cpp
#include "stdafx.h"
#include "CppUnitTest.h"
#include "D:\Projects\FunctionSolver\FunctionSolver\FunctionSolver.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTests
{
    TEST_CLASS(calculate_Tests)
    {
    public:
        TEST_METHOD(calculate_get4and7_14returned)
        {
            int n = 4;
            int x = 7;
            int expected = 14;

            int actual = calculate(n, x);

            Assert::AreEqual(expected, actual);
        }

    public:
        TEST_METHOD(calculate_get15and6_170returned)
        {
            int n = 15;
            int x = 6;
            int expected = 170;

            int actual = calculate(n, x);

            Assert::AreEqual(expected, actual);
        }
    };
```

Figure 5.8 – Unit tests for Calculate function

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

```
36    TEST_CLASS(checkValidParams_Tests)
37    {
38    public:
39        TEST_METHOD(checkValidParams_get10_exceptionNotThrown)
40        {
41            int n = 10;
42
43            try
44            {
45                checkValidParams(n);
46                Assert::IsTrue(true);
47            }
48            catch (...)
49            {
50                Assert::Fail();
51            }
52        }
53
54    public:
55        TEST_METHOD(checkValidParams_get3_exceptionThrown)
56        {
57            int n = 3;
58
59            try
60            {
61                checkValidParams(n);
62                Assert::Fail();
63            }
64            catch (...)
65            {
66                Assert::IsTrue(true);
67            }
68        }
69    };
70    }
```

Figure 5.9 – Unit tests for checkValidParams function

When test methods are developed, you can execute them. To do that, you have to build solution (Build → Build Solution / Ctrl + Shift + B), open Test Explorer if it wasn't opened automatically (Test → Windows → Test Explorer) and run all tests. If everything go smoothly, you will see that all tests passed in the summary sub-window (Figure 5.10).
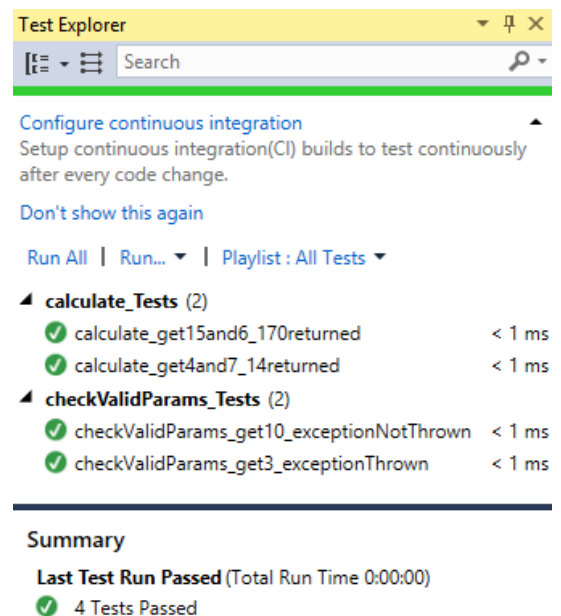

Figure 5.10 – Successful unit tests

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018

Also you should make some changes in unit tests to get them unpassed. For example, change the expected result to the wrong value and run the test again (Figure 5.11 a). In the Test Explorer you will see which tests weren't passed (Figure 5.11 b).


```
TEST_METHOD(calculate_get4and7_14returned)
{
    int n = 4;
    int x = 7;
    int expected = 100000000;

    int actual = calculate(n, x);

    Assert::AreEqual(expected, actual);
}
```
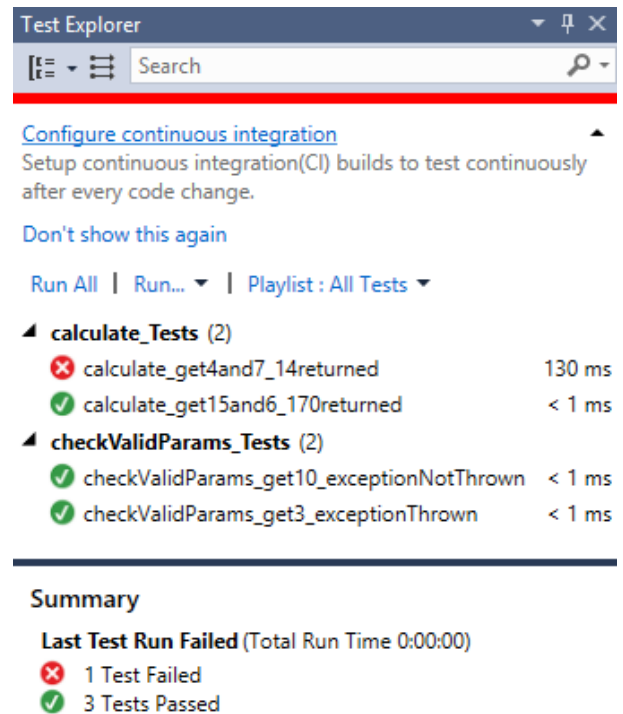
(a)   (b)

Figure 5.11 – Unit test with error

5. Make all necessary actions on xp-dev.com. Show the iteration where you made updating information.

6. Prepare the report of the work

Make a report with all actions according to the tasks from this lab.

Author: Melnyk Karina, associate professor
Department of software engineering and management information technologies, NTU "KhPI"
Kharkiv, 2018