Laboratory Training 2

C++ Operators and Statements

1.1 Programmatic Implementation of Branching Algorithm

Develop a program that implements an algorithm for solving quadratic equation. The program should include checking all possible variants of the source data. In particular, the discriminant should be checked, and it should be checked whether the equation is quadratic. If the equation degenerates into a linear one, it is necessary to provide for finding the root of this linear equation, or to establish the presence of infinite count of solutions (absence of solutions).

1 1 -2	x1 = -2 x2 = 1
1 1 2	No roots
2 2 -4	x1 = -2 x2 = 1
0 1 2	x = -2
0 0 10	No roots
0 0 0	Infinite count of roots

Laboratory Training 2

C++ Operators and Statements

1.2 Programmatic Implementation of Looping Algorithm

Develop a program that implements an algorithm for calculating the following expression:

$$y = 1/(x + 2) + 2/(x + 4) + ... + (k - 1)/(x + 2(k - 1)) + (k + 1)/(x + 2(k + 1)) + ... + n/(x + 2n)$$

Provide a check of possible errors.

x = 1, n = 4, k = 2	y = 1/3+3/7+4/9
x = 1, n = 4, k = 5	y = 1/3+2/5+3/7+4/9

x = 1 n = 4 k = 2	1.20635
x = 1 n = 4 k = 5	1.60635
x = -2 n = 4 k = 2	Error

Laboratory Training 2

C++ Operators and Statements

1.3 Calculating Product

Write a program that reads **x** and **n** and calculates **y**:

$$y = (x + 1)(x - 2)(x + 3)(x - 4) \ldots (x - 2n)$$

x = 1 n = 3	-720
x = 0.5 n = 4	46899.3

Laboratory Training 2

C++ Operators and Statements

1.4 Calculating Sum

Write a program that reads `eps` and calculates `y`:

$$y = 1/2 + 1/4 + 1/8 + 1/16 + ...$$

The loop terminates if new summand is less than `eps`.

0.1	0.875
0.00001	0.999985
0.0000001	1

Laboratory Training 2

C++ Operators and Statements

1.5 Individual Assignment

You should develop a program that calculates values of a function in a given range. The program should implement an algorithm developed in carrying out assignment 1.3 of previous lab.

#	Function	#	Function
1, 17	$y=\begin{cases}\sum_{i=1}^{n}(i+x)^2, x<0\\ \sum_{i=0}^{n-1}\prod_{j=1}^{n}\frac{x+i}{i+j}, x\geq 0\end{cases}$	9	$y=\begin{cases}\sum_{i=2}^{n-1}\frac{x}{i}, x\leq 0\\ \prod_{i=1}^{n-1}\sum_{j=0}^{i}\frac{i}{j+x}, x>0\end{cases}$
2, 18	$y=\begin{cases}\prod_{i=0}^{n-1}\sum_{j=1}^{n}(x-i+j)^3, x<0\\ \sum_{i=1}^{n-1}\frac{x}{i}, x\geq 0\end{cases}$	10	$y=\begin{cases}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{1}{x-i-j}, x<0\\ \prod_{i=0}^{n-3}(-x-i), x\geq 0\end{cases}$
3, 19	$y=\begin{cases}\prod_{j=2}^{n-2}(j+x), x<0\\ \sum_{i=0}^{n-1}\prod_{j=0}^{n-1}(x+i+j^2), x\geq 0\end{cases}$	11	$y=\begin{cases}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\frac{1}{x-i+xj}, x\leq 0\\ \prod_{i=1}^{n}(\frac{1}{x}-\frac{1}{i}), x>0\end{cases}$
4, 20	$y=\begin{cases}x+\prod_{i=0}^{n-1}(i^2-x), x\leq 0\\ \sum_{i=1}^{n-1}\sum_{j=0}^{n-1}\frac{x}{i+j}, x>0\end{cases}$	12	$y=\begin{cases}x-\sum_{i=0}^{n}\prod_{j=i}^{n-1}(i^2+j), x<0\\ \sum_{i=1}^{n-2}(i-x)^3, x\geq 0\end{cases}$
5, 21	$y=\begin{cases}\prod_{i=1}^{n-1}\sum_{j=1}^{n}\frac{i}{j^2-x}, x<0\\ x-\sum_{i=1}^{n-1}i^3, x\geq 0\end{cases}$	13	$y=\begin{cases}x+\prod_{i=0}^{n-1}(i^3+x), x\leq 0\\ \sum_{i=1}^{n-1}\sum_{j=0}^{n-1}\frac{x}{i+j}, x>0\end{cases}$
6, 22	$y=\begin{cases}\sum_{i=0}^{n}(x-i)^2, x\leq 0\\ \prod_{i=1}^{n}\prod_{j=0}^{n-1}(x-i-2j), x>0\end{cases}$	14	$y=\begin{cases}\prod_{j=2}^{n-2}(j^2+x), x<0\\ \sum_{i=0}^{n-1}\prod_{j=0}^{n-1}(x+i^2+j), x\geq 0\end{cases}$
7, 23	$y=\begin{cases}\sum_{i=0}^{n}\prod_{j=i}^{n-1}(i-xj), x<0\\ \sum_{i=1}^{n-2}(i-x)^2, x\geq 0\end{cases}$	15	$y=\begin{cases}\sum_{i=2}^{n-1}\frac{x-1}{i}, x\leq 0\\ \prod_{i=0}^{n-1}\sum_{j=0}^{i}\frac{i+1}{j+x}, x>0\end{cases}$
8, 24	$y=\begin{cases}\prod_{i=1}^{n-1}\sum_{j=1}^{n}\left(x-i^2+j\right)^2, x<0\\ \sum_{i=0}^{n-1}\frac{x-1}{i+1}, x\geq 0\end{cases}$	16	$y=\begin{cases}\prod_{i=0}^{n-1}(i+x), x\leq 0\\ \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\frac{x+j}{i+j+1}, x>0\end{cases}$

Laboratory Training 3

Use of Functions

1.1 Static Local Variables

Write a program that calculates and shows the minimum and maximum of integers as the user inputs those integers. Use static local variables.

11 -5 4 12 0	11 11 11 -5 -5 11 4 -5 11 12 -5 12 0 -5 12
4 5 0	4 4 4 5 4 5 0 0 5
-4 -100 0	-4 -4 -4 -100 -100 -4 0 -100 0

Laboratory Training 3

Use of Functions

1.2 Recursion

Write a program that reads *x* and *n* and calculates *y* using recursive function:

$$y = (x + 1)(x + 2)(x + 3)(x + 4) \ldots (x + n)$$

x = 1 n = 4	120
x = -2 n = 7	0

Laboratory Training 3

Use of Functions

1.3 Default Arguments

Create a function that returns 1, argument, and product of arguments, depending on arguments count. Test this function in main() function. Implement program in two ways: using function overloading and using default arguments.

0	1
1 10	10
2 4 5	20
3 1 1 1	1

Laboratory Training 3

Use of Functions

1.4 Quadratic Equation

Create a function for solving of quadratic equation. Function should return the number of roots (0, 1, or 2) or -1 if the equation has an infinite number of roots. The function should get the coefficients as arguments and return the roots as reference type arguments.

1 1 -2	x1 = -2 x2 = 1
1 1 2	No roots
2 2 -4	x1 = -2 x2 = 1
0 1 2	Root is -2
0 0 10	No roots
0 0 0	Infinite count of roots

Laboratory Training 3

Use of Functions

1.5 Individual Assignment

You should create a program that implements an individual assignment of previous laboratory training. Program should be split into several functions. Function y() should obtain values of x and n as arguments and return value calculated using formula given in an individual assignment. Create a separate function for reading data. Do not use global variables.

Laboratory Training 4

Use of Arrays and Pointers

1.1 Binary Representation

Write a program that reads unsigned short integers and prints their binary representation. Left-hand zeros should not be printed.

34	100010
4	100
0	0

Laboratory Training 4

Use of Arrays and Pointers

1.2 Sum of the Minimum and Maximum Items

Write a program that calculates the sum of the minimum and maximum items of an array of double precision floating point values. Use two separate functions.

Laboratory Training 4

Use of Arrays and Pointers

1.3 Descending Order

Write a program that sorts items of an array of integers in descending order.

Laboratory Training 4

Use of Arrays and Pointers

1.4 Array in Free Store

Write a program that reads from keyboard the size of a two-dimensional array, allocates an array in free store, reads array items from keyboard, calculates sums of rows and puts these sums into a new array.

2 3 1 2 3 4 5 6	6 15

Laboratory Training 4

Use of Arrays and Pointers

1.5 Individual Assignment

You should create a program that defines and initializes a two-dimensional array of *integer* items and then implements following activities:

- transformation of the source array according to step one of the individual assignment
- creation and filling of a new (one-dimensional) array of *double precision floating point* type items according to step two of the individual assignment
- output of both array items

You should create one-dimensional array in free store (heap) using new operator and remove it before end of execution using delete operator. The program should signal errors if transformation or filling are not possible.

Index of variant (from students list)	Rule of a source array transformation (step one)	Rule of filling of a new array using items of transformed source array (step two). Resulting array should contain:	Row count m	Column count n
1	All items with odd values should be doubled	Square roots of minimal positive items of rows	4	3
2	All items with even values should be replaced with their squares	Cubic roots of minimal items of columns	3	5
3	All items with null value should be replaced with ones	Natural logarithms of maximum positive items of rows	3	4
4	All items with even values should be doubled	Natural logarithms of minimal positive items of columns	4	5
5	All items should be replaced with their absolute magnitudes	Minimal items of columns	5	4
6	All items with even values should be tripled	Cubic roots of diagonal items	3	3
7	All positive items should be replaced with integer parts of their Briggs (base ten) logarithms	Sums of negative items of columns	4	5
8	All negative items should be replaced with their squares	Square roots of diagonal items	4	4
9	All positive items should be replaced with integer parts of their Napierian (natural) logarithms	Products of negative items of rows	5	4
10	All positive items should be replaced with integer parts of their square roots	Maximum positive items of rows	3	5
11	All positive items with even values should be doubled	Cubic roots of maximum items of columns	5	4
12	All negative items with odd values should be tripled	Square roots of minimal positive items of columns	3	4
13	All negative items with odd values should be doubled	Sums of Briggs (base ten) logarithms of positive items of rows	4	3
14	All positive items with even values should be tripled	Quotients from division of maximal positive items of columns by their Briggs (base ten) logarithms	3	5