Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №1 дисциплины «Технологии программирования»

Выполнил студент группы ИВТ-22_____/Крючков И. С/
Проверил_____/Долженкова М. Л./

Киров 2022

1. Задание

Написать программу, реализующую умножение матриц, используя числа типов byte и float с использованием перегруженных функций и шаблонов.

2. Листинг программы

```
#include <iostream>
#include <limits>
using namespace std;
void set_matrix(char **&, char **&);
void set_matrix(float **&, float **&);
void get_value_matrix(char &);
void get_value_matrix(float &);
int arows, acols, brows, bcols;
__int16** mx(char**, char**);
float** mx(float**, float**);
float** mx(float** a, float** b) {
	float** c;
	c = new float *[arows];
	for (int i = 0; i < arows; i++) {
		c[i] = new float[bcols];
		for (int j = 0; j < bcols; j++) {
			c[i][j] = 0;
			for (int k = 0; k < brows; k++) {
				c[i][j] += a[i][k] * b[k][j];
			}
		}
	}
	return c;
}
__int16** mx(char** a, char** b) {
	__int16** c;
	c = new __int16* [arows];
	for (int i = 0; i < arows; i++) {
		c[i] = new __int16[bcols];
		for (int j = 0; j < bcols; j++) {
			c[i][j] = 0;
```

```
                for (int k = 0; k < brows; k++) {
                    c[i][j] += a[i][k] * b[k][j];
                }
            }
        }
        return c;
}
void get_value_matrix_size(int &value) {
        int v;
        while (true) {
            if ((cin >> v).good()) {
                if (v >= 1 and v <= 1000) {
                    value = v;
                    break;
                }
                else {
                    cout << "Invalid value" << endl;
                    cin.clear();
                    cin.ignore(numeric_limits<streamsize>::max(), '\n');
                    continue;
                }
            }
            else {
                cout << "Invalid value" << endl;
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                continue;
            }
        }
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
}
void get_value_type(int &value) {
        int v;
        while (true) {
            if ((cin >> v).good()) {
                if (v >= 1 and v <= 3) {
```

```
                value = v;
                break;
            }
            else {
                cout << "Invalid value" << endl;
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                continue;
            }
        }
        else {
            cout << "Invalid value" << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            continue;
        }
    }
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
}
void get_value_matrix(float &value) {
    float v;
    while (true) {
        if (cin >> v) {
            value = v;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            break;
        }
        else {
            cout << "Invalid value" << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            continue;
        }
    }
}
```

```
void get_value_matrix(char &value) {
	char str[5];
	int tv;
	while (true) {
		cin.getline(str, 5);
		bool f = true;
		bool z = false;
		int k = 0;
		while (k < 4) {
			if (!(str[k] >= '0' and str[k] <= '9' or (str[k] == '-' and k == 0))) {
				if (int(str[k]) == 0) {
					if(k == 0){
						f = false;
						z = false;
						cout << "Invalid value " << endl;
					}
					break;
				}
				else {
					f = false;
					if(cin.rdbuf()->in_avail() >= 1){
						z = true;
					}

					cout << "Invalid value " << endl;
					break;
				}
			}
			k++;
		}
		if (str[0] == '-' and k == 1 and f == true) {
			f = false;
			cout << "Invalid value " << endl;
		}
		if (f) {
			tv = atoi(str);
			if (tv >= -128 && tv <= 127) {
```

```
                                        value = tv;
                                        if (cin.rdbuf()->in_avail() > 0) {
                                                cin.clear();
                                                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                                        }
                                        break;
                                }
                                else {
                                        cout << "Invalid value " << endl;
                                        if (cin.rdbuf()->in_avail() > 0) {
                                                cin.clear();
                                                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                                        }
                                }
                        }
                        if (z) {
                                cin.clear();
                                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                        }
                }
        }
        void set_matrix(char **&a, char **&b) {
                a = new char *[arows];
                cout << "Input A matrix" << endl;
                for (int i = 0; i < arows; i++) {
                        a[i] = new char[acols];
                        for (int j = 0; j < acols; j++) {
                                cout << "a[" << i << "][" << j << "] = ";
                                get_value_matrix(a[i][j]);
                        }
                }
                cout << "Matrix A" << endl;
                for (int i = 0; i < arows; i++) {
                        for (int j = 0; j < acols; j++) {
                                cout << int(a[i][j]) << " ";
                        }
                        cout << endl;
```

```
	}
	b = new char *[brows];
	cout << "Input B matrix" << endl;
	for (int i = 0; i < brows; i++) {
		b[i] = new char[bcols];
		for (int j = 0; j < bcols; j++) {
			cout << "b[" << i << "][" << j << "] = ";
			get_value_matrix(b[i][j]);
		}
	}
	cout << "Matrix B" << endl;
	for (int i = 0; i < brows; i++) {
		for (int j = 0; j < bcols; j++) {
			cout << int(b[i][j]) << " ";
		}
		cout << endl;
	}
}
void set_matrix(float **&a, float **&b) {
	a = new float* [arows];
	cout << "Input A matrix" << endl;
	for (int i = 0; i < arows; i++) {
		a[i] = new float[acols];
		for (int j = 0; j < acols; j++) {
			cout << "a[" << i << "][" << j << "] = ";
			get_value_matrix(a[i][j]);
		}
	}
	cout << "Matrix A" << endl;
	for (int i = 0; i < arows; i++) {
		for (int j = 0; j < acols; j++) {
			cout << a[i][j] << " ";
		}
		cout << endl;
	}
	b = new float *[brows];
	cout << "Input B matrix" << endl;
```

```
	for (int i = 0; i < brows; i++) {
		b[i] = new float[bcols];
		for (int j = 0; j < bcols; j++) {
			cout << "b[" << i << "][" << j << "] = ";
			get_value_matrix(b[i][j]);
		}
	}
	cout << "Matrix B" << endl;
	for (int i = 0; i < brows; i++) {
		for (int j = 0; j < bcols; j++) {
			cout << b[i][j] << " ";
		}
		cout << endl;
	}
}
template<class T>
void print_result(T** matrix) {
	cout << "Result matrix" << endl;
	for (int i = 0; i < arows; i++) {
		for (int j = 0; j < bcols; j++) {
			if(typeid(T) != typeid(float)){
				cout << int(matrix[i][j]) << " ";
			}
			else {
				cout << matrix[i][j] << " ";
			}

		}
		cout << endl;
	}
}
int main()
{
	setlocale(LC_ALL, "Russian_Russia.1251");
	int n = 0;
	do {
		cout << "Matrix A rows (1-1000): " << endl;
```

```
		get_value_matrix_size(arows);
		cout << "Matrix A cols (1-1000): " << endl;
		get_value_matrix_size(acols);
		cout << "Matrix B rows (1-1000): " << endl;
		get_value_matrix_size(brows);
		cout << "Matrix B cols (1-1000): " << endl;
		get_value_matrix_size(bcols);
		if (acols != brows) {
			cout << "Error: Matrix A cols != Matrix B rows" << endl;
		}
} while (acols != brows);

cout << "Select type:" << endl
		<< "1 - float" << endl
		<< "2 - byte" << endl
		<< "3 - Exit" << endl;

while (true) {
		get_value_type(n);
		if (n == 1) {
			float **a, **b;
			set_matrix(a, b);
			print_result(mx(a, b));
			break;
		}
		else if (n == 2) {
			char **a, **b;
			set_matrix(a, b);
			print_result(mx(a, b));
			break;
		}
		else if (n == 3) {
			return 0;
		}
		else {
			continue;
		}
```

```
        }
        system("pause");
        return 0;
}
```

3. Экранные формы

```
Matrix A rows (1-1000):
2
Matrix A cols (1-1000):
2
Matrix B rows (1-1000):
2
Matrix B cols (1-1000):
2
Select type:
1 - float
2 - byte
3 - Exit
1
Input A matrix
a[0][0] = 2
a[0][1] = 3
a[1][0] = 4
a[1][1] = 5
Matrix A
2 3
4 5
Input B matrix
b[0][0] = 1
b[0][1] = 1
b[1][0] = 1
b[1][1] = 1
Matrix B
1 1
1 1
Result matrix
5 5
9 9
```

4. Вывод

В ходе выполнения лабораторной работы были получены необходимые знания синтаксиса языка программирования C++. Написана программа на языке C++, выполняющая умножения матриц, используя числа типов byte и float. Освоены принципы построения перегруженных и шаблонных функций, применены на практике при написании программы. Изучены принципы работы стандартных потоков ввода/вывода. Реализованы механизмы валидации входных данных.