MinGW-w64

Установка компиляторов GCC (C++) и Clang в Windows. Интеграция с IDE Code::Blocks и Qt Creator

Методическое руководство

Последнее изменение 28.01.22

http://inf-w.ru

1. О MinGW-w64

MinGW-w64 (Minimalist GNU for Windows) является проектом, который создан для поддержки компилятора GCC в системах Windows. Он обеспечивает поддержку как 32-битных, так и 64-битных архитектур, а также новых API. Сердцевиной проекта Mingw-w64 являются заголовки и библиотеки поддержки для запуска вывода GCC в Windows

Поскольку MinGW-w64 не является ни GCC, ни binutils, существуют насколько проектов предоставляющих установочные бинарные пакеты, которые их объединяют. Перейдите на сайт MinGW-w64 (https://www.mingw-w64.org/) и кликните на ссылке Downloads. На этой странице (Рис. 1.1) перечислены все доступные ресурсы. Среди всех проектов, наиболее актуальные версии компиляторов предоставляют Cygwin и MSYS2 (GCC и Clang).

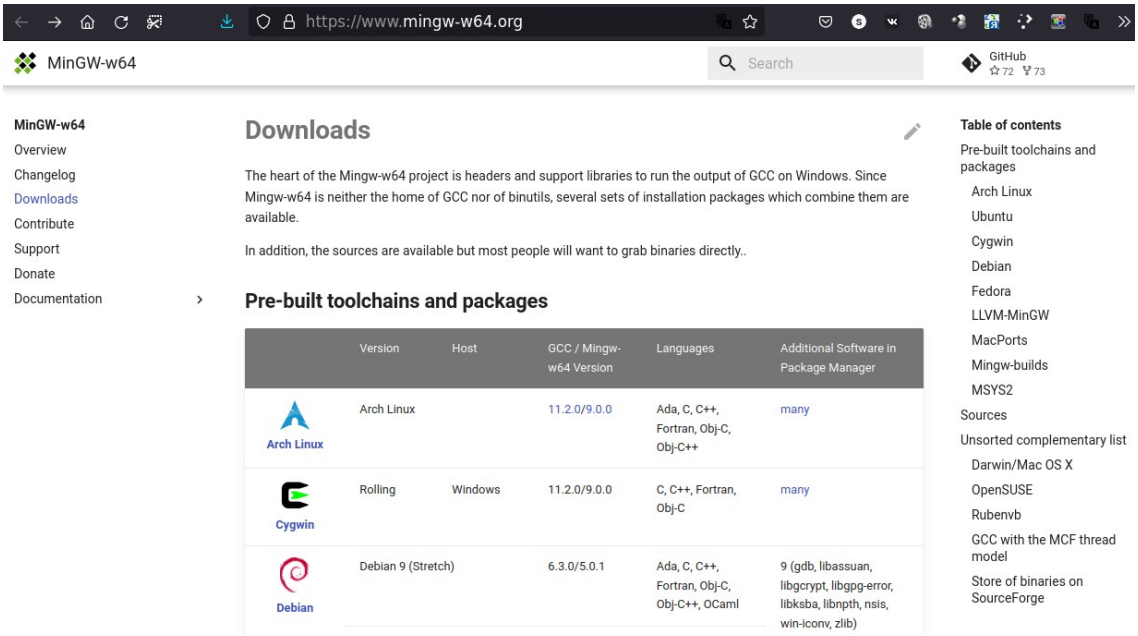

Рис. 1.1

Cygwin и MSYS2, помимо набора компиляторов, предоставляют среду Linux в окружении Windows. Как и Cygwin, MSYS2 состоит из терминала командной строки mintty, bash, систем контроля версий (таких как git и subversion), инструментов (таких как tar и awk), и даже систем сборки (таких как autotools). Все они основаны на модифицированной версии Cygwin. Несмотря на то, что некоторые из этих центральных частей основаны на Cygwin, основное внимание в MSYS2 уделяется предоставлению среды сборки для собственного программного обеспечения Windows, а части, использующие Cygwin, сведены к минимуму.

2. Установка Cygwin

Cygwin — это Linux-подобная среда для Windows. Она состоит из библиотеки, которая действует на уровне эмуляции и обеспечивает функциональность системных вызовов POSIX, и набора инструментов, которые обеспечивают интерфейс Linux. После установки Cygwin пользователи получают доступ ко многим стандартным утилитам UNIX. Их можно использовать в одной из предоставленных оболочек, таких, как bash, или из командной строки Windows. Кроме того, программисты могут писать приложения с графическим интерфейсом, которые используют стандартный Microsoft Win API

1. Перейдите на сайт Cygwin: https://cygwin.com/ и кликните на ссылке Install Cygwin в боковом меню (Рис. 2.1). Скачайте файл setup-x86_64.exe для установки или обновления среды Cygwin. Запустите установщик среды после загрузки (Рис. 2.2)

Рис. 2.1

2. Существует несколько вариантов загрузки пакетов. Выбираем «Install from Internet» (нам все пакеты не нужны) (Рис. 2.3)
3. Выбираем директорию для загрузки (оставляем по умолчанию — C:\cygwin64) и устанавливаем переключатель «All Users» (Рис. 2.4)
4. Также указываем папку для локального хранения загруженных пакетов (используем по умолчанию) (Рис. 2.5)
5. Выбираем способ выхода в сеть (Рис. 2.6)

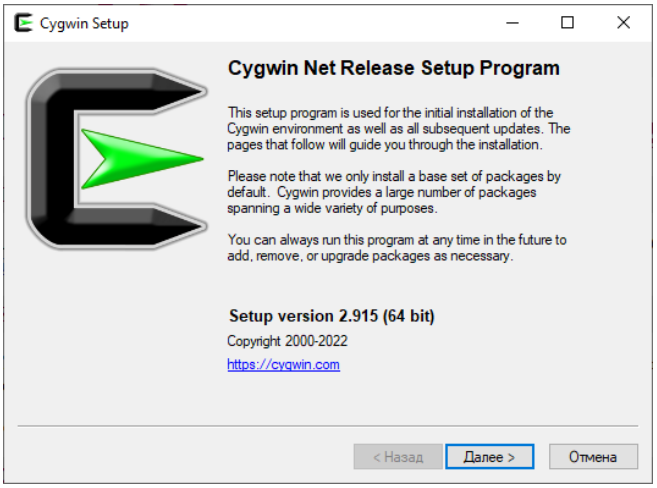

Рис. 2.2

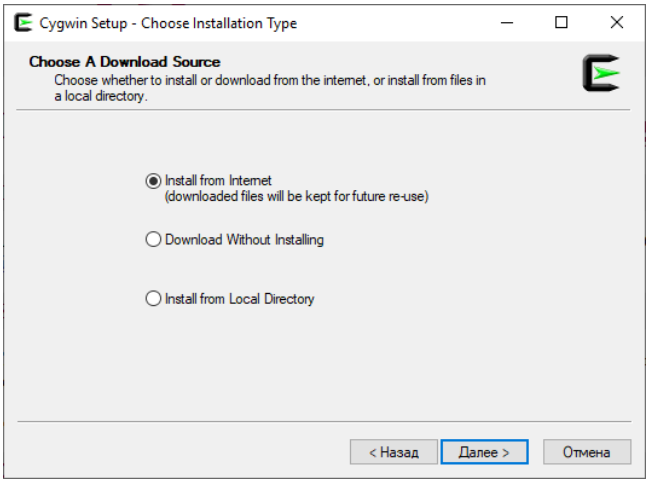

Рис. 2.3

6. Далее следует выбор хранилища (решать вам) (Рис. 2.7)
7. Теперь самая важная часть — выбираем устанавливаемые пакеты. Выберите в раскрывающемся списке View > Category (Рис. 2.8). Найдите категорию Devel («Разработка»). Напротив названия пакетов mingw64-x86_64-g++ и clang, в колонке «New» нажмите на Skip. Появятся версии этих пакетов. Это означает, что вы выбрали эти пакеты для установки. (Устанавливать флажки в колонке Src не нужно!).

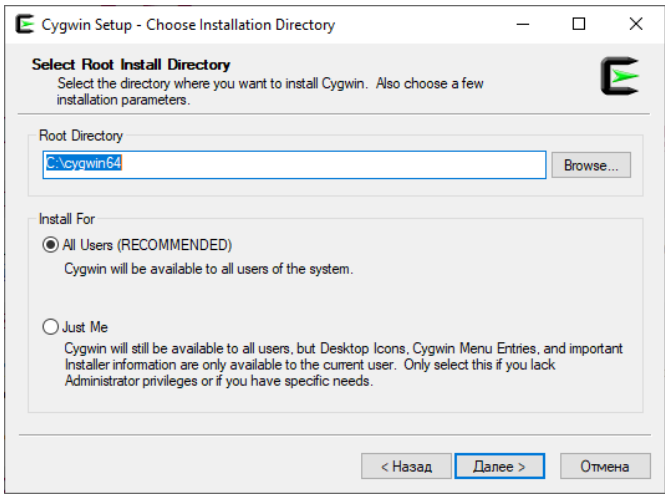

Рис. 2.4

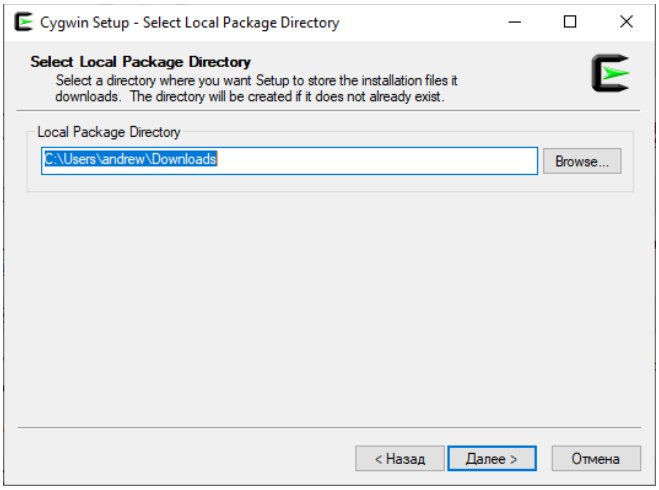

Рис. 2.5

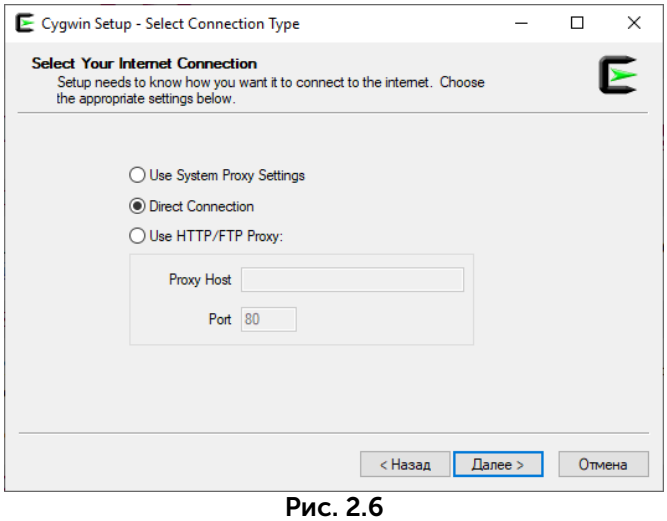

Рис. 2.6

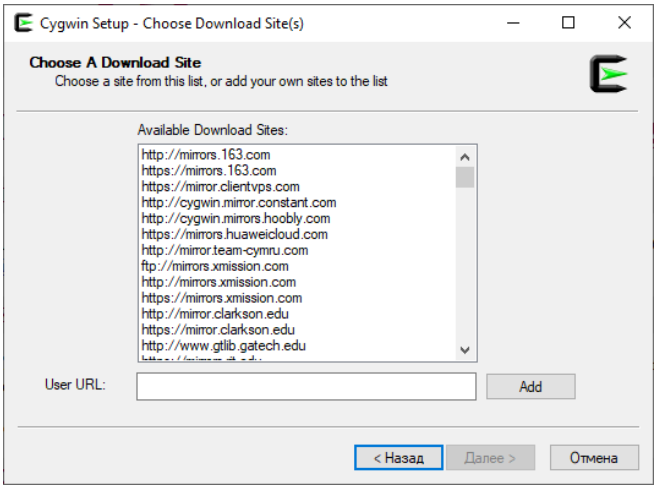

Рис. 2.7

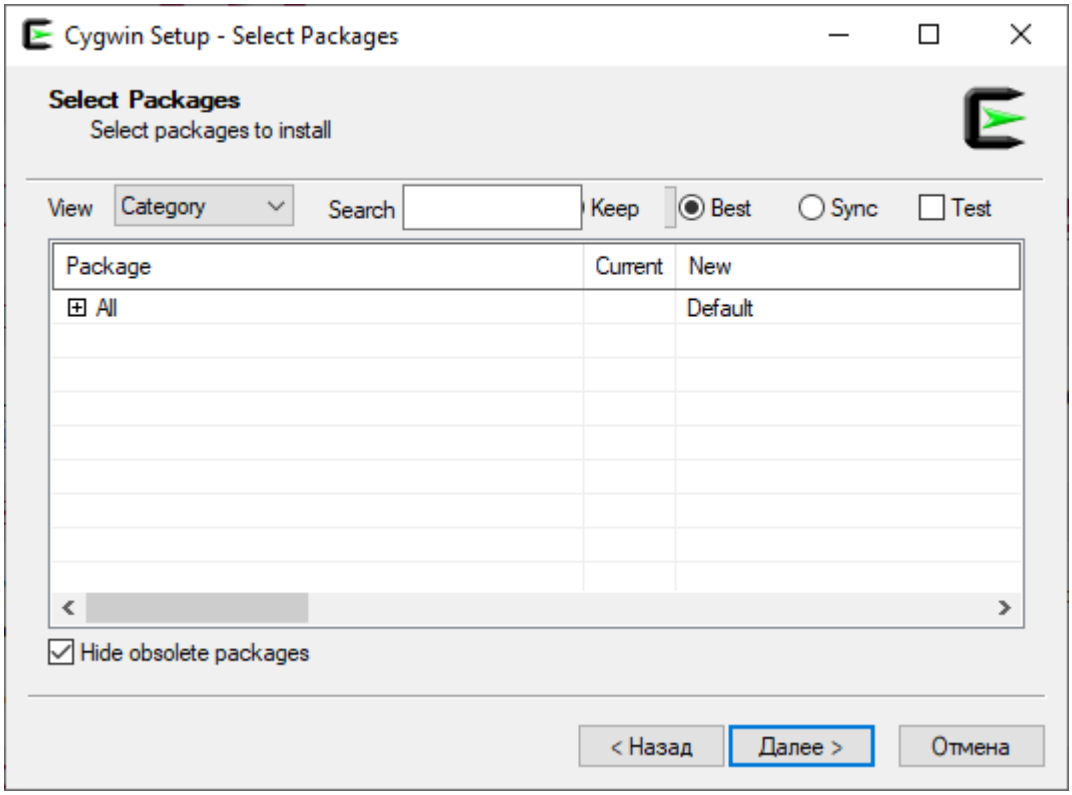
Cygwin Setup - Select Packages
Select Packages
Select packages to install
View
Category
Search
Keep
Best
Sync
Test
Package
Current
New
All
Default
Hide obsolete packages
< Назад
Далее >
Отмена

Рис. 2.8

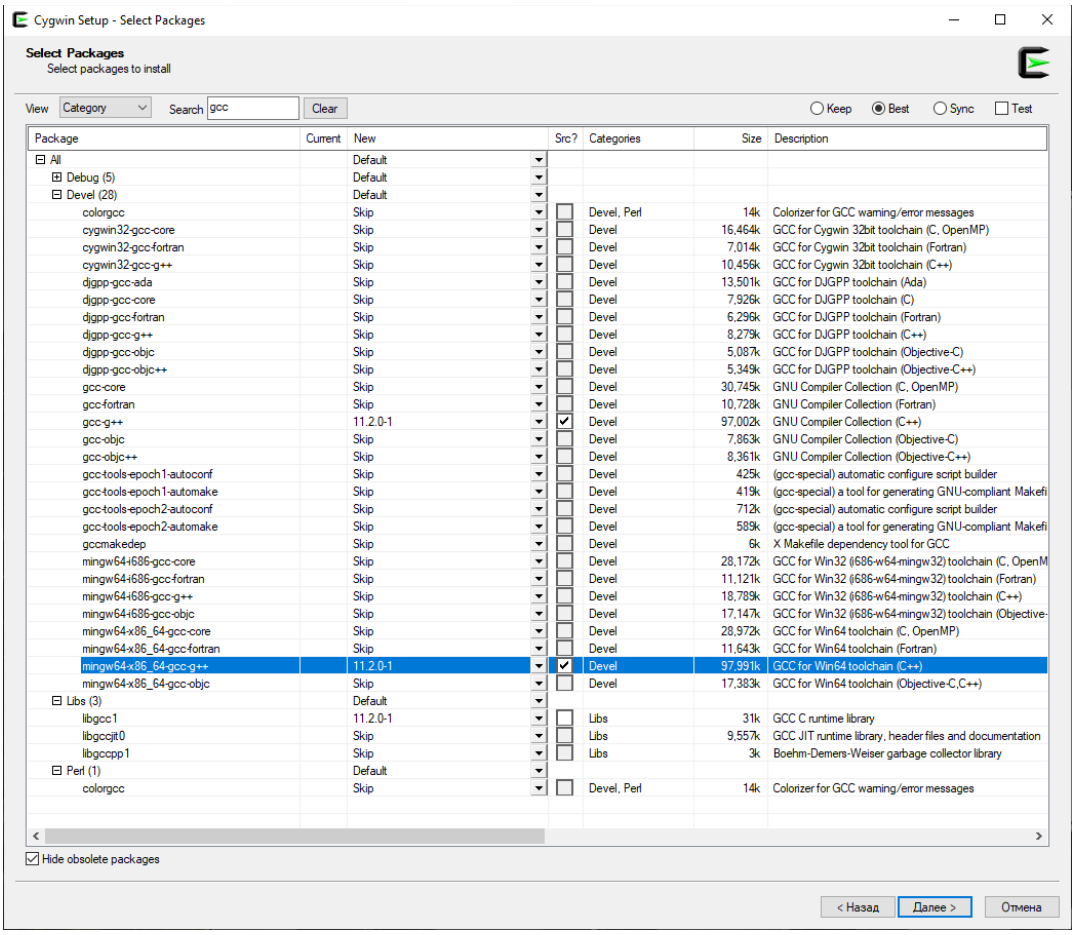
Cygwin Setup - Select Packages
Select Packages
Select packages to install
View
Category
Search
gcc
Clear
Keep
Best
Sync
Test
Package
Current
New
Src?
Categories
Size
Description
All
Default
Debug (5)
Default
Devel (28)
Default
colorgcc
Skip
Devel, Perl
14k
Colorizer for GCC warning/error messages
cygwin32-gcc-core
Skip
Devel
16,464k
GCC for Cygwin 32bit toolchain (C, OpenMP)
cygwin32-gcc-fortran
Skip
Devel
7,014k
GCC for Cygwin 32bit toolchain (Fortran)
cygwin32-gcc-g++
Skip
Devel
10,456k
GCC for Cygwin 32bit toolchain (C++)
djgpp-gcc-ada
Skip
Devel
13,501k
GCC for DJGPP toolchain (Ada)
djgpp-gcc-core
Skip
Devel
7,926k
GCC for DJGPP toolchain (C)
djgpp-gcc-fortran
Skip
Devel
6,296k
GCC for DJGPP toolchain (Fortran)
djgpp-gcc-g++
Skip
Devel
8,279k
GCC for DJGPP toolchain (C++)
djgpp-gcc-objc
Skip
Devel
5,087k
GCC for DJGPP toolchain (Objective-C)
djgpp-gcc-objc++
Skip
Devel
5,349k
GCC for DJGPP toolchain (Objective-C++)
gcc-core
Skip
Devel
30,745k
GNU Compiler Collection (C, OpenMP)
gcc-fortran
Skip
Devel
10,728k
GNU Compiler Collection (Fortran)
gcc-g++
11.2.0-1
Devel
97,002k
GNU Compiler Collection (C++)
gcc-objc
Skip
Devel
7,863k
GNU Compiler Collection (Objective-C)
gcc-objc++
Skip
Devel
8,361k
GNU Compiler Collection (Objective-C++)
gcc-tools-epoch1-autoconf
Skip
Devel
425k
(gcc-special) automatic configure script builder
gcc-tools-epoch1-automake
Skip
Devel
419k
(gcc-special) a tool for generating GNU-compliant Makefi
gcc-tools-epoch2-autoconf
Skip
Devel
712k
(gcc-special) automatic configure script builder
gcc-tools-epoch2-automake
Skip
Devel
589k
(gcc-special) a tool for generating GNU-compliant Makefi
gccmakedep
Skip
Devel
6k
X Makefile dependency tool for GCC
mingw64-i686-gcc-core
Skip
Devel
28,172k
GCC for Win32 (i686-w64-mingw32) toolchain (C, OpenM
mingw64-i686-gcc-fortran
Skip
Devel
11,121k
GCC for Win32 (i686-w64-mingw32) toolchain (Fortran)
mingw64-i686-gcc-g++
Skip
Devel
18,789k
GCC for Win32 (i686-w64-mingw32) toolchain (C++)
mingw64-i686-gcc-objc
Skip
Devel
17,147k
GCC for Win32 (i686-w64-mingw32) toolchain (Objective-
mingw64-x86_64-gcc-core
Skip
Devel
28,972k
GCC for Win64 toolchain (C, OpenMP)
mingw64-x86_64-gcc-fortran
Skip
Devel
11,643k
GCC for Win64 toolchain (Fortran)
mingw64-x86_64-gcc-g++
11.2.0-1
Devel
97,991k
GCC for Win64 toolchain (C++)
mingw64-x86_64-gcc-objc
Skip
Devel
17,383k
GCC for Win64 toolchain (Objective-C,C++)
Libs (3)
Default
libgcc1
11.2.0-1
Libs
31k
GCC C runtime library
libgccjit0
Skip
Libs
9,557k
GCC JIT runtime library, header files and documentation
libgccpp1
Skip
Libs
3k
Boehm-Demers-Weiser garbage collector library
Perl (1)
Default
colorgcc
Skip
Devel, Perl
14k
Colorizer for GCC warning/error messages
Hide obsolete packages
< Назад
Далее >
Отмена

Рис. 2.9

Остальные (зависимые пакеты) будут установлены автоматически. В этом же разделе выберите для установки следующие пакеты make, cmake, gdb, pkgconf и пакеты поддержки Qt5: mingw64-x86_64-qt5* (Если вам нужна поддержка Qt6, то вам следует устанавливать MSYS2, так как Cygwin, на момент написания методички, поддержки Qt6 не имеет). В следующем окне можно увидеть список всех устанавливаемых пакетов:

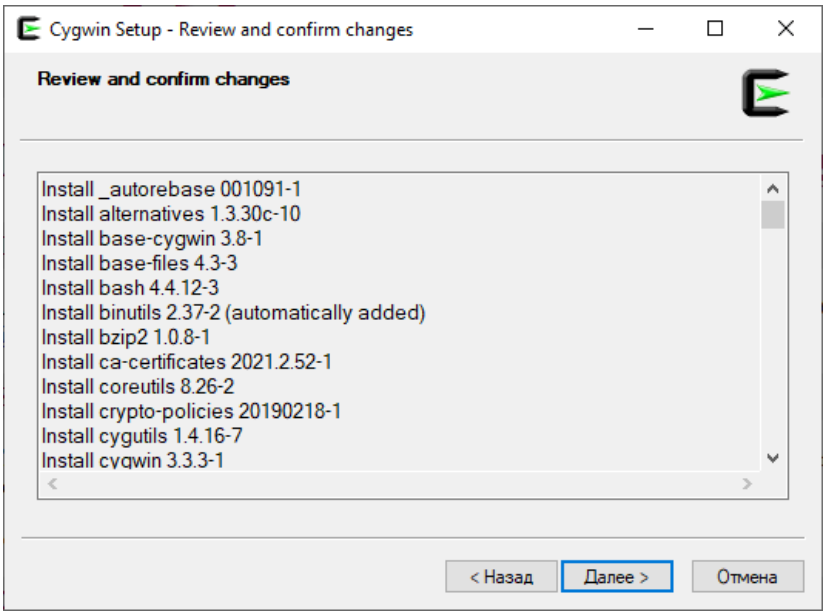

Рис. 2.10

Далее > Начинается установка пакетов.

8. После завершения установки Cygwin, устанавливаем путь к компилятору(-ам) в перемеренной среды. Для этого вызовите контекстное меню на ярлыке «Этот компьютер» и выберите пункт «Свойства». Откроется окно:

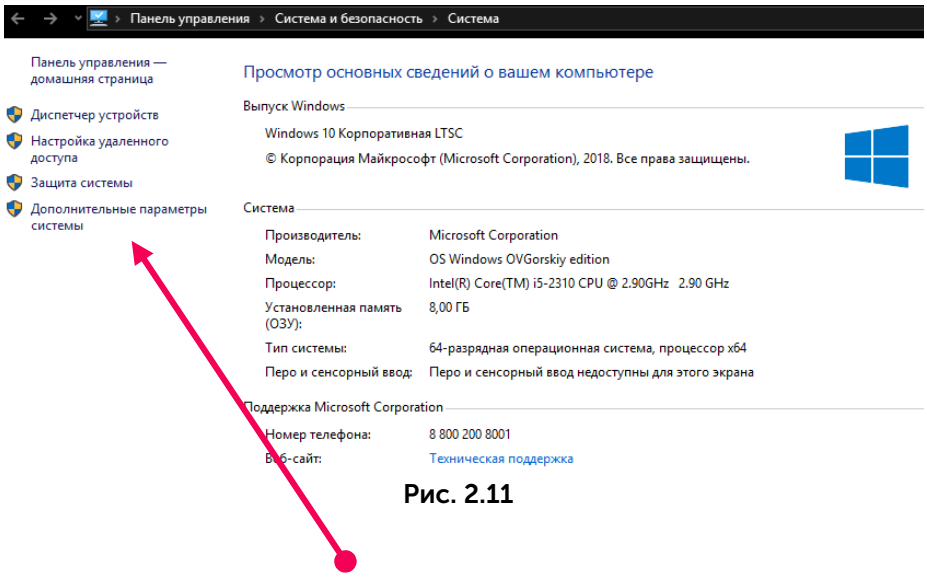

Рис. 2.11

9. Выберите в меню слева пункт «Дополнительные параметры системы». В следующем окне («Свойства системы»): Дополнительно > Переменные среды (Рис. 2.12)

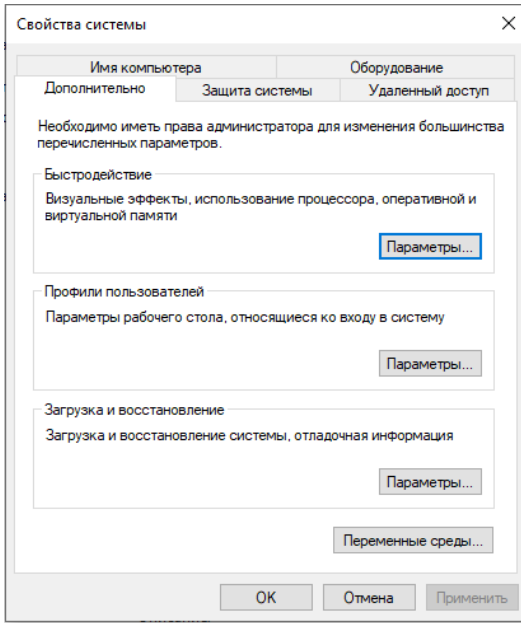

Рис. 2.12

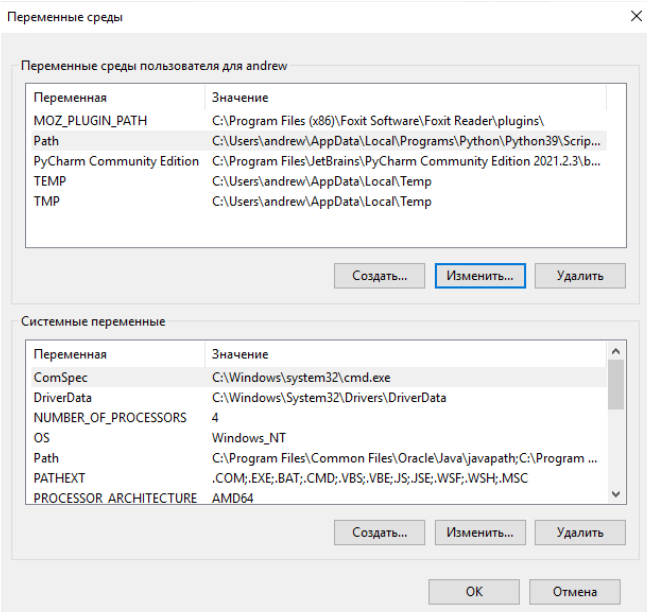

Рис. 2.13

10. В следующем окне изменяем переменную Path (выделяем, нажимаем кнопку «Изменить»). В следующем окне (Рис. 2.14) нажимаем «Создать» и, в появившемся поле, записываем пути к компиляторам: C:\cygwin64\bin
    Сохраняем и закрываем все окна.

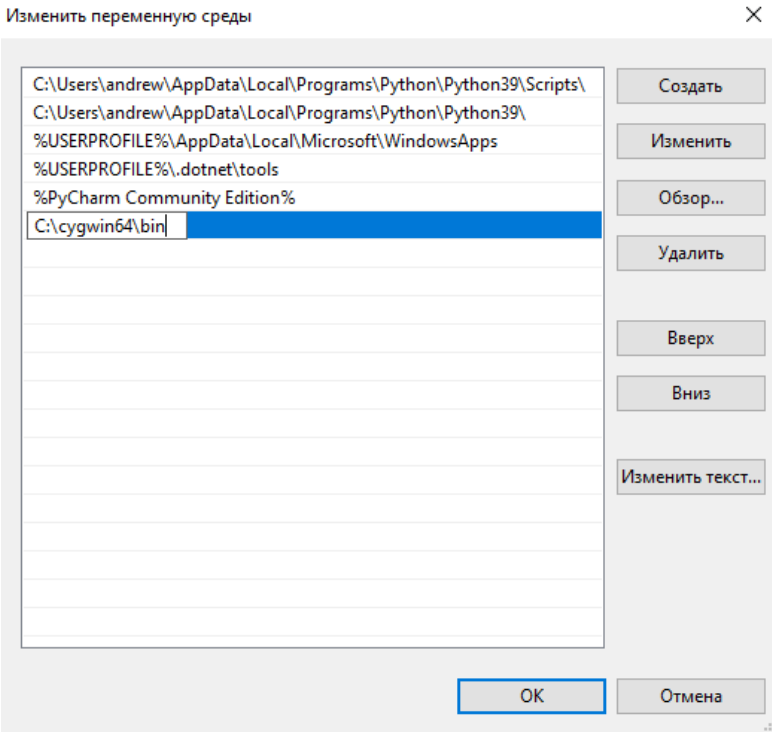

Рис. 2.14

11. На этом установка Cygwin завершена!

3. Установка MSYS2

MSYS (Minimal SYStem) представляет собой легковесную Unix-подобную среду внутри ОС Windows созданную на основе проектов Cygwin (уровень совместимости POSIX) и MinGW-w64 с целью лучшей совместимости с собственным программным обеспечением Windows. Он содержит оболочку bash, менеджер пакетов Pacman и множество инструментов для создания собственных приложений Windows с использованием наборов инструментов MinGW-w64.

1. Установка MSYS2 отличается от установки Cygwin, описанной выше. Перейдете на сайт проекта http://www.msys2.org/:

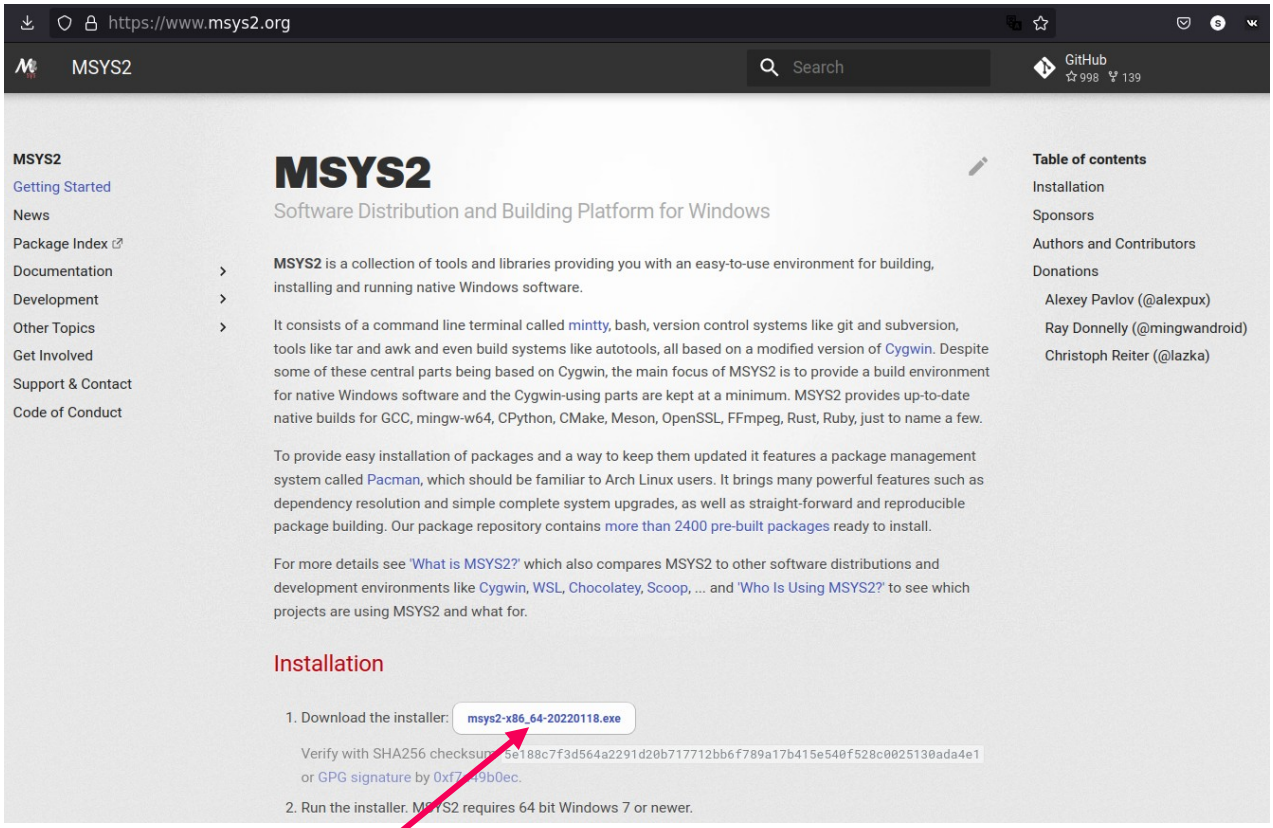

Рис. 3.1

2. Кликните на ссылке для загрузки ядра системы, содержащего менеджер пакетов Pacman. После загрузки, запустите установку программы. Выберите директорию для установки MSYS2 (по умолчанию — это C:\msys64) (Рис. 3.2). По окончании загрузки запустите среду MSYS2. Это произойдет автоматически, если не будет снят флажок «Запустить MSYS2...» (Рис. 3.3). Нажимаем «Завершить».

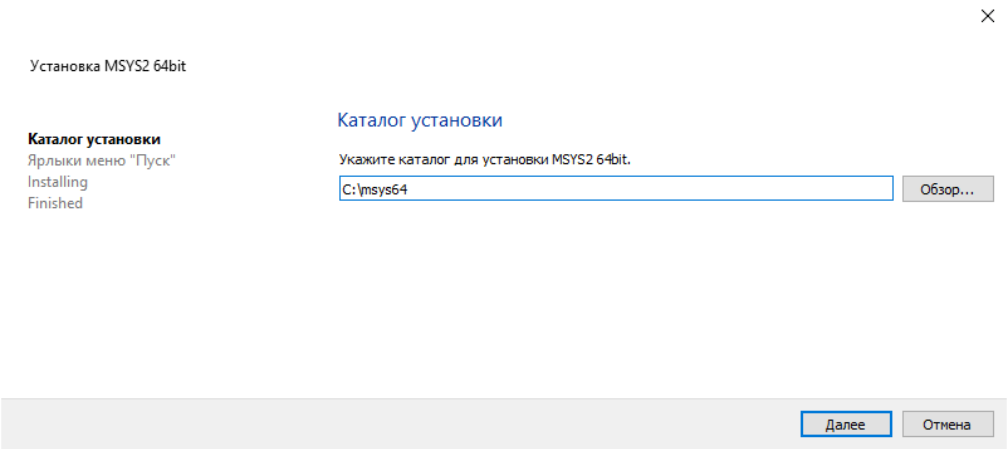

Рис. 3.2

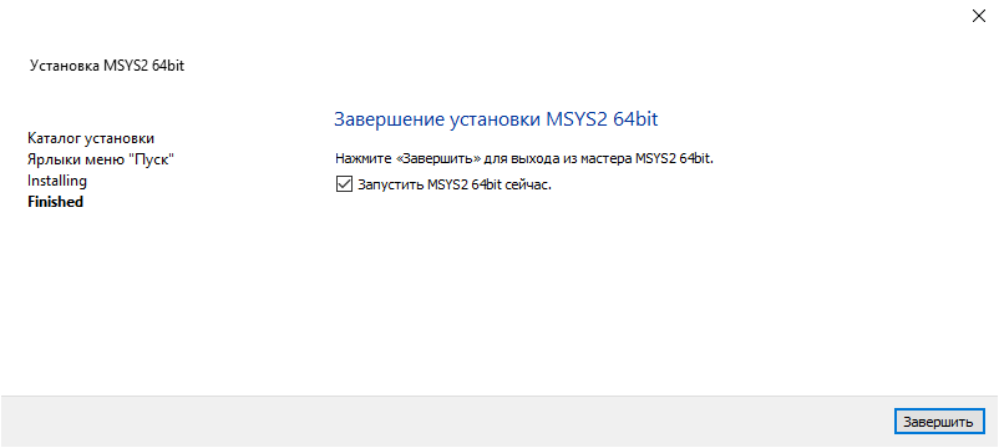

Рис. 3.3

3. Появится консольное окно с командной строкой. В оболочке (Рис. 3.4) последовательно выполните следующие команды для обновления системы:

```
pacman -Syu
pacman -Su
```

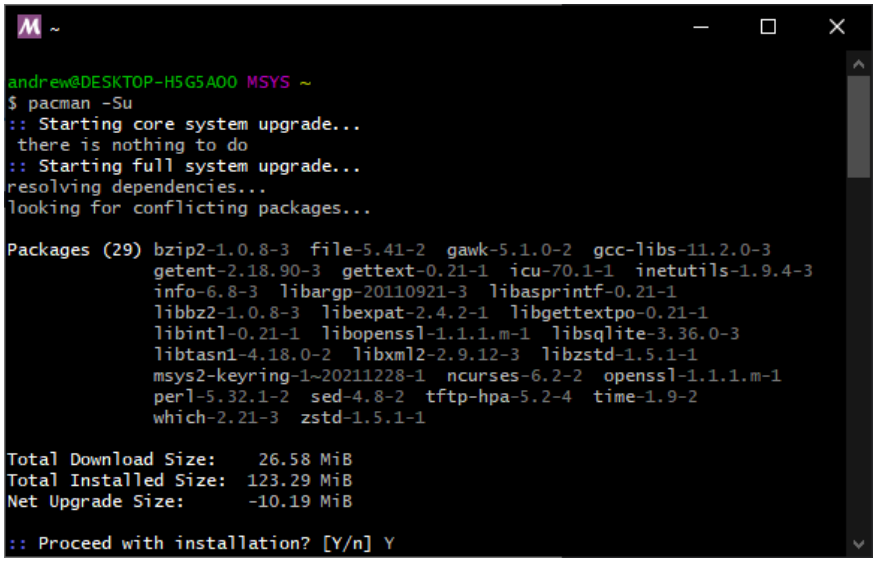

Рис. 3.4

4. После обновления системы, начинаем устанавливать необходимые пакеты. Полный список пакетов содержащихся в среде MSYS2, и распределенных по группам, можно посмотреть здесь: https://packages.msys2.org/group/. Нам необходимо установить пакеты из этих групп:
   1. pacman -S mingw-w64-clang-x86_64-toolchain
   2. pacman -S mingw-w64-clang-x86_64-qt5 (если нужно)
   3. pacman -S mingw-w64-clang-x86_64-qt6
   4. pacman -S mingw-w64-x86_64-toolchain
   5. pacman -S mingw-w64-x86_64-qt5 (если нужно)
   6. pacman -S mingw-w64-x86_64-qt6
   7. pacman -S mingw-w64-clang-x86_64-cmake
   8. pacman -S mingw-w64-x86_64-cmake

   После ввода каждой из этих команд будет отображено содержимое группы (Рис. 3.5). Для всех групп, кроме mingw-w64-x86_64-toolchain, устанавливаем все пакеты (нажимаем Enter в приглашении ввода «Enter a selections...»). MinGW предоставляет набор нескольких компиляторов, но нам нужна поддержка только C/C++, поэтому в строке «Enter a selections...» вводим номер позиции — 3 (mingw-w64-x86_64-gcc). После проверки зависимостей будут отображены все устанавливаемые пакеты (зависимости система разрешает сама). Вводим «Y» и нажимаем Enter.

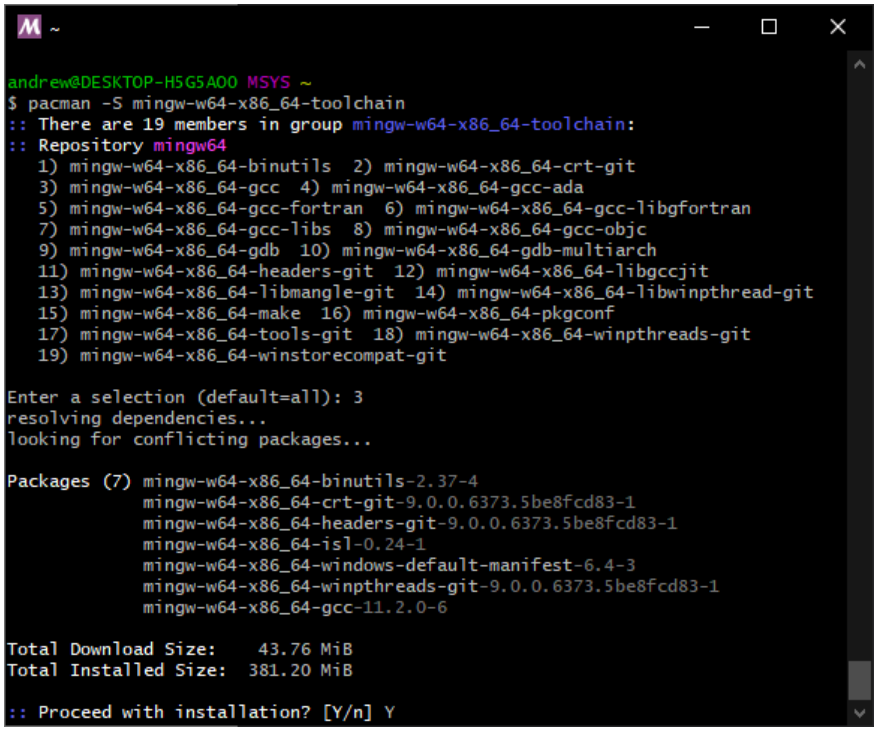

Рис. 3.5

5. Помимо пакета mingw-w64-x86_64-gcc, необходимо загрузить еще и эти пакеты:
   1. pacman -S mingw-w64-x86_64-gdb
   2. pacman -S make
   3. pacman -S mingw-w64-x86_64-cmake

4. pacman -S mingw-w64-x86_64-pkgconf

6. Если вам понадобятся другие пакеты, то вы всегда можете добавить их позже, запустив оболочку MSYS2 с командной строкой в меню «Пуск». Ненужные пакеты можно удалить. Используйте для этого следующие команды:
   Для установки:

```
pacman -S <name of the package>
```

   Для удаления:

```
pacman -R <name of the package>
```

7. На этом установка MSYS2 завершена! Пути к компиляторам (gcc и clang) нужно прописать в переменной среды так, как показано выше для среды Cygwin (см. стр. 6 - 7). Например: C:\msys64\mingw64\bin и C:\msys64\clang64\bin

4. Установка и настройка IDE Code::Blocks с компилятором GCC C++

1. Перейдите на сайт IDE Code::Blocks: https://www.codeblocks.org/ и выберите в меню раздел Downloads (Рис. 4.1).

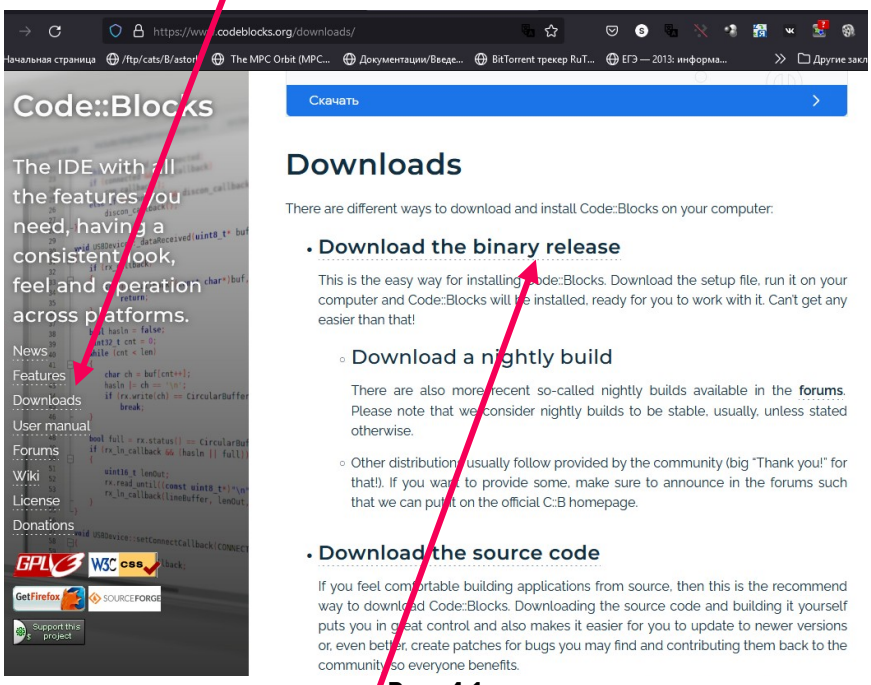

Рис. 4.1

2. Перейдите на страницу Download the binary release и выберите версию инсталлятора,

который не поставляется со встроенным компилятором GCC. На момент написания методички — это версии codeblocks-20.03-setup.exe или codeblocks-20.03-setup-nonadmin.exe. Первая включает Code::Blocks со всеми подключаемыми модулями, а вторая — поставляется для удобства пользователей, не имеющих прав администратора на своих компьютерах (Рис. 4.2).

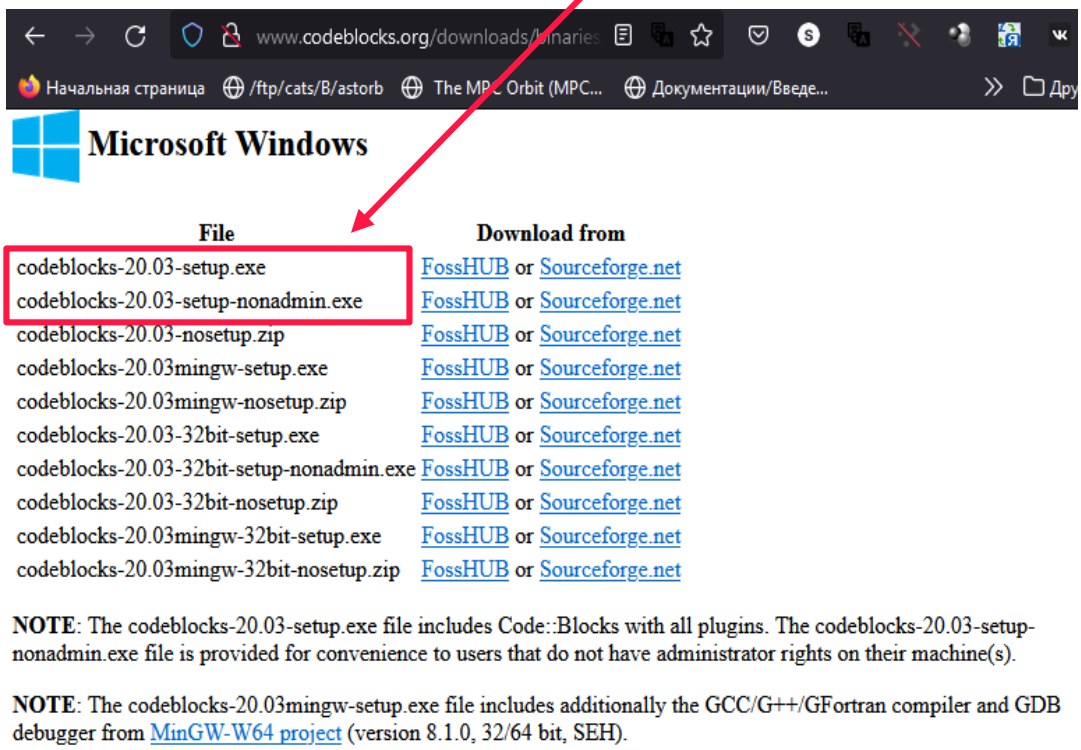

Рис. 4.2

3. После скачивания и установки, запустите программу и перейдите в меню Settings > Compiler... В появившемся окне Global compiler settings выберите вкладку Toolchain executables. В раскрывающемся списке выберите CygwinGCC и нажмите кнопку Auto-detect. Code::Blocks автоматически распознает программы установленные в директории компилятора, которые нужны для компиляции, линковки и отладки (Рис. 4.3). Обратите внимание, что в Cygwin для разных компиляторов используется одна и та же директория C:\cygwin64\bin. Поэтому, если нужен иной компилятор, например, Clang, то все позиции необходимо будет установить вручную, нажимая на кнопки с тремя точками (в MSYS2 компилятор Clang устанавливается в отдельную директорию). Кнопка с тремя точками рядом с кнопкой Auto-detect позволяет установить путь к компилятору вручную (если вы не устанавливали путь в переменной среды). Если вы намерены всегда использовать этот компилятор, то нажмите кнопку Set as default, для установки данного набора для компиляции по умолчанию.
4. Перейдете на вкладку Compiler settings (Рис. 4.4) и установите флажок используемого стандарта (флаг компилятора), например, -std=c++20 (на момент написания методички).

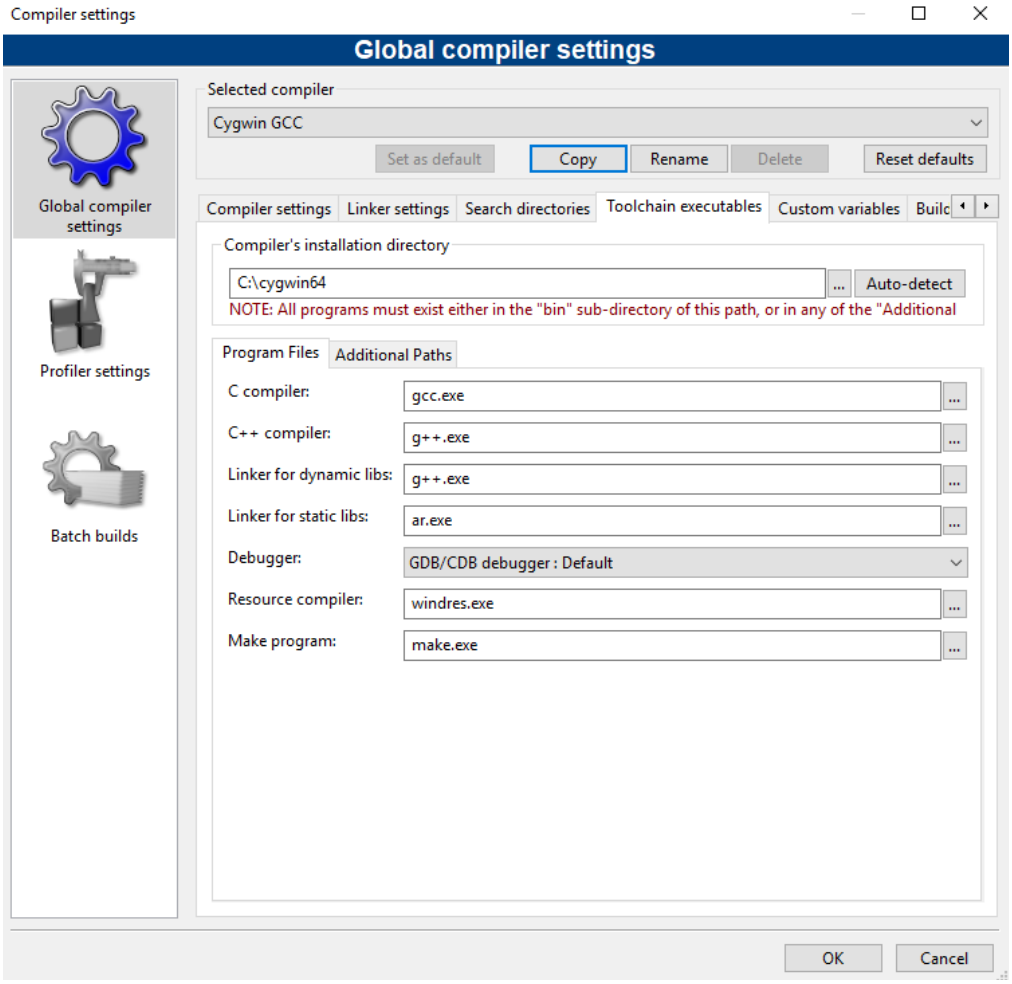
Compiler settings
Global compiler settings
Selected compiler
Cygwin GCC
Set as default
Copy
Rename
Delete
Reset defaults
Global compiler settings
Profiler settings
Batch builds
Compiler settings
Linker settings
Search directories
Toolchain executables
Custom variables
Compiler's installation directory
C:\cygwin64
Auto-detect
NOTE: All programs must exist either in the "bin" sub-directory of this path, or in any of the "Additional
Program Files
Additional Paths
C compiler:
gcc.exe
C++ compiler:
g++.exe
Linker for dynamic libs:
g++.exe
Linker for static libs:
ar.exe
Debugger:
GDB/CDB debugger : Default
Resource compiler:
windres.exe
Make program:
make.exe
OK
Cancel

Рис. 4.3

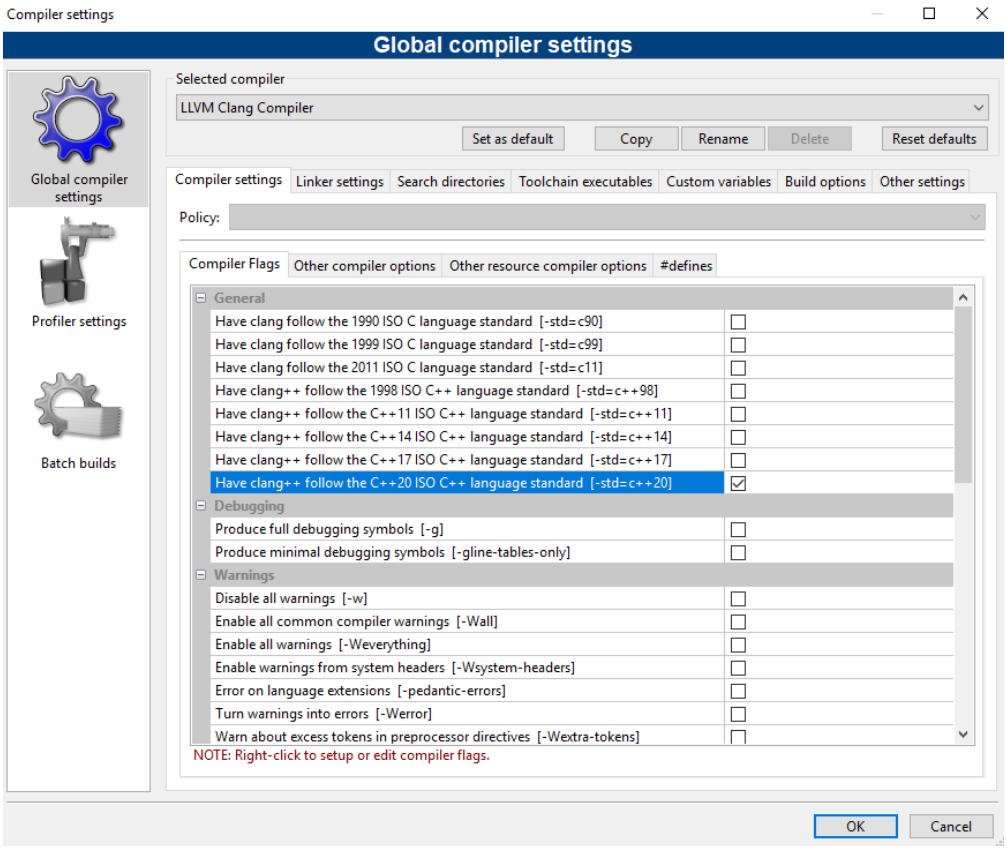
Compiler settings
Global compiler settings
Selected compiler
LLVM Clang Compiler
Set as default
Copy
Rename
Delete
Reset defaults
Global compiler settings
Profiler settings
Batch builds
Compiler settings
Linker settings
Search directories
Toolchain executables
Custom variables
Build options
Other settings
Policy:
Compiler Flags
Other compiler options
Other resource compiler options
#defines
General
Have clang follow the 1990 ISO C language standard [-std=c90]
Have clang follow the 1999 ISO C language standard [-std=c99]
Have clang follow the 2011 ISO C language standard [-std=c11]
Have clang++ follow the 1998 ISO C++ language standard [-std=c++98]
Have clang++ follow the C++11 ISO C++ language standard [-std=c++11]
Have clang++ follow the C++14 ISO C++ language standard [-std=c++14]
Have clang++ follow the C++17 ISO C++ language standard [-std=c++17]
Have clang++ follow the C++20 ISO C++ language standard [-std=c++20]
Debugging
Produce full debugging symbols [-g]
Produce minimal debugging symbols [-gline-tables-only]
Warnings
Disable all warnings [-w]
Enable all common compiler warnings [-Wall]
Enable all warnings [-Weverything]
Enable warnings from system headers [-Wsystem-headers]
Error on language extensions [-pedantic-errors]
Turn warnings into errors [-Werror]
Warn about excess tokens in preprocessor directives [-Wextra-tokens]
NOTE: Right-click to setup or edit compiler flags.
OK
Cancel

Рис. 4.4

Ранее для этого использовалась отдельная вкладка, но в новой версии появилась возможность настраивать эти строки правой кнопкой мыши, из контекстного меню (Рис. 4.5).

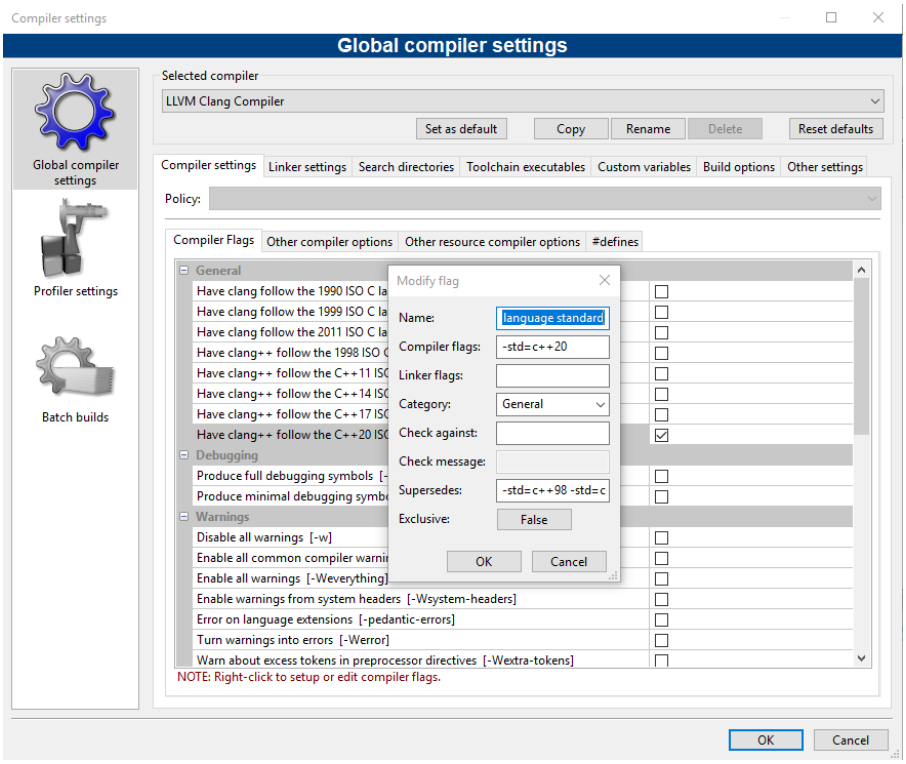

Рис. 4.5

5. Сохраните настройки и перейдите к окну программы Start here (Рис. 4.6). Нажмите Create a new project (Создать новый проект).

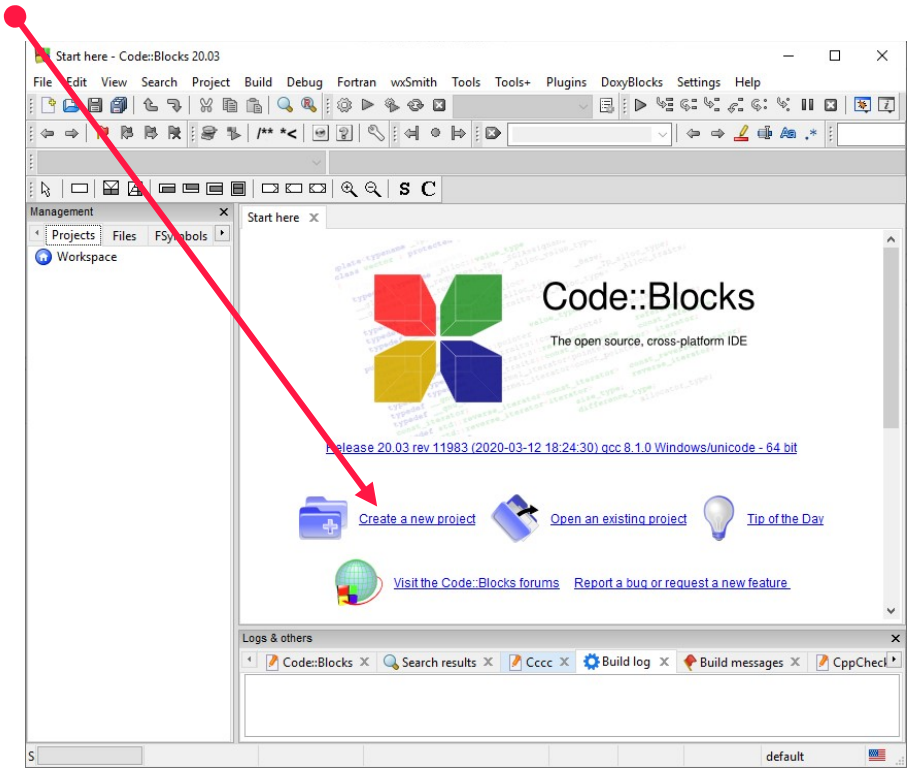

Рис. 4.6

В окне New from template выберите категорию проекта: Console application (Рис. 4.7).

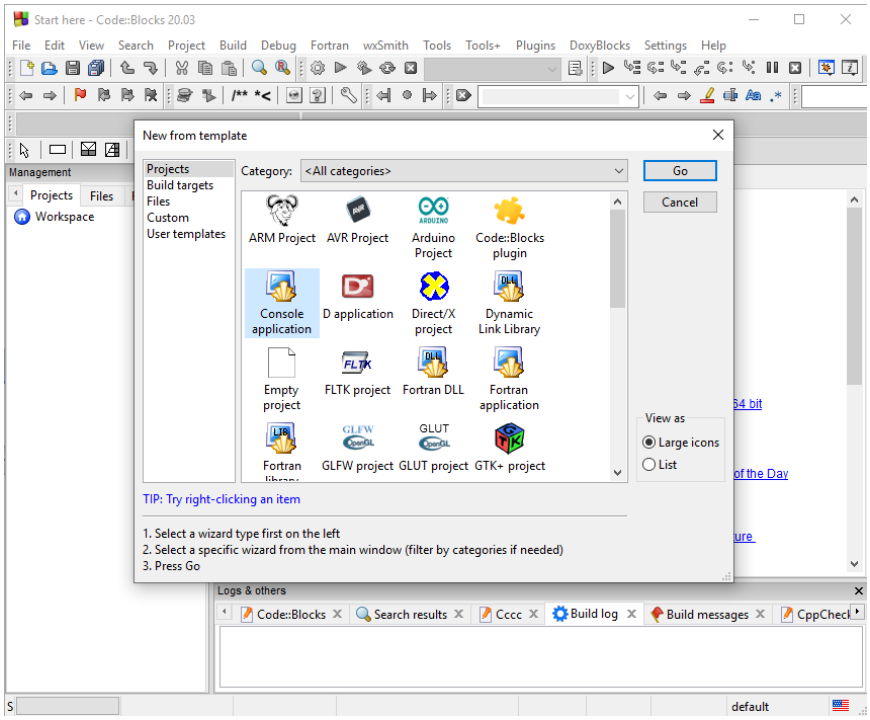

Рис. 4.7

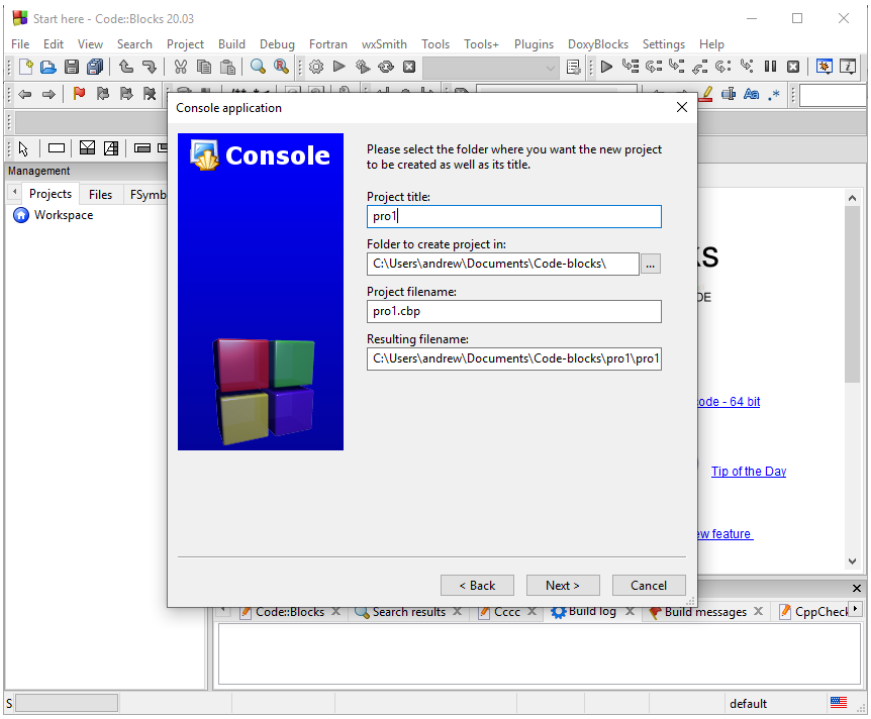

Рис. 4.8

Далее указываем название проекта и (обязательно, если это первый проект!) папку для сохранения проектов. Это должна быть директория в личном пользовательском пространстве (Рис. 4.8). В последнем окне настраиваются директории Debug и Release (оставляем настройки по умолчанию). Закончив настройки проекта, нажимаем Finish. На этом интеграция компилятора MinGW (GCC) Cygwin в среде Code::Blocks завершена!

5. Настройка Qt Creator с компилятором GCC C++

1. Запустите программу Qt Creator и перейдите в меню Инструменты > Параметры. Выбираем в боковом меню «Комплекты». Комплекты — это инструменты для сборки и компиляции проектов. Комплекты создаются под различные задачи. Комплектов может быть сколь угодно много. Создадим свой комплект под именем MSYS2 (в MSYS2 осуществляется поддержка Qt6). Нажмите на кнопку «Добавить» и заполните все необходимые поля (Рис. 5.1).

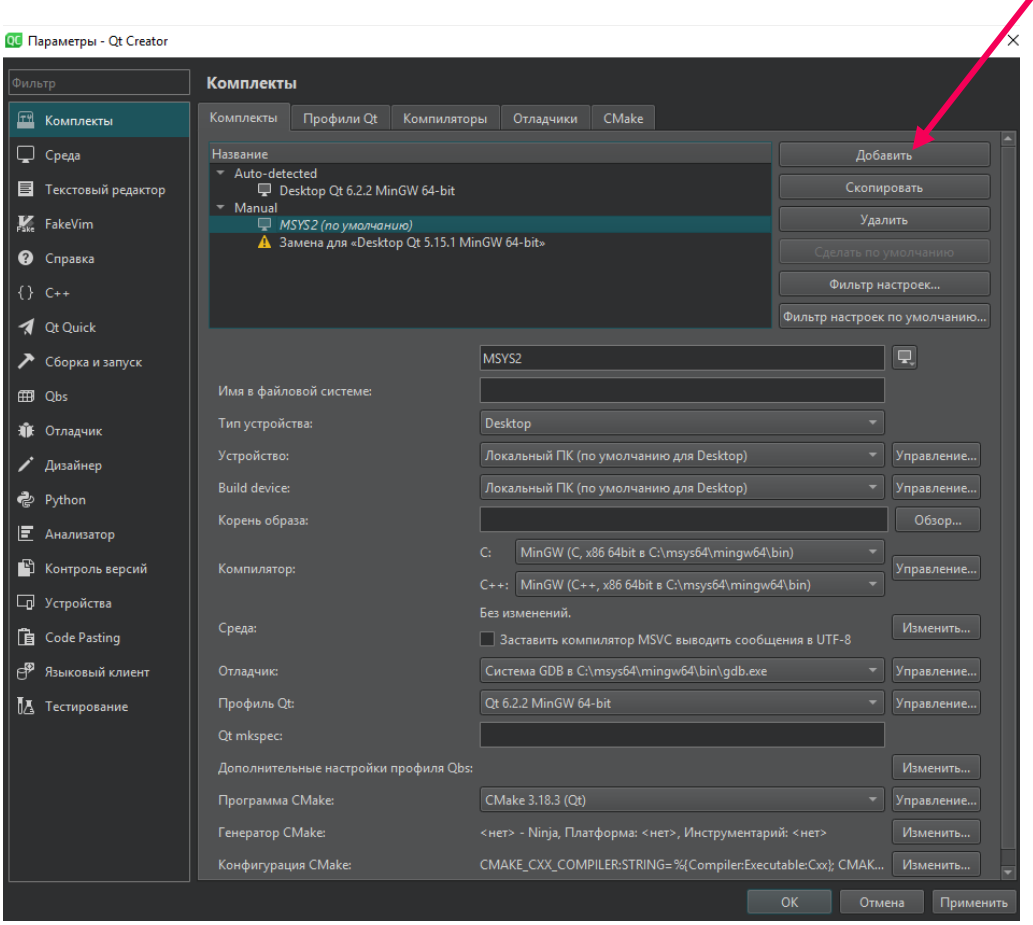

Рис. 5.1

Обратите внимание, что если комплект содержит ошибки, то рядом с именем комплекта появится восклицательный знак. Все ошибки необходимо исправить, иначе сборка проекта может быть не выполнена.

2. Введите имя комплекта (например, Desktop-MSYS2), тип устройства. Корень образа (sysroot — корневая директория для поиска заголовков и библиотек) не указываем, оставляем по умолчанию системные пути (этот путь будет необходим для создания кроссплатформенного приложения). Указываем, где находятся компиляторы языков C и C++. Указываем отладчик GDB системы MSYS2. Выбираем соответствующий

профиль Qt и программу CMake. На вкладках перечисленных компонентов (в верхней части окна) можно изменить настройки для каждого из них. Если этот комплект вы намерены использовать постоянно, то сделайте его проектом по умолчанию, нажав «Сделать по умолчанию». Сохраните изменения.

3. Выберите в меню (слева) «Сборка и запуск». Переключатель в положении «Каталог». Укажите директорию для сохранения проекта (доступную для пользователя).
4. Создадим первое приложение в Qt Creator. В основном окне Начало > Проекты > +Создать (Рис. 5.2)

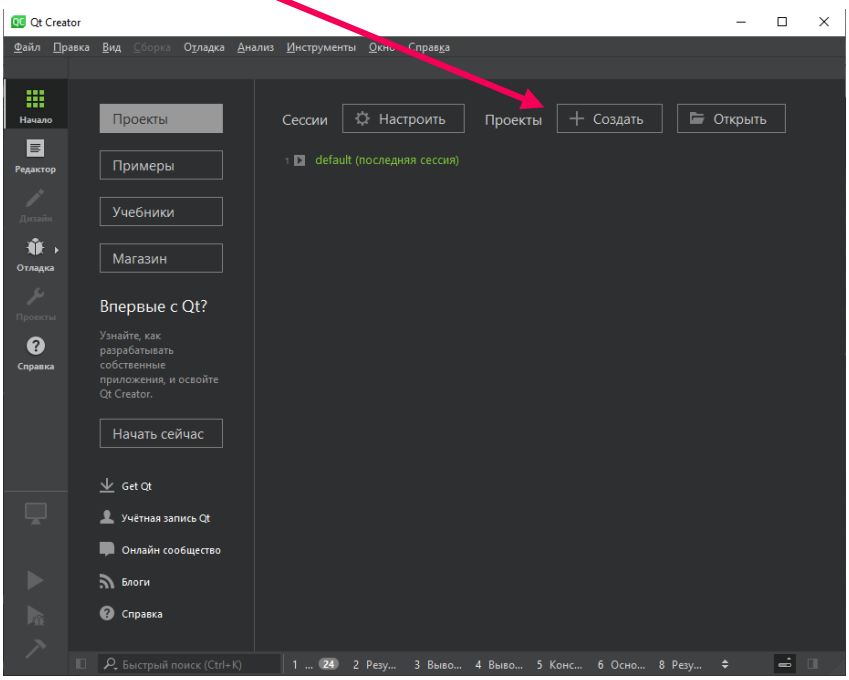

Рис. 5.2

5. В окне настройки Нового проекта (Рис. 5.3) выберите Проект без Qt > Приложение на языке C++

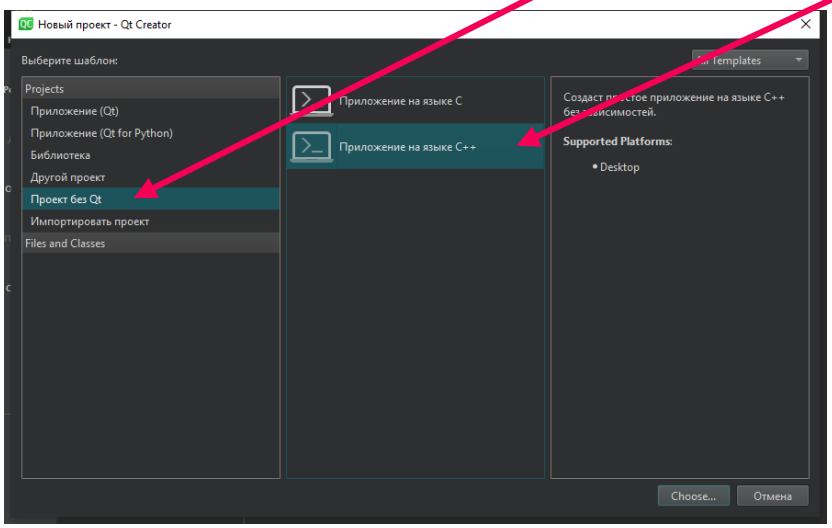

Рис. 5.3

6. Нажимаем Choose… и перейдите к следующему окну. Введите название проекта. Нажимаем Далее (Рис. 5.4).

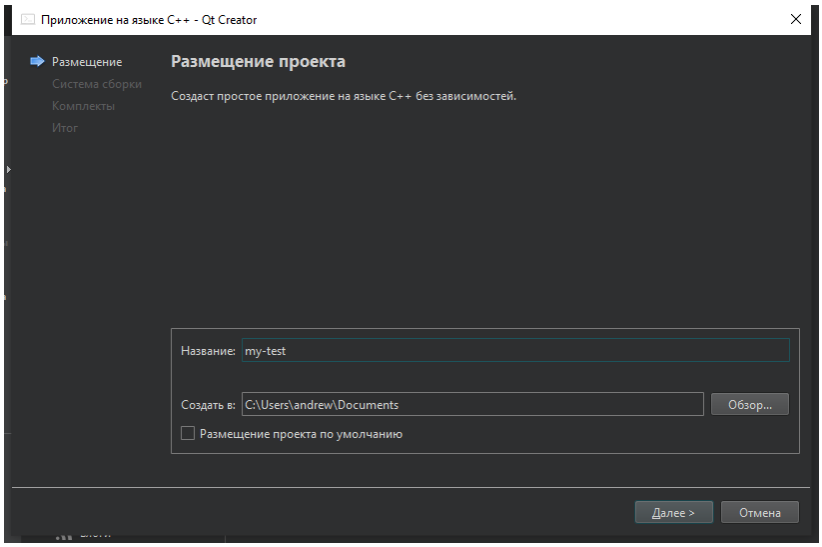

Рис. 5.4

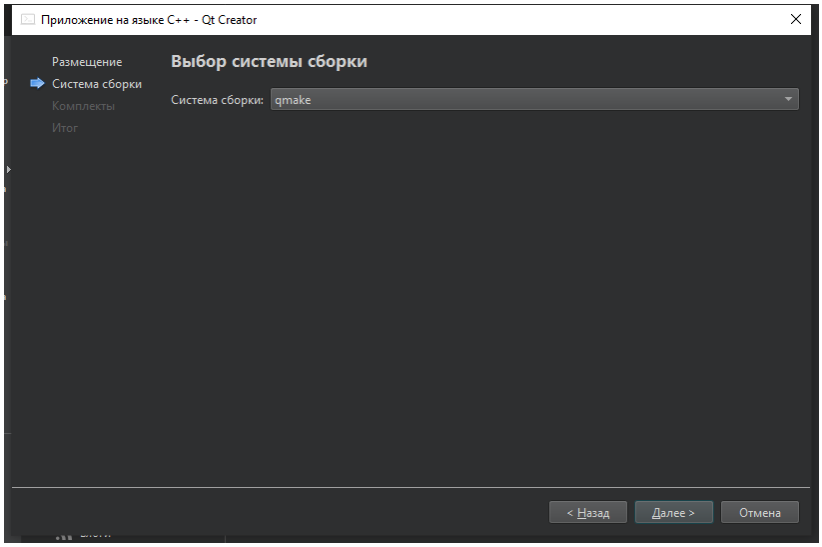

Рис. 5.5

7. Оставляем систему сборки qmake (по умолчанию). Далее выбираем комплект для сборки и компиляции, который мы установили ранее (Рис. 5.6):

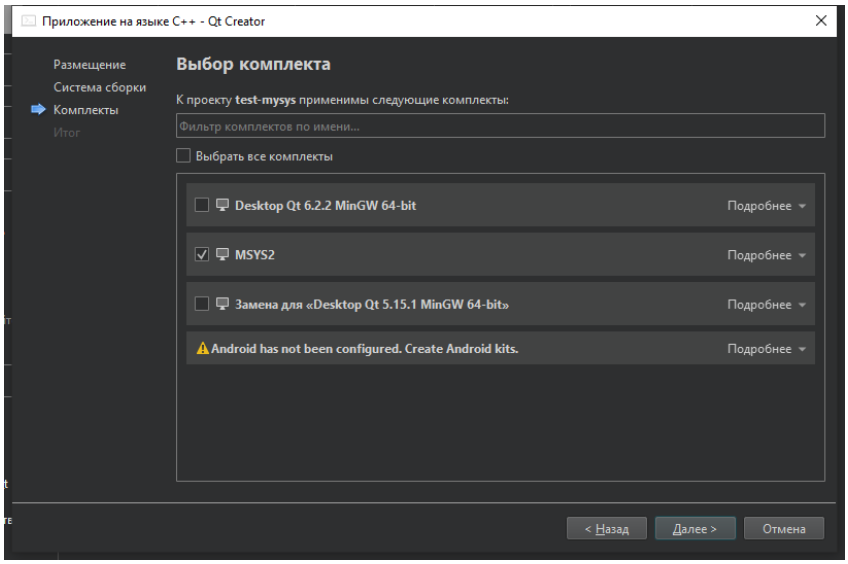

Рис. 5.6

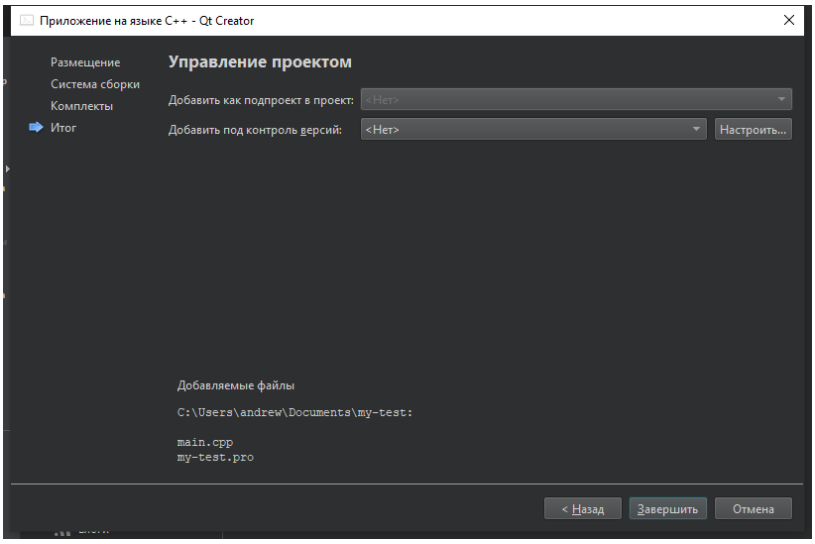

Рис. 5.7

8. Завершаем настройку (добавление контроля версий будет актуальным в больших проектах) и переходим к окну с проектом. В меню (слева) нажмите на кнопку «Проекты». Выберите: Сборка и запуск > Запуск (Рис. 5.8). Перейдите к секции «Запуск» и установите флажок «Запускать в терминале».

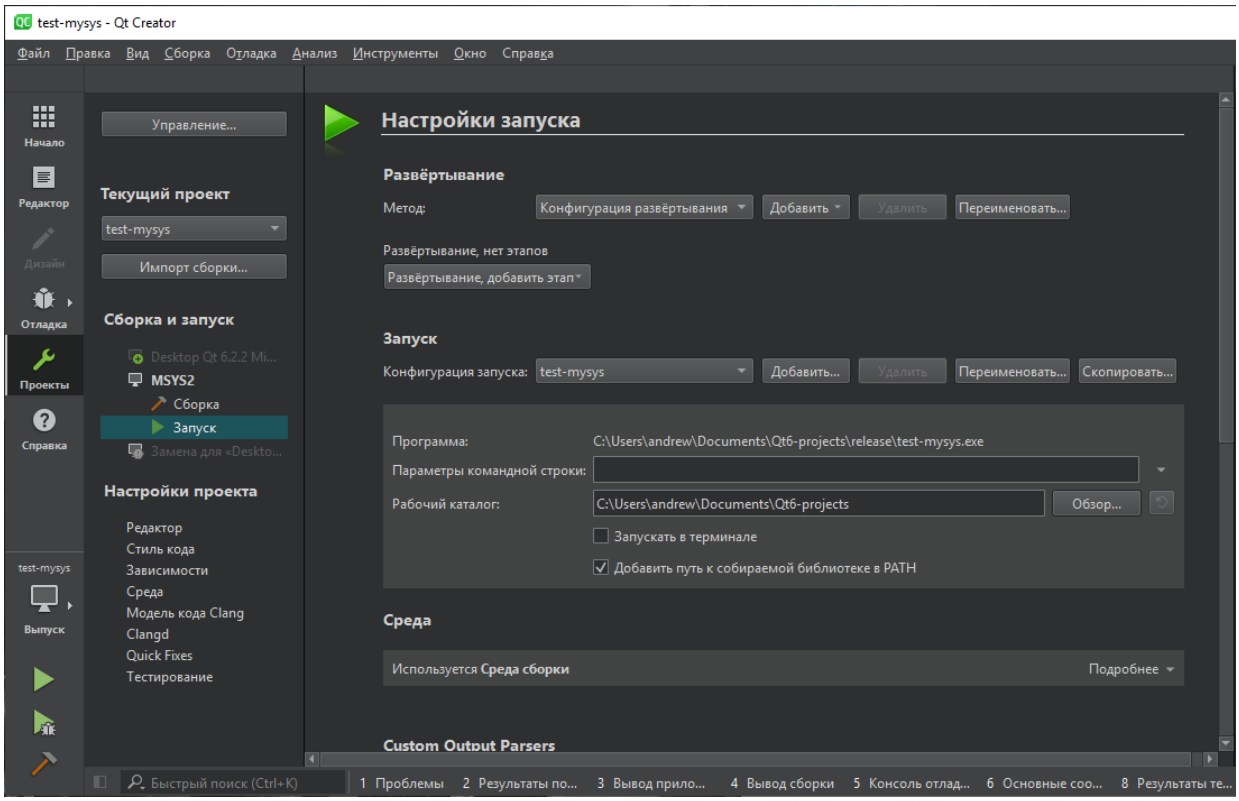

Рис. 5.8

9. На этом интеграция компилятора MinGW MSYS2 в среде Qt Creator завершена!

6. Настройка Qt Creator и Code::Blocks с компилятором Clang

Настройки компилятора Clang в Code::Blocks и Qt Creator выполняются аналогично настройке компилятора GCC C++. Список программ необходимых для компиляции, сборки и отладки вы можете видеть на рисунках ниже.

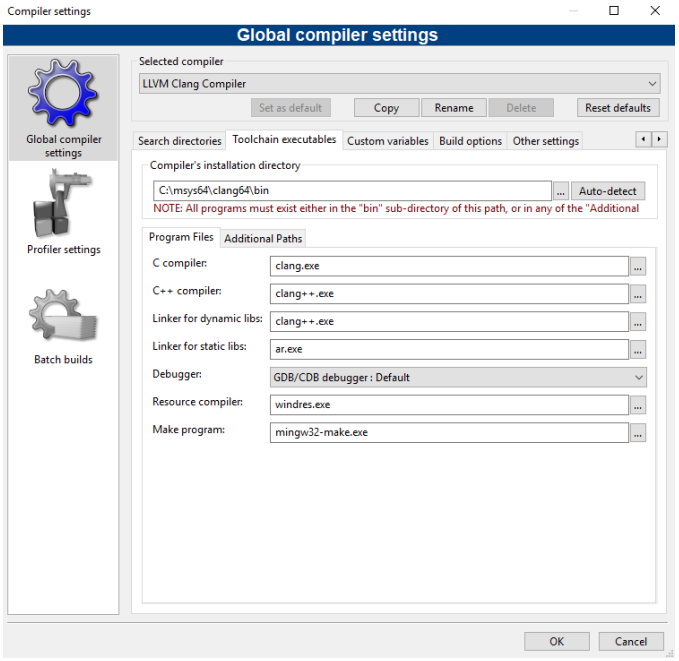

Рис. 6.1

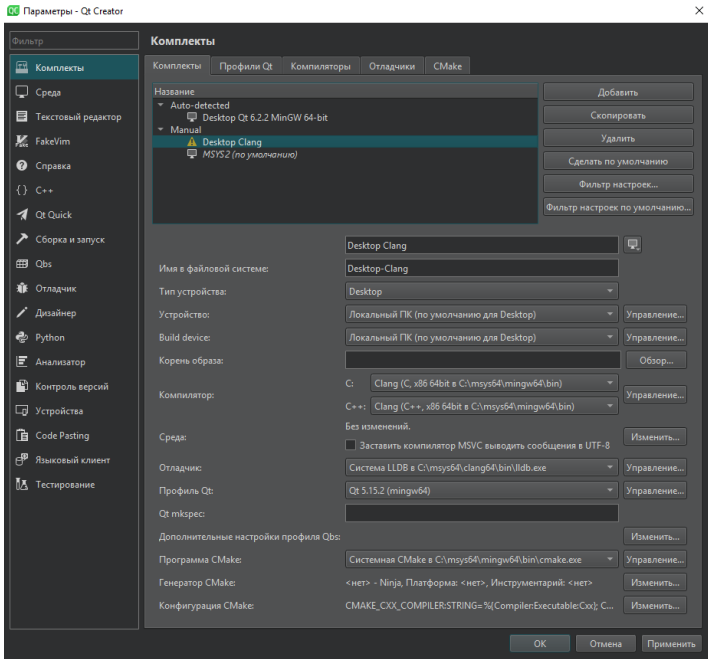

Рис. 6.2

Перейдите в меню Code::Blocks: Settings > Compiler. В списке Selected compiler выберите > LLVM Clang Compiler. С помощью кнопки «...» найдите директорию установки компилятора Clang. Установите пути к программам компилятора (Рис. 6.1).

Создайте новый комплект в Qt Creator: Инструменты > Параметры > Добавить (например, Desktop-Clang). Заполните поля как показано на Рис. 6.2. Обратите внимание, что стандартным отладчиком Clang является программа lldb.

7. Настройка консоли для вывода кириллицы

Решений данного вопроса очень много, но, к сожалению, универсального решения не существует. В любом случае, консоль нужно настроить для вывода в кодировке UTF-8. Это определяется в настройках операционной системы (правкой реестра). В Windows 10 можно поступить следующим образом. Перейдите в меню «Пуск» > Панель управления и выберите раздел «Часы и регион»

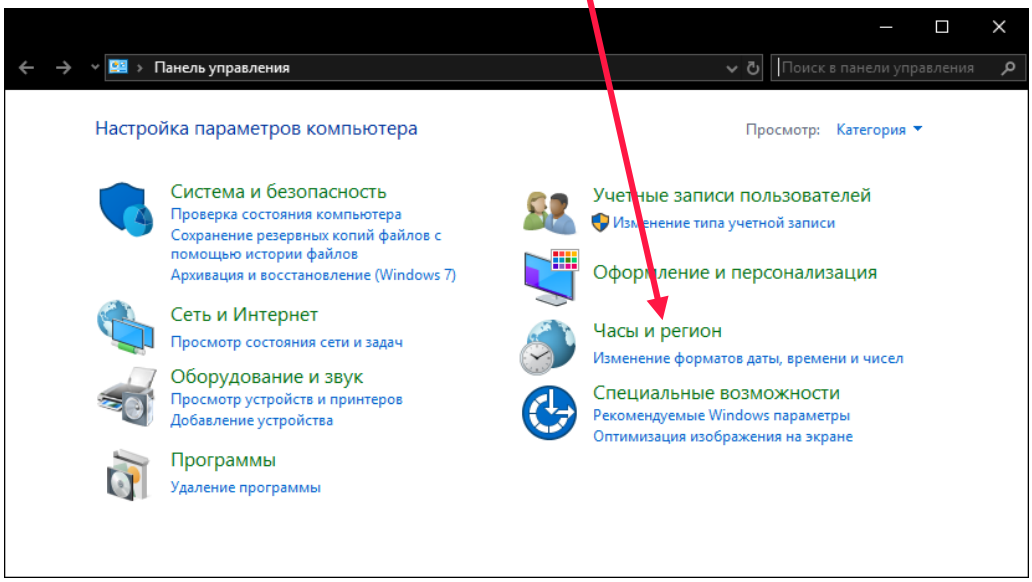

Рис. 7.1

Выберите в окне (Рис. 7.2) вкладку Дополнительно > Изменить язык системы и установите флажок > Бета-версия: Использовать Юникод (UTF-8)… > Ок (Рис. 7.3).

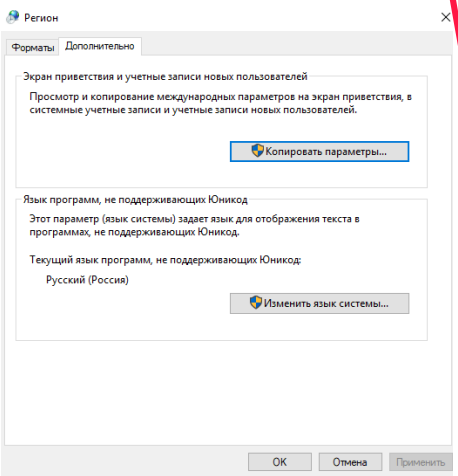

Рис. 7.2

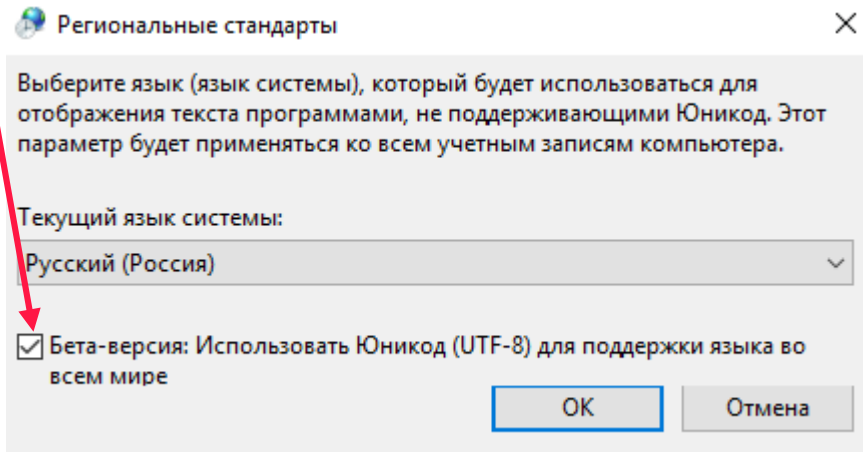

Рис. 7.3