

Часть 1. Консольные приложения на языке C++ в среде Qt Creator

Цель работы: изучить среду разработки *Qt Creator 4-12 Community*.

Часть 1. Однофайловый проект

Цель: создать однофайловый проект в среде разработки “*Qt Creator 4-12 Community*” и научиться работать с ним.

Была создана заготовка консольного приложения на языке C в среде “*Qt Creator 4-12 Community*” с системой сборки *qmake*, названная *prnod* и размещенная в папке Загрузки (Рисунок 1-5).

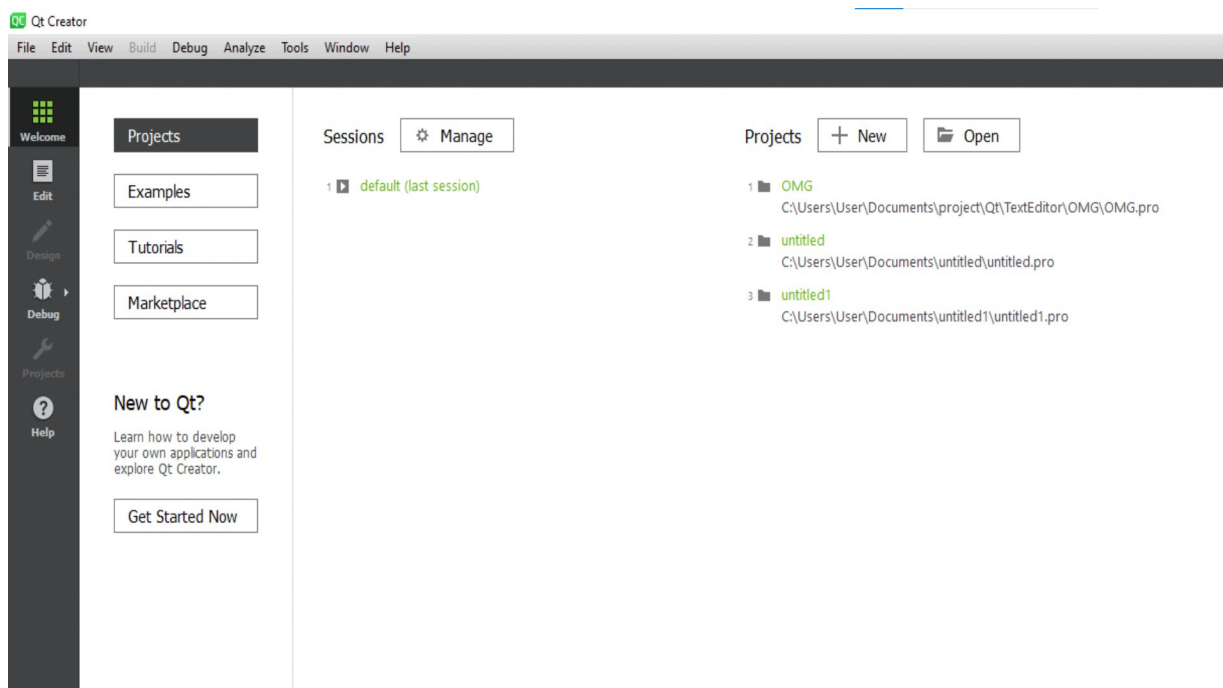


Рисунок 1 – Вид главного окна среды при создании проекта

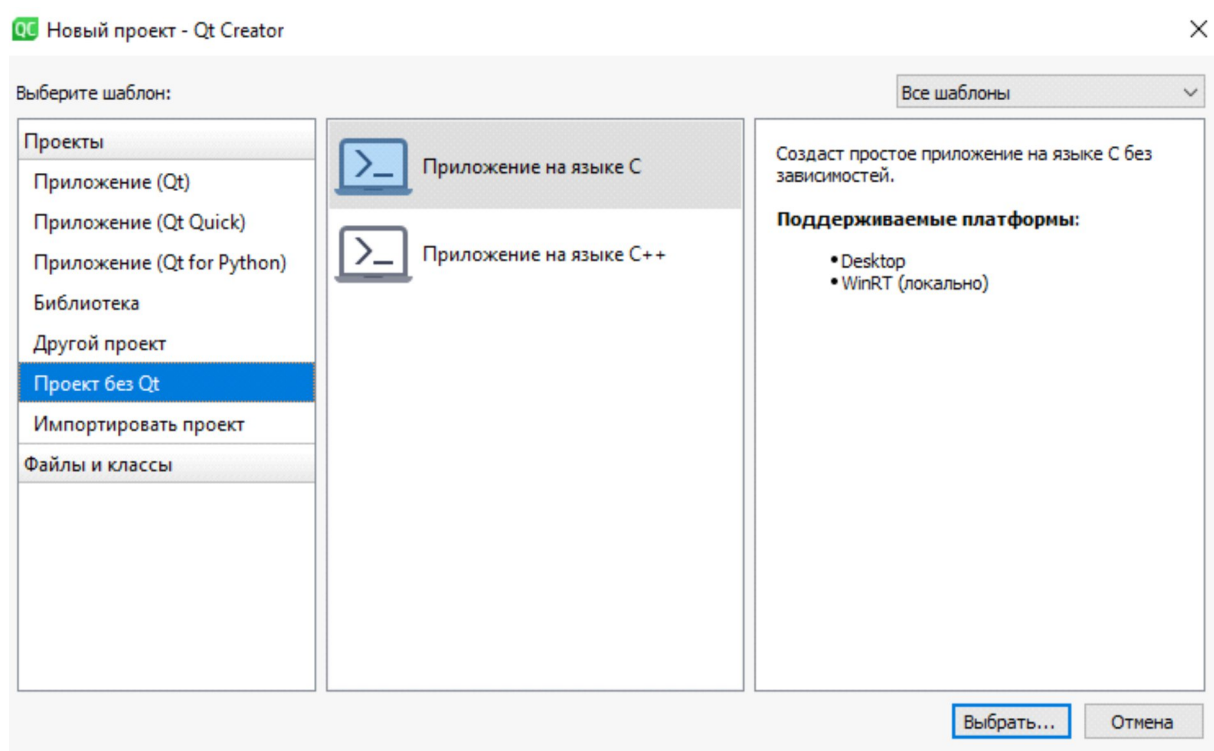


Рисунок 2 – Вид окна выбора типа проекта

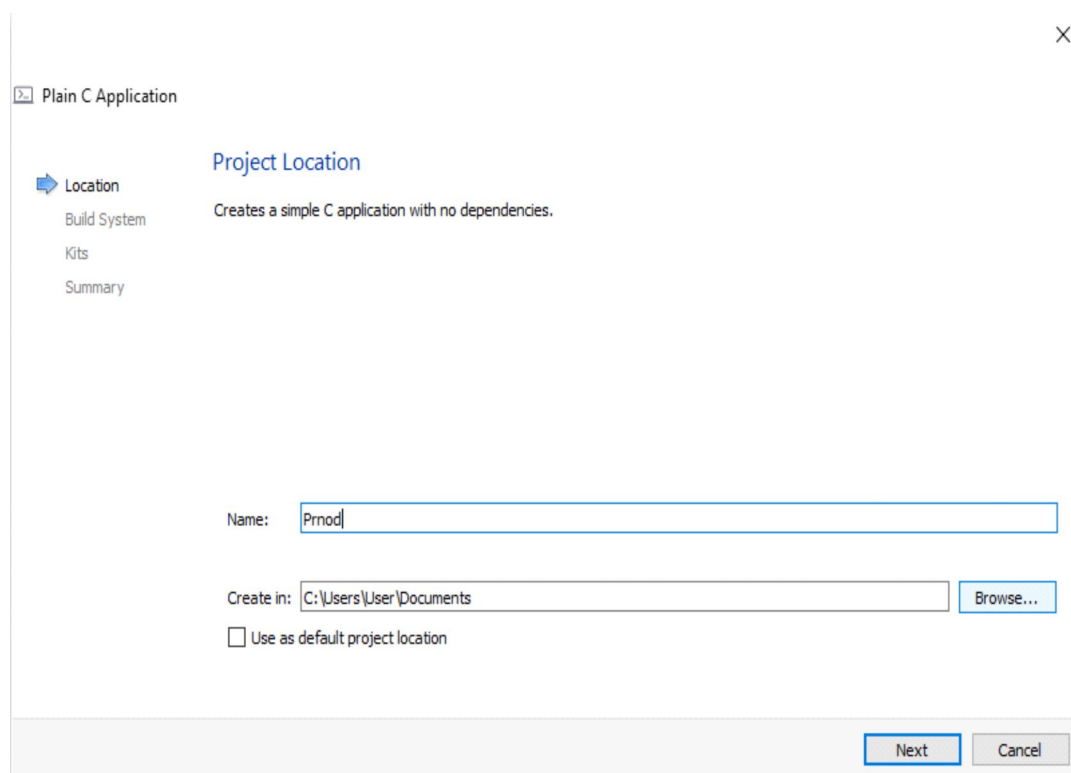


Рисунок 3 – Вид окна выбора имени проекта и задания его местоположения

Define Build System

Build system:

Рисунок 4 – Вид окна выбора системы сборки

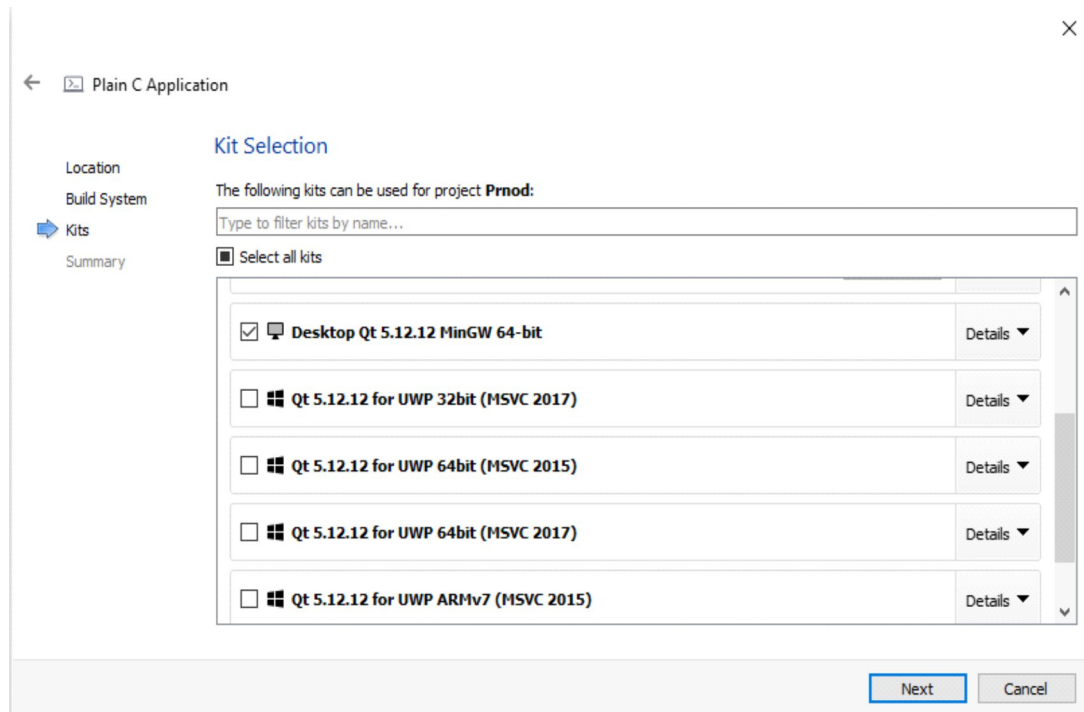


Рисунок 5 – Вид окна выбора комплектов

Затем в окно редактора была введена программа вычисления наибольшего общего делителя (НОД) двух целых чисел (Рисунок 6). Был произведен тестовый запуск программы (Рисунок 7).

Код программы:

```
#include <stdio.h> // подключение процедур ввода вывода
int nod(int x, int y)
{
    while (x!=y)
        if (x>y) x=x-y;
```

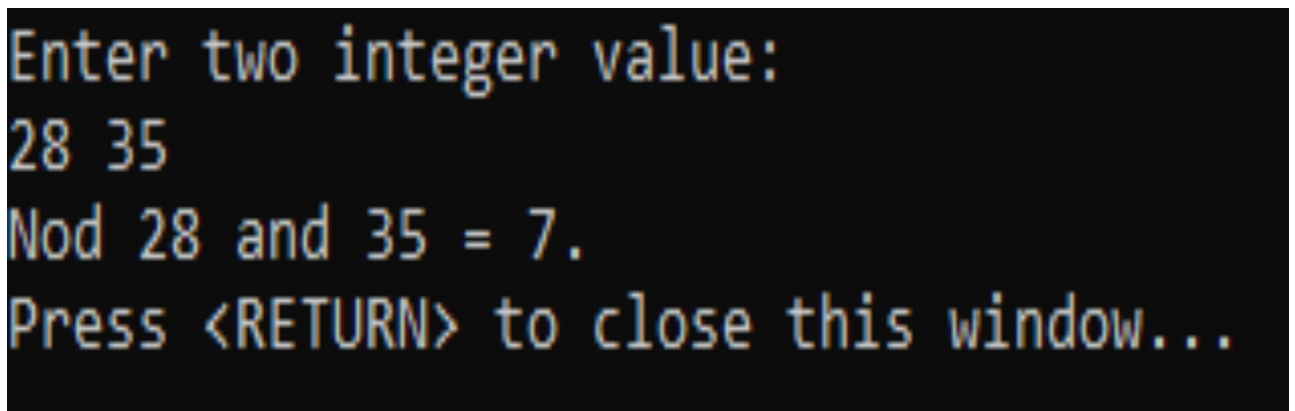
```

        else y=y-x;
    return y;
}
int main()
{
    int a,b;

    puts("Enter two integer value:");
    scanf_s("%d %d",&a,&b);
    printf("Nod %d и %d = %d.\n",a,b,nod(a,b));
    return 0;
}

```

Рисунок 6 – Программа вычисления НОДа в окне редактора



```

Enter two integer value:
28 35
Nod 28 and 35 = 7.
Press <RETURN> to close this window...

```

Рисунок 7 – Окно консоли с выводом результата работы программы

Вывод: было получено представление о среде *Qt Creator 4-12 Community* и базовые навыки работы с ней: создание однофайлового приложения, запуск программ и его настройка, обнаружение ошибок программы.

Часть 2. Многофайловый проект.

Цель: создать многофайловый проект и научиться работать с ним.

Создан и добавлен в проект заголовочный файл *Nod.h*, который с помощью строки **int nod(int x, int y)** (заголовка функции нахождения НОДа двух целых чисел) будет вызывать модуль с телом функции **nod**.

Код модуля *Nod.h*:

```
-----  
#ifndef NOD_H  
#define NOD_H  
int nod(int x, int y);  
#endif // NOD_H  
-----
```

Тем же способом был создан файл реализации модуля *Nod.cpp*, расширение которого в дальнейшем было исправлено на *.c* для соответствия выбранному языку программирования C.

Функция *nod()* была перенесена из файла программы в файл реализации модуля *Nod.c* и подключена к заголовочному файлу модуля *Nod.h* с помощью строки **#include "Nod.h"**.

Код модуля *Nod.c*:

```
-----  
#include <stdio.h>  
#include "Nod.h"  
int nod(int x, int y)  
{  
    while (x!=y)  
        if (x>y) x=x-y;  
        else y=y-x;  
}
```

```
        return y;
    }

```

Вместо функции *nod()* в файл основной программы *main.c* была добавлена ссылка на заголовочный файл *Nod.h*: **#include "Nod.h"**.

Код модуля *main.c*:

```
#include "Nod.h"
int main()
{
    int a,b;

    puts("Enter two integer value:");
    scanf_s("%d %d",&a,&b);
    printf("Nod %d и %d = %d.\n",a,b,nod(a,b));
    return 0;
}

```

После запуска созданного многофайлового проекта был получен тот же результат, что и при первом запуске программы (однофайлового проекта)

Вывод: были получены навыки создания и добавления в проект файлов.

Часть 3. Отладка

Цель: ознакомиться с механизмом пошаговой отладки программ и научиться его использованию.

С помощью подпункта меню «Отладка» **Запустить и встать на main** была запущена пошаговая отладка программы.

С помощью клавиши **F10** был выполнен шаг программы с вводом.

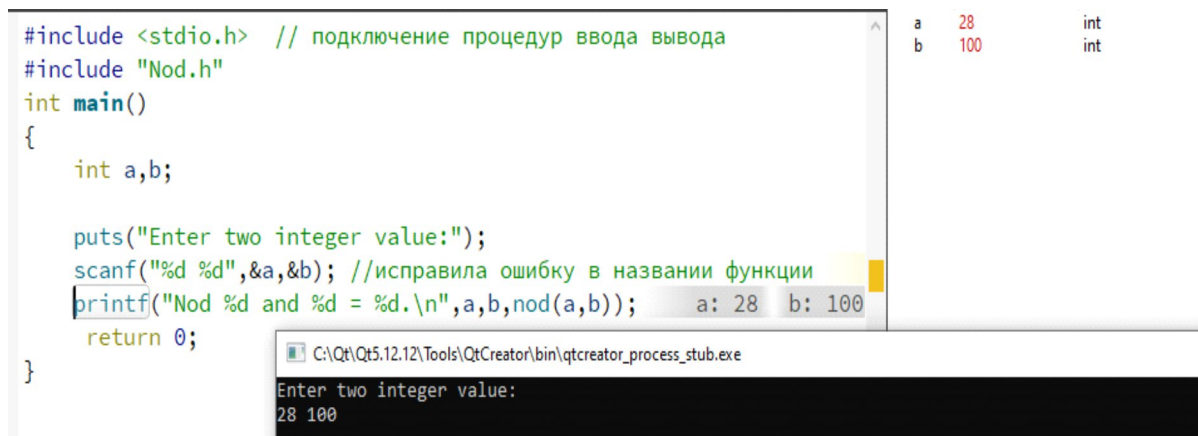


Рисунок 8 – Окно после выбора режима пошаговой отладки

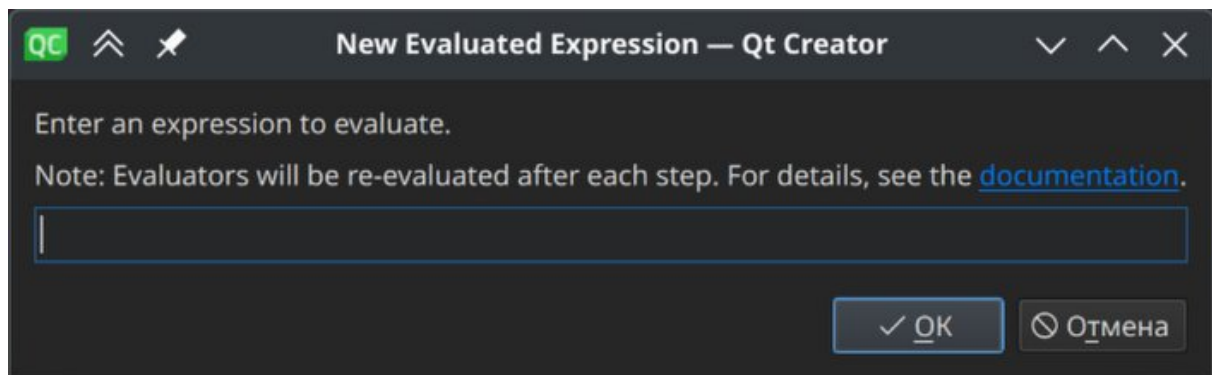


Рисунок 9 – окно добавления переменной/выражения

С помощью пункта меню **Отладка/Добавить вычисляемое выражение** были добавлены выражения и переменные для просмотра их текущего значения в ходе отладки (Рисунок 8, 9).

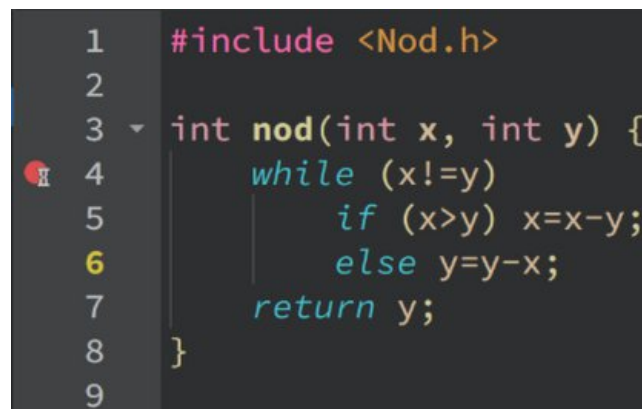


Рисунок 10 – точка останова

С помощью нажатия на серое поле слева от интересующей строки была создана точка останова. Клавишами **F10**, **F11** была произведена отладка программы с точки останова. (Рисунок 10)

Вывод: были получены навыки пошаговой отладки программы с использованием захода в подпрограмму, точек останова и просмотром текущих значений добавленных выражений.

Заключение: в результате проделанной работы я научился создавать однофайловые и многофайловые проекты в среде *Qt Creator 4-12 Community*, а также использовать средства отладки консольных приложений.

Часть 2. Создание схем алгоритмов в визуальных средах

Цель работы: Приобретение практических навыков создания схем алгоритмов, с использованием Microsoft Visio 2016 и Draw.io.

Задание 1. Создание схем алгоритмов средствами Microsoft Visio 2016

Цель задания 1: создать схему алгоритма и овладеть базовыми навыками работы в Microsoft Visio 2016.

Была запущена программа Microsoft Visio 2016. В открывшемся окне выбрана «Простая схема», как показано на рисунке 9. Таким образом создается пустой документ.

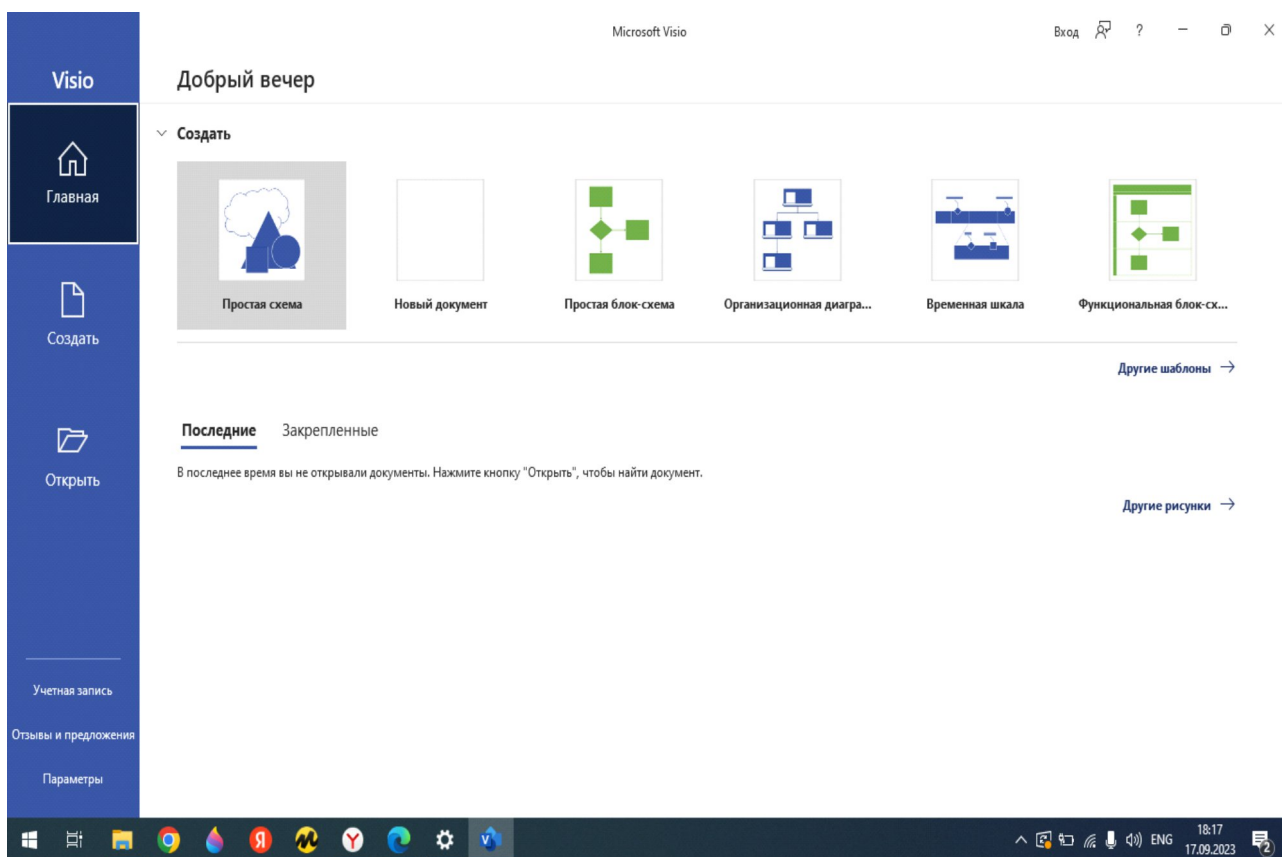


Рисунок 9 – Создание пустого документа в Microsoft Visio 2016

Была создана схема алгоритма с использованием раздела «Фигуры». Основные элементы схемы находятся в подразделе «Простые фигуры», представленном на рисунке 10. Для того, чтобы добавить необходимую фигуру, необходимо перетащить ее из раздела «Фигуры» на рабочее пространство. У добавленной фигуры можно изменить размер, цвет заливки и контура, редактировать форму (закругление, толщина, ширина). Двойным щелчком левой кнопкой мыши по фигуре добавляется текст, который можно также форматировать через пункт меню «Главная».

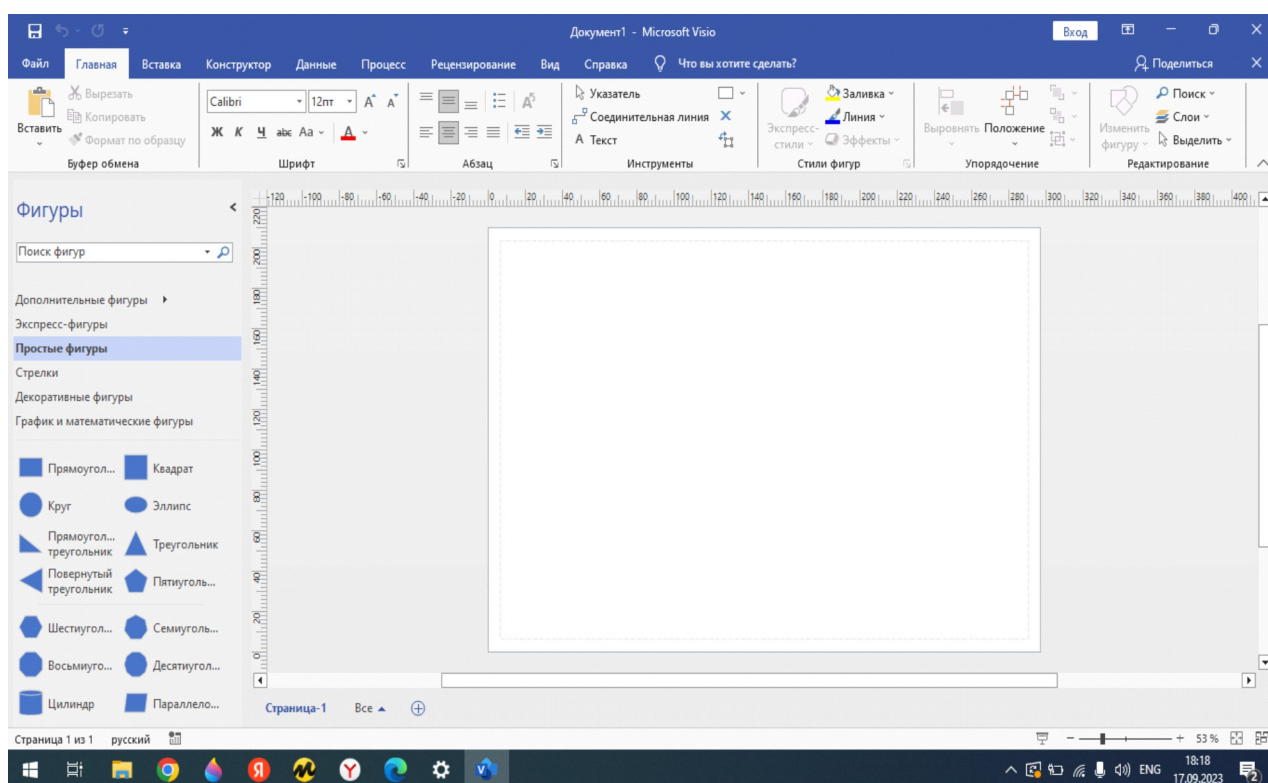


Рисунок 10 – Подраздел «Простые фигуры» Microsoft Visio 2016

Далее были соединены фигуры с помощью прямых и стрелок. Для этого используется подраздел «Стрелки» раздела «Фигуры». Чтобы добавить прямую, следует выбрать «Прямая линия» и перетащить данный элемент в рабочее пространство, как показано на рисунке 11.

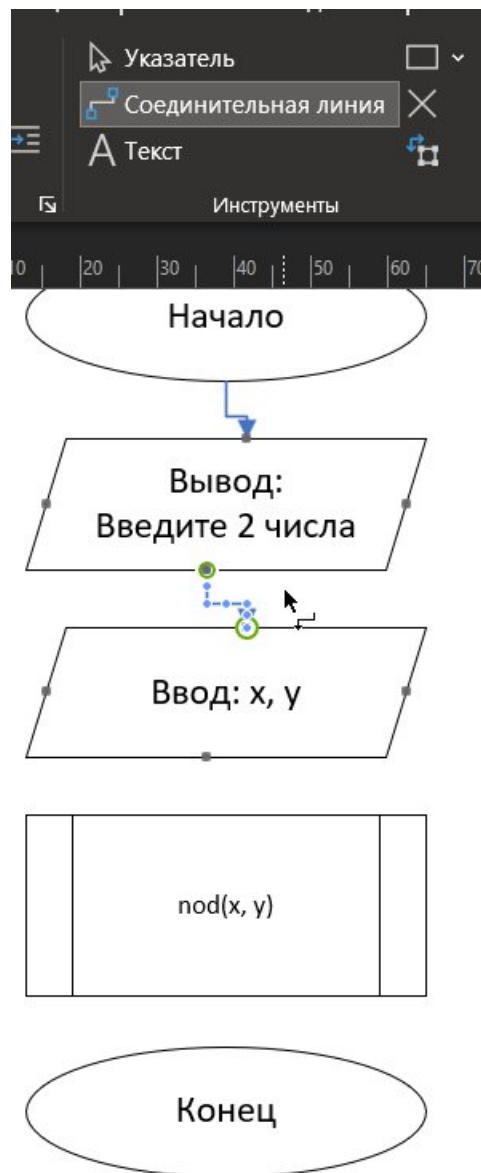


Рисунок 11 – Добавление линий Microsoft Visio 2016

В нужных местах были добавлены стрелки, для чего необходимо выбрать «Линия со стрелкой». После того, как стрелка добавлена и перетащена в нужное место, следует указать для нее направление. Для этого делается щелчок правой кнопкой мыши по стрелке и выбирается необходимое направление (Рисунок 12).

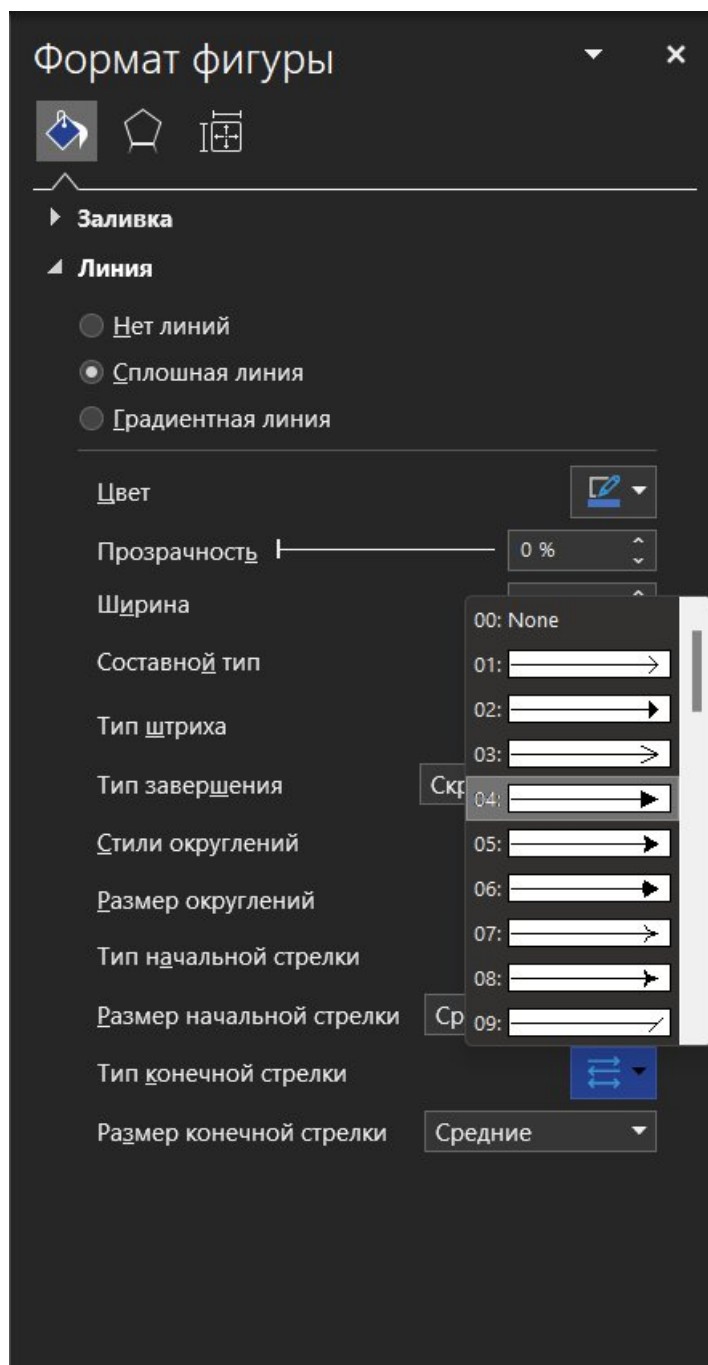


Рисунок 12 – изменение стрелки

По завершению работы документ был сохранен через подпункт «Сохранить как...» пункта меню «Файл». Таким образом получена готовая схема алгоритма, представленная на рисунке 13.

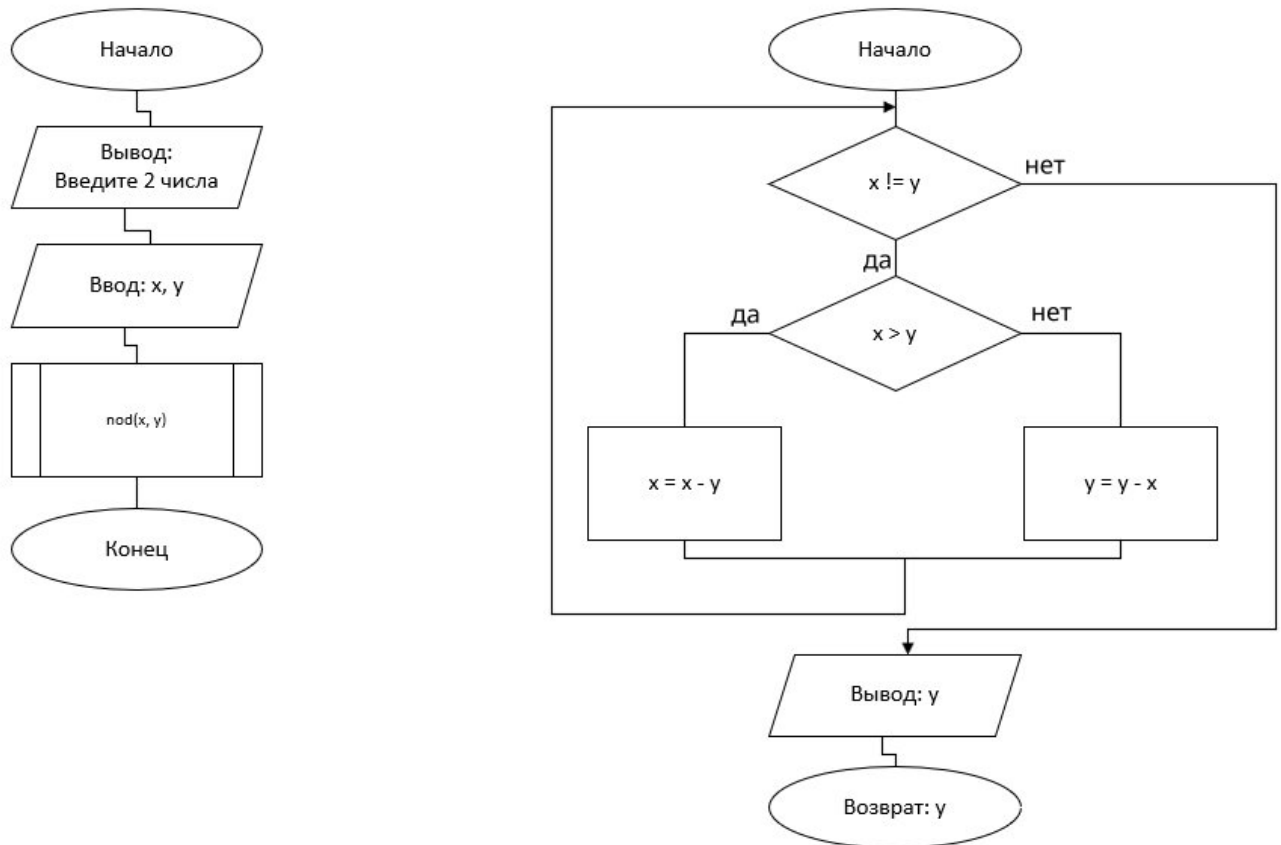


Рисунок 13 – Схема алгоритма, созданная в Microsoft Visio 2016

Вывод к заданию 1: в ходе выполнения задания была создана схема алгоритма и получены базовые навыки работы в Microsoft Visio 2016.

Задание 2. Создание схем алгоритмов средствами Draw.io

Цель задания 2: создать схему алгоритма и получить базовые навыки работы в Draw.io.

Была запущена программа Draw.io. В появившемся окне выбрано «Это устройство», «Создать новую диаграмму», как показано на рисунке 14.

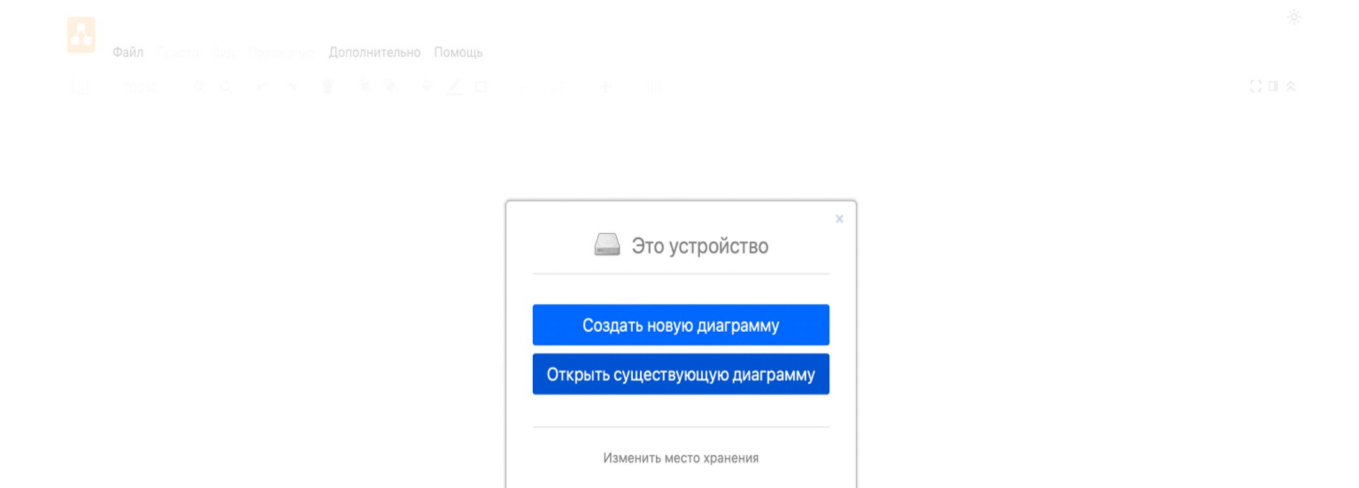


Рисунок 14– Создание новой диаграммы Draw.io

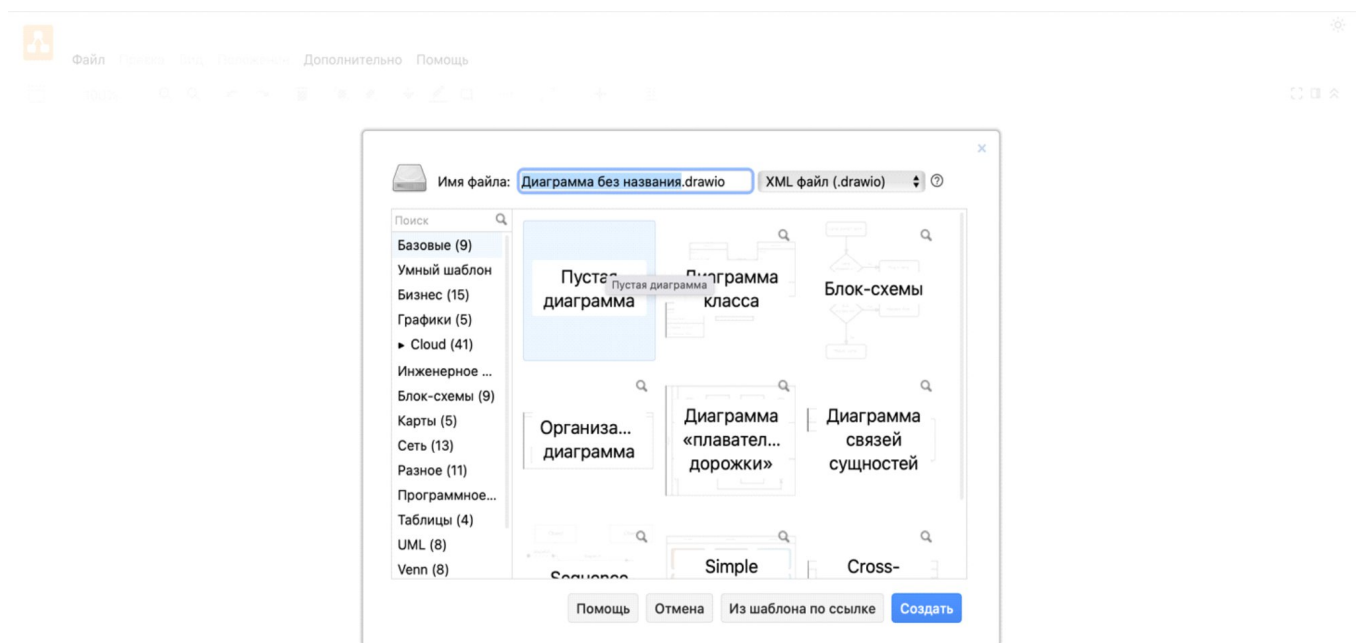


Рисунок 15 – Выбор вида диаграммы Draw.io

Далее выбрано «Пустая диаграмма» в окне, представленном на рисунке 15. Таким образом создается пустой документ, в котором будет рисоваться схема алгоритма. Вид этого документа представлен на рисунке 16.

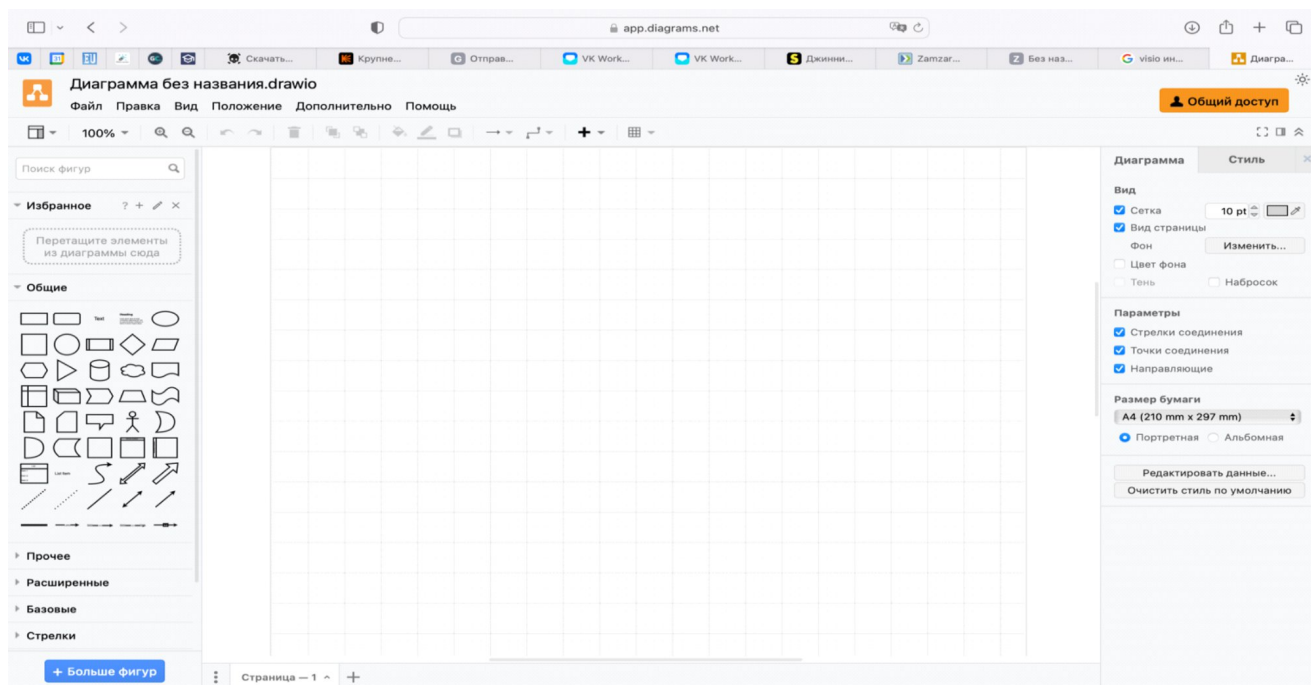


Рисунок 16 – Пустой документ Draw.io

Для добавления нужных элементов в рабочее пространство, используется панель с фигурами, раздел «Общие». Необходимые элементы перетаскиваются в рабочее пространство.

При добавлении фигуры справа появляется панель с инструментами для редактирования и форматирования фигуры и ее текста. Можно изменить размер, цвет заливки и контура, форматировать форму фигуры (ширина, толщина). Чтобы добавить текст в элемент, необходимо дважды щёлкнуть левой кнопкой мыши по фигуре. Возможности редактирования фигуры представлены на рисунке 17.

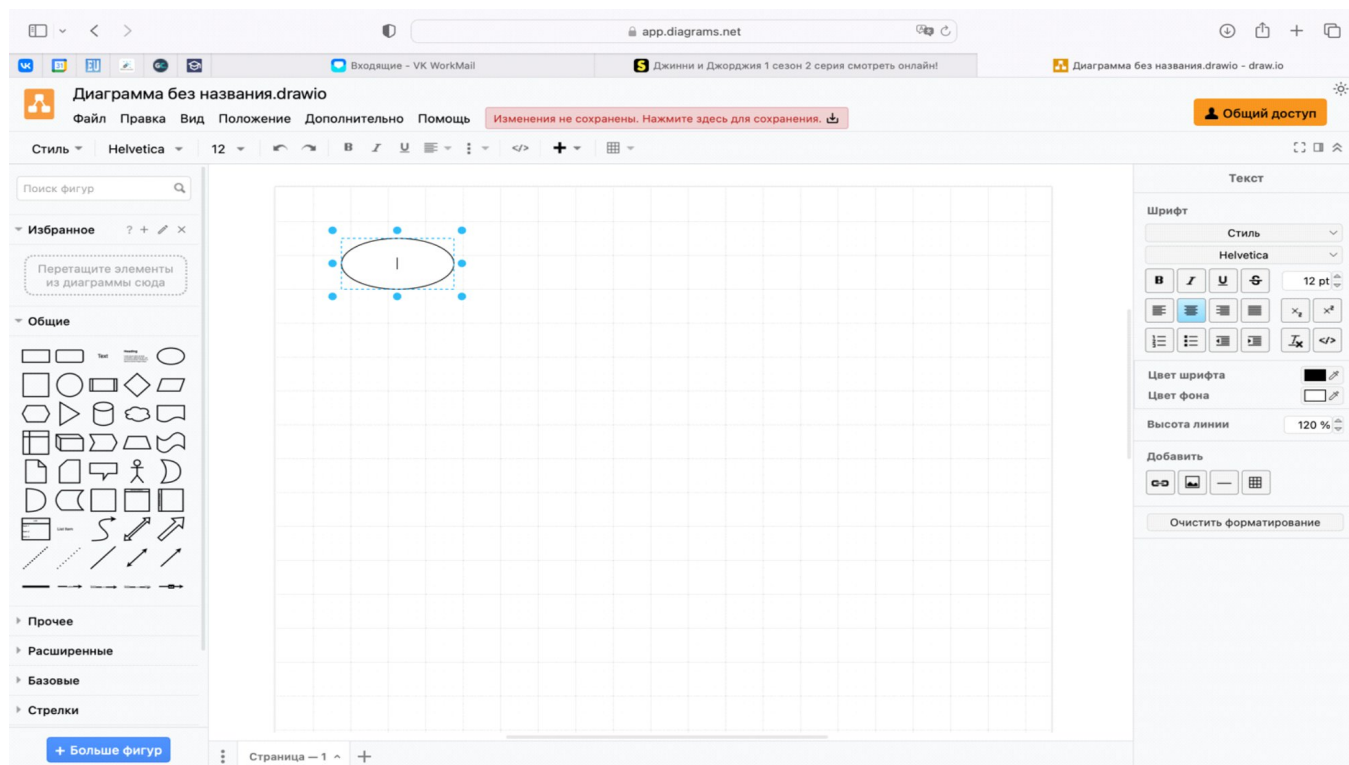


Рисунок 17 – Добавление фигуры Draw.io

Для соединения элементов между собой, добавляются прямые и стрелки из того же раздела. Чтобы выбрать направление стрелки, нужно переместить ее подходящим образом в рабочем пространстве. Вид добавленной линии показан на рисунке 18.

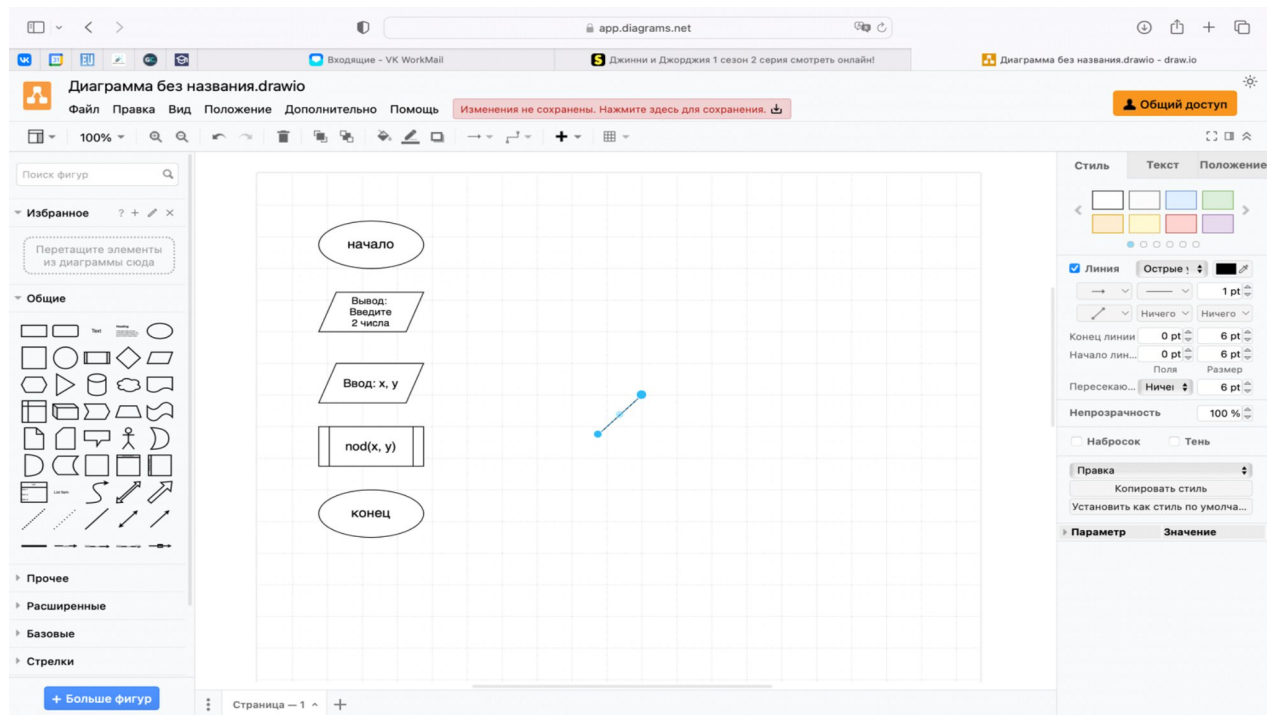


Рисунок 18 – Добавление прямой Draw.io

Чтобы прямую или стрелку изогнуть необходимое количество раз, нужно потянуть за кружки на линии, которые не обозначают конец линии или уже существующий изгиб. Изгибы линии представлены на рисунке 19.

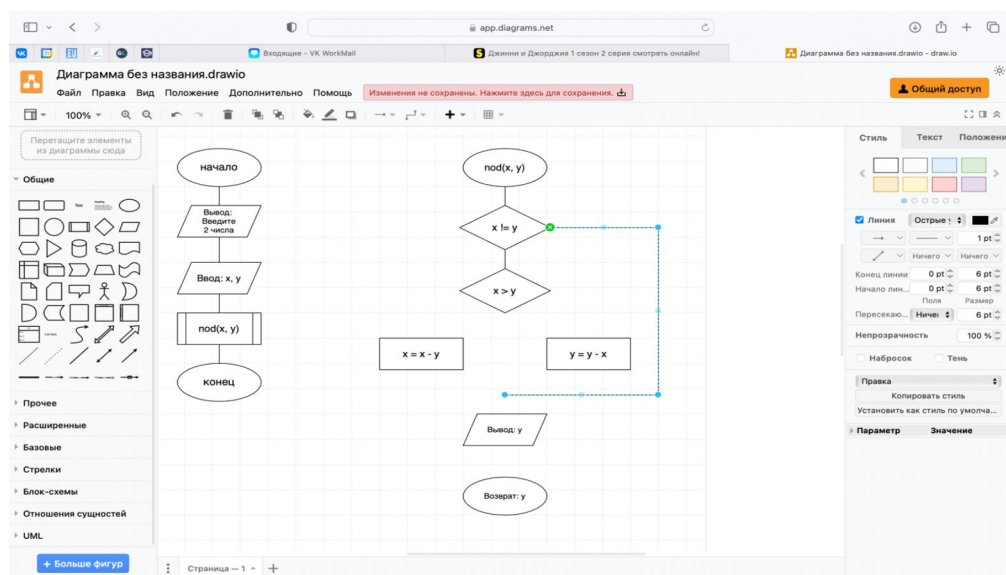


Рисунок 19 – Добавление изгибов линии Draw.io

Чтобы добавить текст, не принадлежащий фигуре, необходимо из панели с фигурами, раздела «Общие» перетащить значок, обозначающие текст, вид которого представлен на рисунке 20.

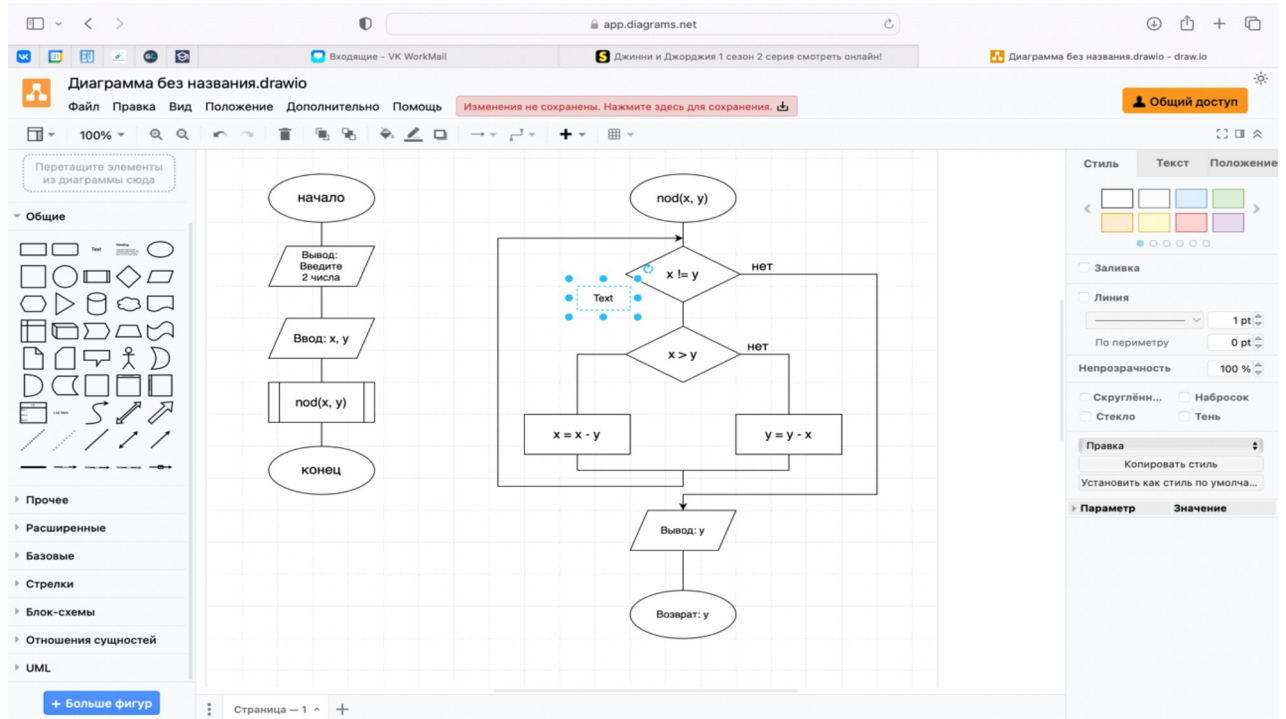


Рисунок 20 – Добавление текста Draw.io

После выполнения действий документ был сохранен через пункт меню «Файл». Полученная готовая схема алгоритма представлена на рисунке 21.

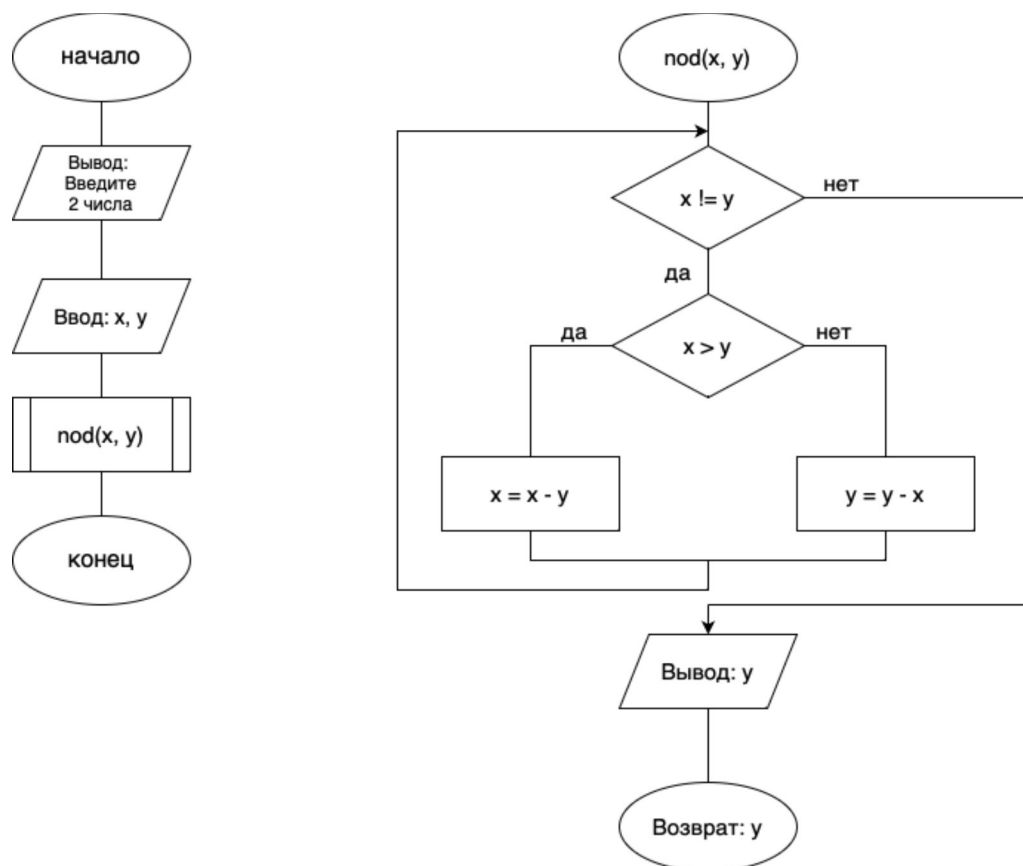


Рисунок 21 – Схема алгоритма Draw.io

Вывод к заданию 2: в ходе выполнения задания была создана схема алгоритма и получены базовые навыки работы в Draw.io.

Вывод: в результате лабораторной работы были приобретены навыки создания схем алгоритмов в Microsoft Visio 2016 и Draw.io.

Был сделан вывод, что для создания схемы алгоритма удобнее использовать среду Draw.io. В ней проще работать с прямыми и стрелками, можно легко добавлять любое количество изгибов. Также здесь присутствует разметка страницы, что способствует созданию ровных схем.

Выводы к лабораторной работе: в результате выполнения лабораторной работы были приобретены базовые навыки обращения со средой “Qt

Creator 4-12 Community” и создания схем алгоритма в Microsoft Visio 2016 и Draw.io.

В ходе лабораторной работы в среде “*Qt Creator 4-12 Community*” была написана и разбита на подпрограммы в файлы модулей простейшая программа. Также на примере данной программы были приобретены навыки использования средств отладки.

В ходе лабораторной работы были созданы схемы алгоритма в программах Microsoft Visio 2016 и Draw.io, а также выявлена наиболее подходящая для данной задачи программа. В Draw.io легче работать с некоторыми из необходимых в построении схем элементов, поэтому эта программа выбрана как наиболее удобная.