Лабораторна робота №5
Робота з IDE Code::Blocks:
компіляція та відлагодження програми

Мета роботи: Вивчення основних прийомів роботи з інтегрованим середовищем розробки *Code::Blocks*. Отримання навичок компіляції та відлагодження програми.

Зміст:

Короткі теоретичні відомості

Інтегроване середовище розробки
(Integrated Development Environment, IDE)

В загальному випадку для створення програм потрібно мати такі компоненти:

- ✓ текстовий редактор;
- ✓ компілятор та/або інтерпретатор та редактор зв'язків;
- ✓ бібліотеки функцій.

Сучасні системи програмування включають в себе всі зазначені компоненти і називаються інтегрованими середовищами розробки.

Code::Blocks – це безкоштовне середовище розробки (*IDE*) програм на *C/C++* та *Fortran*, досить зручне для користувачів. Ця середовище розробки створено із можливостями для розширення та гнучким налаштуванням.

Створений за концепцією модулів, що підключаються, (плагінів) *Code::Blocks* може бути легко розширений. Будь-яка функціональність може бути доданою шляхом інсталяції певних плагінів.

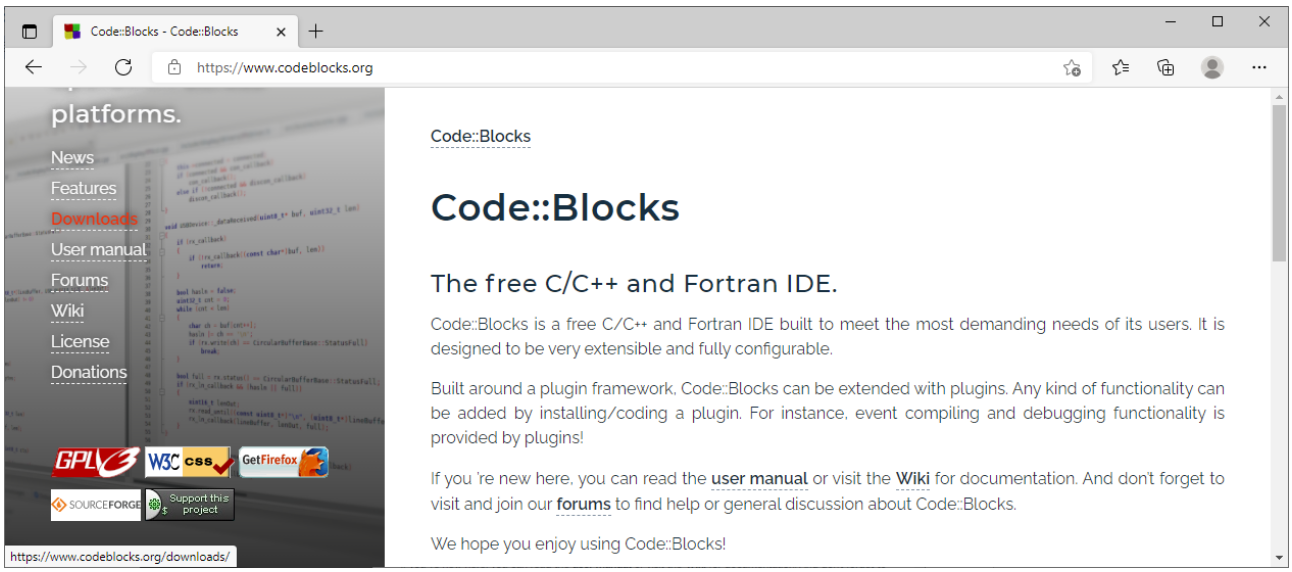
Інсталяція *Code::Blocks*

IDE Code::Blocks можна встановити на власний комп'ютер. Для скачування інсталяційного файлу із *IDE Code::Blocks* потрібно перейти за посиланням:

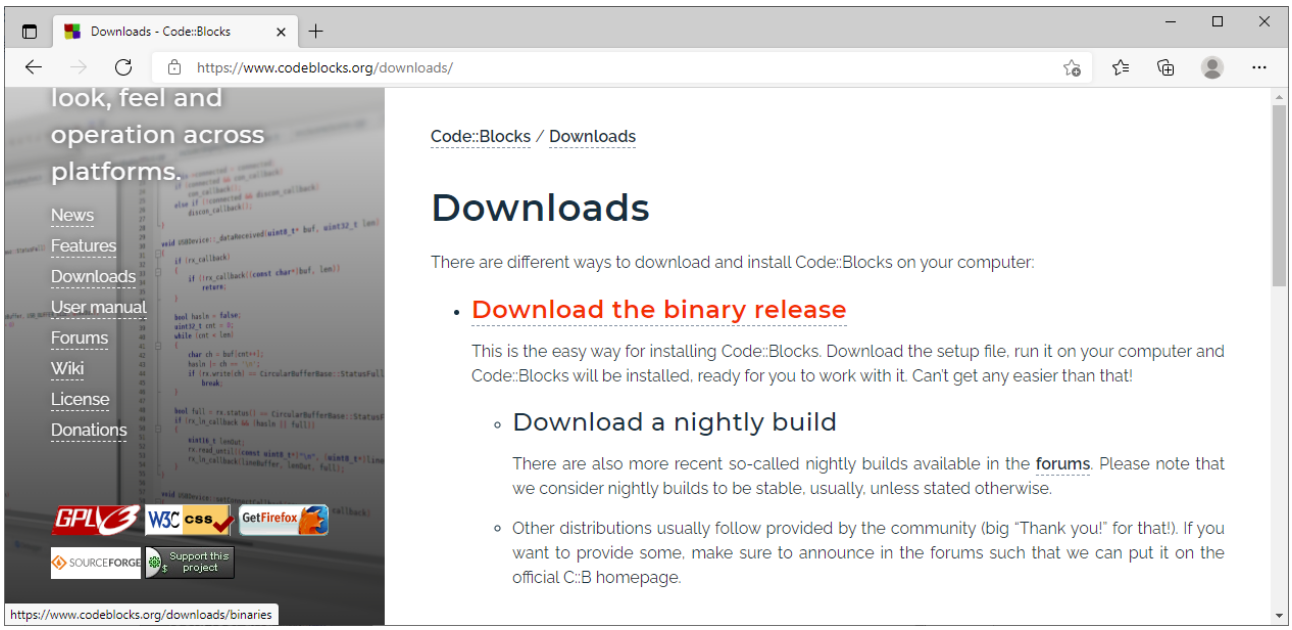
https://www.codeblocks.org/

За посиланням, наведеним вище, відкривається головна сторінка, на якій можна отримати основну інформацію про цей програмний продукт.

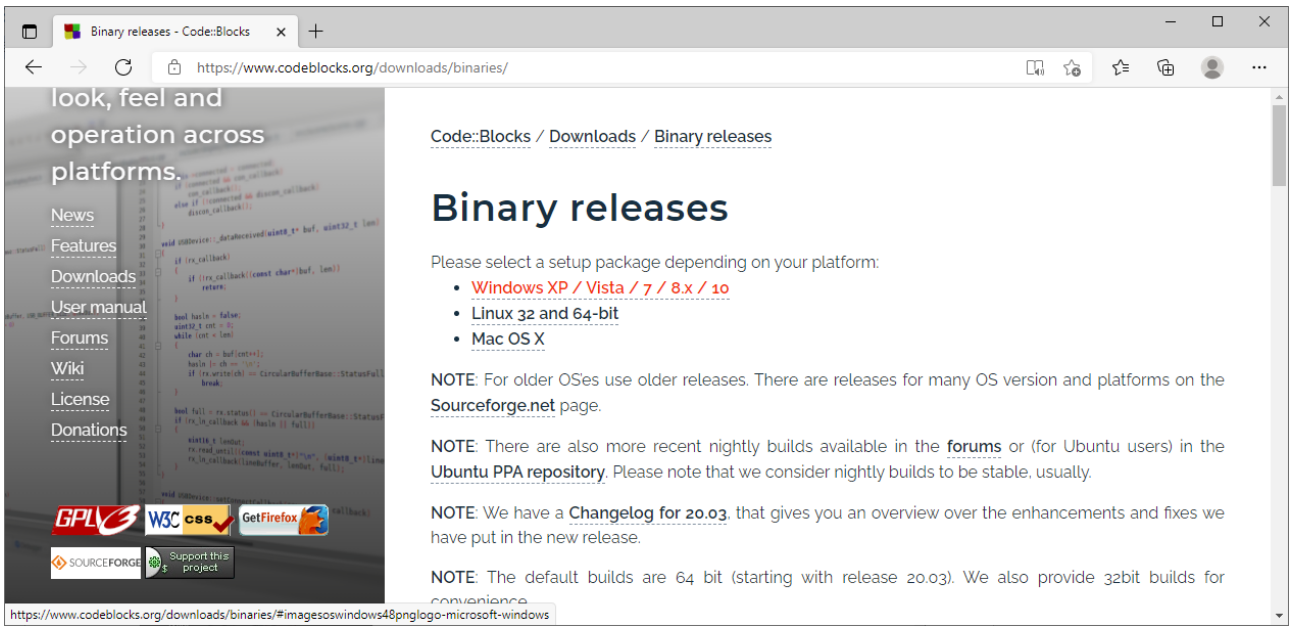
Для завантаження інсталяційного файлу *IDE Code::Blocks* потрібно натиснути *Download* в панелі, розташованій ліворуч на головній сторінці (помічено червоними літерами на рисунку нижче):



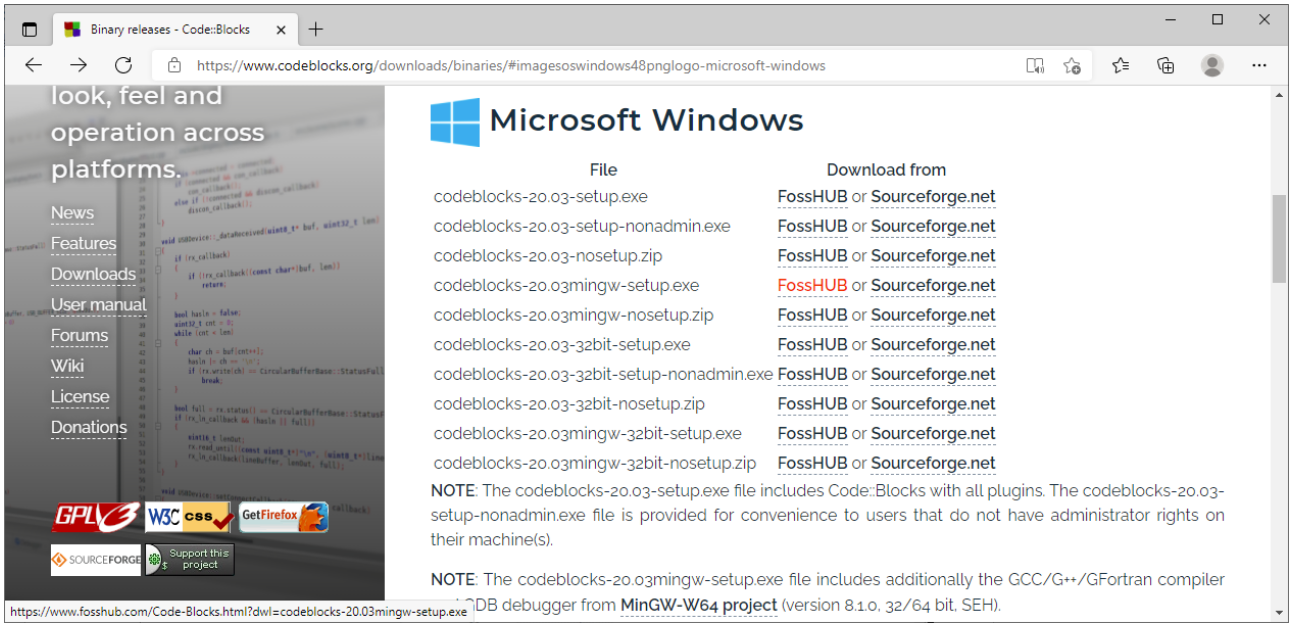
На сторінці *Download* потрібно натиснути *Download the binary release*:



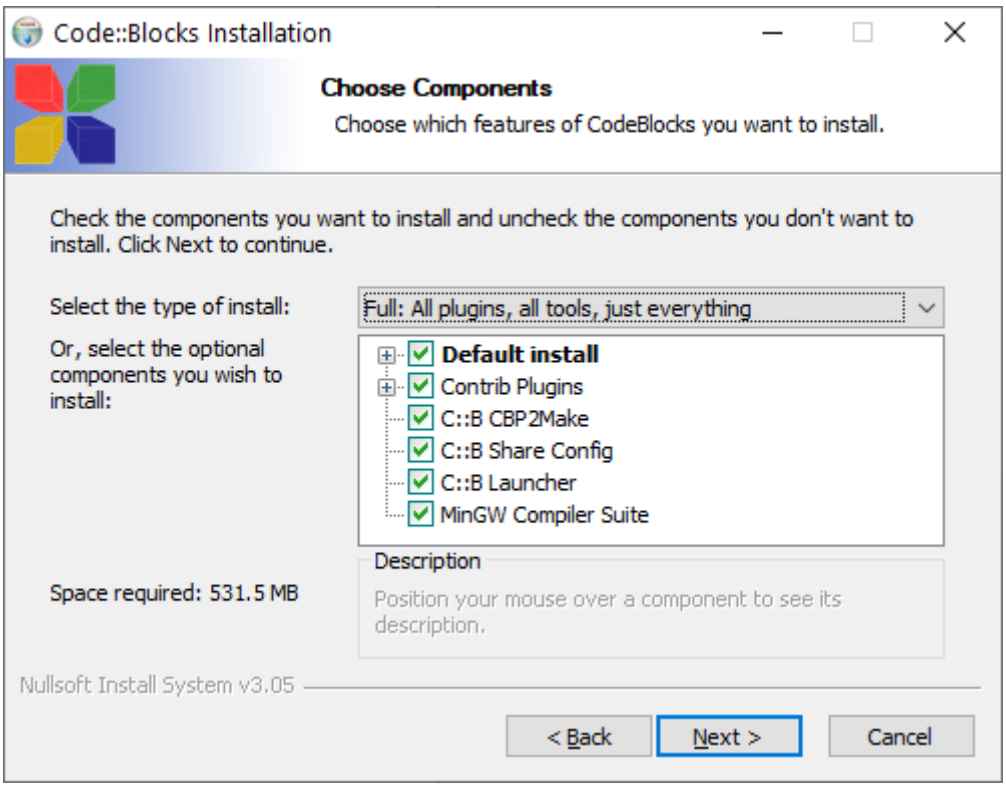
На сторінці *Binary releases* обирається операційна система (*Windows*, *Linux* чи *Mac OS*), яка встановлена на машині, на яку інсталюється *Code::Blocks*:



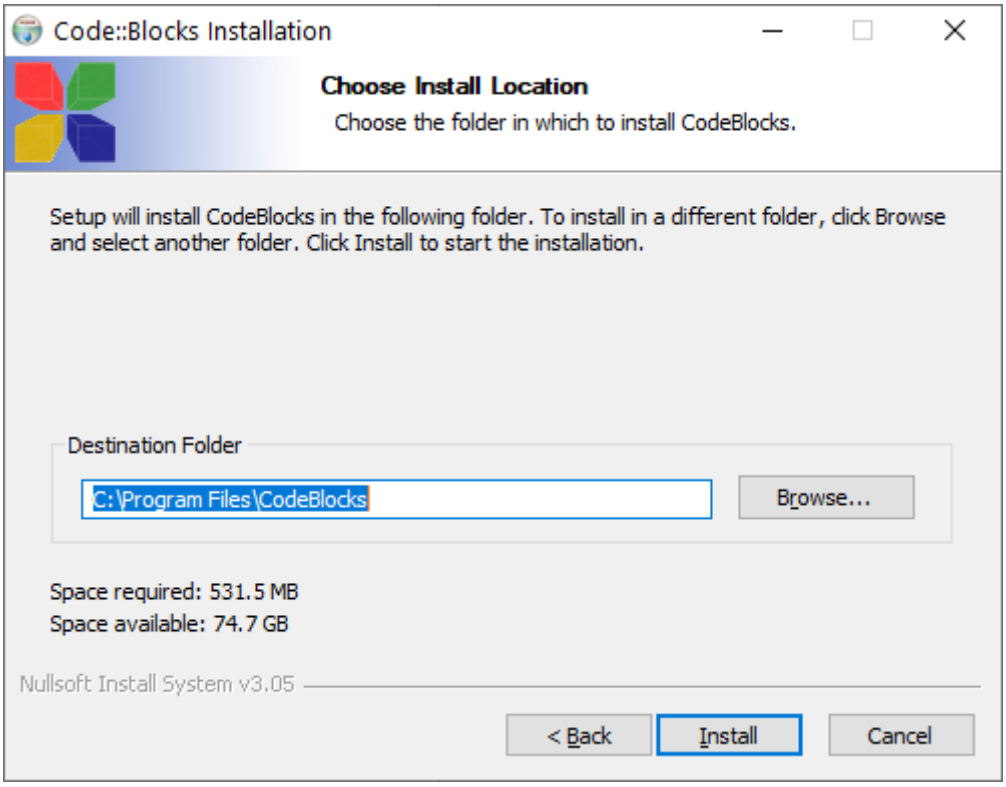
Для збірки та налагодження написаних програм потрібен компілятор та засоби налагодження. Якщо ви не впевнені, чи є на вашому комп'ютері компілятор і налагоджувач для *C*, обирайте варіант *codeblocks*-версія-*mingw-setup*. Цей варіант окрім власне *IDE* доповнений також компілятором:



Після завантаження інсталяційного файлу, його потрібно запустити на виконання та погодитися із ліцензійними умовами. На сторінці *Choose Components* потрібно обрати ті компоненти, які потрібно інсталювати:



та встановити шлях до каталогу, в якому буде інстальований програмний продукт:



Запуск *Code::Blocks* на віддаленому сервері

Із *IDE Code::Blocks* можна працювати не тільки на локальній машині, попередньо інсталювавши це середовище на неї, але і запустивши це *IDE* на віддаленому сервері. Наприклад, на сервері *sandbox.ee.kpi.ua,* на якому ми працювали на попередніх лабораторних роботах, також встановлено *IDE Code::Blocks.*

Для запуску *IDE Code::Blocks* на віддаленому сервері, окрім SSH-клієнта, також використовується протокол для відображення та передачі зображень *X Windows System*. Цей протокол дозволяє відображати графіку програмного забезпечення як на локальних дисплеях, так і як завгодно віддалених у мережевій доступності.

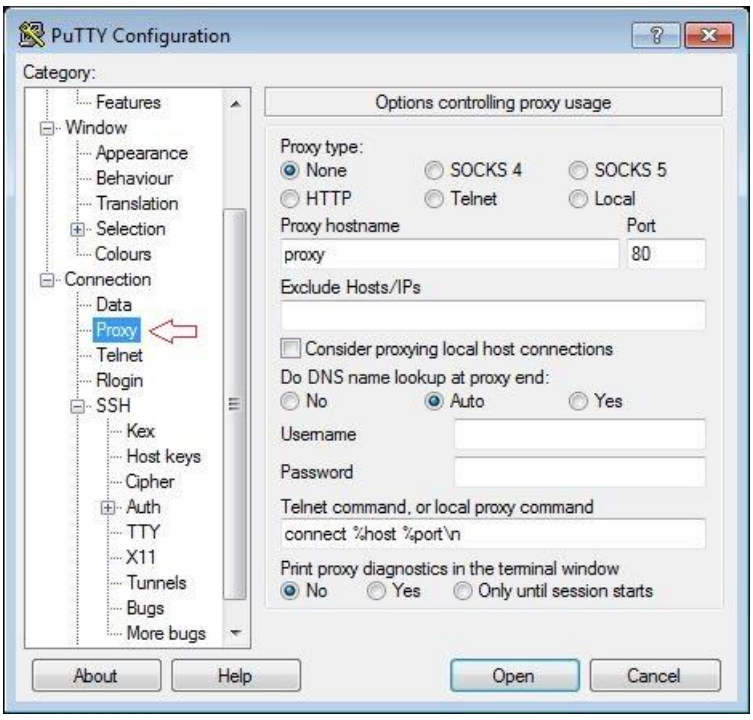
Для відображення графіки програмного забезпечення на віддаленому робочому місці, необхідне налаштування спеціалізованого сервера на робочому місці.

Для підключення до віддаленого сервера та відображення вікон програмного забезпечення на робочому місці необхідні:

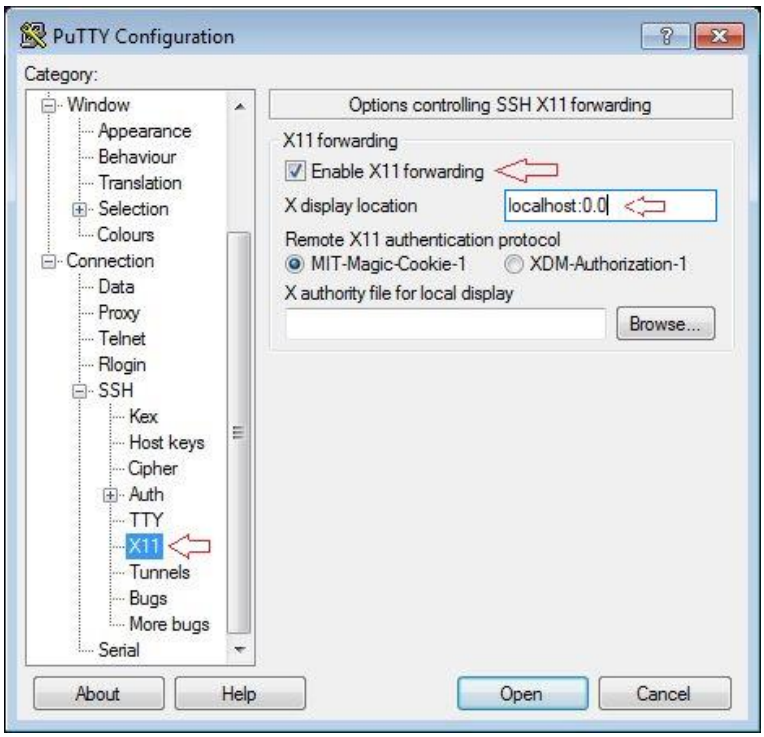
1. Акаунт (логін та пароль) до сервера.
2. SSH клієнт, що забезпечує підключення до сервера, шифрування каналу передачі, авторизацію на сервері та відображення текстової консолі. Клієнт має підтримувати функцію «*X-forwarding*».
3. Локальний *X-Server*, що забезпечує правильне відображення графічної інформації, переданої з сервера, у віконному середовищі клієнта.

Інсталяція, налаштування SSH-клієнта та робота із ним описані в лабораторній роботі «Основи роботи с UNIX-подібними системами». Але окрім раніш описаних налаштувань, для режиму роботи із відображенням графіки, потрібно ще виконати додаткові налаштування:

a. За необхідності встановити налаштування проксі

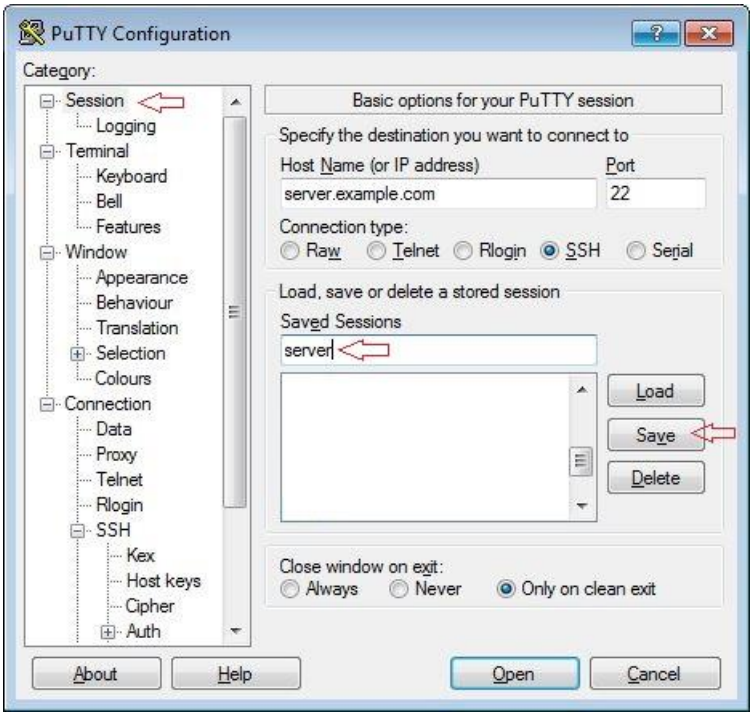


b. Встановити дозвіл на транспорт протоколу X11 та адресу дисплея (`localhost:0.0`)



c. При роботі через повільні канали може бути корисним встановити стиснення даних у властивостях SSH, у локальній мережі у цьому нема необхідності.

d. І нарешті зберегти профіль з’єднання для подальшого використання під новим ім’ям:



Найпоширенішими серверами *X11* для *Windows* є *VcXsrv* та *XMing*. Останні версії цих програм можна скачати по посиланням:

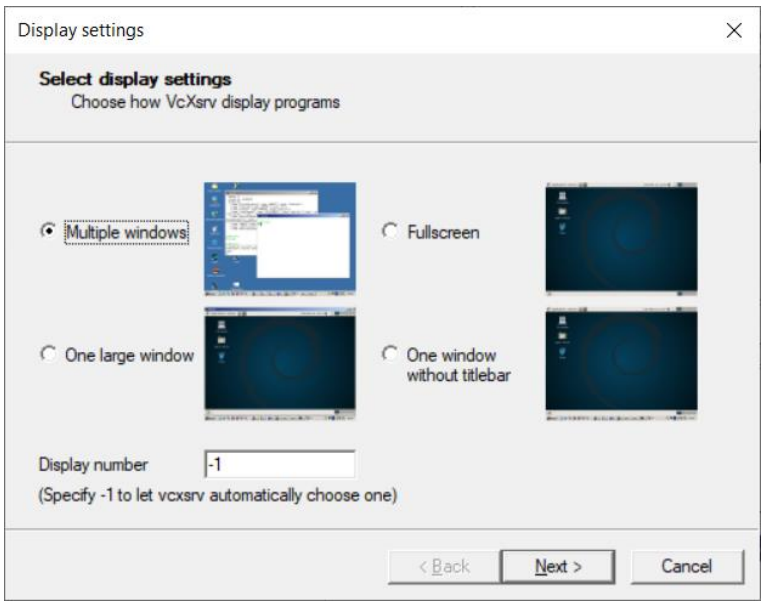
VcXsrv: https://sourceforge.net/projects/vcxsrv/

XMing: https://sourceforge.net/projects/xming/

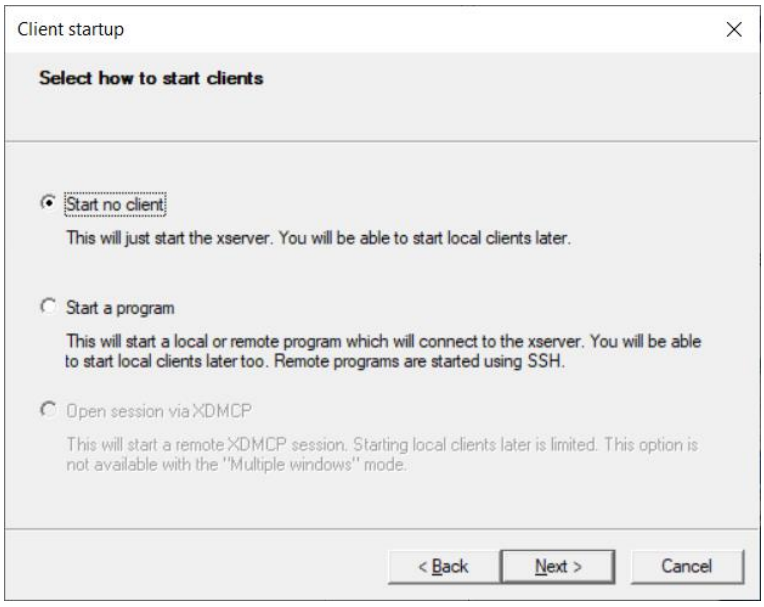
Після встановлення однієї із цих програм слід запустити програму *XLaunch*

та виконати налаштування, як показано на скріншотах нижче:

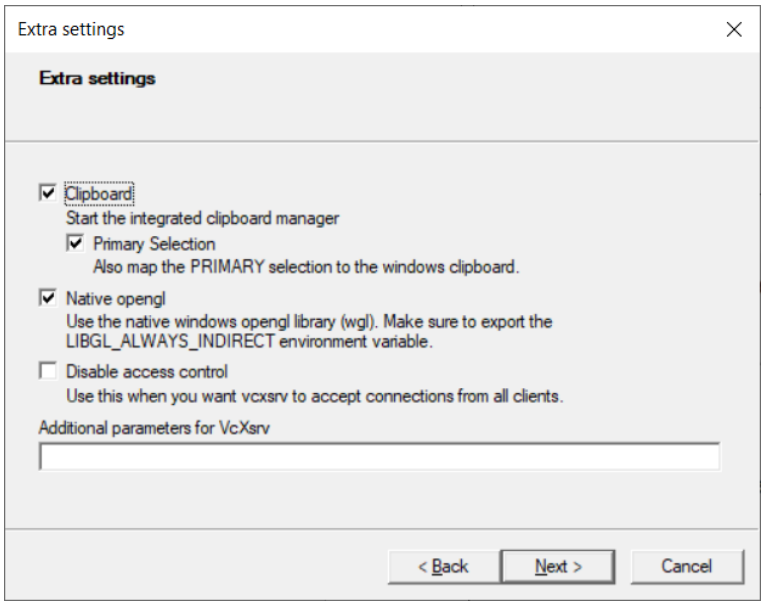
a.

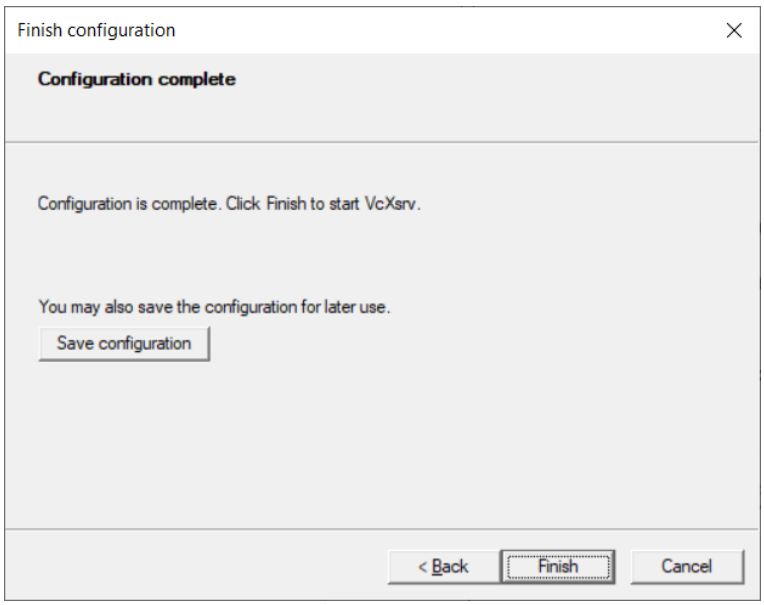

Display settings
Select display settings
Choose how VcXsrv display programs
Multiple windows
Fullscreen
One large window
One window without titlebar
Display number
-1
(Specify -1 to let vcxsrv automatically choose one)
< Back
Next >
Cancel

b.


Client startup
Select how to start clients
Start no client
This will just start the xserver. You will be able to start local clients later.
Start a program
This will start a local or remote program which will connect to the xserver. You will be able to start local clients later too. Remote programs are started using SSH.
Open session via XDMCP
This will start a remote XDMCP session. Starting local clients later is limited. This option is not available with the "Multiple windows" mode.
< Back
Next >
Cancel

c.


Extra settings
Extra settings
Clipboard
Start the integrated clipboard manager
Primary Selection
Also map the PRIMARY selection to the windows clipboard.
Native opengl
Use the native windows opengl library (wgl). Make sure to export the LIBGL_ALWAYS_INDIRECT environment variable.
Disable access control
Use this when you want vcxsrv to accept connections from all clients.
Additional parameters for VcXsrv
< Back
Next >
Cancel

d. 

На останньому кроці можна зберегти конфігураційний файл у доступному місці, наприклад на робочому столі (після натискання кнопки *Save configuration*). Після цього натискаємо кнопку *Finish*. Це був початковий етап установки. Надалі достатньо запускати збережений конфігураційний файл перед початком кожного сеансу роботи із сервером *sandbox.ee.kpi.ua*. Наприклад, якщо конфігураційний файл було збережено на робочому столі, достатньо двічі клікнути на ньому. Можна також щоразу проходити процедуру запуску Xlaunch.

Упевнитись, що клієнт працює, можна за іконкою у треї:

Після запуску *XMing* або *VcXsrv* треба підключитися до сервера за допомогою SSH-клієнта.

Після під’єднання до сервера, для запуску *IDE Code::Blocks,* в командному рядку потрібно набрати `codeblocks` або `codeblocks &` і далі можна працювати із цією *IDE* так само, які і на локальній машині.

Довідка. Зверніть увагу, що власне програма та файли, які нею створюються, знаходяться на віддаленому сервері, ви не можете їх знайти на своєму комп’ютері. Для їх отримання слід скористатися програмою обміну файлами, наприклад *WinSCP*.

Можливості та засоби розробки програм *IDE Code::Blocks*

Проекти як контейнери

Для розробки програми на *C* в *IDE Code::Blocks* необхідно створити проект. Проект – це контейнер, який об’єднує файли вихідного коду, бібліотеки та налаштування для програми, що розробляється. Проект містить посилання на ці файли, налаштування компілятора та з інших елементів конфігурації програми.

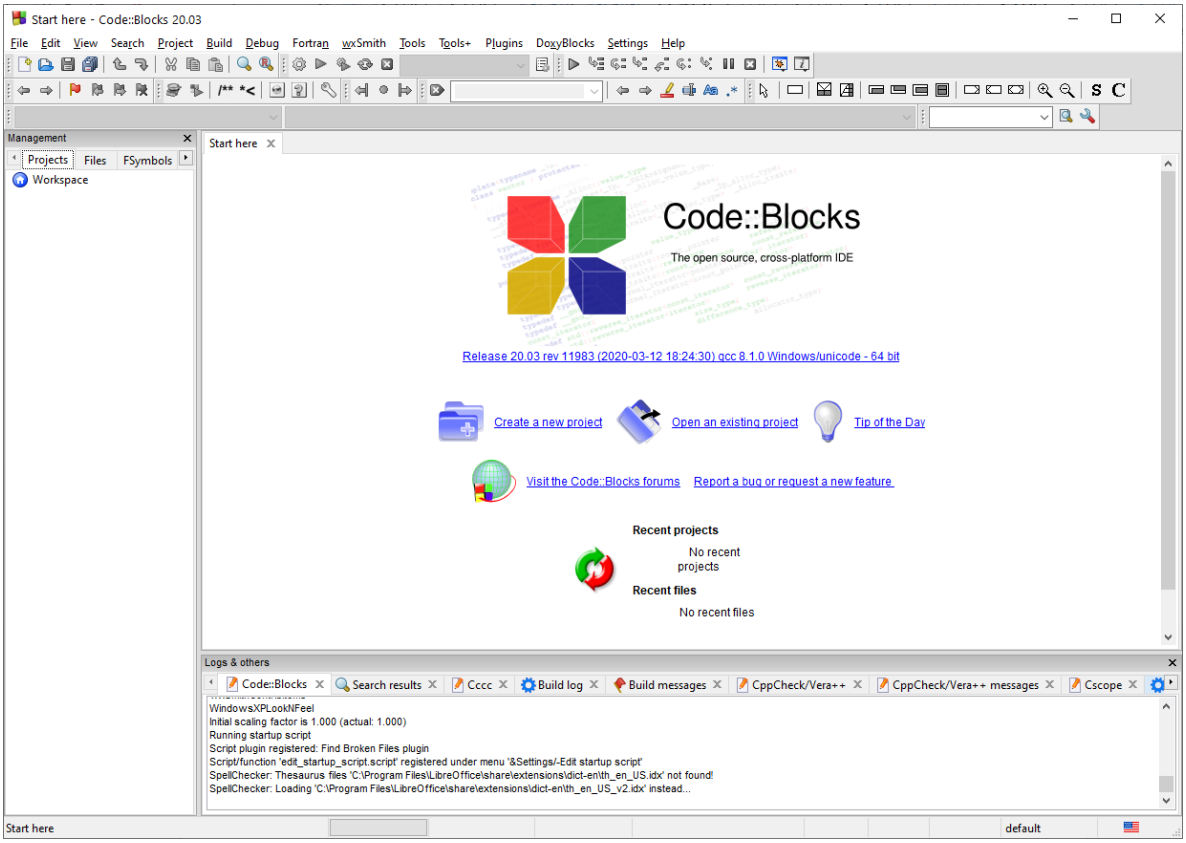
Для кожного проекту створюється окремий каталог, де розміщуються файли проекту. Також цей каталог містить файл з конфігурацією проекту і має ім'я проекту з розширенням *.cbp*. При перенесенні проекту на інший комп'ютер необхідно брати саме весь цей каталог, проте підкаталоги *bin* (містить файли що виконуються для програми що розроблялась) та *obj* (містить об'єктні файли програми що створювалась) не потрібні.

Головна сторінка, відкриття та створення проектів

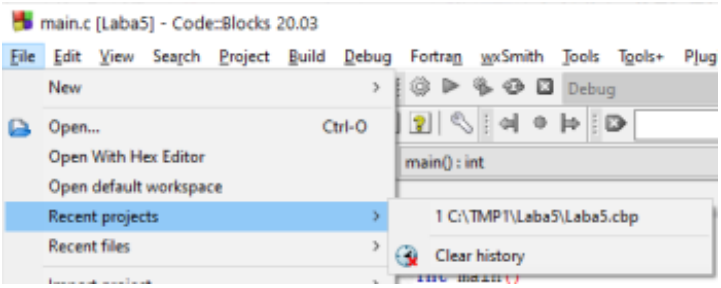
Після запуску *IDE Code::Blocks* відкривається стартова сторінка, яка показана на рисунку нижче. Вона містить декілька областей:

- область *Management* із менеджером проектів (ліворуч), де можна подивитися проекти та файли, які є частинами цих проектів,
- область редагування файлів, в якій можна подивитися вміст файлів, які були виділені в менеджері проектів,

- область *Logs&others*, де розташовані вікна із службовою інформацією:

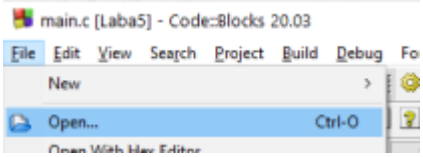


Для відкриття існуючих проектів із якими нещодавно працювали, потрібно обрати опції меню *File: Recent projects*:



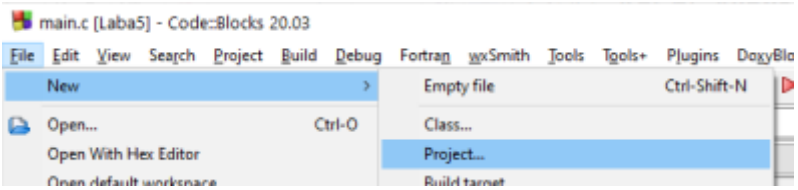
І далі із списку обрати потрібний проект.

Для відкриття існуючих проектів із якими працювали раніше, потрібно обрати опції меню *File: Open*:

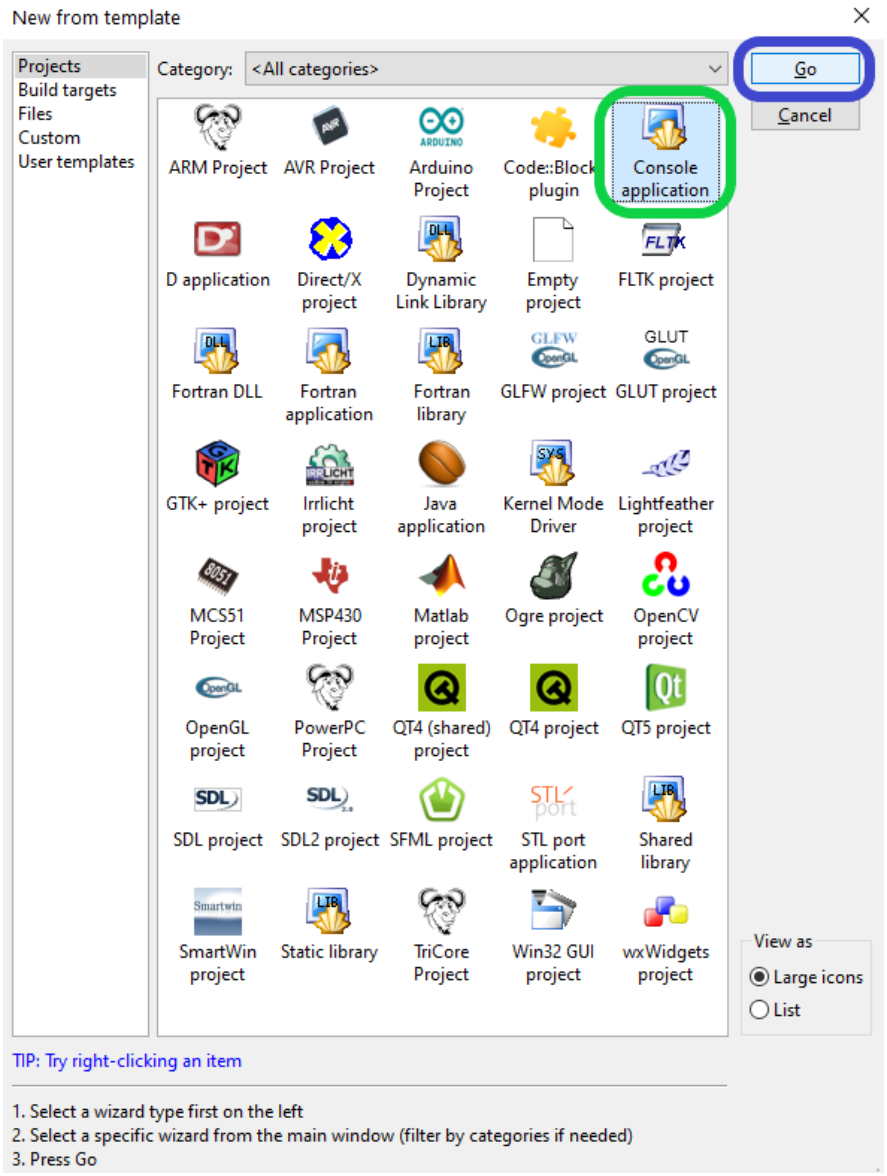


І далі у вікні провідника відкрити файлі із розширенням *.cbp* , який відповідає потрібному проекту.

Для створення нового проекту потрібно обрати опції меню *File: New: Project…*:

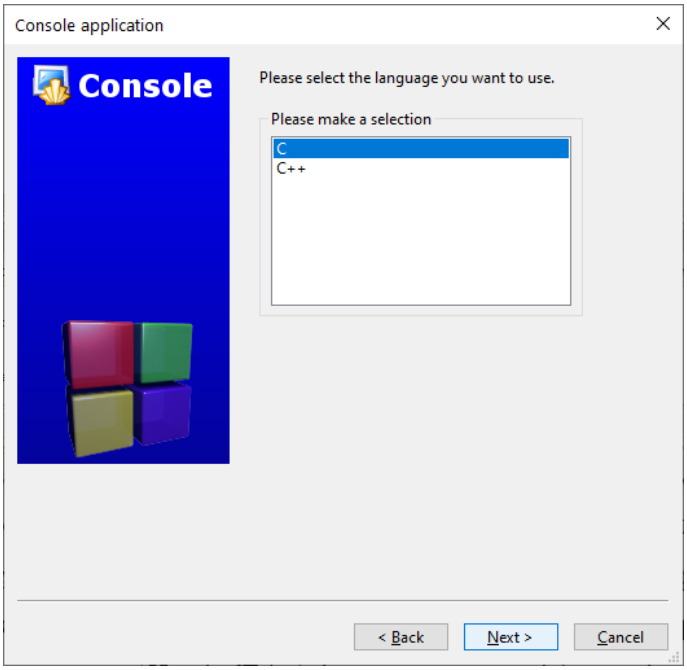


Далі у вікні вибору шаблону проекту (*New from template*) потрібно обрати тип шалону, який підходить для цілей розробки. Наприклад, для програм, які будуть створюватися в межах цього курсу підходить шаблон *Console application* (наведено зеленим на рисунку нижче). Після вибору шаблону потрібно натиснути кнопку *Go* (наведено синім на рисунку нижче):

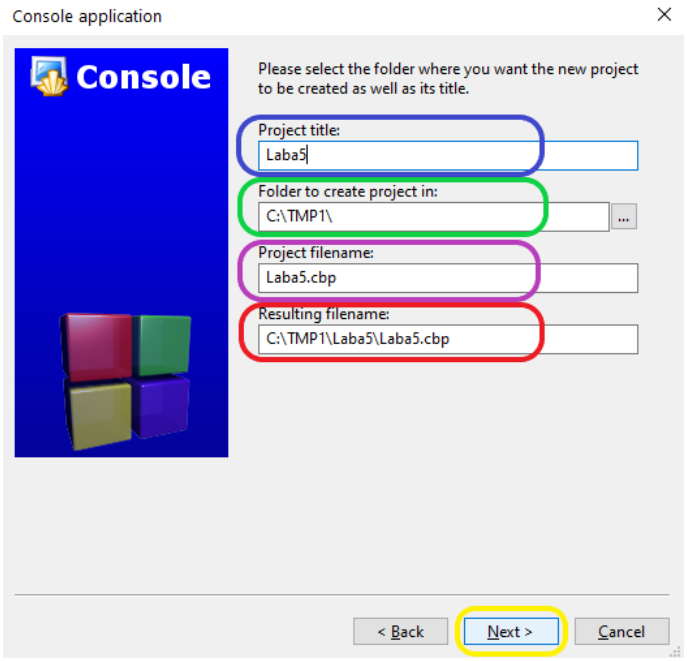


При створенні проекту в діалоговому вікні потрібно вказати, синтаксису якої мови повинен відповідати новий проект що створюється. Наприклад, для програм, які

будуть створюватися в межах цього курсу підходить опція *C*, як це показано на рисунку нижче:

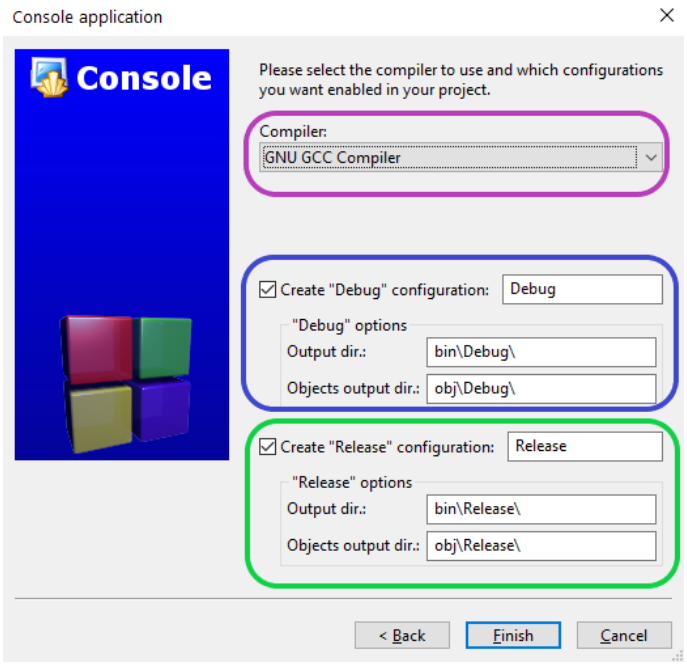


Далі потрібно вказати ім’я проекту (наведено синім на рисунку нижче), вказати каталог, де буде зберігатися проект (наведено зеленим), при цьому поля із іменем файлу проект (наведено фіолетовим) та результуючим ім’ям (наведено червоним) будуть заповнені автоматично:



При створенні проекту на сервері, проект повинен обов’язково знаходитися в домашньому каталозі студента (/home/username). Скористайтеся кнопкою «…» для пошуку потрібного каталогу.

Наступний крок – це вибір компілятора, який буде використано при компіляції проекті (на рисунку нижче наведено фіолетовим). Також в цьому діалоговому вікні можна обрати який тип конфігурації будуть мати вихідні файли – *Debug* (наведено синім на рисунку нижче), *Release* (наведено зеленим на рисунку нижче), чи обидва разом. Окрім цього можна також виконати налаштування назв каталогів, в яких будуть зберігатися вихідні файли для *Debug* чи *Release* конфігурацій:

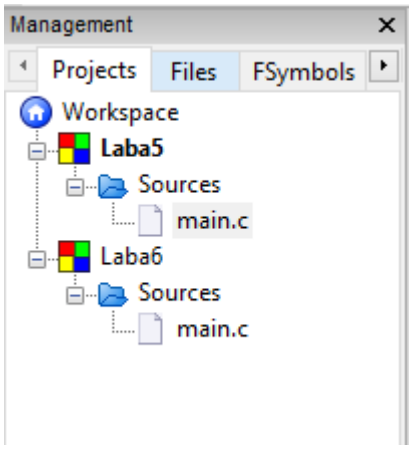


Довідка. *Для налагодження програми такими засобами, як gdb, у виконуваний файл програми має бути додана додаткова інформація. Її наявність збільшує розмір виконуваного файла, а іноді призводить до неможливості належної оптимізації. Така конфігурація найчастіше називається Debug. Після всестороннього дослідження програми за допомогою налагодження, створюється окрема версія програми (Release), яка не містить надлишкової інформації і може бути оптимізована. Ця версія програми надається кінцевим споживачам. Перемикання налаштувань Debug та Release можна здійснити із випадаючого списку на панелі інструментів. Для цілей нашої дисципліни достатньо Debug-конфігурації.*

Після створення проекту, він буде містити тільки один файл із ім'ям *main.c* , в якому буде записано найпростіша програма, яка при запуску на виконання, друкує на екран напис: «*Hello world!*»:

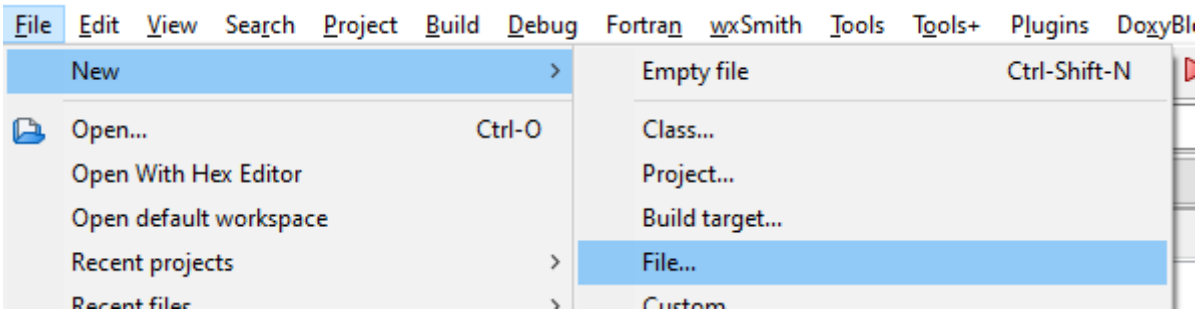
```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

При створенні декількох проектів, усі вони з'являються в менеджері проектів. Проект, який є активним у поточний момент часу, наведений більш темним кольором (на рисунку нижче це проект *Laba5*):

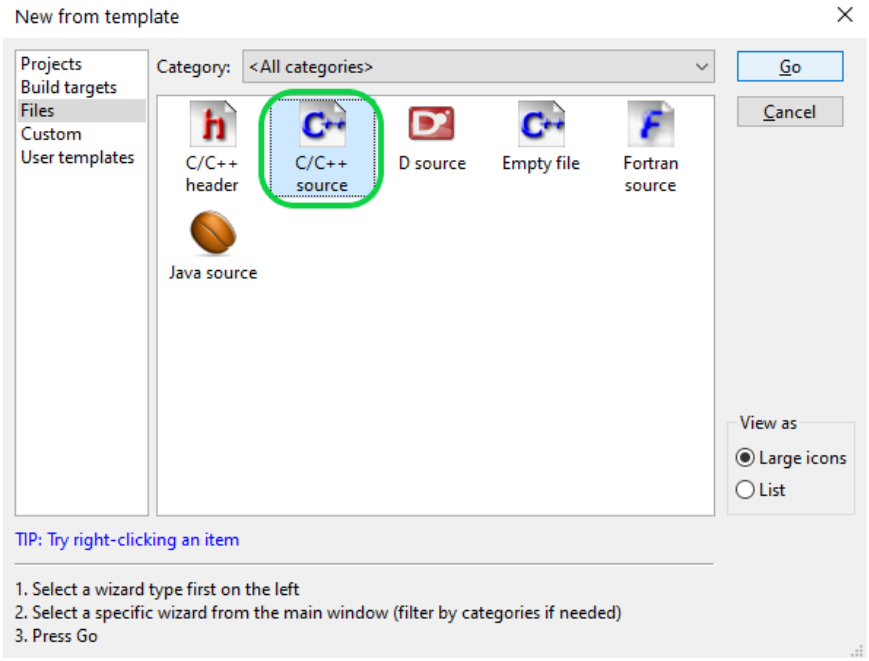


Додавання файлів до проекту

Для додавання файлу до проекту потрібно обрати опції меню *File: New: File…*:

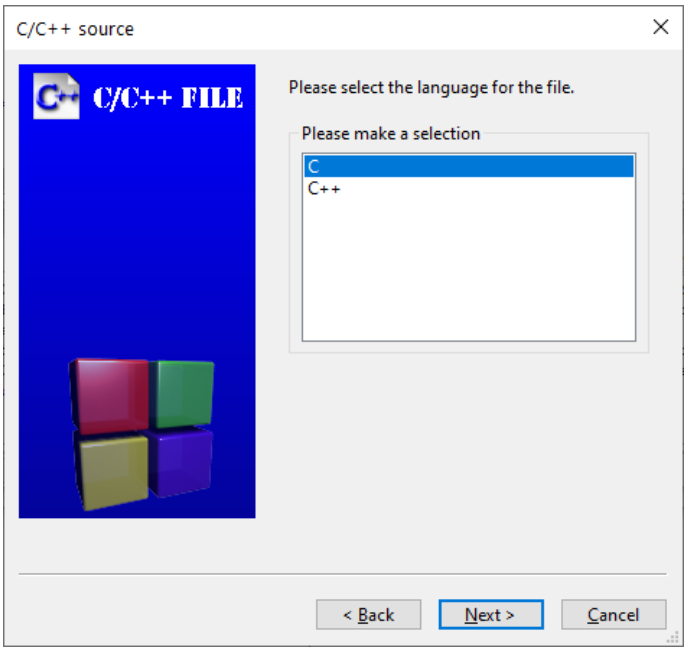


Далі у діалоговому вікні вказати тип файлу, який потрібно створити. Наприклад, для програм, які будуть створюватися в межах цього курсу підходить опція *C/C++ source* (наведена зеленим на рисунку нижче):

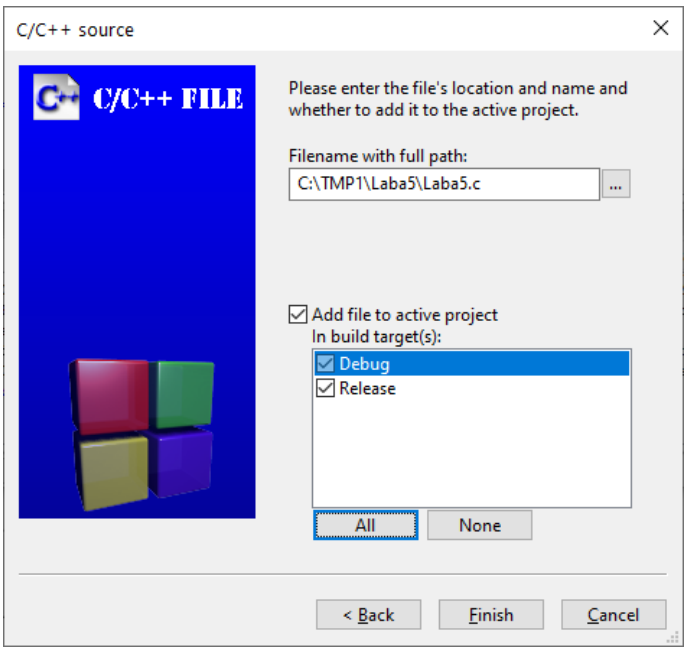


Для файлів із заголовками функцій потрібно буде використовувати *C/C++ header*.

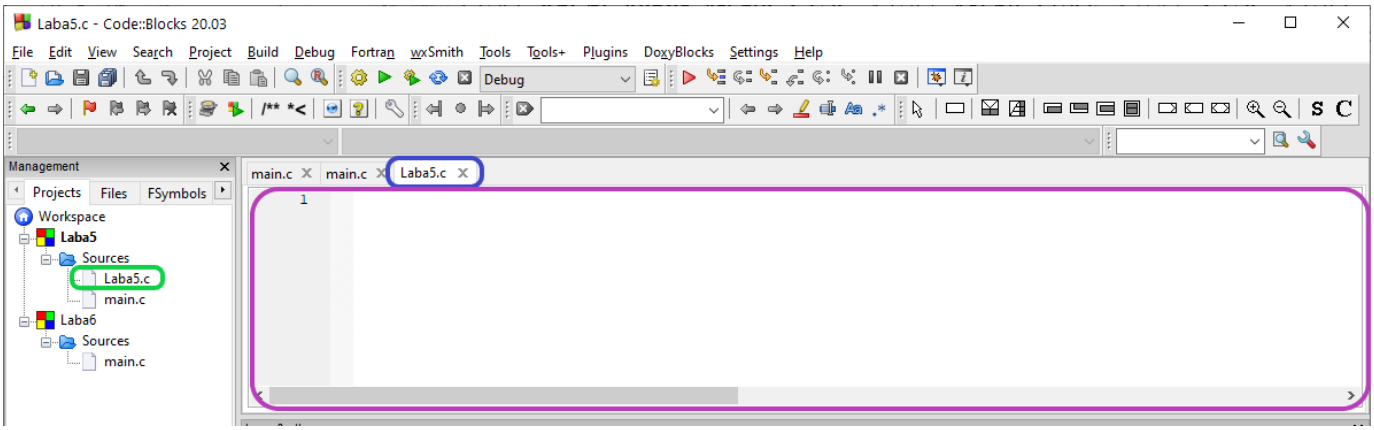
На наступному кроці потрібно обрати мову, якій буде відповідати файл що створюється. Обрана мова повинна відповідати мові, яка буда задана при створенні проекту:



Далі задається ім’я файлу, що створюється і місце його розташування:



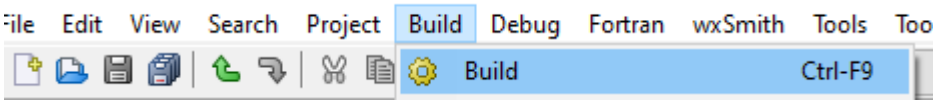
Якщо усі кроки були виконані коректно, то серед списку файлів проекту з’явиться назва створеного файлу (наведено зеленим на рисунку нижче). А в області редагування файлів можна буде побачити нову закладку із відповідним ім’ям (наведено синім на рисунку нижче). Якщо зайти в цю закладку, то відкриється область редагування новоствореного файлу (наведено фіолетовим на рисунку нижче):



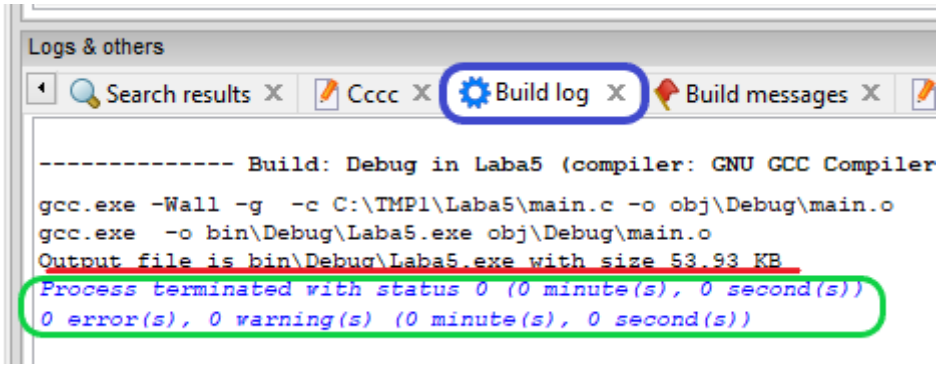
Засоби побудови

У середовищі *IDE Code::Blocks* є засоби побудови та налагодження.

Для побудови (збірки) програми потрібно вибрати опції меню *Build*:*Build*:



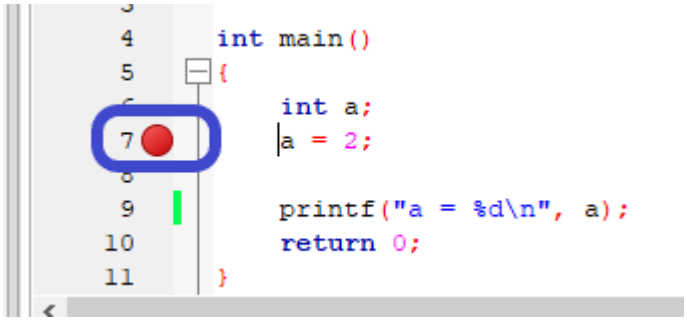
Інформацію стосовно збірки проекту буде відображено у вікні *Build log* області *Logs&others* (наведено синім на рисунку нижче). В останніх рядках цього вікна буде зазначена інформація стосовно кількості помилок та попереджень (наведено зеленим на рисунку нижче), а також назву і розмір вихідного файлу (підкреслено червоним):



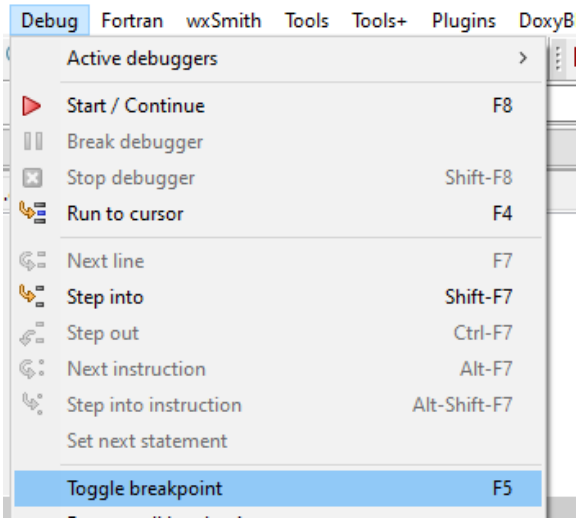
Засоби налагодження

Частіш за все, усі синтаксичні помилки знаходяться на етапі побудови проекту. Для знаходження логічних та семантичних помилок та помилок які виникають під час виконання, зручніш за все використовувати засоби налагодження. Для того, щоб відслідковувати як змінюються змінні в процесі виконання програми, зручно використовувати режим призупинення виконання (*Break Point*). В цьому режимі виконання програми призупиняється перед рядком, в якому встановлено *Break Point* і користувач має час, щоб переглядати вміст змінних за допомогою вікна *Watches.*

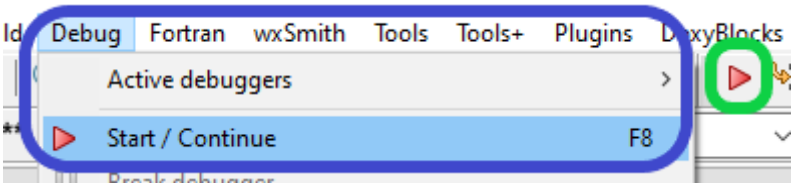
Для того, щоб припинити виконання програми до певного рядка, в цьому рядку потрібно встановити *Break Point.* Його можна встановити, клікнувши лівою кнопкою миші на сірій області, яка розташована ліворуч від тексту програми, навпроти обраного рядка (на рисунку нижче *Break Point* – це червоний круг):



Також *Break Point* можна встановити розташувавши курсор в потрібному рідку та обравши опції меню *Debug: Toggle breakpoint* або натиснувши гарячу клавішу **F5**:

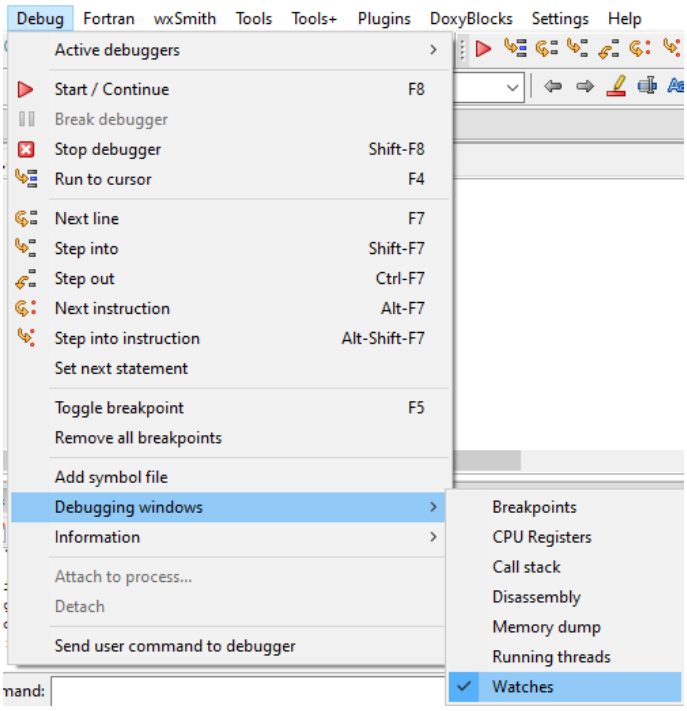


Для запуску програми на виконання у режимі налагодження потрібно або обрати опції меню *Debug: Start/Continue* (наведено синім на рисунку нижче), або натиснути гарячу клавішу **F8**, або натиснувши відповідно іконку головного меню (наведено зеленим на рисунку нижче):

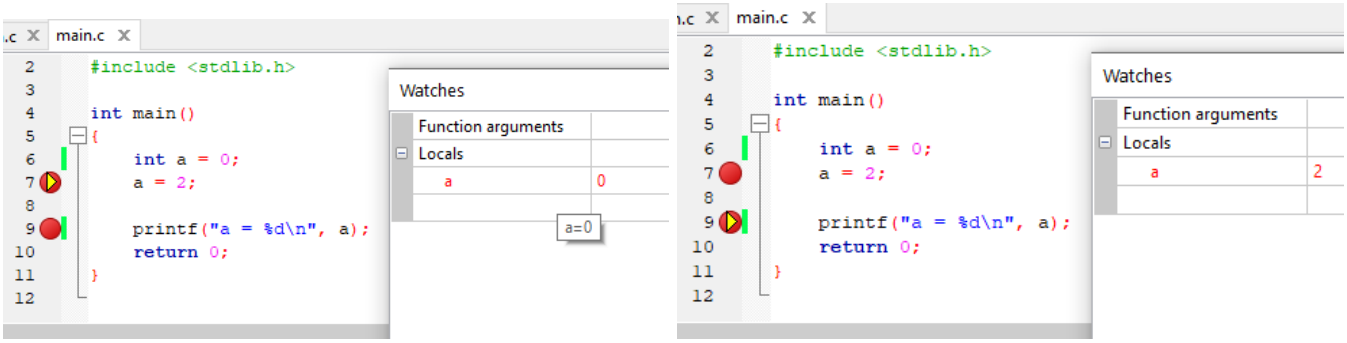


Якщо в тексті програми встановлені *Break Point,* то виконання програми буде призупинено. При цьому в рядку, перед яким програму було призупинено, з’явиться жовтий трикутник ( ). Для продовження виконання потрібно виконати такі ж самі дії і програма продовжить своє виконання або до наступного *Break Point,* або до кінця програми, якщо в тексті програми не встановлено жодного *Break Point.*

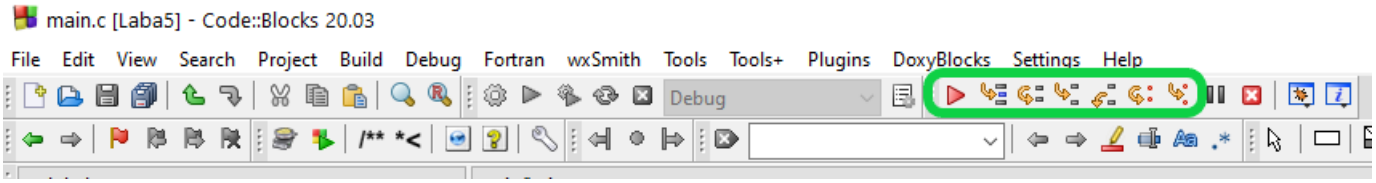
Вміст змінних можна подивитися у вікні *Watches,* яке можна відкрити обравши опції меню *Debug: Debugging windows: Watches*:



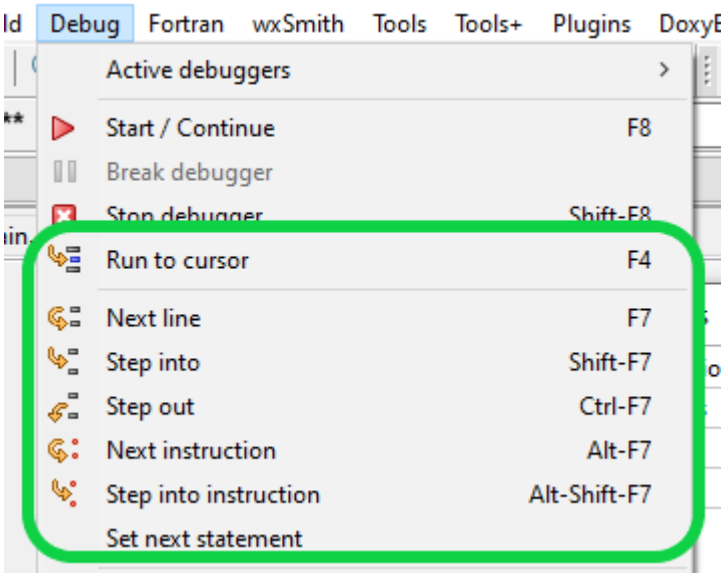
Вікно *Watches* демонструє поточне значення, яке зберігається в змінній. Наприклад, на рисунку нижче ліворуч показано значення змінної *a* до виконання присвоювання *a = 2*, а на рисунку праворуч – після.



IDE Code::Blocks дозволяє виконувати покрокове виконання програми. Для цього можна скористатися іконками головного меню:



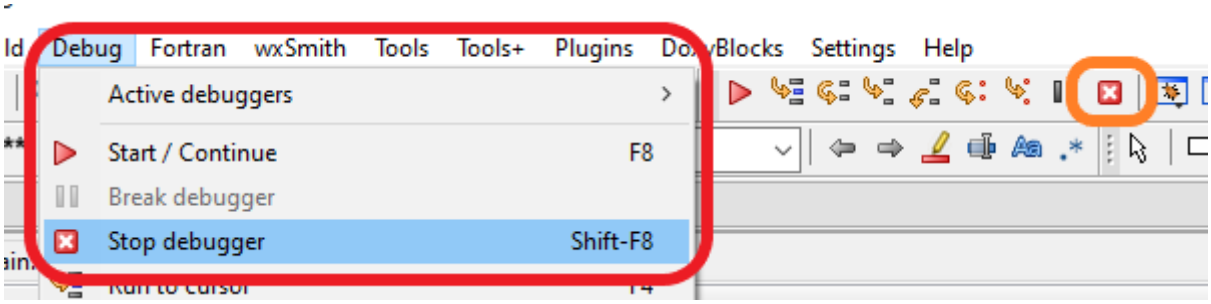
Або використавши опцію меню *Debug* та обравши одну із потрібних команд (наведено зеленим на рисунку нижче):



- Опція меню *Debug:Run to cursor* або гаряча клавіша **F4** призводять до виконання програми до рядка, в якому розташовано курсор,
- Опція меню *Debug:Next line* або гаряча клавіша **F7** призводять до виконання тільки одного рядка, на який показує жовтий трикутник,
- Опція меню *Debug:Step into* або комбінація клавіш **Shift+F7** призводять до виконання тільки одного рядка, на який показує жовтий трикутник, із заходом всередину функції, якщо така зустрічається в цьому рядку,
- Опція меню *Debug:Step out* або комбінація клавіш **Ctrl+F7** призводять до виконання тільки одного рядка, на який показує жовтий трикутник, без заходу всередину функції, якщо така зустрічається в цьому рядку.

Дострокове завершення роботи програми в налагоджувальному режимі

Для дострокового завершення роботи в програми, запущеної в налагоджувальному режимі в потрібно обрати опції меню *Debug*:*Stop Debugger* (або натисніть **Shift** + **F8**):



Більш детальну інструкцію по використанню *IDE Code::Blocks* можна отримати за посиланням:

https://www.codeblocks.org/docs/manual_codeblocks_en.pdf

Деякі функції стандартного вводу-виводу

Функції стандартного вводу-виводу реалізовані в стандартній бібліотеці, а їх прототипи описані у файлі ***stdio.h***. Для його підключення в текст програми, зазвичай на її початку, слід додати директиву препроцесора:

```
#include <stdio.h>
```

printf() – форматний вивід на екран:

```
int printf(const char *format [, argument]... );
```

Перший параметр є символьним рядком, який задає специфікації формату. Інші параметри – змінні та виразів, значення яких виводяться. Кожна специфікація формату має такий вигляд (параметри у квадратних дужках необов'язкові):

```
%[flags][width][.precision]type
```

де type специфікатор типу:

- d або i - ціле десяткове число зі знаком
- u - десяткове число без знака
- x - ціле шыстнадцяткове число без знака
- f - число з плаваючою комою
- e - число в Е-формі
- g - число з плаваючою комою або в Е-формі
- c - один символ
- s - рядок
- % - символ %

flags ознака вирівнювання:

+ або пусто - вирівнювання по правому краю

– - вирівнювання по лівому краю

width - ціле число - загальна ширина поля. Якщо це число починається з цифри 0, при друці число доповнюється зліва нулями до заданої ширини. У задану ширину входять всі символи, включаючи знак, дробову частину і т.п.

precision - ціле число - кількість знаків після крапки при виведенні дійсних чисел.

Функція повертає кількість надрукованих символів.

Приклади використання

```
int int_val;

/* друкує ціле число */
printf( "Integer decimal formats %d\n", int_val );

/* друкує ціле число таким чином, щоб число містило не менше 6 розрядів */
/* (якщо число містить менше число розрядів, то вони доповнюються нулями зліва)*/
printf( "Integer decimal formats %.6d\n", int_val );

/* друкує ціле беззнакове число */
printf( "Unsigned integer %u\n", int_val );

/* друк числа в 16-вій системі числення */
printf( "Hex: %Xh or 0x%x\n", int_val, int_val );

/* друк числа в 8-вій системі числення */
printf( "Octal: %o\n", int_val );
```

```
float   float_val;

/* друк дійсного числа */
printf( "Real numbers: %f\n", float_val );
/* друк дійсного числа у експоненціяльному форматі */
printf( "Real numbers: %e\n", float_val );
```

scanf() – форматне введення з клавіатури

```
int scanf(char *format, <список вводу>);
```

Перший параметр є символьним рядком, який задає специфікації формату (див. функцію *printf()*). Інші параметри – список адрес змінних, в які вводяться дані. У цьому списку перед іменами всіх змінних, крім тих, які вводяться за специфікацією типу *%s*, повинен стояти символ *&*. Функція повертає кількість успішно перетворених полів.

Приклади використання

```
int x;
float y;
/* читання цілого числа */
printf("Input x:");
scanf("%d", &x);

/* читання дійсного числа */
printf("Input y:");
scanf("%f", &y);
```

Найпростіші програми мовою Сі

```
#include <stdio.h>

int main(void)
{
 printf("Hello, world!");
}
```

Програма друкує на екрані слова:

Hello, world!

Програма для розв’язання квадратного рівняння виглядає наступним чином:

```
#define _CRT_SECURE_NO_WARNINGS
#define _USE_MATH_DEFINES

#include <stdio.h>
#include <math.h>

/* Розв’язання рівняння a*x^2 + b*x + c = 0.      */
/* Коефіцієнти a, b, c вводяться з клавіатури.    */
/* Перевірка на існування коренів не проводиться! */

int main(void)
{
    float a, b, c, D, x1, x2;

    /* Читання коефіцієнтів рівняння з клавіатури */
    printf("a = ");
    scanf("%f", &a);
    printf("b = ");
    scanf("%f", &b);
```

```
    printf("c = ");
    scanf("%f", &c);

    D = b*b - 4*a*c; /* дискримінант */
    x1 = (-b - sqrtf(D)) / (2 * a);
    x2 = (-b + sqrtf(D)) / (2 * a);

    printf("x1 = %f\n", x1);
    printf("x2 = %f\n", x2);
}
```

Робоче завдання

1. Створити новий порожній проект або на локальній машині, або на сервері за бажанням.
2. В файл із назвою *main.c* скопіювати (між фігурних скобок) вміст таблиці згідно варіанту. Результат обчислень вивести на екран (використовувати функцію *printf()*).
3. Скомпілювати проект.
4. Провести покрокове налагодження:
    a. Встановити *Break Point* на певному рядочку всередині програми.
    b. Подивитися віконця *Watch* значення змінних *a, b, c, d.*
    c. Запустити на виконання програму у режимі налагодження.
    d. Виконати декілька рядків коду дивлячись як змінюються на кожному кроці значення змінних.
    e. Виконати програми до кінця та продемонструвати результат обчислення задачі згідно варіанту.

Варіант	Завдання

1.	int a = 0;
	int b = 0;
	int c = 0;

Варіант	Завдання

	int d = 0;
	
	a = 2;
	b += 3;
	c = a + b;
	d = a*b + c - d

2.	int a = 0;
	int b = 0;
	int c = 0;
	int d = 0;
	
	a = 2;
	b -= 3;
	c = a + b;
	d = a*b + c - d

3.	int a = 0;
	int b = 0;
	int c = 0;
	int d = 0;
	
	a = 2;
	b += 3;
	c = a - b;
	d = a*b + c - d

4.	int a = 0;
	int b = 0;
	int c = 0;
	int d = 0;
	
	a = 2;
	b += 3;
	c = a + b;
	d = a*b - c - d

5.	int a = 0;
	int b = 0;
	int c = 0;
	int d = 0;
	
	a = 2;
	b += 3;
	c = a + b;
	d = a*b + c + d

6.	int a = 10;
	int b = 5;
	int c = 2;
	int d = 1;
	
	a = 2;
	b += 3;
	c = a + b;
	d = a*b + c - d

7.	int a = 10;
	int b = 5;
	int c = 2;
	int d = 1;

Варіант	Завдання

	a = 2;
	b -= 3;
	c = a + b;
	d = a*b + c - d

8.	int a = 10;
	int b = 5;
	int c = 2;
	int d = 1;
	
	a = 2;
	b += 3;
	c = a - b;
	d = a*b + c - d

9.	int a = 10;
	int b = 5;
	int c = 2;
	int d = 1;
	
	a = 2;
	b += 3;
	c = a + b;
	d = a*b - c - d

10.	int a = 10;
	int b = 5;
	int c = 2;
	int d = 1;
	
	a = 2;
	b += 3;
	c = a + b;
	d = a*b + c + d

11.	int a = 10;
	int b = 5;
	int c = 2;
	int d = 1;
	
	a = 2;
	b -= 3;
	c = a - b;
	d = a*b - c - d

12.	float a = 0;
	float b = 0;
	float c = 0;
	float d = 0;
	
	a = 100.;
	b += 50.0;
	c = a + b;
	d = a/b + c*d

13.	float a = 0;
	float b = 0;
	float c = 0;
	float d = 0;

Варіант	Завдання

	a = 100.;
	b += 25.0;
	c = a + b;
	d = a/b + c*d

14.	float a = 0;
	float b = 0;
	float c = 0;
	float d = 0;
	
	a = 100.;
	b += 10.0;
	c = a + b;
	d = a/b + c*d

15.	float a = 0;
	float b = 0;
	float c = 0;
	float d = 0;
	
	a = 100.;
	b += 50.0;
	c = a - b;
	d = a/b + c*d

16.	float a = 0;
	float b = 0;
	float c = 0;
	float d = 0;
	
	a = 100.;
	b += 50.0;
	c = a + b;
	d = a*b + c/d

17.	float a = 0;
	float b = 0;
	float c = 0;
	float d = 0;
	
	a = 100.;
	b += 50.0;
	c = a + b;
	d = a/b - c*d

18.	float a = 0;
	float b = 0;
	float c = 0;
	float d = 0;
	
	a = 100.;
	b += 50.0;
	c = a + b;
	d = a*b - c/d

19.	float a = 0;
	float b = 0;
	float c = 0;
	float d = 0;
	
	a = 100.;

Варіант	Завдання

	b += 50.0;
	c = a - b;
	d = a*b - c/d

20.	float a = 0;
	float b = 0;
	float c = 0;
	float d = 0;
	
	a = 10.;
	b += 5.0;
	c = a + b;
	d = a*b + c/d

21.	float a = 0;
	float b = 0;
	float c = 0;
	float d = 0;
	
	a = 200.;
	b += 20.0;
	c = a + b;
	d = a*b + c/d

Контрольні питання

1. Що таке *IDE Code::Blocks* ?
2. Які засоби відлагодження в середовищі розробки *IDE Code::Blocks* вам відомі?
3. Чим відрізняються режими *Debug:Step into* та *Step out*?
4. Які функції стандартного вводу мови *C* вам відомі?
5. Які функції стандартного виводу мови *C* вам відомі?
6. Які ключ/ключі потрібно використовувати для друку/сканування чисел в Е-форматі?