**ЛАБОРАТОРНАРОБОТА№1**

**ВИМОГИ БЕЗПЕКИ ПІД ЧАС РОБОТИ В ЛАБОРАТОРІЇ ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ**

1. Ці правила є обов'язковими для всіх студентів та інших осіб, які працюють в лабораторіях постійно чи тимчасово.

2. Перевірка знань цих правил проводиться:

3. обслуговуючого персоналу – завідувачем лабораторією;

4. студентів – викладачами, які є керівниками лабораторних робіт.

5. Після перевірки знань цих правил, кожен з тих, хто працює в лабораторії ставить свій підпис в спеціальному журналі. Без цього підпису ніхто до роботи в лабораторії не повинен бути допущеним.

6. Дотримання правил з техніки безпеки повинно гуртуватися на високій свідомості всіх, хто працює в лабораторіях. Кожен, хто помітить порушення правил, а також несправність, яка являє собою небезпеку для людей і обладнання, повинен сповістити про це керівника.

7. Робота в лабораторії по виконанню конкретного завдання може проводитися тільки після ретельного ознайомлення студентів з обладнанням і роботою, та чіткого уявлення про те, які елементи установки будуть під напругою та дотик до яких є небезпечними у стані роботи.

8. Забороняється виконання ремонтних робіт на обладнанні, яке знаходиться під напругою.

9. Забороняється наближатися або торкатися до струмоведучих частин, що обертаються, або усувати несправності без відключення установок.

10. Забороняється проводити переключення в схемі, яка знаходиться під напругою.

11. Перевірку наявності напруги дозволяється проводити тільки за допомогою вольтметра.

12. Апарати управління та вимірювальні прилади слід розташовувати так, щоб було зручно вести спостереження за приладами, не перегинаючись через проводи та апарати.

13. У випадку виникнення будь-яких несправностей необхідно негайно вимкнути живленим установки та сповістити керівника занять про це.

14. Кнопки управління, рубильники встановлювати в легкодоступних місцях для швидкого виключення схеми.

15. У випадку припинення подачі електроенергії в лабораторію всі установки в лабораторії обов'язково вимикаються на робочих місцях.

16. В лабораторіях категорично забороняється:

* Палити в усіх приміщеннях, крім спеціально відведених для цього місць;
* Прокладати без дозволу постійні та тимчасові лінії;
* Користуватися побутовими електронагрівальними приладами;
* Користуватися зіпсованим електрообладнанням, саморобними запобіжниками, провідниками із зіпсованою ізоляцією та саморобними електросвітильниками;
* Проводити в непристосованих приміщеннях обмивку та фарбування деталей горючими рідинами та фарбниками;
* Зберігати паливно-мастильні матеріали, хімікати та інші горючі речовини;
* Загромаджувати проходи в лабораторіях;

17. Всі, хто працює в лабораторії повинні знати, де знаходиться аптечка з медикаментами для наданим першої допомоги.

18. При ураженні людини електричним струмом треба негайно вимкнути напругу, надати першу допомогу та покликати лікаря.

19. Порятунок осіб, які постраждали, залежить від того, як швидко вони будуть звільнені від електричного струму та як швидко їм буде надано першу допомогу.

20. Першу допомогу необхідно надати негайно на місці події.

21. Переносити людину, яка постраждала в інше місце необхідно тільки в тих випадках, коли небезпека продовжує загрожувати або надання допомоги на місці неможливе.

22. При відсутності у постраждалого дихання, серцебиття, пульсу ніколи не треба ставити під сумнів необхідність першої допомоги, тому що при ураженні електричним струмом смерть часто буває несправжньою. Тільки лікар може дати висновок про смерть постраждалого.

23. До приїзду лікаря постраждалому необхідно надати допомогу і провести штучне дихання з дотриманням всіх правил надання першої допомоги.

24. Про випадок негайно треба сповістити керівництво кафедри, деканату та інституту.

25. Недотримання цих вимог не дозволяється. Якщо розпорядження суперечить діючим правилам, необхідно надати роз'яснення з приводу неухильною виконання цих правил і довести це до відома керівництва.

# Введення в розробку програмного забезпечення

Перед написанням і виконанням нашої першої програми, ми повинні зрозуміти, з чого взагалі складається розробка програмного забезпечення на [**мові програмування C++**](https://acode.com.ua/urok-2-vvedennya-v-movy-programuvannya-s-i-s/)**.**

Проілюструємо процес розробки програмного забезпечення (скор. “ПЗ”):



## **Крок №1: Визначте проблему, яку потрібно вирішити**

Цей крок називається «**Що?**». Тут ви повинні зрозуміти, що ж саме ви хочете, щоб ваша програма виконувала. Цей крок може бути як найпростішим, так і найскладнішим. Все, що вам потрібно — це чітко сформулювати головну ідею вашої програми. Тільки після цього ви зможете перейти до наступного кроку.

Ось декілька прикладів виконання кроку №1:

* “Я хочу написати програму, яка буде вираховувати середнє арифметичне переданих їй чисел.”
* “Я хочу написати програму, в якій буде 2D-лабіринт, по якому користувач зможе пересуватися.”
* “Я хочу написати програму, яка аналізуватиме ціни акцій на біржах і прогнозуватиме скачки цін на акції вгору чи вниз.”

## **Крок №2: Визначте, як ви вирішуватимете проблему**

Тут ми відповідаємо на запитання «**Як?**». Як саме ми вирішуватимемо проблему з кроку №1? Цим кроком досить часто нехтують при розробці програмного забезпечення. Справа в тому, що способів вирішення проблеми може бути багато, тільки от деякі з цих способів можуть бути хорошими, деякі погані, а деякі просто жахливі. Ви повинні навчитися розрізняти якість пропонованих рішень. Дуже часто можна спостерігати ситуацію, коли у програміста виникає ідея і він відразу ж сідає за стіл програмувати. Як ви вже могли б здогадатися, такий сценарій далеко не завжди призводить до хороших рішень і хорошого коду.

Як правило, **більшість хороших рішень мають наступні характеристики**:

* простота;
* наявність документації (з інструкціями і коментарями);
* модульність: будь-яка частина програми може бути повторно використана або змінена пізніше без впливу на інші частини коду;
* надійність: наявність системи обробки помилок і екстрених випадків.

Коли ви відразу ж починаєте програмувати після виникнення геніальної ідеї, ви сідаєте за стіл і думаєте, що зробите це, ось це і ще ось це. Таким чином ви приймаєте рішення, які дозволять вам швидше виконати поставлене завдання. Однак це може призвести до того, що ви отримаєте програму, яку пізніше буде важко модифікувати, підтримувати або взагалі отримаєте десятки (якщо не сотні) помилок на етапі компіляції.

Згідно з [**законом Парето**](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF_%D0%9F%D0%B0%D1%80%D0%B5%D1%82%D0%BE), програміст витрачає близько 20% часу на написання програми і близько 80% часу на її відлагодження (виправлення помилок) або підтримку (додання нових функціональних можливостей). Тому цілком логічно, що краще витратити додатковий час на обдумування кращого способу вирішення проблеми перед фактичним написанням коду, ніж потім витрачати решту 80% часу на пошук та виправлення помилок.

## **Крок №3: Напишіть програму**

Для того, щоб написати програму, вам необхідно мати знання вибраної вами мови програмування та середовище розробки.

Програму можна написати, використовуючи будь-який редактор, навіть той же Блокнот в Windows або текстовий редактор Vi в Unix. Проте ми настійно рекомендую використовувати середовище розробки.

Редактор типового програміста, як правило, має наступні особливості, які полегшують процес програмування:

**Нумерація рядків**. Ця функція надзвичайно корисна при відлагодженні програми, коли компілятор повідомляє нас про наявність помилок. Типова помилка компіляції складається з “помилка” і “№ рядка“, де ця помилка сталася, наприклад: “помилка перевизначення змінної x, рядок 80“. Без нумерації рядків шукати той самий 80-й рядок коду було б трохи важко, чи не так?

**Підсвічування синтаксису**. Підсвічування синтаксису змінює колір різних частин програми і коду, що покращує сприйняття як цілої програми, так і окремих її частин.

**Спеціальний шрифт**. В звичайних шрифтах дуже часто виникає плутанина через певні символи, коли незрозуміло, що саме знаходиться перед вами. Наприклад, цифра 0 чи буква O, цифра 1 чи буква l (нижній регістр L), чи, можливо, буква I (верхній регістр i). Ось саме в таких випадках і потрібен спеціальний шрифт, в якому можна чітко відрізнити дані символи, запобігаючи випадкове використання одного символу замість іншого.

Файли з вашими програмами на мові C++ слід називати **name.cpp**, де name замінюється назвою вашої програми, а розширення .cpp повідомляє компілятору (і вам також), що це ваш файл коду, який містить інструкції на мові програмування C++. Слід звернути увагу, що деякі програмісти використовують розширення .cc замість .cpp, проте рекомендую використовувати саме .cpp.

Також варто відзначити, що програми в C++ можуть складатися з декількох .cpp-файлів. Хоча більшість програм, які ви створюватимете, не будуть надто великими, у вашій подальшій практиці ви матимете змогу навчитися писати програми, які складатимуться з десятків, якщо не з сотень окремих .cpp-файлів .

## **Крок №4: Виконайте компіляцію програми**

Для того, щоб виконати компіляцію програми, нам потрібен компілятор. **Робота компілятора складається з двох частин:**

Перевірка програми на відповідність правилам мови C++ (перевірка синтаксису). Якщо вона буде невдалою, то компілятор повідомить вас про помилки, які потрібно буде виправити.

Конвертація кожного вашого файлу з кодом в **об’єктний файл** (або “об’єктний модуль”) на [**машинній мові**](https://acode.com.ua/urok-1-vvedennya-v-programuvannya/#toc-0). Об’єктні файли, як правило, називаються **name.o** або **name.obj**, де name повинне бути таке ж, як і назва вашого вихідного .cpp-файлу. Якщо ваша програма складається з трьох .cpp-файлів, то компілятор згенерує 3 об’єктних файли.



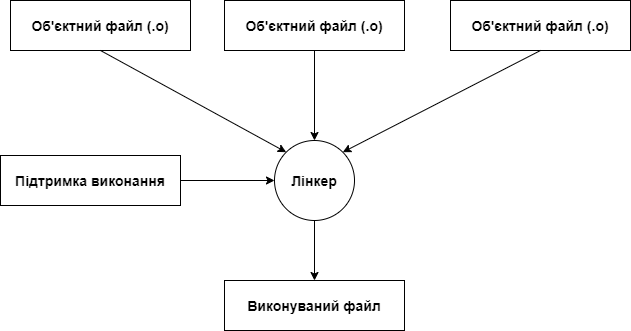
Варто зазначити, що такі операційні системи як Linux і macOS мають вже вбудований компілятор C++, який називається **g++**. Для компіляції файлів з командного рядка за допомогою g++ вам необхідно написати наступне:

g++ -c file1.cpp file2.cpp file3.cpp

Таким чином ми створили file1.o, file2.o и file3.o. -c означає “тільки скомпілювати”, тобто створити .o (об’єктні) файли. Також є і інші компілятори для Linux, Windows та інших операційних систем.

## **Крок №5: Виконайте лінкінг програми (з’єднайте об’єктні файли)**

**Лінкінг** — це процес *зв’язування* всіх об’єктних файлів, що генеруються компілятором, в єдину виконувану програму, яку ви потім зможете запустити (виконати). Це можна зробити за допомогою програми, яка називається **лінкер** (або ***“компоновщик”***).



Крім об’єктних файлів, лінкер також підключає файли зі Стандартної бібліотеки С++ (або будь-якої іншої бібліотеки, яку ви використовуєте). Сама по собі мова С++ є досить невеликою та простою. Проте до неї підключається велика бібліотека додаткових функцій, які можуть використовувати ваші програми, і ці функції знаходяться в Стандартній бібліотеці C++. Наприклад, якщо ви хочете вивести що-небудь на екран, то у вас в коді повинна бути спеціальна команда, яка повідомить компілятору, що ви плануєте використати функцію виведення інформації на екран зі Стандартної бібліотеки C++.

Після того, як лінкер закінчить з’єднувати всі об’єктні файли (за умови, що не буде помилок), ви отримаєте виконуваний файл. Знову ж таки, щоб зв’язати *.o-*файли, які ми створили вище в Linux або в macOS, ми можемо скористатися g++:

g++ -o prog file1.o file2.o file3.o

Команда -o повідомляє g++, що ми хочемо отримати виконуваний файл з іменем prog з наступних файлів: file1.o, file2.o і file3.o. При бажанні, компіляцію і лінкінг можна об’єднати в один крок:

g++ -o prog file1.cpp file2.cpp file3.cpp

Результатом виконання стане виконуваний файл з іменем prog.

## **Крок №6:** Виконайте тестування та відлагодження програми

Ви вже можете запустити виконуваний файл і подивитися, чи все працює так як треба. Якщо ні, то настав час для відлагодження вашої програми.

Зверніть увагу, що для виконання кроків №3-№6 вам знадобиться спеціальне програмне забезпечення. Хоча ви можете використовувати окремі програми на кожному з цих кроків, середовище розробки (скор. **“IDE”** від англ. “**I**ntegrated **D**evelopment **E**nvironment”) об’єднує всі ці програми в собі. Зазвичай з IDE ви отримуєте редактор коду з нумерацією рядків і підсвічуванням синтаксису, а також компілятор та лінкер. А коли вам потрібно буде виконати відлагодження програми, то ви зможете використати вбудований відлагоджувач. Крім того, IDE має ще ряд інших корисних можливостей: доповнення коду, система контролю версій тощо.

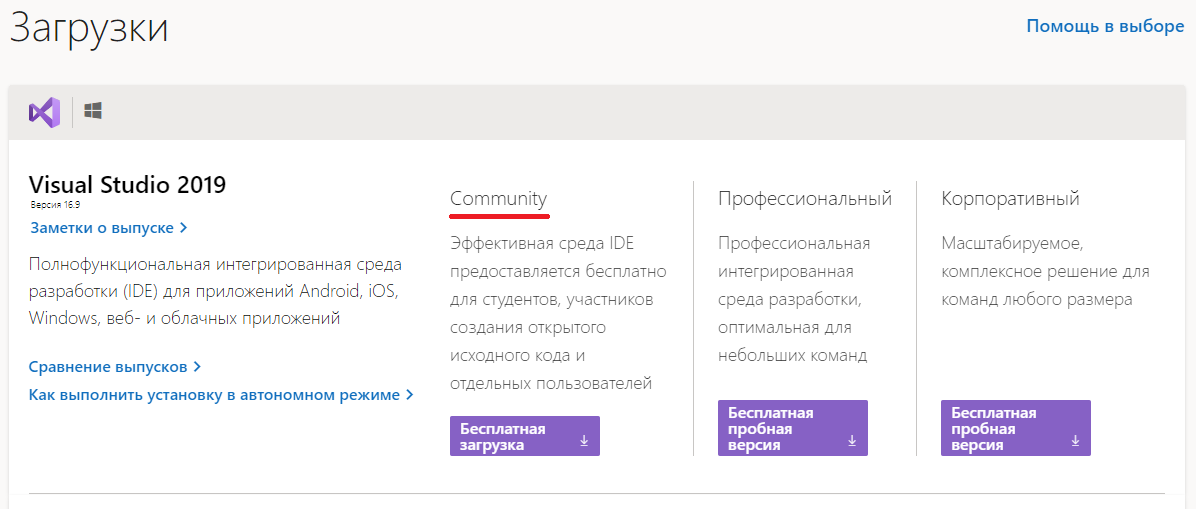
**Встановлення IDE (Інтегрованого Середовища Розробки)**

**Інтегроване середовище розробки** (скор. **“IDE”** від англ. “***I***ntegrated ***D***evelopment ***E***nvironment”) — це програмне забезпечення, яке містить все необхідне для розробки, компіляції, [**лінкінгу**](https://acode.com.ua/urok-3-vvedennya-v-rozrobku-programnogo-zabezpechennya/#toc-4) та відлагодження коду програм. Нам необхідно встановити одну з таких IDE для написання програм мовою С++.

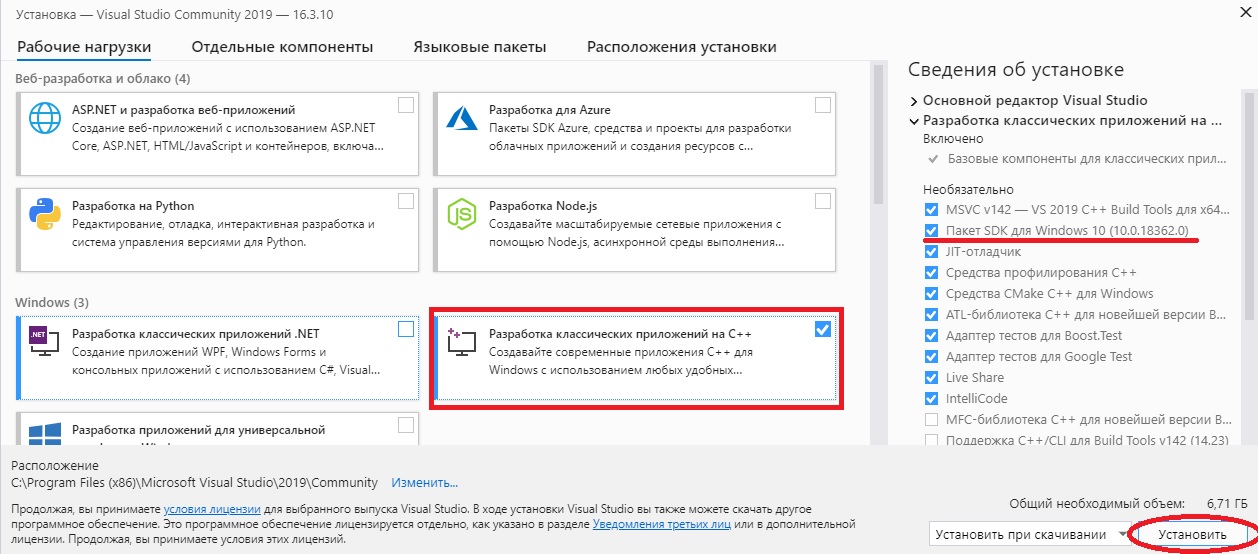
Але яке саме інтегроване середовище розробки вибрати? Я рекомендую Visual Studio від Microsoft (для користувачів Windows) або Code::Blocks (для користувачів Linux/Windows). Також ви можете встановити і будь-яку іншу IDE. Основні концепції, які розглядатимуться, працюватимуть у всіх середовищах розробки. Втім, іноді код може частково відрізнятися в різних IDE, тому вам доведеться самостійно шукати більш детальну інформацію про специфіку роботи в обраній вами IDE.

**IDE для користувачів Windows**

Якщо ви користувач Windows, то можете встановити [**Visual Studio 2019**](https://visualstudio.microsoft.com/ru/downloads/) версію “Community”, яка є безкоштовною (всі інші версії платні):



Після того, як ви скачаєте і запустите інсталятор, вам потрібно буде вибрати пункт "Разработка классических приложений на C++". Пункти, вибрані за замовчуванням в правій частині екрана, не варто чіпати — там все добре, тільки переконайтеся, що поставлена галочка біля пункту "Пакет SDK для Windows 10". Цей пакет може використовуватися і в більш ранніх версіях Windows, тому не переживайте, якщо у вас Windows 7 або Windows 8 — все працюватиме. Після цього натискаємо "Установить":



При бажанні ви можете вказати галочки і біля інших пунктів для скачування, але врахуйте, що тоді розмір завантажуваного вами середовища розробки буде збільшено.

# Компіляція вашої першої програми

Перед написанням нашої першої програми ми повинні ще дещо дізнатися.

**Теорія**

По-перше, незважаючи на те, що код наших програм знаходиться в *.cpp-*файлах, ці файли додаються в проект. **Проект** містить всі необхідні файли вашої програми, а також зберігає вказані вами налаштування вашої [**IDE**](https://acode.com.ua/urok-4-vstanovlennya-ide-integrovanogo-seredovyshha-rozrobky/). Кожен раз, при відкритті вашого проекту, він запускається з того моменту, на якому ви зупинилися минулого разу. При компіляції програми, проект повідомляє компілятору і лінкеру, які файли необхідно скомпілювати, а які зв’язати. Варто відзначити, що файли проекту однієї IDE не працюватимуть в іншій IDE. Вам доведеться створювати новий проект (в іншій IDE).

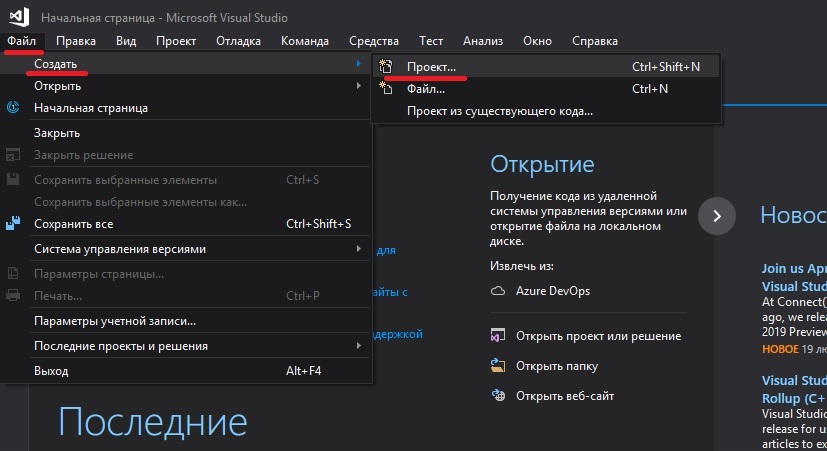
По-друге, є різні типи проектів. При створенні нового проекту, вам потрібно буде вибрати його тип. Всі проекти, які ми створюватимемо, будуть **консольного типу**. Це означає, що вони запускаються в консолі (аналог командного рядка). За замовчуванням, консольні додатки не мають графічного інтерфейсу користувача (скор. ***“GUI”*** від англ. *“****G****raphical* ***U****ser* ***I****nterface”*) і компілюються в автономні виконувані файли. Це ідеальний варіант для вивчення C++, оскільки він зводить всю складність до мінімуму.

По-третє, при створенні нового проекту більшість середовищ розробки автоматично додадуть ваш проект в робочий простір. **Робочий простір** — це своєрідний контейнер, який може містити один або декілька пов’язаних проектів. Незважаючи на те, що ви можете додавати декілька проектів в один робочий простір, все ж рекомендується створювати окремий робочий простір для кожної програми. Це набагато простіше для початківців.

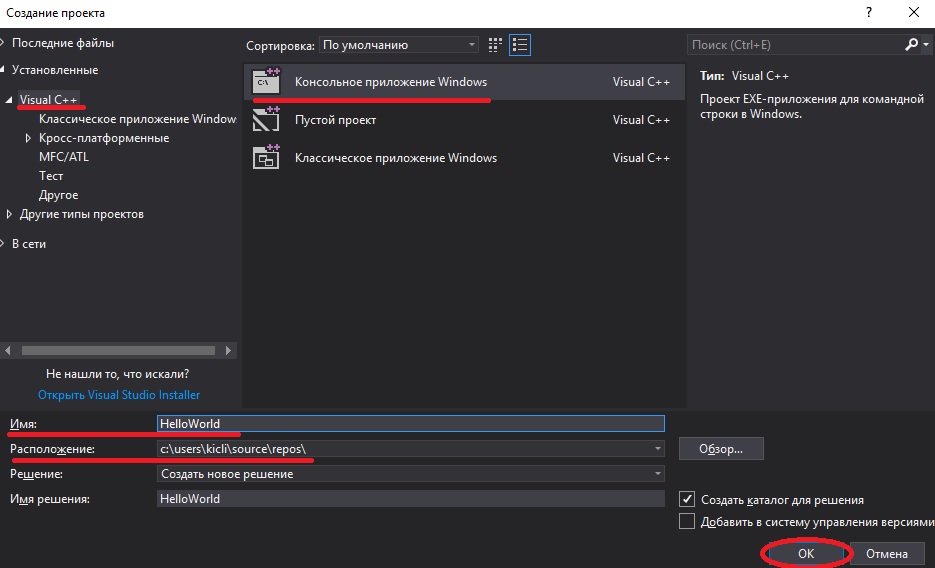
Традиційно, нашою першою програмою на мові С++ буде всім відома програма «Hello, world!». Ми не будемо порушувати традиції

**Користувачам Visual Studio**

Для створення нового проекту в Visual Studio, вам необхідно спочатку запустити Visual Studio, а потім вибрати "Файл" > "Создать" > "Проект":



Далі з’явиться діалогове вікно, де вам потрібно буде вибрати "Консольное приложение Windows" на вкладці "Visual C++" і натиснути"ОК"

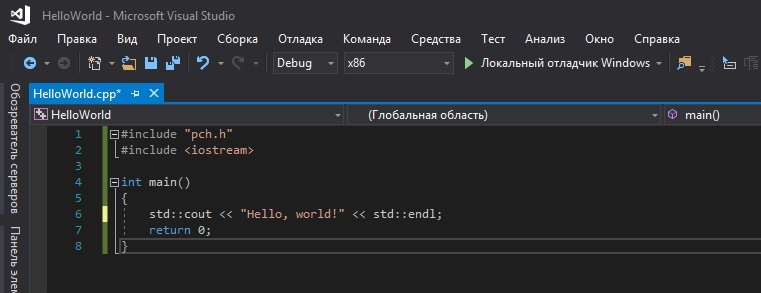


Також ви можете вказати ім’я проекту (будь-яке) і його місцезнаходження (рекомендую нічого не змінювати) в відповідних полях.

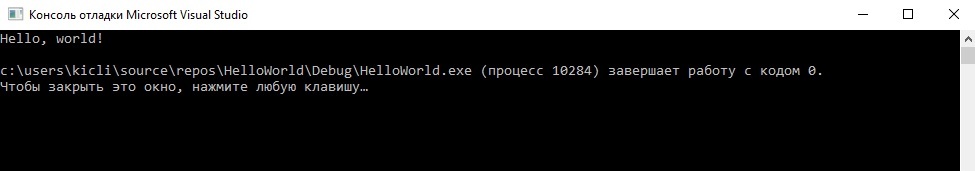
В текстовому редакторі ви побачите, що в вашому робочому просторі вже є написаний текст з кодом. Видаліть все, що там написано і напишіть наступний код:



Ось, що у вас повинно бути:



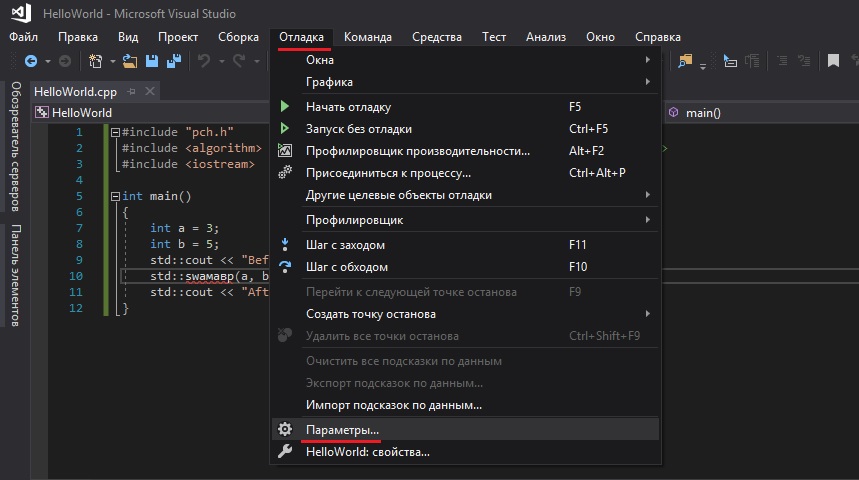
Щоб запустити програму в Visual Studio, натиснітьCtrl+F5. У вас повинно вийти наступне:



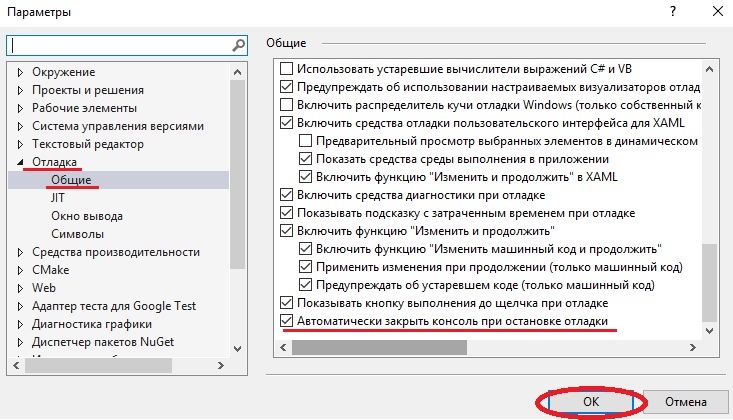
Це означає, що компіляція виконалася успішно і результат виконання вашої програми:

Hello, world!

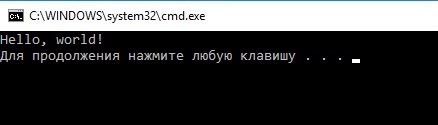
Щоб видалити рядок "...завершает работу с кодом 0...", вам необхідно перейти в "Отладка" > "Параметры":



Потім "Отладка" > "Общие" і поставити галочку біля "Автоматически закрыть консоль при остановке отладки" і натиснути "ОК":



Тепер ваше консольне вікно виглядатиме наступним чином:



Вітаю вас! Ви навчилися компілювати програми в Visual Studio!

# Вирішення найбільш поширених проблем початківців в С++

**Як використовувати кирилицю в програмах С++?**

**Рішення**

Щоб використовувати кирилицю в програмах на мові C++, вам необхідно підключити заголовковий файл Windows.h:



І прописати два наступних рядки в функції main():



**При запуску програми з’являється чорне консольне вікно, а потім відразу зникає.**

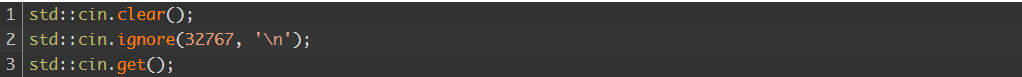
**Рішення**

Деякі компілятори (наприклад, Bloodshed’s Dev C++) автоматично не затримують консольне вікно після того, як програма завершила своє виконання. Якщо проблема в компіляторі, то наступні два кроки допоможуть вам вирішити цю проблему:

***Крок №1*: Додайте наступний рядок коду в верхню частину вашої програми:**

******

***Крок №2*: Додайте наступний код в кінець функції main() (перед оператором** return**):**



Таким чином ваша програма очікуватиме натискання клавіші, щоб закрити консольне вікно. Ви отримаєте додатковий час, щоб добре все роздивитися/вивчити. Після натискання будь-якої клавіші, консольне вікно закриється.

Інші рішення, такі як system("pause");, можуть працювати не на всіх типах операційних систем, тому краще використовувати вищезазначений варіант.

***Примітка***: Visual Studio не затримує консольне вікно, якщо виконання програми запущено з відлагодженням ("Отладка" > "Начать отладку" або F5). Якщо ви хочете, щоб була пауза, то скористайтеся вищезазначеним рішенням, або запустіть програму без відлагодження ("Отладка" > "Запуск без отладки" або Ctrl+F5).

# Налаштування компілятора: Режими конфігурації “Debug” і “Release”

**Конфігурація збірки** (англ. “build configuration”) — це набір налаштувань проекту, які визначають принцип його побудови. Конфігурація збірки складається з:

імені виконуваного файлу;

директорії виконуваного файлу;

в яких директоріях IDE шукатиме код та заголовкові файли;

інформації про відлагодження та параметри оптимізації вашого проекту.

Ваше [**інтегроване середовище розробки**](https://acode.com.ua/urok-4-vstanovlennya-ide-integrovanogo-seredovyshha-rozrobky/) має дві конфігурації збірки: “Debug” (Дебаг/Відлагодження) і “Release” (Реліз).

**Конфігурація “Debug”** призначена для відлагодження вашої програми. Ця конфігурація відключає всі налаштування по оптимізації та включає інформацію про відлагодження, що робить ваші програми більшими і повільнішими, але спрощує проведення відлагодження. Режим “Debug” зазвичай використовується в якості конфігурації збірки за замовчуванням.

**Конфігурація “Release”** використовується для побудови програми з метою її подальшого публікування. Програма оптимізується за розміром і продуктивністю і не містить додаткової інформації про відлагодження.

## Перемикання режимів “Debug” і “Release” в Visual Studio

Найпростіший спосіб змінити конфігурацію проекту — це вибрати відповідну конфігурацію зі списку на панелі швидкого доступу:

# 

## Перемикання режимів “Debug” і “Release” в Code::Blocks

У Code::Blocks на панелі швидкого доступу також є випадаючий список, де ви можете вибрати відповідний режим конфігурації:

**Висновки**

Використовуйте конфігурацію “Debug” при розробці програм, а конфігурацію “Release” при їх публікації (коли ви вже будете готові представити вашу програму на загальний огляд).

# Налаштування компілятора: Розширення компілятора

**Розширення компілятора**

Стандарт C++ визначає правила щодо того, як програми повинні себе поводити в конкретних ситуаціях. І в більшості випадків компілятори також дотримуються цих правил. Однак багато компіляторів впроваджують власні зміни в мову програмування, часто для підвищення сумісності з іншими версіями мови (наприклад, C99). Ці поведінкові зміни, які є різними в різних компіляторах, називаються **розширеннями компілятора**.

Використовуючи розширення компілятора ви отримуєте можливість написання програм, які є несумісними зі стандартом C++. Програми, які використовують нестандартні розширення, зазвичай, не компілюватимуться в інших компіляторах (які не підтримують ці ж розширення), або працюватимуть некоректно.

Дуже часто розширення компілятора є включеними за замовчуванням. Особливо це стосується початківців, коли вони можуть вважати поведінку, специфічну для певного компілятора, як частину офіційного стандарту C++ (коли насправді це не так).

Оскільки розширення компілятора рідко коли потрібні і можуть зробити ваші програми невідповідними стандарту C++, то я раджу вимикати розширення компілятора.

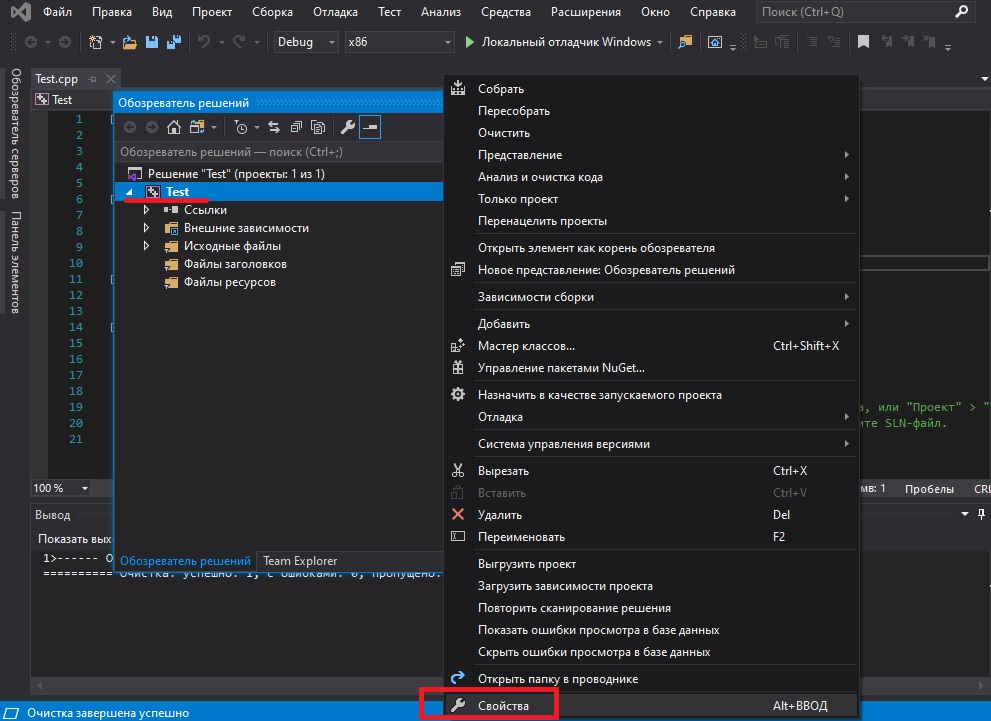
***Порада*:** Вимикайте розширення компілятора, щоб бути певними, що ваші програми залишаються сумісними зі стандартом C++ і працюватимуть у будь-якій системі.

***Примітка***: Ці налаштування застосовуються до кожного проекту окремо. Вам потрібно буде це все повторювати для кожного новоствореного вами проекту, або створити шаблон проекту з цими налаштуваннями і тоді вже по ньому створювати нові проекти.

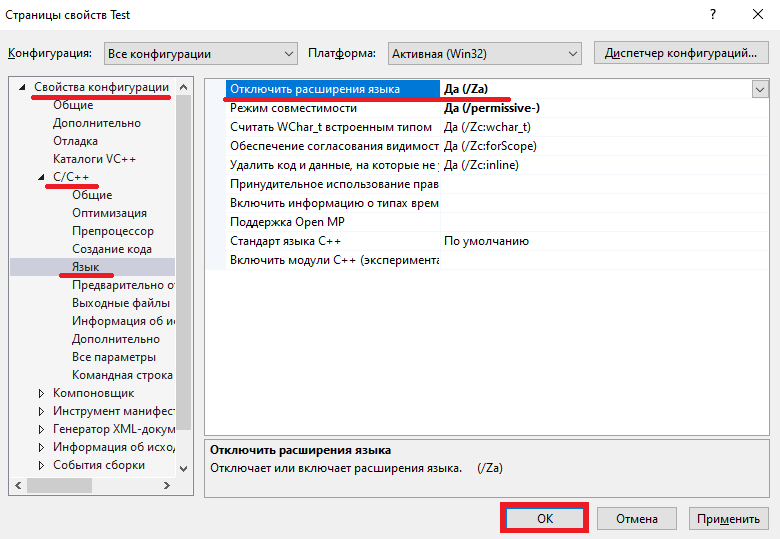
**Вимкнення розширень компілятора**

**Користувачам Visual Studio**

Щоб вимкнути розширення компілятора в Visual Studio, клацніть правою кнопкою миші по назві вашого проекту в "Обозреватель решений" > "Свойства":



У діалоговому вікні проекту переконайтесь, що в полі "Конфигурация" вибрано пункт "Все конфигурации". Потім перейдіть на вкладку "C/C++" > "Язык" та встановіть значення "Да (/Za)" для пункту "Отключить расширения языка", після цього натисніть "OK":



# Вибір стандарту мови С++

Як із величезної кількості різних версій C++ (C++98, C++03, [**C++11**](https://acode.com.ua/cpp-11-novovvedennya/), [**C++14**](https://acode.com.ua/cpp-14-novovvedennya/), [**C++17**](https://acode.com.ua/cpp-17-novovvedennya/), [**C++20**](https://acode.com.ua/cpp-20-novovvedennya/)) компілятор розуміє, яку з них йому слід використовувати? Як правило, компілятор вибирає стандарт мови програмування за замовчуванням (дуже часто це не є найновішою версією мови). Якщо ви хочете використовувати інший стандарт, то вам необхідно буде внести певні зміни в налаштування вашої [**IDE**](https://acode.com.ua/urok-4-vstanovlennya-ide-integrovanogo-seredovyshha-rozrobky/)/компілятора. Варто зазначити, що дані налаштування застосовуються лише до поточного проекту. При створенні нового проекту вам доведеться все робити заново.

**Кодові імена версій мови С++**

Зверніть увагу, що стандарти мови С++ названі в честь тих років, в які вони були завершені/опубліковані (наприклад, C++17 був завершений та опублікований в 2017 році).

Однак, коли узгоджується новий стандарт мови програмування, то дуже часто буває незрозуміло, в якому році вдасться його завершити, тому поточному стандарту мови, який розробляється, дають кодове ім’я, яке потім замінюється на фактичну назву версії при публікуванні. Наприклад, C++11 мав кодове ім’я c++1x, коли над ним вели роботу. Ви і досі можете натикатися на подібні кодові імена версій С++, особливо, коли мова заходить про майбутню версію мови С++, у якої ще немає остаточної назви.

Нижче подані кодові імена версій С++ і їх остаточні назви:

c++1x = C++11

c++1y = C++14

c++1z = C++17

c++2a = C++20

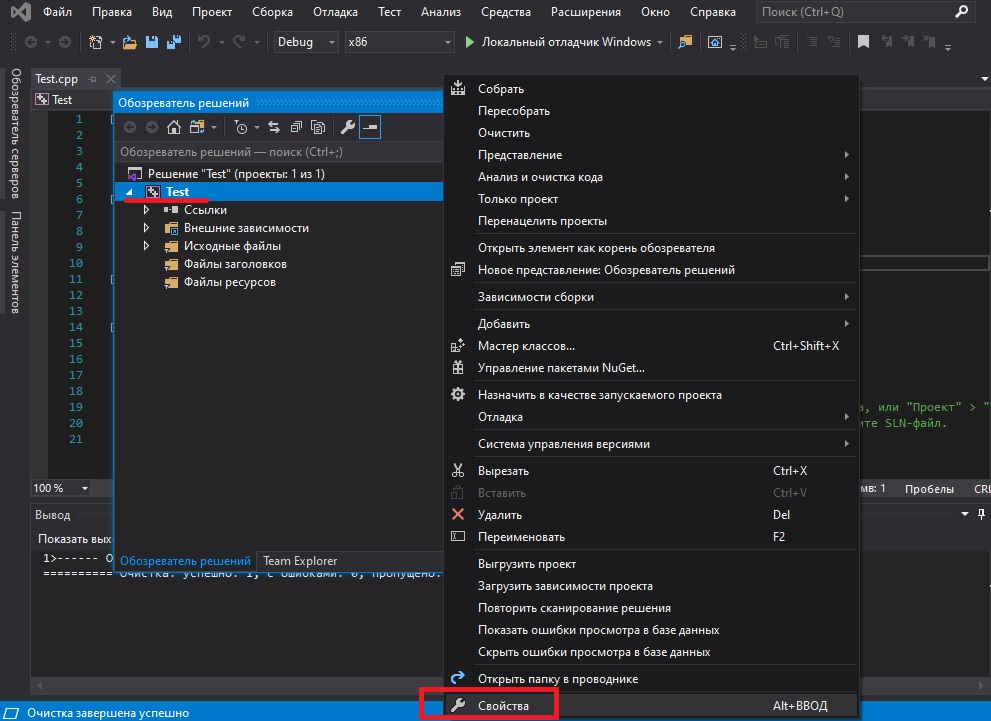
Наприклад, якщо ви зустріли назву c++1z, то знайте, що це є синонімом стандарту мови C++17.

**Вибір стандарту мови С++ в Visual Studio**

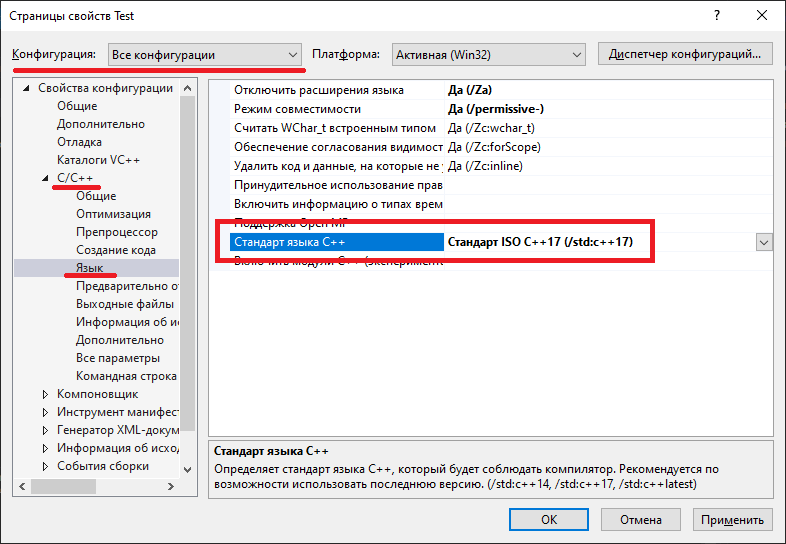
На момент написання вказівки, Visual Studio 2019 за замовчуванням використовує функціонал версії C++14, що не дозволяє використовувати нові можливості, представлені в C++17 (і в C++20).

Щоб використовувати новий функціонал, вам необхідно підключити новий стандарт мови. На жаль, зараз немає способу зробити це глобально — ви повинні робити це для кожного проекту окремо.

Щоб використовувати новий стандарт мови С++ в Visual Studio, відкрийте ваш проект, клацніть правою кнопкою миші по назві вашого проекту в "Обозреватель решений" > "Свойства":



В діалоговому вікні вашого проекту переконайтеся, що в пункті "Конфигурация" вибрано значення "Все конфигурации". Після цього перейдіть на вкладку "C/C++" > "Язык" і в пункті "Стандарт языка С++" виберіть ту версію С++, яку ви хотіли б використовувати:



# 1.3. Контрольні запитання

1. Які є основні кроки для розроблення програм?
2. Що робить компілятор?
3. Що робить компонувальник?
4. Що робить відлагоджувач?
5. Чим об’єктний код відрізняється від виконуваного?
6. Що таке процедурне програмування?
7. Що таке інтегроване середовище розробки програмного забезпечення?
8. Які дії необхідно виконати для створення консольного проекту в інтегрованому середовищі розробки VS?

# 1.4. Лабораторне завдання

1. Прослухати інформацію про порядок роботи в комп’ютерному класі. Проаналізувати інформацію на робочому столі Вашого ПК, запам’ятати порядок роботи (включення, ввід паролю, виключення).
2. Запустити середовище VS, розглянути всі команди, запам’ятати найбільш необхідні для роботи команди.
3. У каталозі **Student** на диску **D** створити каталог з назвою своєї групи. В ньому створити каталог з назвою свого прізвища. В ньому створити консольний проект під назвою "**Lab1**". Усі назви не повинні містити в собі пробіли, крапки, коми та інші розділові знаки.
4. У проекті створити файл коду програми на мові С++.
5. Написати код програми "Hello world!", скомпілювати, скомпонувати та запустити програму до виконання.
6. Модифікувати програму так, щоб вона виводила на консоль ваше прізвище.
7. Скомпілювати, скомпонувати та запустити на виконання програму з допомогою консольних команд.

**1.5. Зміст звіту** 1. Титульний аркуш.

1. Мета роботи.
2. Хід виконання лабораторного завдання.
3. Текст модифікованої програми.
4. Результати виконання модифікованої програми.
5. Аналіз результатів та висновки.