**Отчет ПО ЛАБОРАТОРНЫМ РАБОТАМ**

**По информатике**

Лабораторная работа №1.

Порядок сборки проекта на языке C++

(наименование лабораторной работы в соответствии с учебным планом)

***Цель и задачи работы:***

Изучение порядка сборки проекта на языке С++ с консоли.

***Задание №1***

1. Выучить основные команды командной строки (cmd): cd, chdir, chkdsk, cls, copy, del, dir, erase, find, help, md, mkdir, move, rd, rename, replace, rmdir, type, xcopy.

2. Научиться работать в любом файловом менеджере без использования мыши.

3. Составить и добавить в отчёт таблицу с перечнем файловых менеджеров и датой их последнего обновления.

4. Добавить в отчёт описание типов данных.

***Теоретические сведения:***

Основные команды командной строки:

Сd - вывод имени либо смена текущей папки.

Chdir - вывод имени либо смена текущей папки.

Chkdsk - проверка диска и вывод статистики.

Cls - очистка экрана.

Copy - копирование одного или нескольких файлов файлов в другое место.

Del - удаление одного или нескольких файлов.

Dir - вывод списка файлов и подпапок из указанной папки.

Erase - удаляет один или несколько файлов.

Find - ищет текстовую строку в одном или нескольких файлах.

Help - выводит справочную информацию о командах Windows.

Md - создает каталог.

Mkdir - создает каталог.

Move - перемещает один или несколько файлов из одного каталога в другой.

Rd - удаляет каталог.

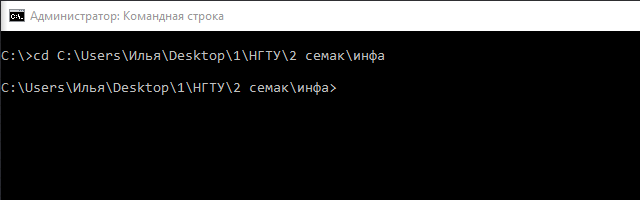
Rename - переименовывает файлы.

Replace - заменяет файлы.

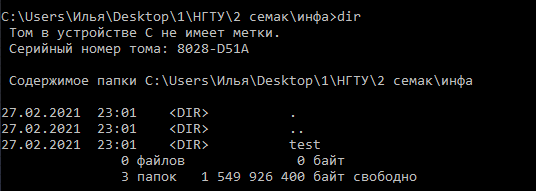
Rmdir - удаляет каталог.

Type - отображает содержимое текстовых файлов.

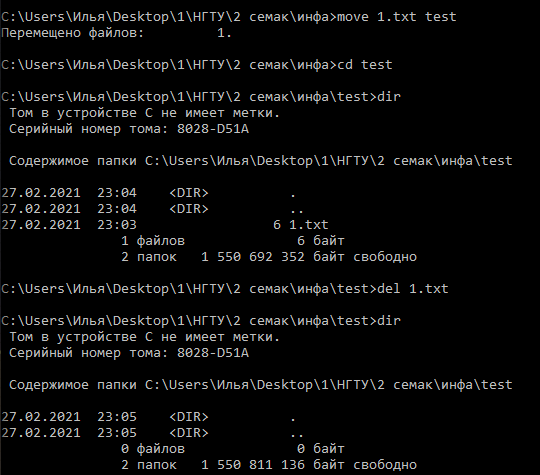
Xcopy - копирует файлы и деревья папок.



*(рис.1 - пример работы команды cd)*



*(рис.2 - пример работы команд mkdir и dir)*



*(рис.3 - пример использования команд move , del)*

**Файловый менеджер**  — компьютерная программа, предоставляющая интерфейс пользователя для работы с файловой системой и [файлами](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB). Файловый менеджер позволяет выполнять наиболее частые операции над файлами — создание, открытие/проигрывание/просмотр, редактирование, перемещение, переименование, копирование, удаление, изменение атрибутов и свойств, поиск файлов и назначение прав. Помимо основных функций, многие файловые менеджеры включают ряд дополнительных возможностей.

|  |  |  |  |
| --- | --- | --- | --- |
| **Файловый менеджер** | **Особенности** | **Операционная система** | **Последняя версия** |
| Midnight Commander | Поддержка многоязыкового интерфейса. Настраиваемое пользовательское меню. Список каталогов для быстрого доступа. Операции копирования и перемещения файлов в фоновом режиме | GNU/Linux, macOS, Windows | 4.8.26 (10 января 2021) |
| Cloud Commander | Клиент-серверная архитектура позволяющая подключаться удаленно к файловому менеджеру. Работа в браузере. Настраиваемая авторизация. Использование технологий diff и [Web Storage](https://ru.wikipedia.org/wiki/Web_Storage) для ускорения операций считывания и записи файлов. | Windows, Linux, Mac OS, Android | 10.2.0 (6 июня 2018) |
| Double Commander | Поддержка Unicode. Все операции выполняются в фоновом режиме | Linux,Windows,Mac OS | 0.9.9 (27 июля 2020) |
| MuCommander | Работа с локальными дисками, с архивами. Гибкая настройка «горячих клавиш» | Java Runtime Environment | 0.9.4-2 (8 марта 2020) |
| Far Manager | Поддерживает работу с файловой системой NTFS, различные кодировки текстов, может использовать системные функции для копирования файлов, имеет многоязычный интерфейс и систему помощи. | Windows, Linux | 3.0.05742.1996 (18 февраля 2021) |
| Total Commander | Внутренние распаковщики архиваторов. Многоязычный графический интерфейс. Настраиваемые сочетания клавиш. Вкладочный интерфейс. | Windows, Windows Mobile, Android | 3.20 (25 марта 2020) |

**Типы данных**

Каждая переменная имеет определенный тип. Этот тип определяет, какие значения может иметь переменная, какие операции с ней можно производить и сколько памяти она будет занимать. В C++ определены следующие типы данных:

**bool**: логический тип. Может принимать одну из двух значений **true** (истина) и **false** (ложь). Размер занимаемой памяти для этого типа точно не определен.

**char**: представляет один символ в кодировке ASCII. Занимает в памяти 1 байт. Может хранить любое значение из диапазона от -128 до 127, либо от 0 до 255.

**short**: представляет целое число в диапазоне от –32768 до 32767. Занимает в памяти 2 байта.

**int**: представляет целое число. В зависимости от архитектуры процессора может занимать 2 байта (16 бит) или 4 байта (32 бита). Диапазон предельных значений соответственно также может варьироваться от –32768 до 32767 (при 2 байтах) или от −2 147 483 648 до 2 147 483 647 (при 4 байтах).

**long**: представляет целое число в диапазоне от −2 147 483 648 до 2 147 483 647. Занимает в памяти 4 байта.

**float**: представляет вещественное число ординарной точности с плавающей точкой в диапазоне +/- 3.4E-38 до 3.4E+38. В памяти занимает 4 байта.

**double**: представляет вещественное число двойной точности с плавающей точкой в диапазоне +/- 1.7E-308 до 1.7E+308. В памяти занимает 8 байт.

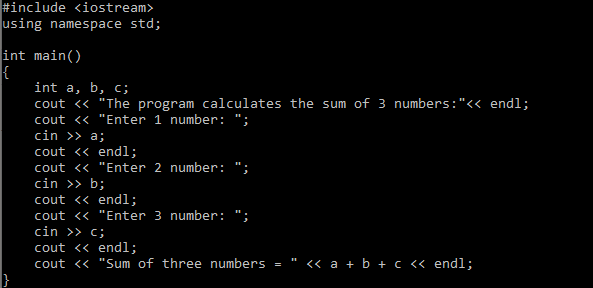
**void**: тип без значения.

В таблице приведены ограничения для обычных числовых представлений.

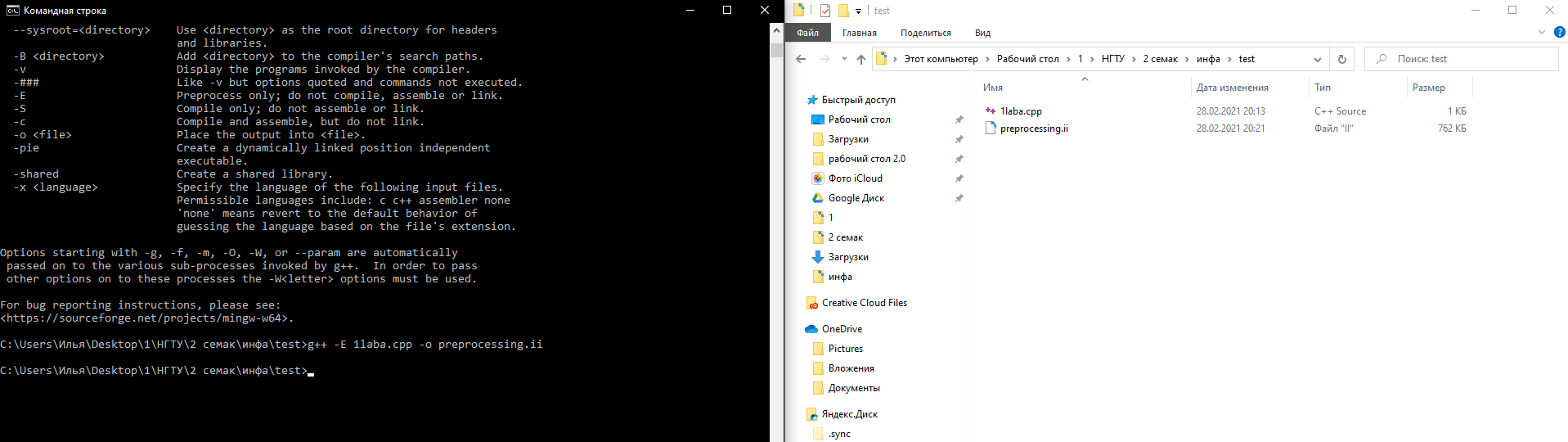
|  |  |  |
| --- | --- | --- |
| **Имя типа** | **Байты** | **Диапазон значений** |
| Int | 4 | От -2147483648 до 2147483847 |
| Unsigned int | 4 | От 0 до 4294967295 |
| Bool | 1 | False или true |
| Char | 1 | От -128 до 127 по умолчанию |
| Short | 2 | От -32768 до 32767 |
| Long | 4 | От -2147483648 до 2147483647 |
| Float | 4 | 3,4\*10^(-38) до 3,4 \* 10^(+38) |
| Double | 8 | 1,7\*10^(-308) до 1,7\*10^(+308) |
| Void |  | Без значения |

**Задание №2.**

Написать программу сложения трех чисел на языке С++. Показать последовательный порядок сборки проекта в командной строке (препроцессинг, компиляция, ассемблирование, линковка, запуск исполняемого файла) с использованием g++. Объяснить содержимое формируемых промежуточных файлов.

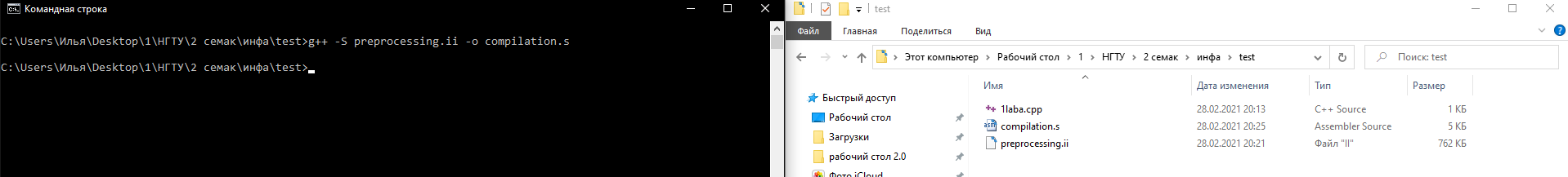


*(рис. 4 – программа сложения трех чисел)*



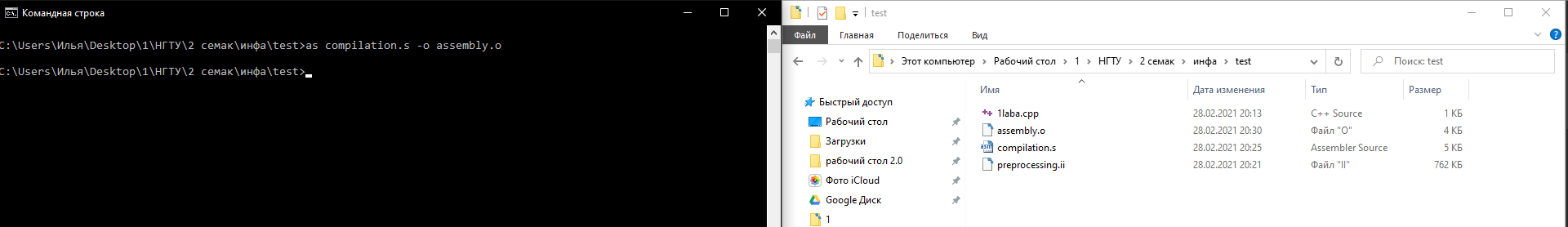
*(рис. 5 – этап препроцессинга)*

**Препроцессор** — это макро процессор, который преобразовывает программу для дальнейшего компилирования. На данной стадии происходит работа с препроцессорными директивами. Например, препроцессор добавляет хэдеры в код (**#include**), убирает комментирования, заменяет макросы (**#define**) их значениями, выбирает нужные куски кода в соответствии с условиями **#if**, **#ifdef** и **#ifndef**. Получим код в выходной файл **preprocessing.ii** (прошедшие через стадию препроцессинга C++ файлы имеют расширение **.ii**), используя флаг **-E**, который сообщает компилятору, что компилировать файл не нужно, а только провести его препроцессинг.



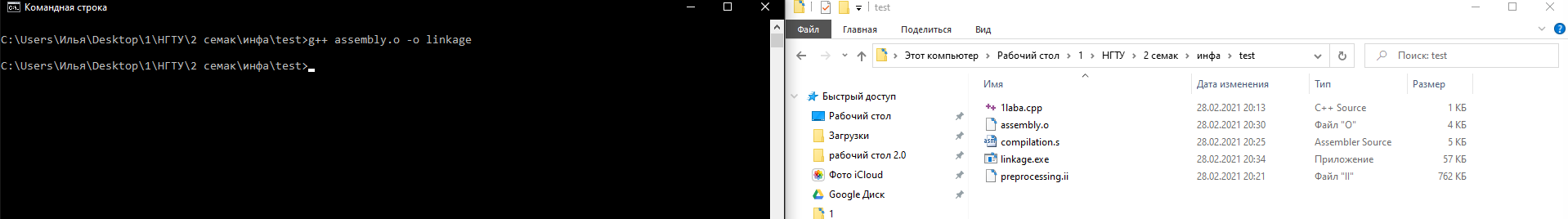
*(рис. 6 – этап компиляции)*

На данном шаге происходит компиляция, то есть преобразование полученного на прошлом шаге кода без директив в ассемблерный код. Это промежуточный шаг между высокоуровневым языком и машинным кодом.



*(рис. 7 – этап ассмеблирования)*

Ассемблер преобразовывает ассемблерный код в машинный, сохраняя его в объектном файле. **Объектный файл** — это созданный ассемблером промежуточный файл, хранящий кусок машинного кода. Этот кусок машинного кода, который еще не был связан вместе с другими кусками машинного кода в конечную выполняемую программу, называется объектным кодом. Получим машинный код с помощью ассемблера (**as**) в выходной объектный файл **assembly.o**.



*(рис. 7 – этап компоновки)*

Компоновщик связывает все объектные файлы и статические библиотеки в единый исполняемый файл, который мы и сможем запустить в дальнейшем.

Изучив каждый этап компиляции по отдельности, можно вывести общую цепь, по которой «передвигается» исходный код в процессе компиляции. Сначала программа (исходный код) передается препроцессору, который выполняет директивы. Итоговый код программы поступает к компилятору, который выделяет лексемы, а затем распознает выражения и операторы, построенные из этих лексем. Выделяет ошибки, если нет, то строит объектный код. Компоновщик формирует исполняемый модуль программы, подключая библиотечные файлы.

**Вывод:** Изучена большая часть команд командной строки. Была построена (написана) программа сложения трёх чисел на языке С++. Во время компиляции проекта в командной строке были изучены этапы и их порядок сборки программы в исполняемый файл.