

Compulsory 2

Generated by Doxygen 1.8.18

1 Programming I: Compulsory 2	1
1.1 Introduction	1
1.2 Instructions	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Polynomial Class Reference	7
4.1.1 Constructor & Destructor Documentation	7
4.1.1.1 Polynomial() [1/2]	8
4.1.1.2 Polynomial() [2/2]	8
4.1.2 Member Function Documentation	8
4.1.2.1 Derivate()	8
4.1.2.2 operator*()	8
4.1.2.3 operator+()	8
4.1.2.4 operator-()	9
4.1.2.5 Print()	9
4.1.3 Member Data Documentation	9
4.1.3.1 coefficients	9
4.1.3.2 constant	9
4.1.3.3 isSolved	9
4.1.3.4 isTwoSolutions	9
4.1.3.5 solutions	10
4.2 Principia Class Reference	10
4.2.1 Constructor & Destructor Documentation	10
4.2.1.1 Principia() [1/2]	10
4.2.1.2 Principia() [2/2]	10
5 File Documentation	11
5.1 CMakeLists.txt File Reference	11
5.2 inputfunctions.cpp File Reference	11
5.2.1 Function Documentation	11
5.2.1.1 GetDouble()	11
5.2.1.2 GetInteger()	12
5.2.1.3 GetString()	12
5.2.1.4 GetUnsignedInteger()	12
5.3 inputfunctions.h File Reference	12
5.3.1 Function Documentation	12
5.3.1.1 GetDouble()	12
5.3.1.2 GetInteger()	13

5.3.1.3 GetString()	13
5.3.1.4 GetUnsignedInteger()	13
5.4 main.cpp File Reference	13
5.4.1 Function Documentation	13
5.4.1.1 Calc()	14
5.4.1.2 Factorial()	14
5.4.1.3 main()	14
5.4.1.4 SolvePolynomial()	14
5.5 main.h File Reference	14
5.5.1 Function Documentation	15
5.5.1.1 Calc()	15
5.5.1.2 Factorial()	15
5.5.1.3 SolvePolynomial()	15
5.5.2 Variable Documentation	15
5.5.2.1 princ	15
5.6 polynomial.cpp File Reference	15
5.7 polynomial.h File Reference	16
5.7.1 Variable Documentation	16
5.7.1.1 MAX_COEFFCIANTS	16
5.8 principia.cpp File Reference	16
5.9 principia.h File Reference	16
5.10 README.md File Reference	16
Index	17

Chapter 1

Programming I: Compulsory 2

1.1 Introduction

For this Compulsory, you will have to create a program using a Procedural Programming Paradigm approach to implement a calculator that will perform the following operations:

- Factorial (use recursion)
- Solving 3rd degree polynomial equations (addition, subtraction, multiplication)
- Simple math equations (addition, subtraction, multiplication and division).

1.2 Instructions

In order to do this, create a menu system that will allow the user to choose between these three options, as well as offer a choice to terminate the program. Use functions to keep the code as clear and modular as possible. Control for unexpected inputs (strings where numbers are expected). Use XML documentation practices for all defined functions. The submission will be a PDF with Surname_Name_Compulsory2 and have the following:

Link to GitHub repository (public!) with all the project files and code Brief paragraph discussing the benefits of using recursion in certain scenarios versus an iterative approach. How you feel you have progressed in programming since the start of the semester.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Polynomial	7
Principia	10

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

inputfunctions.cpp	11
inputfunctions.h	12
main.cpp	13
main.h	14
polynomial.cpp	15
polynomial.h	16
principia.cpp	16
principia.h	16

Chapter 4

Class Documentation

4.1 Polynomial Class Reference

```
#include <polynomial.h>
```

Public Member Functions

- void [Derivate](#) ()
Will print out the derivation of the polynomial.
- void [Print](#) ()
Prints out some of the attributes of the polynomial.
- [Polynomial](#) ()
Default constructor: Creates a polynomial with all attributes set to 0.
- [Polynomial](#) (double third, double second, double first, double c)
Constructor: Creates a polynomial where you can decide on some attributes.
- [Polynomial operator+](#) (const [Polynomial](#) &rhs)
- [Polynomial operator-](#) (const [Polynomial](#) &rhs)
- [Polynomial operator*](#) (const [Polynomial](#) &rhs)

Public Attributes

- double [coefficients](#) [[MAX_COEFFCIANTS](#)]
- double * [constant](#)
- double [solutions](#) [2]
- bool [isTwoSolutions](#)
- bool [isSolved](#)

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Polynomial() [1/2]

```
Polynomial::Polynomial ( )
```

Default constructor: Creates a polynomial with all attributes set to 0.

4.1.1.2 Polynomial() [2/2]

```
Polynomial::Polynomial (
    double third,
    double second,
    double first,
    double c )
```

Constructor: Creates a polynomial where you can decide on some attributes.

4.1.2 Member Function Documentation

4.1.2.1 Derivate()

```
void Polynomial::Derivate ( )
```

Will print out the derivation of the polynomial.

4.1.2.2 operator*()

```
Polynomial Polynomial::operator* (
    const Polynomial & rhs ) [inline]
```

4.1.2.3 operator+()

```
Polynomial Polynomial::operator+ (
    const Polynomial & rhs ) [inline]
```

- operator overload. Add to polynomials

4.1.2.4 operator-()

```
Polynomial Polynomial::operator- (
    const Polynomial & rhs ) [inline]
```

- operator overload. Subtract to polynomials

4.1.2.5 Print()

```
void Polynomial::Print ( )
```

Prints out some of the attributes of the polynomial.

4.1.3 Member Data Documentation

4.1.3.1 coefficients

```
double Polynomial::coefficients[MAX_COEFFCIANTS]
```

4.1.3.2 constant

```
double* Polynomial::constant
```

4.1.3.3 isSolved

```
bool Polynomial::isSolved
```

4.1.3.4 isTwoSolutions

```
bool Polynomial::isTwoSolutions
```

4.1.3.5 solutions

```
double Polynomial::solutions[2]
```

The documentation for this class was generated from the following files:

- [polynomial.h](#)
- [polynomial.cpp](#)

4.2 Principia Class Reference

```
#include <principia.h>
```

Public Member Functions

- [Principia](#) ()
- [Principia](#) (std::string s)

4.2.1 Constructor & Destructor Documentation

4.2.1.1 Principia() [1/2]

```
Principia::Principia ( )
```

4.2.1.2 Principia() [2/2]

```
Principia::Principia (
    std::string s )
```

The documentation for this class was generated from the following files:

- [principia.h](#)
- [principia.cpp](#)

Chapter 5

File Documentation

5.1 CMakeLists.txt File Reference

5.2 inputfunctions.cpp File Reference

```
#include "inputfunctions.h"
```

Functions

- int [GetInteger](#) (string message)
Returns an integer, will ask user for an integer until a whole number is typed.
- unsigned int [GetUnsignedInteger](#) (string message)
Returns an integer, will ask user for an integer until a whole positive number is typed.
- double [GetDouble](#) (string message)
- string [GetString](#) (string message)

5.2.1 Function Documentation

5.2.1.1 GetDouble()

```
double GetDouble (  
    string message )
```

Returns an double (high precision float), will ask user for an decimal number until a decimal number is typed NOTE: Decimal numbers are like this: 1.1 2.3 1.5 etc. (with a .) Not: 1,1 2,3 1,5 etc. (C++ uses the American system)

5.2.1.2 GetInteger()

```
int GetInteger (
    string message )
```

Returns an integer, will ask user for an integer until a whole number is typed.

5.2.1.3 GetString()

```
string GetString (
    string message )
```

Returns a string (letters), will ask user for letter characters until they have been typed

5.2.1.4 GetUnsignedInteger()

```
unsigned int GetUnsignedInteger (
    string message )
```

Returns an integer, will ask user for an integer until a whole positive number is typed.

5.3 inputfunctions.h File Reference

```
#include <iostream>
#include <string>
```

Functions

- int [GetInteger](#) (string message="")
Returns an integer, will ask user for an integer until a whole number is typed.
- double [GetDouble](#) (string message="")
- string [GetString](#) (string message="")
- unsigned int [GetUnsignedInteger](#) (string message)
Returns an integer, will ask user for an integer until a whole positive number is typed.

5.3.1 Function Documentation

5.3.1.1 GetDouble()

```
double GetDouble (
    string message )
```

Returns an double (high precision float), will ask user for an decimal number until a decimal number is typed NOTE: Decimal numbers are like this: 1.1 2.3 1.5 etc. (with a .) Not: 1,1 2,3 1,5 etc. (C++ uses the American system)

5.3.1.2 GetInteger()

```
int GetInteger (
    string message = "" )
```

Returns an integer, will ask user for an integer until a whole number is typed.

5.3.1.3 GetString()

```
string GetString (
    string message )
```

Returns a string (letters), will ask user for letter characters until they have been typed

5.3.1.4 GetUnsignedInteger()

```
unsigned int GetUnsignedInteger (
    string message )
```

Returns an integer, will ask user for an integer until a whole positive number is typed.

5.4 main.cpp File Reference

```
#include "main.h"
#include "inputfunctions.h"
#include "polynomial.h"
```

Functions

- int [main](#) ()
The main entry point of the program.
- unsigned long long int [Factorial](#) (unsigned int n)
Calculate factorial (n!) recursively.
- void [SolvePolynomial](#) ()
- void [Calc](#) ()
Do basic calculations.

5.4.1 Function Documentation

5.4.1.1 Calc()

```
void Calc ( )
```

Do basic calculations.

5.4.1.2 Factorial()

```
unsigned long long int Factorial (
    unsigned int n )
```

Calculate factorial (n!) recursively.

5.4.1.3 main()

```
int main ( )
```

The main entry point of the program.

5.4.1.4 SolvePolynomial()

```
void SolvePolynomial ( )
```

Function which allows user to put in two polynomials and do addition, subtraction, multiplication and division

5.5 main.h File Reference

```
#include <iostream>
#include <string>
#include "principia.h"
```

Functions

- unsigned long long [Factorial](#) (unsigned int n)
Calculate factorial (n!) recursively.
- void [Calc](#) ()
Do basic calculations.
- void [SolvePolynomial](#) ()

Variables

- [Principia princ](#)

5.5.1 Function Documentation

5.5.1.1 Calc()

```
void Calc ( )
```

Do basic calculations.

5.5.1.2 Factorial()

```
unsigned long long Factorial (
    unsigned int n )
```

Calculate factorial (n!) recursively.

5.5.1.3 SolvePolynomial()

```
void SolvePolynomial ( )
```

Function which allows user to put in two polynomials and do addition, subtraction, multiplication and division

5.5.2 Variable Documentation

5.5.2.1 princ

```
Principia princ
```

5.6 polynomial.cpp File Reference

```
#include "polynomial.h"
#include <cmath>
#include <iostream>
```

5.7 polynomial.h File Reference

Classes

- class [Polynomial](#)

Variables

- const int [MAX_COEFFCIANTS](#) = 7

5.7.1 Variable Documentation

5.7.1.1 MAX_COEFFCIANTS

```
const int MAX_COEFFCIANTS = 7
```

5.8 principia.cpp File Reference

```
#include "principia.h"
```

5.9 principia.h File Reference

```
#include <string>
```

Classes

- class [Principia](#)

5.10 README.md File Reference

Index

- Calc
 - main.cpp, 13
 - main.h, 15
- CMakeLists.txt, 11
- coefficients
 - Polynomial, 9
- constant
 - Polynomial, 9
- Derivate
 - Polynomial, 8
- Factorial
 - main.cpp, 14
 - main.h, 15
- GetDouble
 - inputfunctions.cpp, 11
 - inputfunctions.h, 12
- GetInteger
 - inputfunctions.cpp, 11
 - inputfunctions.h, 12
- GetString
 - inputfunctions.cpp, 12
 - inputfunctions.h, 13
- GetUnsignedInteger
 - inputfunctions.cpp, 12
 - inputfunctions.h, 13
- inputfunctions.cpp, 11
 - GetDouble, 11
 - GetInteger, 11
 - GetString, 12
 - GetUnsignedInteger, 12
- inputfunctions.h, 12
 - GetDouble, 12
 - GetInteger, 12
 - GetString, 13
 - GetUnsignedInteger, 13
- isSolved
 - Polynomial, 9
- isTwoSolutions
 - Polynomial, 9
- main
 - main.cpp, 14
- main.cpp, 13
 - Calc, 13
 - Factorial, 14
 - main, 14
 - SolvePolynomial, 14
- main.h, 14
 - Calc, 15
 - Factorial, 15
 - princ, 15
 - SolvePolynomial, 15
- MAX_COEFFCIANTS
 - polynomial.h, 16
- operator*
 - Polynomial, 8
- operator+
 - Polynomial, 8
- operator-
 - Polynomial, 8
- Polynomial, 7
 - coefficients, 9
 - constant, 9
 - Derivate, 8
 - isSolved, 9
 - isTwoSolutions, 9
 - operator*, 8
 - operator+, 8
 - operator-, 8
 - Polynomial, 7, 8
 - Print, 9
 - solutions, 9
- polynomial.cpp, 15
- polynomial.h, 16
 - MAX_COEFFCIANTS, 16
- princ
 - main.h, 15
- Principia, 10
 - Principia, 10
- principia.cpp, 16
- principia.h, 16
- Print
 - Polynomial, 9
- README.md, 16
- solutions
 - Polynomial, 9
- SolvePolynomial
 - main.cpp, 14
 - main.h, 15