МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»

# МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт

**з курсів: «Інформаційні технології і програмування»**
**«Інформаційні процеси в телекомунікаційних системах»**

# ПРОГРАМУВАННЯ НА МОВІ С++

для студентів спеціальностей:
151 Автоматизація та комп'ютерно-інтегровані технології
172 Телекомунікації та радіотехніка

затверджено
редакційно-видавничою
радою університету,
протокол №3 від 26.10.2022 р.

Харків
НТУ «ХПІ»
2022

Методичні вказівки для виконання лабораторних робіт «Програмування на мові С++» для студентів спеціальностей 151 Автоматизація та комп'ютерно-інтегровані технології, 172 Телекомунікації та радіотехніка / Уклад. А.О. Зуєв, Д.Г. Караман, М.А. Денисенко - Х.: НТУ «ХПІ», 2022. - 35 с.

Укладачі: А. О. Зуєв
Д. Г. Караман
М.А. Денисенко

Рецензент Д.А. Гапон

Кафедра автоматизації та кібербезпеки енергосистем

# ВСТУП

Розвиток сучасних технологій неможливий без використання комп'ютерної техніки та програмного забезпечення. Підготовка фахівців в області систем управління, автоматики та телекомунікаційних систем вимагає широких знань і навичок володіння обчислювальною технікою, а так само знання основ алгоритмізації та мов програмування високого рівня.

Методичні вказівки призначені для вивчення основ мови програмування С++ як на практичних і лабораторних роботах, так і для самостійного вивчення. У методичних вказівках на прикладах розглядаються основні принципи програмування та методи написання програм на мові С++, а саме: вбудовані і користувацькі типи даних, структури і масиви та їх ініціалізація і використання, оператори мови С++, логічні і математичні функції, цикли і умовні оператори, функції, передача аргументів і повернення результату роботи функції, шаблони і обробка виключних ситуацій, а також основи роботи з потоками введення-виведення.

Наведено індивідуальні завдання для виконання практичних і лабораторних робіт, а також завдання для самостійного вивчення мови програмування. Розглянуті приклади та завдання допоможуть в ефективному освоєнні програмування на мові С++.

# ЗАГАЛЬНІ ВІДОМОСТІ

Кожна програма на мові C++ складається з двох типів файлів:

1) заголовних (з розширенням .h або .hpp), в яких зазвичай містяться описи констант, глобальних змінних, класів, структур і функцій, підключення бібліотек.

2) файлів реалізації (з розширенням .cpp), в яких міститься компільований код.

Заголовні файли підключаються директивою #include до файлів реалізації. Після директиви вказують в кутових дужках назву системної бібліотеки, або в лапках назву файлу з проекту що розробляється.

Будь-яка програма на мові C++ містить як мінімум одну функцію з назвою main (_tmain), з якої починається її виконання.

Кожен рядок програми закінчується символом крапки з комою. Код програми і опис класів та структур обертається в фігурні дужки {}. Також фігурні дужки використовуються для позначення тіл операторів, або довільних блоків коду.

Багаторядкові коментарі обмежуються символами /* на початку і */ в кінці. Однорядковий коментар починається з символів // і закінчується в кінці рядка.

## Лабораторна робота №1
## Вивчення принципів побудови консольних додатків у середовищі MS Visual Studio

**Мета роботи:** Навчитися використовувати середовище MS Visual Studio для створення консольних додатків в операційній системі Windows.

**Необхідне обладнання та комплектуючі, ПЗ:** персональний комп'ютер з встановленою операційною системою (ОС) Windows, середовище MS Visual Studio.

### Загальні відомості

Програма на мові C++ складається з двох типів файлів, заголовкових (розширення **.h**, **.hpp**, **.inl**) та файлів реалізації (розширення **.cpp**). Компілюються тільки файли реалізації, а заголовки повинні бути підключені до них за допомогою директиви **#include**, після якої пишеться ім'я файлу (і шлях до нього якщо потрібно). Якщо ім'я записано в подвійних лапках - при компіляції вони шукаються в папках проекту, а якщо в кутових дужках - то пошук починається з системних каталогів та бібліотек.

Програма на мові C++ містить як мінімум одну функцію з якої починається її виконання. Зазвичай вона має в назві слово **main** і автоматично створюється при створенні проекту у середовищі для розробки. Функція повертає ціле число (типу **int**) - код помилки виконання. Якщо помилки немає повертається 0.

```
int main()
{
	return 0;
}
```

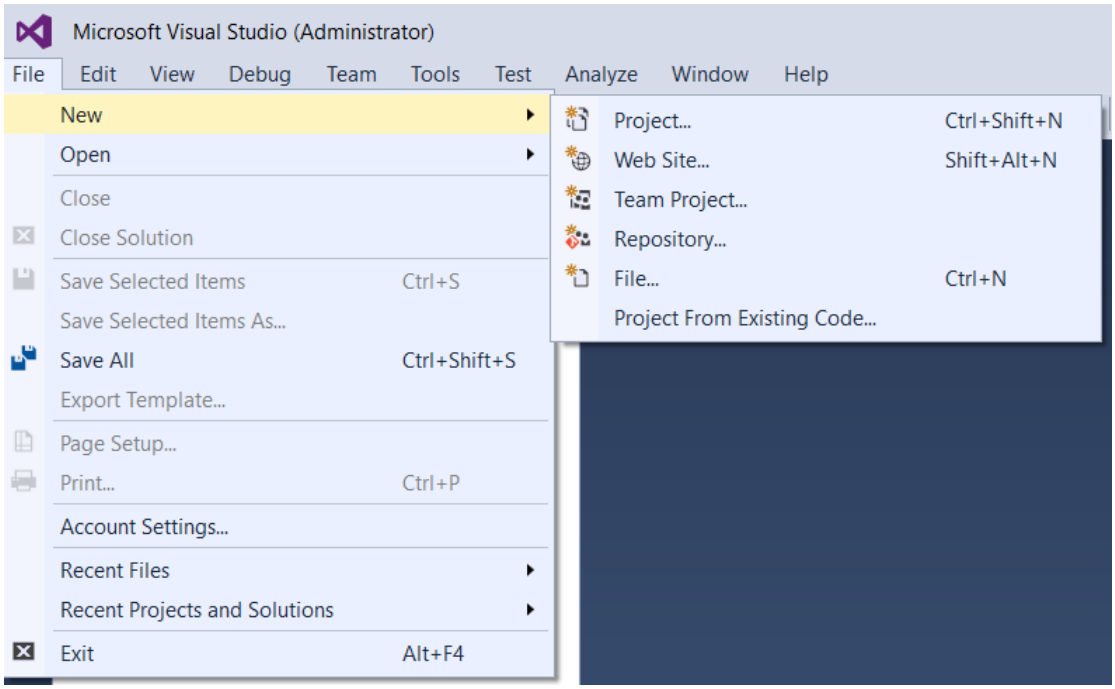
Будь яка інша функія програми має бути викликана з цієї функції, або з іншої функції, що викликається з **main**.

Мова C++ потребує компіляції програми, це означає що перед запуском програми, необхідно її “перекласти” з мови C++ до машинного коду, який має бути собраний до спеціального файлу, що виконується (зазвичай з розширенням **.exe**). Після чого, якщо в процесі “перекладу” не було помилок, отриману програму можна запустити під управлінням операційної системи. Процес “перекладу” складається з двох частин, власне компіляції (перетворення програми до машинного коду) і процесу лінкування, в якому різні фрагменти програми збираються в єдиний файл (або файли). Якщо на якомусь з етапів виникне помилка, або бракує якихось фрагментів, процес зупиняється і файл, що виконується, не буде отриманий.

Така, роздільна компіляція, дозволяє збирати програму по частинах або заміняти тільки ті частини, що змінювалися. Це також дозволяє здійснювати процес компіляції паралельно. Файли, що раніше були откомпільовані, можуть бути використані при складанні програми - цей механізм використовується для створення статичних бібліотек коду.

## Хід виконання роботи

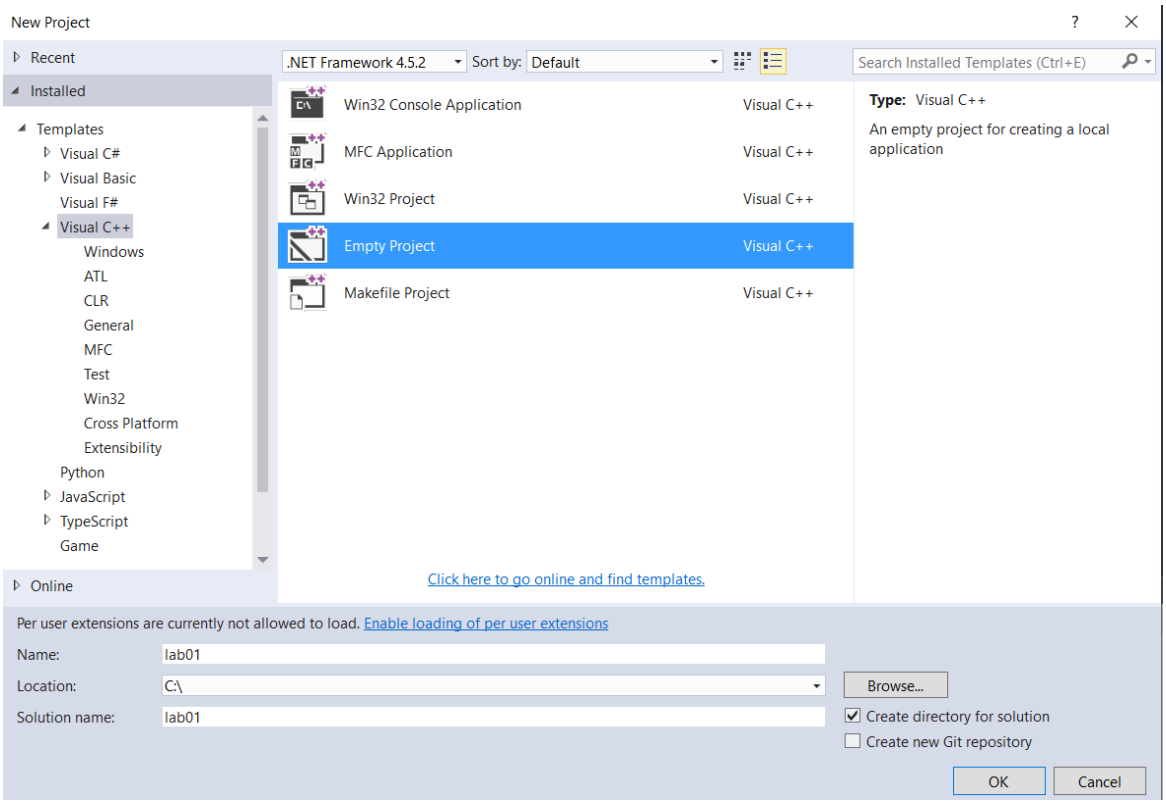
1. Запустити середовище розробки MS Visual Studio в якому вибрати пункт меню **File/New/Project.**



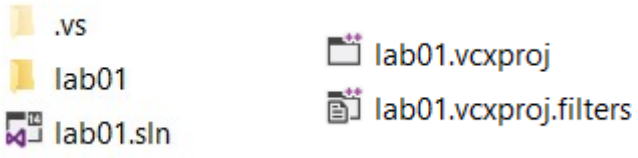

2. Вибрати тип проекту **Visual C++\Empty Project.**

3. Ввести назву проекту в полі **Name: - lab01.**

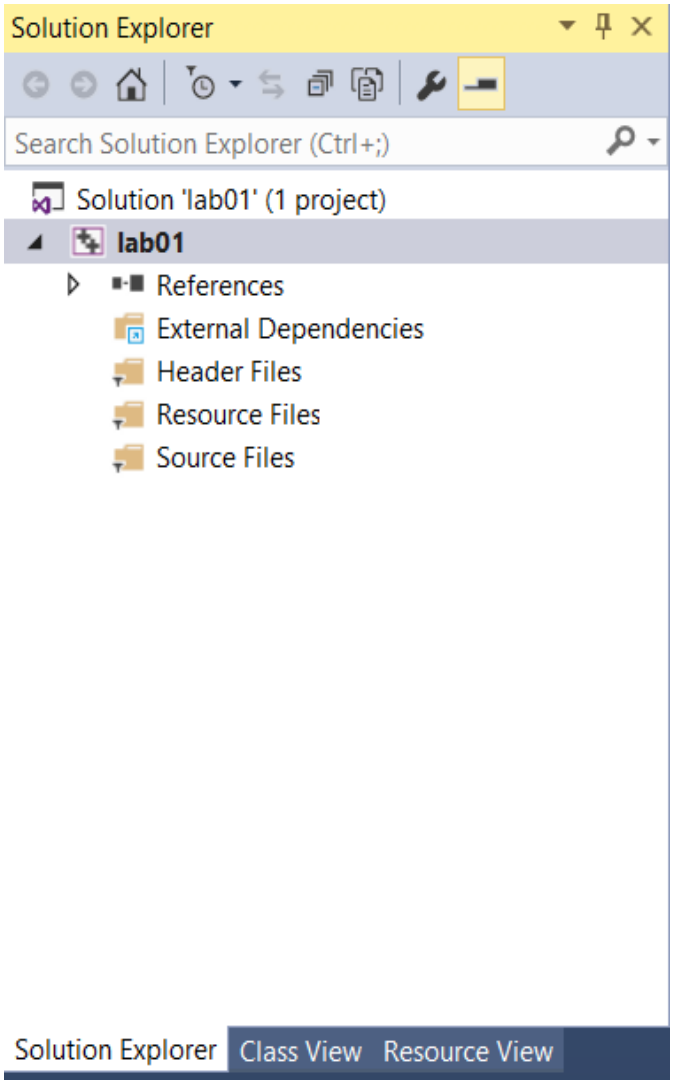
4. Вибрати місцезнаходження проекту (папку проекту) в полі **Location:**, для цього натисніть кнопку **Browse….**



5. У вибраній папці з'явиться файл рішення **(lab01.sln)** необхідний для роботи. А також папка проекту **lab01** з файлом опису проекту і фільтром **(lab01.vcxproj, lab01.vcxproj.filters).**

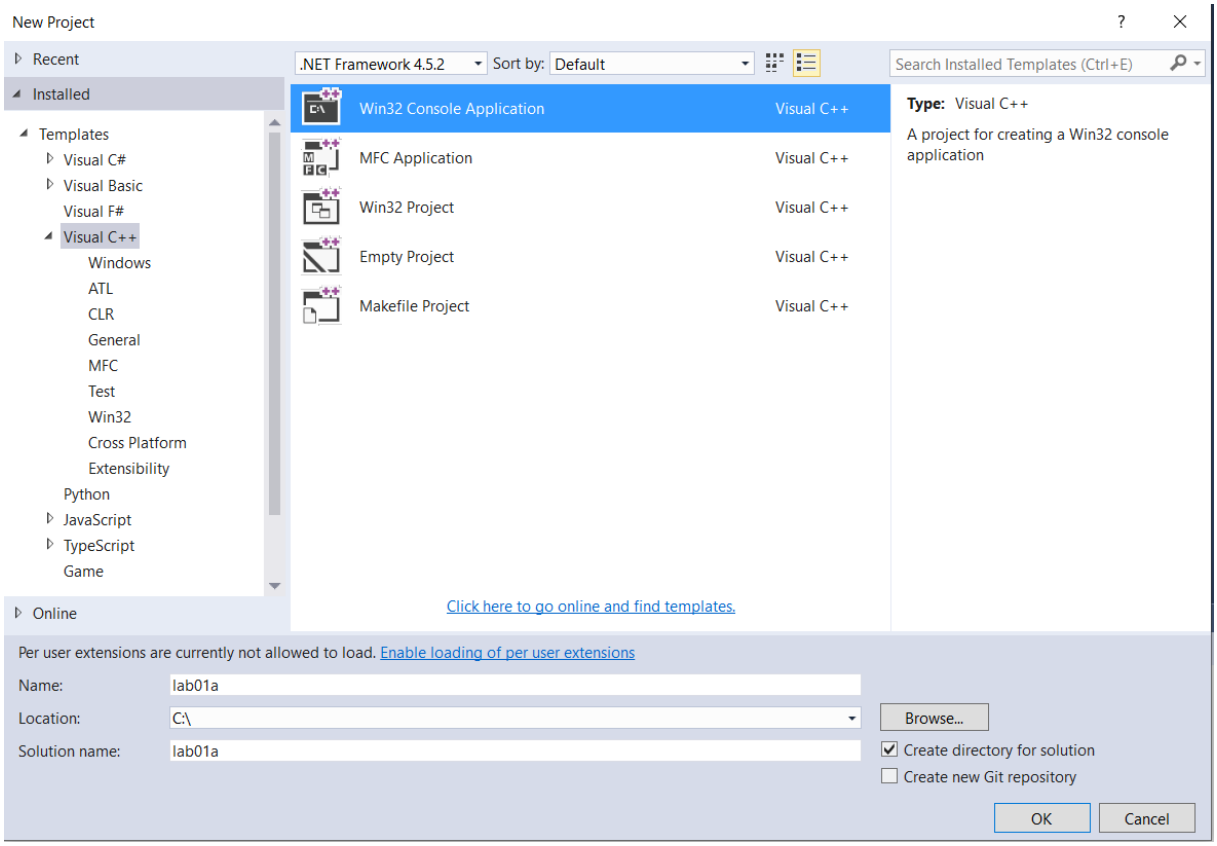



6. У проекті за замовчуванням (на панелі **Solution Explorer**) будуть показані папки і файли, що входять до проекту. За замовчуванням створюється три папки: **Header Files** для заголовкових файлів **(.h, .hpp)**, для файлів ресурсів **(.rc)**, для файлів реалізації **(.cpp)**. Також є окремий розділ **External Dependencies** де показані зовнішні залежності (бібліотеки) які використовуються в проекті.

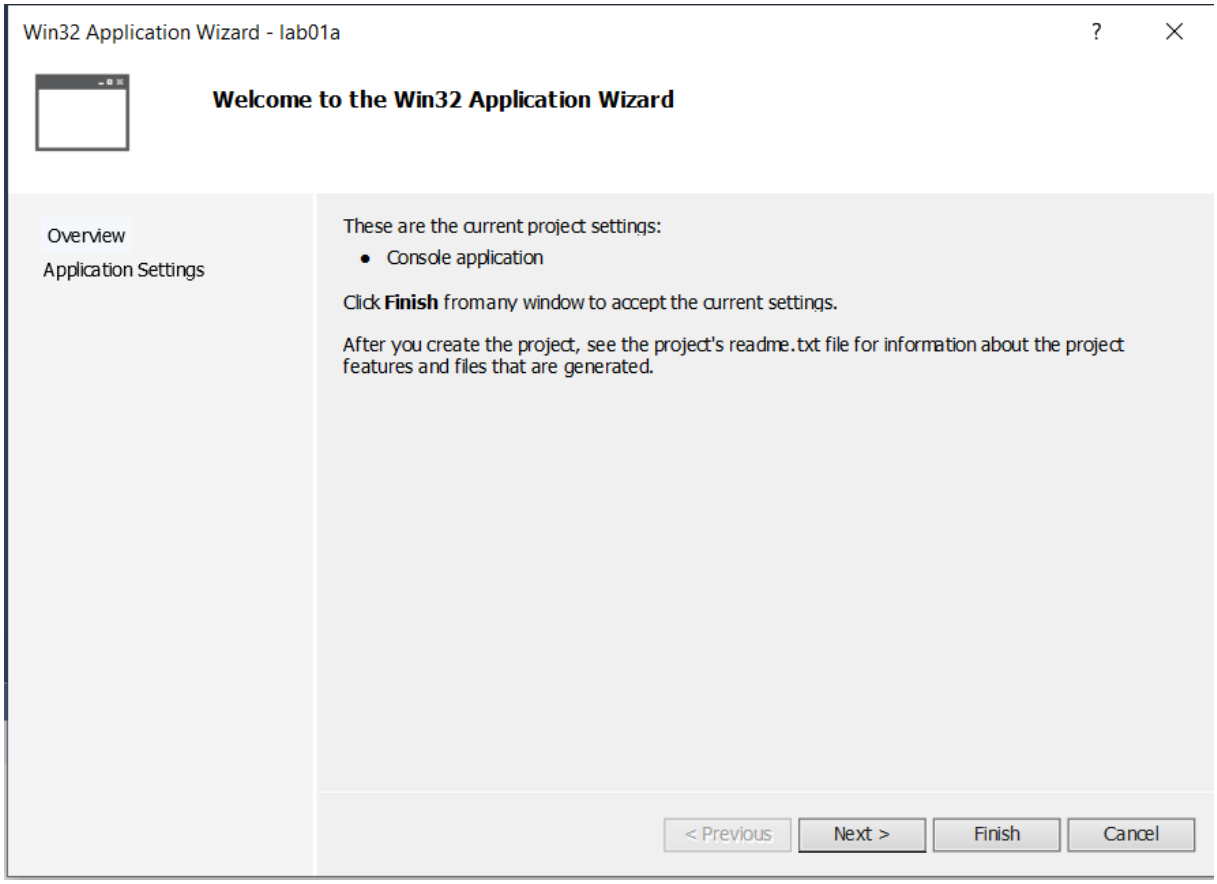



7. Зробіть ще один проект, цього разу - консольний. Для цього потрібно вибрати тип проекту **Visual C++\Win32 Console Application.**
8. Ввести назву проекту в полі **Name: - lab01a.**

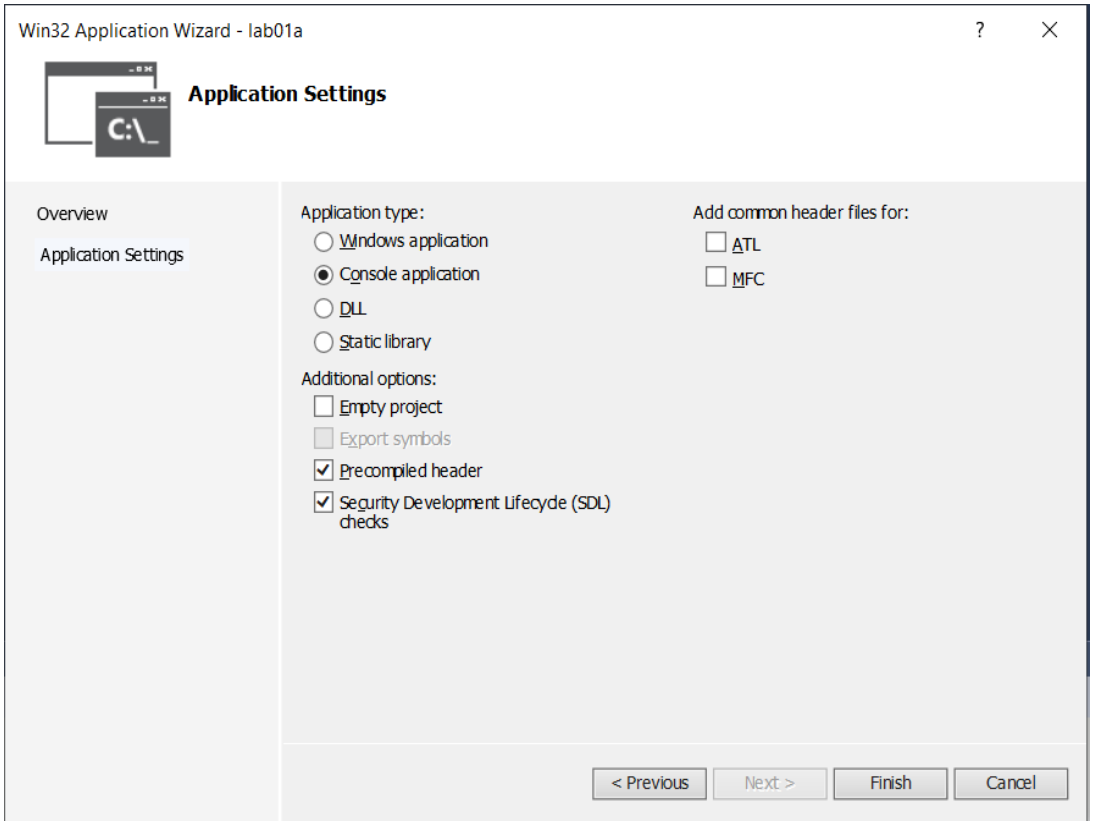
9. Вибрати місцезнаходження проекту (папку проекту) в полі **Location:**, для цього натисніть кнопку **Browse….**



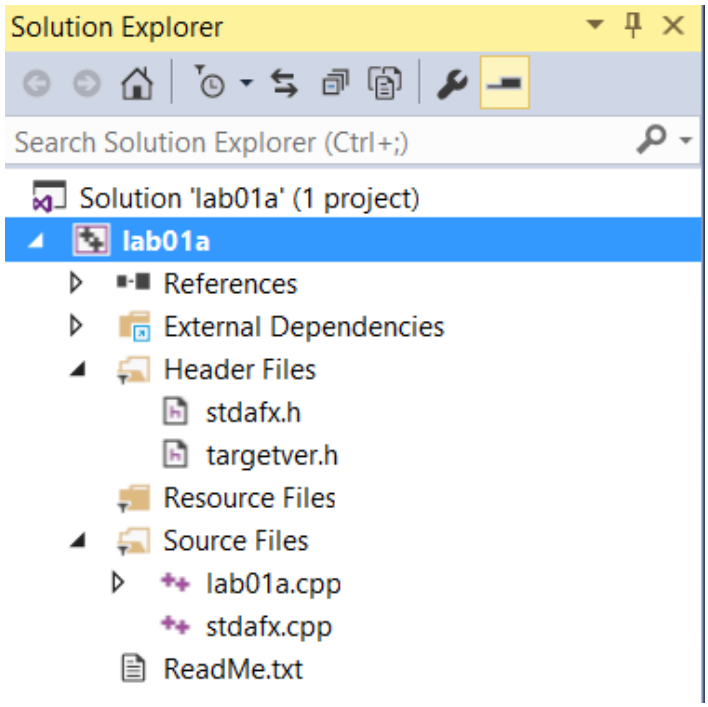

10. Виберіть пункт **Application Settings.**



11. Переконайтеся, що буде обраний тип програми (**Application type**) - **Console application**. Встановлено додаткові опції як на малюнку нижче (**Precompiled header, SDL checks**).




12. Натисніть кнопку **Finish**. Проект буде создано за деякий час.




13. Цього разу в проекті з'являться файли. Файл **stdafx.h** (та **stdafx.cpp**) потрібен для додавання бібліотек, які використовуються у всьому проекті. Цей файл необхідно включати на початку кожного файлу реалізації. Він компілюється один раз при зміні (для

чого в проекті є файл реалізації **stdafx.cpp**), таким чином, загальний час компіляції проекту стає значно меншим.

14. Зміст файлу **stdafx.h**

```
#pragma once
#include "targetver.h"

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

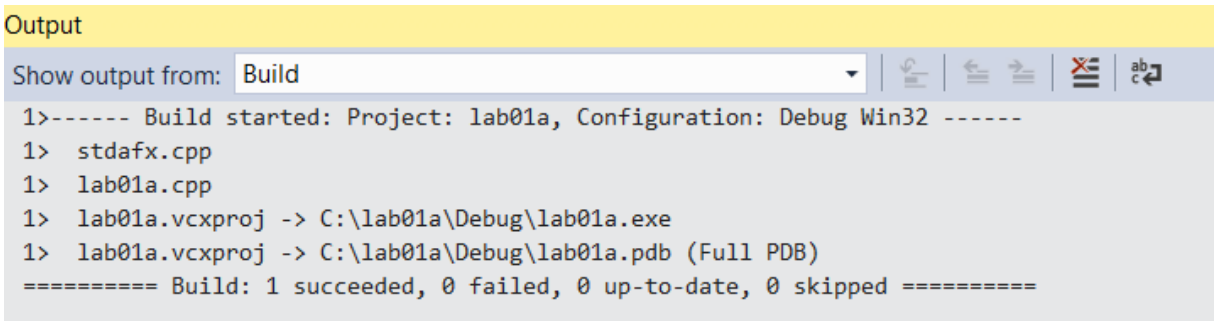
Можна побачити, що заголовки які знаходяться в самому проекті **(**файл **targetver.h)** включаються директивою **#include** в подвійних лапках "". А файли зовнішніх бібліотек **(stdio.h, tchar.h)** в кутових дужках <>.

15. Зміст файлу **lab01.cpp**

```
#include "stdafx.h"

int main()
{
	return 0;
}
```

Можна побачити, що в цьому файлі знаходиться тільки одна функція **main()**, з якої починається виконання програми. І всі функції програми так чи інакше повинні бути викликані з цієї функції. Також бачимо, що на початку файлу підключений заголовковий файл **stdafx.h** з проекту (в подвійних лапках).

16. Виконайте компіляцію проекту за допомогою меню **Build\Build Solution**.
17. В окні **Output** буде відображено хід процесу компіляції. Якщо вона пройшла успішно, буде виведено строку **“Build: 1 succeeded, 0 failed..”,** це означає, що компіляція для 1-го проекту була успішною, і немає (0) проектів які мають помилки.

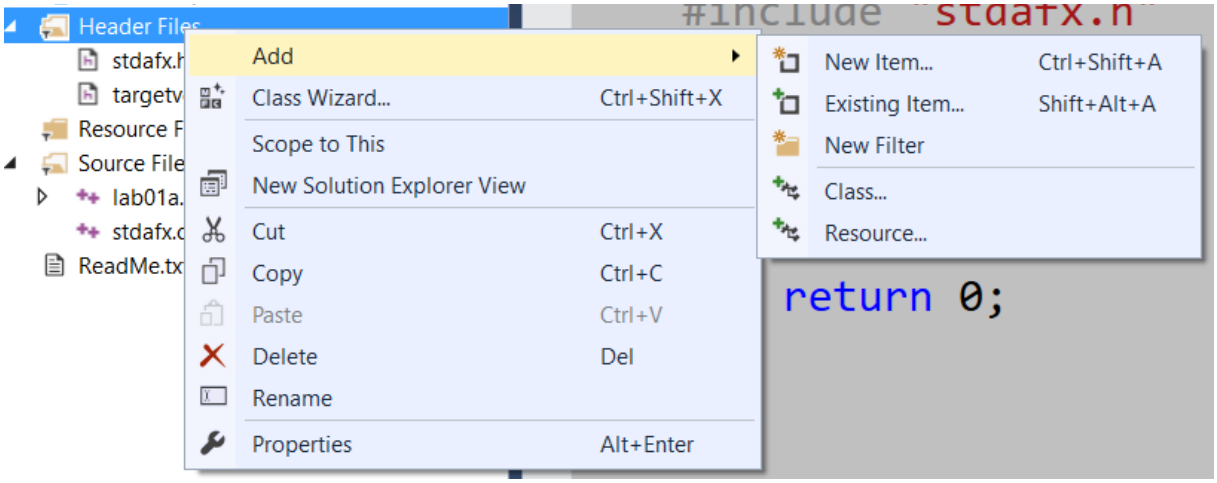



18. Пункт меню **Build\Rebuild Solution** компілює весь проект повністю, не враховуючи попередні результати компіляції. **Build\Build Solution** - компілює тількі ті файли, що зазнали змін з попередньої компіляції.

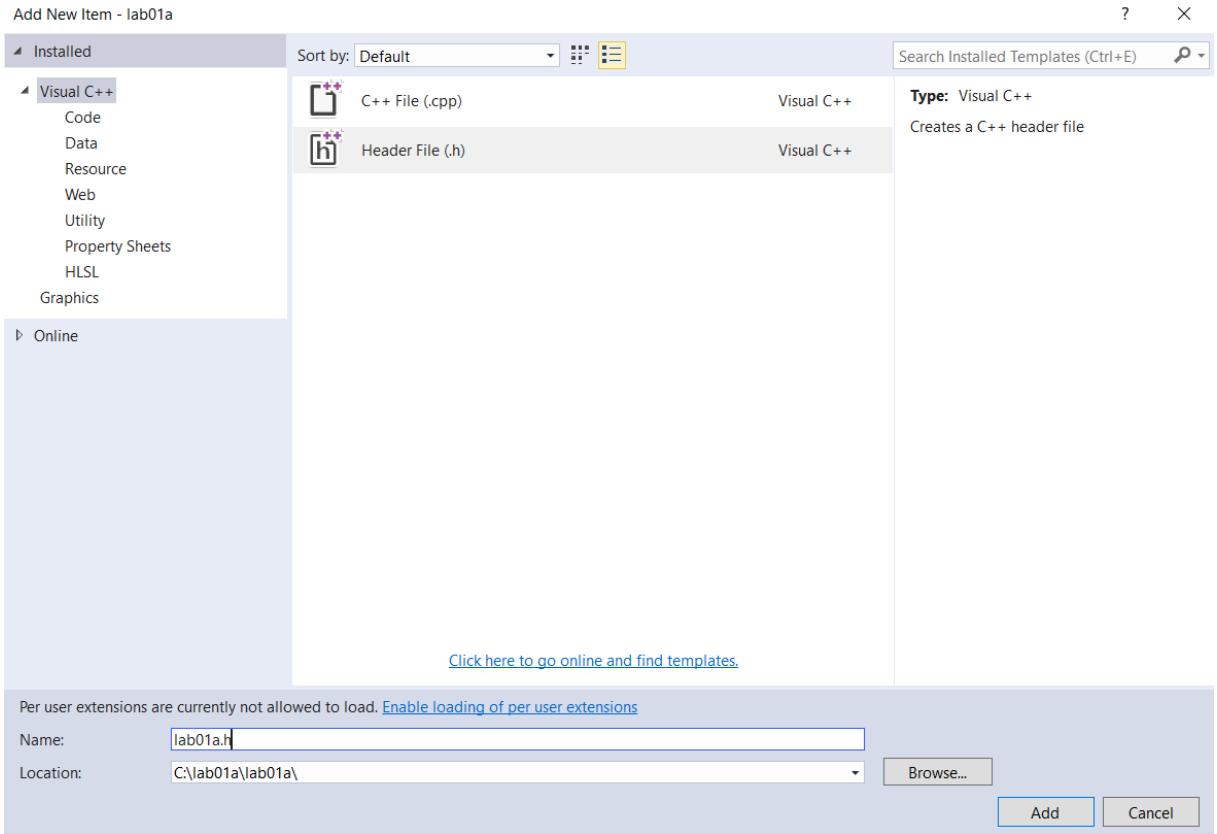
19. Випадаючі списки на верхній панелі:


дозволяють вибрати тип компіляції: **Debug** - для відлагодження, **Release** - для отримання фінальної програми з максимальною швидкодією, та архітектуру цільового процесору: **x86** - 32 бітні додатки, **x64** - 64 бітні додатки.

20. Якщо клацнути на папці в проекті правою кнопкою, у випадаючому меню можна додати нові файли **Add\New Item**, або папки **Add\New Filter**. Також можливо додавати вже існуючи файли **Add\Existing Item**.




21. Додайте до проекту в папку Header Files файл **lab01a.h**. Оберіть розділ **Header File (.h)**, введіть ім’я файлу в **Name:**, після чого натисніть кнопку **Add**. В проекті з’явиться файл **lab01a.h**.



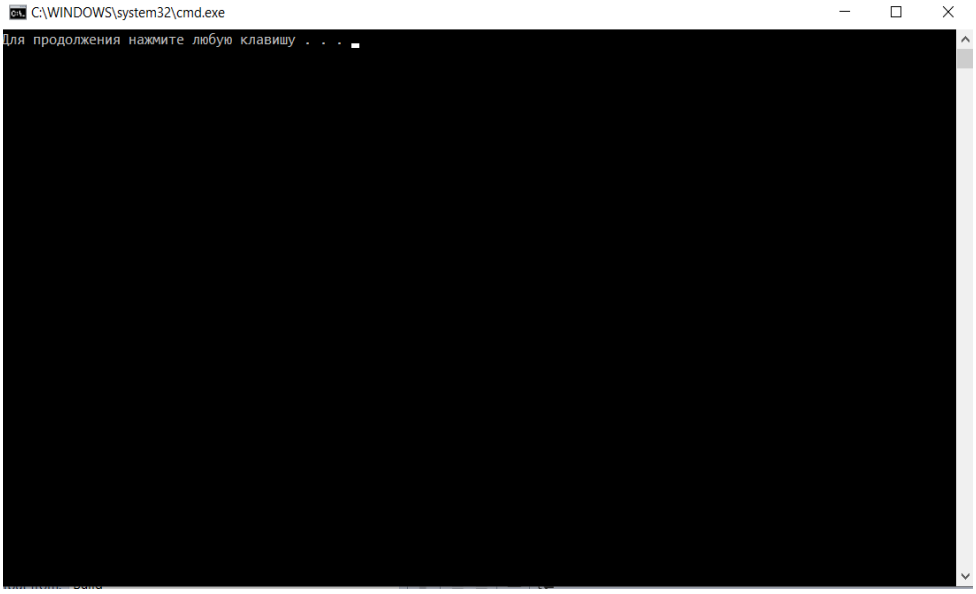
22. Підключіть файл **lab01a.h** до файлу **lab01a.cpp** (після **stdafx.h**), та виконайте компіляцію проекту знову. Ви побачите, що буде проведено компіляцію тількі для файлу **lab01a.cpp**, який було змінено, а для файлу **stdafx.cpp** - ні, для нього буде використано попередню версію машинного коду.

23. Додайте до початку функції **main** код, який создає два об'єкта **i** та **j**:

```
int i = 0;
int j = i + 10;
```

Знову виконайте компіляцію для отриманої програми.

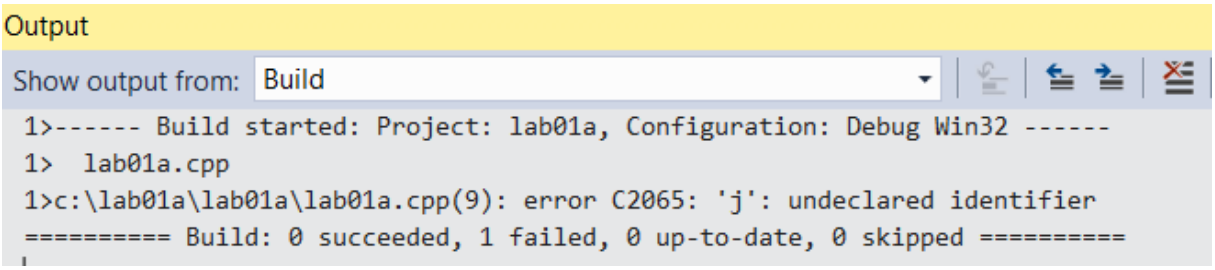
24. Щоб запустити програму застосуйте пункт меню **Debug\Start Without Debugging**. Ви побачите окно консолі з написом **"Для продовження натисніть будь-яку клавішу…"**. Це дозволяє побачити результати якщо вони були виведені до консолі програмою. Якщо ви натиснете будь-яку клавішу на клавіатурі, консоль буде закрито, та програму завершено.




25. Внесіть помилку до коду програми, для чого замініть строку

int i = 0; на int i = j;

Та знову виконайте компіляцію програми. Таким чином, ви отримаєте опис помилки (**error**) в окні **Output**:




Цей опис означає, що в 9 рядку програми було застосовано об'єкт **j**, який ще не було визначено. Якщо подвійно клацнути на рядку з помилкою, курсор у вікні з текстом програми буде переміщено до рядку (та файлу) в програмі де знайдено цю помилку. Виправте її, для чого поверніть попередній рядок. Тепер програма знову може бути зібрана.

## Звіт про лабораторну роботу

**Звіт повинен містити:**

1) титульний лист - назва курсу, номер та назва роботи, ФІБ та групу студента, рік.
2) завдання та мету.
3) короткий опис дій які було зроблено (якщо потрібно з рисунками).
4) текст програми (файли lab01a.h, lab01a.cpp).
5) висновки та результати.

**Звіт необхідно навести в форматі PDF.**

## Контрольні питання

1. Що таке об’ява? Наведіть приклади об’яв допустимих в мові C++.
2. Що таке тип і що він визначає?
3. Які вбудовані типи даних є в мові C++?
4. Які вимоги і обмеження пред'являються до цілочисельних типів даних в мові C++?
5. Чим відрізняються цілочисельні типи даних від дійсних?
6. Для чого призначений логічний тип даних?
7. Визначення розміру типів даних в мові C++?
8. Які основні операції що можуть бути застосовані до об'єктів цілочисельних типів даних?
9. Що таке літеральні константи і які правила їх об’яви?
10. Чим відрізняються об’яви змінних і констант?
11. Що таке масив?
12. Перерахуйте особливості ініціалізації елементів масиву.
13. Як визначити кількість елементів в масиві?
14. Які особливості доступу до елементів масиву?
15. Для чого застосовується модифікатор static?
16. Чим визначається час існування об'єктів в програмі?
17. Для чого застосовується оператор кома?
18. Що таке простір імен і для чого він застосовується?
19. Які способи приведення типів є в мові C++?
20. Що таке приведення типів, для чого воно потрібне?

## Лабораторна робота №2
## Генерація вихідних даних і відлагодження додатків в середовищі MS Visual Studio

**Мета роботи:** Навчитися генерувати псевдовипадкові послідовності для дослідження алгоритмів обробки даних. Навчитися використовувати відлагоджувач для контролю роботи програми.

**Необхідне обладнання та комплектуючі, ПЗ:** персональний комп'ютер з установленою операційною системою (ОС) Windows, середовище MS Visual Studio.

### Загальні відомості

Для дослідження алгоритмів обробки даних необхідно отримувати різні послідовності даних заданої довжини і типу, що також задовольняють певним критеріям, наприклад впорядкованості. Для цього доцільно використовувати функції, які автоматично генерують такі послідовності по заданому закону. Зазвичай, для отримання таких послідовностей використовують генератори псевдовипадкових чисел (ГПВЧ), отримані від генератора числа приводять до необхідного типу і діапазону. Також можуть використовуватися різні функціональні залежності, наприклад, тригонометричні функції для отримання періодичних послідовностей.

Для отримання послідовності випадкових чисел (за допомогою стандартного ГПВЧ мови С++) необхідно ініціалізувати його довільним випадковим числом, наприклад, значенням лічильника часу який пройшов з моменту запуску операційної системи або значенням системного часу з функції time(), яка являє собою монотонно зростаючий лічильник. Для того, щоб зробити це, використовується функція srand(), в яку передається початкове значення на старті програми:

```
srand( (unsigned int) time(0) );
```

Це дозволяє отримувати під час кожного запуску різні послідовності, оскільки лічильник часу постійно змінюється. Якщо ж потрібно отримувати завжди однакову послідовність, то в цю функцію передається фіксоване значення, наприклад:

```
srand( 0x123456U );
```

Далі, за допомогою функції rand(), генерується послідовність псевдовипадкових чисел (ПВЧ) необхідної довжини. Отримані від ГПВЧ числа знаходяться у діапазоні від 0 до RAND_MAX, тому їх необхідно відмасштабувати до необхідного діапазону [L, H] і привести до заданого типу (якщо необхідно). Таке масштабування відбувається в два етапи, спочатку ПВЧ R приводиться до діапазону [0, 1] з [0, RAND_MAX], а потім з [0, 1] до [L, H]:

```
float R0_1 = float( R - 0 ) / ( 0 - RAND_MAX );
float RL_H = R0_1 * ( H - L ) + L;
```

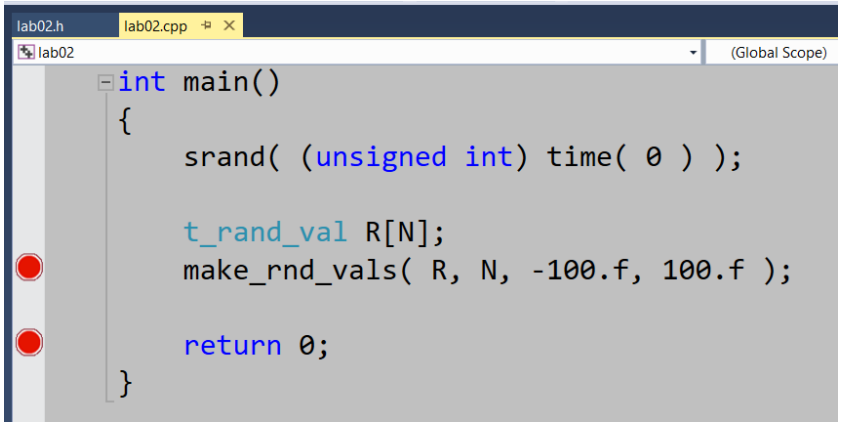
Результат перетворення зберігається у масиві.

Для контролю виконання програми доцільно використовувати вбудований відлагоджувач середовища Visual Studio. Для цього необхідно вибрати профіль компіляції Debug, запустити програму в режимі відлагодження, поставити точки переривання в ключових місцях. Програма дійшовши до точки переривання буде зупинена, в налагоджувальному вікні (Watch) будуть показані поточні значення змінних. Далі можна продовжувати виконання програми до наступної точки переривання, або в покроковому режимі — програма буде зупинятися після виконання кожного рядка.

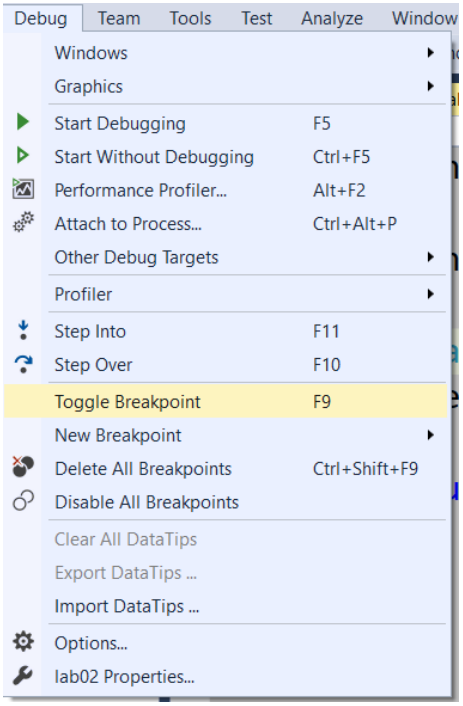
## Хід виконання роботи

1. Запустити середовище розробки MS Visual Studio, в якому створити новий консольний проект lab02 (не пустий) - див. лабораторну роботу №1, пп.7 - 15.
2. Додати до проекту заголовковий файл lab02.h та зв’язати його за файлом lab02.cpp за допомогою директиви #include “lab02.h”.
3. Підключити до файлу lab02.h стандартну бібліотеку time.h для того, щоб були доступні функції роботи з часом, а також бібліотеку stdlib.h для роботи з генератором псевдовипадкових чисел (ГПВЧ).
4. До функції main() додати початкову установку ГПВЧ, за допомогою функції srand() та функції time(0) як параметра. Таким чином, кожного разу буде отримана інша послідовність.
5. Реалізувати функцію, яка буде заповнювати масив чисел за допомогою ГПВЧ, для цього потрібно об’явити функцію в файлі lab02.h, та зробити її реалізацію в файлі lab02.cpp. Розмір послідовності, тип та діапазон чисел взяти з таблиці 2.1 згідно з номером варіанту (номер за списком у журналі академічної групи). Функція повинна отримувати як аргументи: массив, де буде збережено послідовність ПВЧ (передавати через покажчик), кількість елементів послідовності, діапазон, в якому будуть відмасштабовані елементи послідовності. Для заповнення послідовності доцільно застосувати цикл for.
6. Зробити нове ім'я t_rand_val (в файлі lab02.h) для типу даних послідовності згідно з варіантом за допомогою директиви typedef. Це ім’я застосувати у всіх функціях.
7. Об’явити розмір послідовності за допомогою константи N типу int (в файлі lab02.h) .
8. Об’явити масив для послідовності в функції main(), зробити виклик функції, що була зроблена у п.5, задати в якості аргументів: масив, розмір послідовності, діапазон.
9. Виконати компіляцію програми, виправити помилки якщо вони з'явилися.

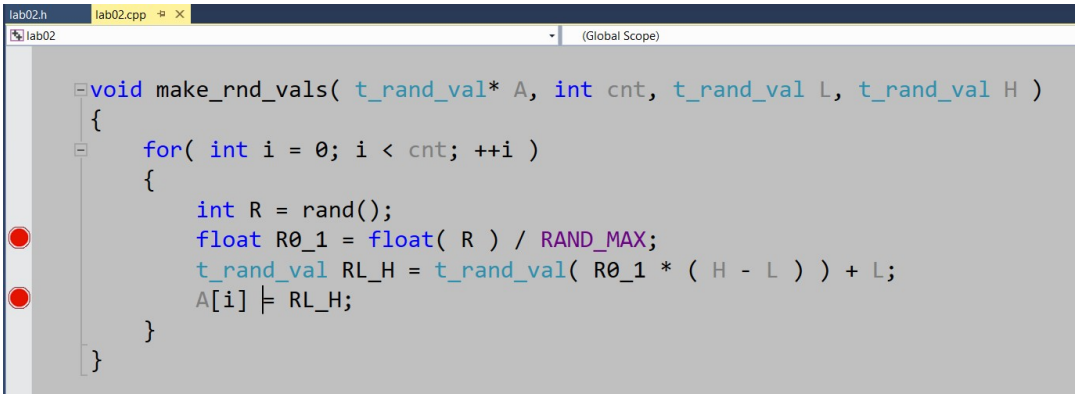
10. Встановити точки переривання в програмі, в функції main(): після об’яви масиву та на виході з функції (на операторі return).



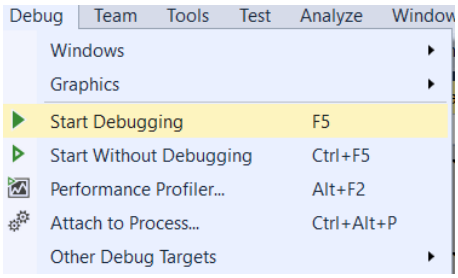

Щоб встановити точку переривання необхідно перевести курсор на відповідний рядок, і в меню Debug вибрати пункт Toggle Breakpoint. Повторний вибір зніме точку переривання.



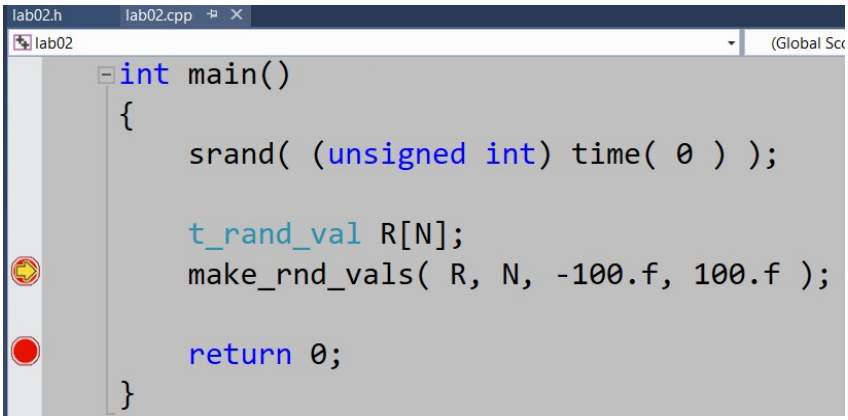

11. Встановити точки переривання в функції генерації (з п.5): після отримання ПВЧ за допомогою rand(), та на рядку запису масштабованого ПВЧ до масиву.



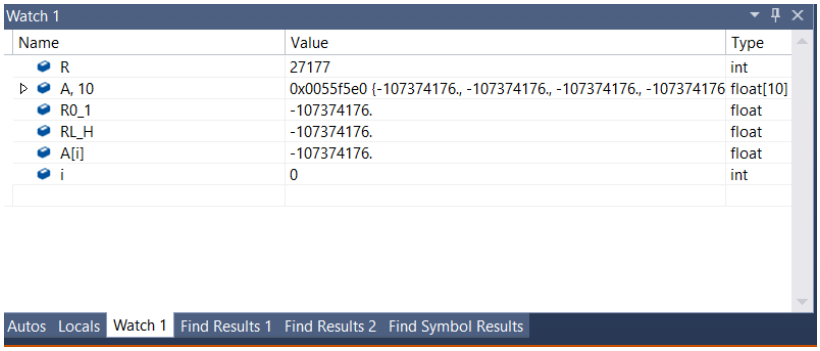
12. Запустити програму в режимі відлагодження Debug \ Start Debugging,



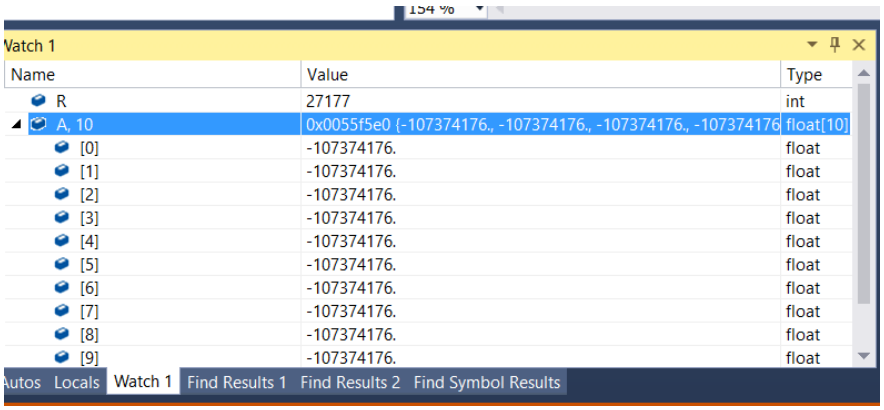

після деякого часу вона зупиниться на першій точці переривання, яка буде відмічена спеціальним курсором відлагодження — жовтою стрілкою. Цей курсор показує точку, у який було припинено виконання програми, вказує на строку коду, яку буде виконано після команди продовжити відлагодження (Step Into або Step Over).




13. На панелі Watch (Watch 1) можно відслідковувати стан змінних в програмі. Для чого додайте до цієї панелі назви змінних з вашої програми: масив, де буде зберігатися послідовність ПВЧ (у прикладі - **R[N]**), змінна в функції, що буде зберігати ПВЧ отримане з функції rand() (у прикладі - **R**), змінна, що буде зберігати ПВЧ в діапазоні від 0 до 1 (у прикладі - **R0_1**), та змінна, що буде зберігати відмасштабоване ПВЧ, яке буде записано до масиву (у прикладі - **RL_H**).

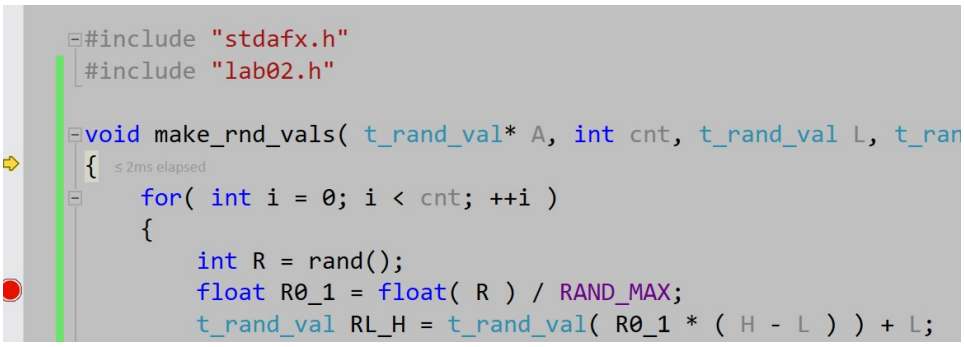



14. Якщо після назви змінної, через крапку поставити число — буде показано відповідне число елементів масиву. Додати змінну-покажчик, через який передається масив для зберігання послідовності, та задайте 10, тобто, показувати 10 елементів.




15. Щоб продовжити виконання програми покроково, виберіть пункт меню Debug\ Step Into,


курсор відлагодження (жовта стрілка) перейде до функції, на якій була встановлена точка переривання або зупинилося виконання програми після останнього кроку.

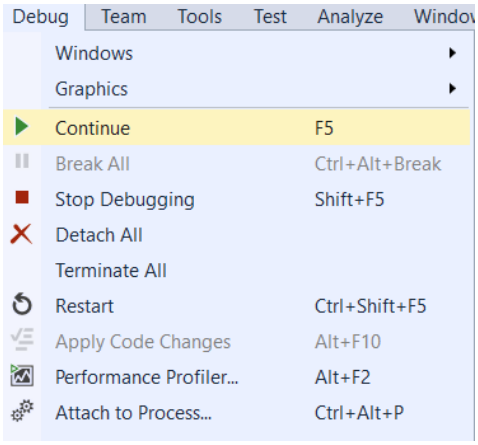



16. Якщо обрати пункт меню Debug\ Step Over, то буде зроблено один крок, без переходу до внутрішнього коду функції (функцію буде виконано як єдину команду).

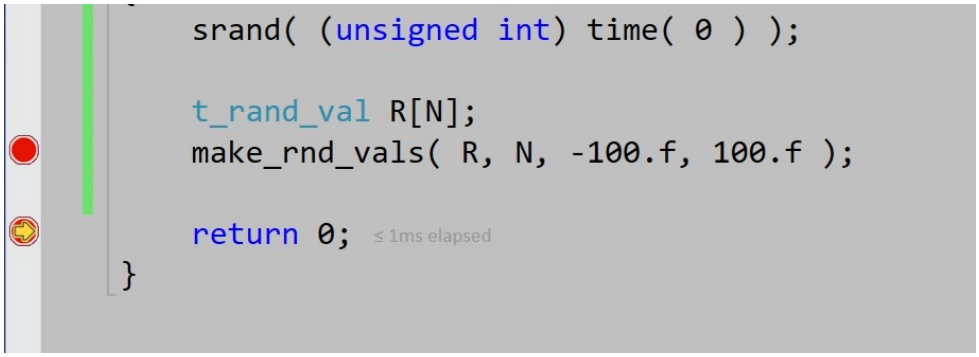

17. Виконувати функцію крок за кроком, та відслідковувати зміни — вони позначаються червоним кольором.

Як можна побачити, спочатку змінні та елементи масиву будуть неініціалізовані (заповнені довільними числами), а упродовж виконання програми отримають потрібні значення. Виконайте кілька ітерацій циклу і переконайтеся що отримані числа належать до потрібного діапазону.

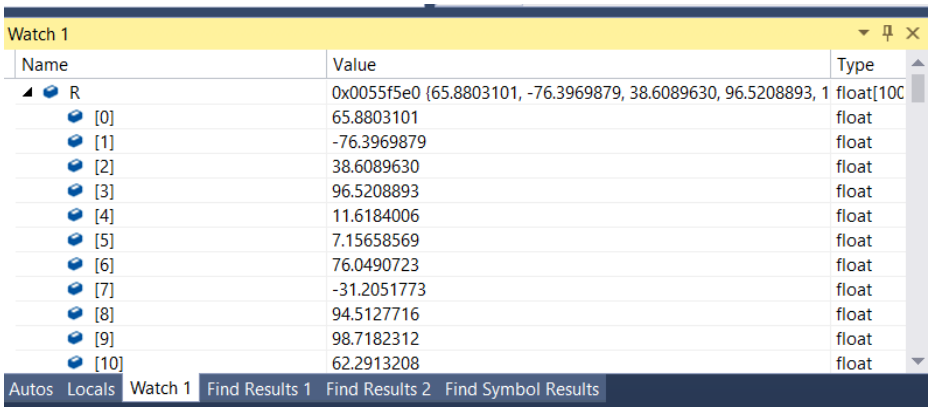
18. Щоб швидко перейти до наступної точки переривання, необхідно вибрати Debug\ Continue




програма почне безперервно виконуватись доти, поки не буде зустріне нову точку переривання.

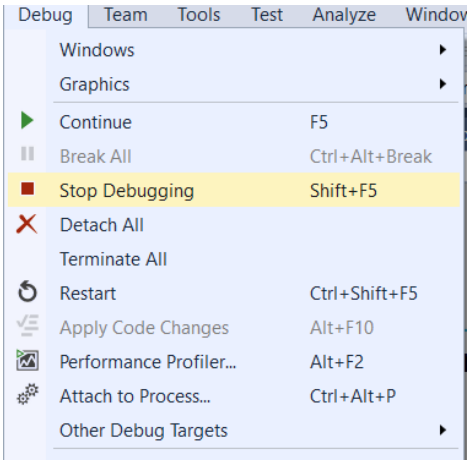



19. Подивіться на отриману послідовність чисел в кінці програми (в масиві) і запишіть цю послідовність до звіту.




Всі елементи масиву повинні містити числа в заданому діапазоні.

20. Для того щоб вийти з режиму відлагодження виберіть Debug\ Stop Debugging.




Приклад тексту програми наведено у **додатку А**.

## Звіт до лабораторної роботи

**Звіт повинен містити:**

1) титульний лист - назва курсу, номер та назва роботи, ПІБ та групу студента, рік.
2) мету та завдання.
3) короткий опис дій, які було зроблено (для наочності доповнити опис рисунками).
4) текст програми (файли lab02.h, lab02.cpp).
5) послідовність ПВЧ, яку було отримано після виконання програми.

6) висновки та аналіз результатів.

**Звіт необхідно зберегти в форматі PDF.**

Таблиця 2.1 — Параметри послідовності за варіантом

| № | тип | розмір | діапазон | № | тип | розмір | діапазон |
|---|---|---|---|---|---|---|---|
| **1** | int | 10 | -10, 20 | **11** | float | 21 | -10, 10 |
| **2** | long | 15 | -100, 100 | **12** | unsigned char | 26 | 32, 48 |
| **3** | float | 25 | -1, 3.5 | **13** | unsigned long | 17 | 1000, 2000 |
| **4** | char | 11 | ‘0’, ‘9’ | **14** | int | 29 | -500, -200 |
| **5** | unsigned short | 31 | 45, 280 | **15** | short | 31 | -5, 5 |
| **6** | unsigned int | 28 | 0, 148 | **16** | double | 16 | 300, 9000.31 |
| **7** | double | 14 | -10, 150 | **17** | float | 9 | -123, 321 |
| **8** | long | 20 | -100, 100 | **18** | int | 12 | -50, 1000 |
| **9** | short | 13 | -3500, 3500 | **19** | short | 15 | -32000, 2500 |
| **10** | int | 32 | -2, 70 | **20** | long | 28 | -1, 1 |

## Додаток А
## Програма генерування псевдовипадкової послідовності
(із 100 елементів типу float в діапазоні від -100 до 100)

**Лістинг коду у файлі lab02.h**

```
#include <time.h>
#include <stdlib.h>

typedef float t_rand_val;
const int N = 100;
void make_rnd_vals( t_rand_val* A, int cnt, t_rand_val L, t_rand_val H );
```

**Лістинг коду у файлі lab02.cpp**

```
#include "stdafx.h"
#include "lab02.h"

void make_rnd_vals( t_rand_val* A, int cnt, t_rand_val L, t_rand_val H )
{
   for( int i = 0; i < cnt; ++i )
   {
      int R = rand();
      float R0_1 = float( R ) / RAND_MAX;
      t_rand_val RL_H = t_rand_val( R0_1 * ( H - L ) ) + L;
      A[i] = RL_H;
   }
}

int main()
{
   srand( (unsigned int) time( 0 ) );

   t_rand_val R[N];

   make_rnd_vals( R, N, -100.f, 100.f );
   return 0;
}
```

## Контрольні питання

1. Як працює оператор присвоєння і для чого він потрібен в програмі?
2. Як працює скорочення запису арифметичних операцій?
3. Для чого застосовуються операції порівняння і логічні операції?
4. Для чого застосовуються умовні оператори?
5. Як замінити умовний оператор if за допомогою тернарного оператора?
6. Як замінити тернарний оператор умовним оператором if?
7. Для чого застосовується оператор вибору?
8. Які варіанти виконання блоків допустимі для оператора вибору?
9. Для чого застосовується оператор break всередині оператора вибору?
10. Для чого потрібні оператори циклу?
11. Які оператори циклу застосовуються в мові C++? У чому між ними різниця?
12. Для чого застосовуються оператори break і continue всередині циклів?
13. У чому полягає різниця між циклом while і do..while?
14. Як виглядає об’ява функції?
15. Чим відрізняється функція від процедури?
16. Яка різниця між реалізацією і об’явою функції?
17. Для чого використовується оператор return в процедурах і функціях?
18. Які основні варіанти передачі аргументів до функції?
19. Які особливості передачі аргументу за посиланням?
20. Для чого використовується передача аргументу за покажчиком?
21. Які особливості передачі аргументу за значенням?
22. Які існують варіанти повернення результатів з функції?
23. Які існують варіанти повернення результатів з процедури?
24. Чим відрізняється передача аргументу за значенням і за константним посиланням?
25. Для чого потрібен виклик функцій?
26. Як зберегти результат що повертається функцією?
27. Чим обмежений час існування аргументів функції переданих за значенням?
28. У якому випадку компілятор вважає функції різними?
29. Що таке аргументи функції за замовчуванням і як їх визначити?
30. Як в програмі створити виняткову ситуацію?
31. Що таке рекурсія і для чого вона застосовується?
32. Як зберегти значення локальної змінної функції до наступного виклику?
33. Чим відрізняється передача аргументів за покажчиком і за значенням?
34. Чим відрізняється виклик функції від її об’яви?

## Лабораторна робота №3
## Введення-виведення даних у файл та у консоль

**Мета роботи:** Навчиться вводити дані з файлу і консолі у програму; навчитися виводити відформатовані дані у файл та до консолі.

**Необхідне обладнання та комплектуючі, ПЗ:** персональний комп'ютер з установленою операційною системою (ОС) Windows, середовище MS Visual Studio.

### Загальні відомості

Для введення-виведення даних різних типів в мові С++ використовуються так звані потоки введення-виведення. У мові визначені два стандартних потоки **std::cout** і **std::cin**, які призначені для виведення на екран до спеціальної області або вікна (консоль) і введення з терміналу (клавіатури) відповідно. Для роботи з потоками необхідно підключити бібліотеку iostream, з файлами — бібліотеку fstream. Для виведення використовується потік типу ostream, а для введення — потік типу istream. Для виведення в потік використовується оператор виведення **<<**, а для введення оператор **>>**.
Наприклад, команда:

```
std :: cout << 10 << "ABCD";
```

Виведе на консоль рядок `10ABCD`.
Команда:

```
int x; std :: cin >> x;
```

чекатиме введення користувачем цілого числа, введення відбудеться після натискання клавиші Enter.

Для роботи із строковими даними доцільно застосовувати спеціалізований строковий тип даних **std::string**, для чого потрібно підключити бібліотеку string.

Наприклад, код для введення рядка довільної довжини з клавіатури буде виглядати так:

```
std::string s;
cin >> s;
```

Для отримання рядка у вигляді покажчика на масив символів (const char*) використовується метод c_str(). Для строкового типу даних визначено оператори введення-виведення в потоки, а також оператор + для конкатенації рядків. Конструктор std::string дозволяє створити рядок з масиву символів.
Для управління форматом виведення в потік використовуються так звані маніпулятори введення-виведення, це спеціальні об'єкти-функції, які змінюють стан потоку при своєму виклику. Для роботи з маніпуляторами необхідно підключити бібліотеку **iomanip**. Виклик відбувається за допомогою оператора виведення **<<**, як і для звичайного тексту.

Наприклад, код:

```
std::cout << std::endl;
```

здійснює перехід на новий рядок та очищення буфера потоку, що призведе до відправлення даних, що були буферизовані, для виводу на екран або для збереження в файл.

Основні маніпулятори:

1) std::setw( int n ) — задає розмір колонки в n символів, діє тільки на наступне поле;
2) std::fixed, std::scientific — переключають режим виведення чисел з плаваючою комою між форматом з фіксованою комою і «науковим представленням» (за допомогою мантиси та експоненти);
3) std::setprecision( int p ) — задає точність (кількість символів після коми) для дійсних чисел;
4) std::right, std::left — задає вирівнювання в колонці праворуч або ліворуч відповідно;
5) std::setfill( char c ) — задає символ-заповнювач для невидимих символів колонки, за замовчуванням використовується пробіл;
6) std::setbase( int x ) — задає систему обчислення, відповідно до якої будуть виводитися цілі числа, для 8-, 10- і 16-річних систем є окремі маніпулятори: std::oct, std::dec і std::hex;
7) std::showbase, std::noshowbase — включає і відключає показ префікса системи обчислення для цілих чисел.

В **додатку Б** надано код для виведення відформатованої таблиці у консоль.

## Хід виконання роботи

1. Запустити середовище розробки MS Visual Studio, в якому створити новий консольний проект lab03 (не пустий) — див. лабораторну роботу №1, пп.7-15.
2. Додати до проекту заголовковий файл lab03.h та зв’язати його за файлом lab03.cpp за допомогою директиви #include “lab03.h”.
3. Підключити до файлу lab03.h стандартні бібліотеки iostream та fstream для того, щоб були доступні функції роботи з потоками введення-виведення, а також бібліотеки time.h та stdlib.h для роботи з часом та генератором псевдовипадкових чисел (ГПВЧ) (див. лабораторну роботу №2).
4. До функції main() додати початкову установку ГПВЧ, за допомогою функції srand() та функції time(0) як параметра. Це забезпечить генерацію нової послідовності випадкових чисел, що не повторюється, кожного разу, коли буде виконуватись програма.
5. Об’явити максимальний розмір послідовності за допомогою константи N типу int (в файлі lab03.h).
6. Реалізувати введення з консолі за допомогою клавіатури (потік cin) кількості елементів, з перевіркою що введене число не перевищує N та не менше за 1. Для цього доцільно скористатися циклом do…while.

7. Реалізувати функції для заповнення масиву структур за допомогою ГПВЧ. Для цього потрібно об’явити ці функції в файлі lab03.h, та зробити їх реалізацію в файлі lab03.cpp. Максимальний розмір послідовності та поля структури взяти з таблиці 3.1 згідно з номером варіанту (номер за списком у журналі академічної групи).

**Перша функція** для синтезу послідовності повинна отримувати як аргументи: масив структур, кількість елементів послідовності, та заповнювати їх за допомогою ГПВЧ. Для заповнення структур доцільно застосувати цикл for. Поля необхідно заповнювати ПВЧ відповідно до типу в наступних діапазонах:

1) char — -100 до 100
2) short — -1000 до 1000
3) int — -10000 до 10000
4) unsigned int — 0 до 10000
5) float — -1.0 до 1.0
6) double — -10000.0 до 10000.0

Для заповнення полів структур доцільно використати функцію з лабораторної роботи №2 для отримання ПВЧ.

**Друга функція** повинна виконувати виведення в файл, вона також повинна отримувати масив структур та ім'я файлу, в який потрібно форматовано вивести масив структур (передавати як const char*). Для виводу чисел з плаваючою комою треба застосувати формат виводу fixed. Форматування має бути наступним:

1) номер структури починаючи з 1, ширина — 2 символи, вирівнювання праворуч, десяткова система, повинен закінчуватися комою;
2) поля структури (див. таблицю 3.2);
3) наступний рядок (std::endl).

8. Об’явити масив для структур в функції main(), зробити послідовний виклик функцій, що були зроблені у п.7, задати в якості аргументів: масив, розмір послідовності, ім'я файлу.

9. Виконати компіляцію програми, виправити помилки, якщо вони були виявлені.

10. Встановити точки переривання в програмі, на початку та в кінці функції main(), та крок за кроком подивитися процес виконання програми.

11. Після закінчення програми, відкрити отриманий файл з відформатованими результатами виводу і перевірити результат.

Приклад тексту програми за варіантом 20 наведено у **додатку В**.

## Звіт до лабораторної роботи

**Звіт повинен містити:**

1) титульний лист за стандартною формою — обов'язково вказати назву курсу, номер та назву роботи, ПІБ та групу студента, рік.
2) мету та завдання.
3) короткий опис дій, які було зроблено (для наочності доповнити опис рисунками).
4) текст програми (файли lab03.h, lab03.cpp).
5) файл з відформатованим виводом.

6) висновки та аналіз результатів.

**Звіт необхідно зберегти в форматі PDF.**

Таблиця 3.1 — Поля структури та максимальний розмір послідовності за варіантом

| № | поля структури | максимальний розмір | № | поля структури | максимальний розмір |
|---|---|---|---|---|---|
| **1** | float, int, bool | 10 | **11** | float, double, float | 30 |
| **2** | float[2], char | 15 | **12** | double, float | 25 |
| **3** | double, float | 20 | **13** | int[3], char[2] | 15 |
| **4** | unsigned int, short[3] | 30 | **14** | unsigned int, short[3] | 10 |
| **5** | float[2], char | 20 | **15** | int, char, short | 20 |
| **6** | int, char, short | 15 | **16** | float, double, float | 15 |
| **7** | int[3], char[2] | 20 | **17** | float[2], char | 30 |
| **8** | double, float | 25 | **18** | int[3], char[2] | 25 |
| **9** | float, double, float | 15 | **19** | unsigned int, short[3] | 20 |
| **10** | unsigned int, short[3] | 30 | **20** | int, char, short | 10 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |

Таблиця 3.2 — Параметри форматування при виводі у файл за типом

| № | тип поля структури | ширина/ цифр після коми (setw/ setprecision) | вирівнювання (left, right) | система числення (setbase або dec, oct, hex) | заповнювач (setfill) |
|---|---|---|---|---|---|
| **1** | bool | 5 | ліво | - | пробіл |
| **2** | char | 4 | право | 10 | пробіл |
| **3** | short | 8 | ліво | 8 | 0 |
| **4** | int | 8 | право | 10 | пробіл |
| **5** | unsigned int | 12 | право | 16 | 0 |
| **6** | float | 8/3 | ліво | 10 | пробіл |
| **7** | double | 15/5 | право | 10 | пробіл |

## Додаток Б

## Приклад виведення форматованої таблиці на консоль

**Лістинг коду у файлі lab03.h**

```
#include <iostream>
#include <iomanip>
using namespace std;

struct s_human
{
    char name[30];
    float  weight;
};
```

**Лістинг коду у файлі lab03.cpp**

```
#include "stdafx.h"
#include "lab03.h"

const int N = 50;
s_human A[N];
…

cout << fixed << setprecision( 2 ) << dec << setfill( ‘ ‘ );

for( int i = 0; i < N; ++i )
{
    cout << right << setw( 3 ) << ( i + 1 ) << “. ”;
    cout << left << setw( 30 ) << A[i].name << “ “;
    cout << right << setw( 6 ) << A[i].weight << endl;
}
…
```

## Додаток В

## Приклад форматованого виведення у файл для варіанту 20

**Лістинг коду у файлі lab03.h**

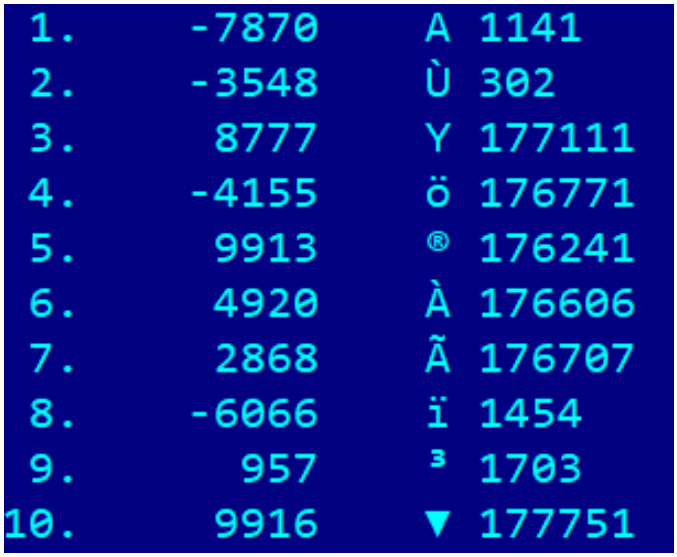
```
#pragma once
#include <iostream>
#include <fstream>
#include <iomanip>
#include <time.h>
#include <stdlib.h>

const int N = 10;        //з таблиці 3.1 (п.п.5)

struct s_my_struct
{
    int i;               //з таблиці 3.1
    char c;
    short s;
};

void fill_array( s_my_struct* A, int cnt );
int output_array( const char* filename, const s_my_struct* A, int cnt );
using namespace std;
```

**Лістинг коду у файлі lab03.cpp**

```
#include "lab03.h"

//з лабораторної №2
int rnd_int( int mn, int mx )
{
    float r = float( rand() ) / ( RAND_MAX - 0 );
    return mn + int( ( mx - mn ) * r );
}

void fill_array( s_my_struct* A, int cnt )
{
    for( int i = 0; i < cnt; ++i )
    {
        //з п.п.7
        A[i].i = rnd_int( -10000, 10000 );
        A[i].c = (char) rnd_int( -100, 100 );
        A[i].s = (short) rnd_int( -1000, 1000 );
    }// for( int i = 0; i < cnt; ++i )
}
```

```
int output_array( const char* filename, const s_my_struct* A, int cnt )
{
    std::fstream f( filename, ios_base::out | ios_base::trunc );
    if( !f.is_open() )
        return -1;

    for( int i = 0; i < cnt; ++i )
    {
        //з п.п.7
        f << dec << right << setfill( ' ' ) << setw( 2 ) << ( i + 1 ) << ". ";
        //з таблиці 3.2
        f << dec << right << setw( 8 ) << A[i].i << " ";
        f << setw( 4 ) << A[i].c << " ";
        f << setw( 8 ) << left << oct << A[i].s;
        f << endl;

    }// for( int i = 0; i < cnt; ++i )
    return 0;
}

int main()
{
    //з п.п.4
    srand( (unsigned int) time( 0 ) );

    //з п.п.6
    cout << "Enter the number of elements in the sequence (in range 1 - " <<
            N << "):" << endl;
    int cnt = N;
    do
    {
        cin >> cnt;
    }
    while( cnt > N || cnt < 1 );

    //з п.п.8
    s_my_struct A[N];

    //з п.п.7
    fill_array( A, cnt );
    int res = output_array( "c:\\lab03\\output.txt", A, cnt );
    return res;
}
```

**Лістинг файлу з прикладом результатів виводу (output.txt) для 10 елементів:**




**Контрольні питання**

1. Як здійснюється введення-виведення даних?
2. Чим відрізняється виведення даних до файлу від виведення даних в стандартний потік std::cout?
3. Що таке маніпулятори введення-виведення?
4. Як задати кількість цифр після коми для виведених на екран дійсних чисел?
5. Як задати вирівнювання в колонці таблиці при форматованому виведення в файл?
6. Як записати 10 байт в двійковий файловий потік?
7. Як зчитати 100 байт з двійкового файлового потоку?
8. Як зчитати будь-яку кількість елементів будь-якого типу з файлового потоку?
9. Чим відрізняється форматований вивід даних в текстовий потік, від запису в бінарний потік?
10. Як визначаються оператори введення-виведення для користувацьких типів даних?
11. Як зберегти структуру даних в текстовий файл?
12. Як зчитати структуру даних з текстового файлу?
13. Як зчитати масив із двійкового і текстового файлів?
14. Як зберегти масив структур в двійковий файл?
15. Як зберегти масив структур в текстовий файл?

## Список рекомендованої літератури

1. Stroustrup, Bjarne. The C++ programming language / Bjarne Stroustrup.— Fourth edition. Published by Pearson Education, Inc. 2013. 1360 p. ISBN 978-0-321-56384-2.

2. Грицюк Ю., Рак Т. Програмування мовою С++. Навчальний посібник. / Юрій Грицюк, Тарас Рак. Львів. Вид-во ЛДУ БЖД 2011. 290 с. ISBN 978-966-3466-85-9.

3. Grimes R. Beginning C++ Programming. Richard Grimes. Packt Publishing Ltd. Birmingham, UK. 2017. 499 p. ISBN 978-1-78712-494-3.

4. Татарчук Д.Д., Діденко Ю.В. Програмування мовами С та С++: навч. посіб. / Д.Д. Татарчук, Ю.В. Діденко. – К.: 2012.–112 с.

5. Bancila M. The Modern C++ Challenge. Marius Bancila. Packt Publishing Ltd. Birmingham, UK. 2018. 309 p. ISBN 978-1-78899-386-9.

6. Бублик В.В. Об’єктно-орієнтоване програмування: [Підручник] / В.В. Бублик. – К.: ІТ книга, 2015. – 624 с. ISBN 978-966-97182-1-1.

# ЗМІСТ

**ВСТУП**.....3
**ЗАГАЛЬНІ ВІДОМОСТІ**.....4
1 Вивчення принципів побудови консольних додатків у середовищі MS Visual Studio.....5
2 Генерація вихідних даних і відлагодження додатків в середовищі MS Visual Studio.....14
**Додаток А** Програма генерування псевдовипадкової послідовності.....22
3 Введення-виведення даних у файл та у консоль.....24
**Додаток Б** Приклад виведення форматованої таблиці на консоль.....29
**Додаток В** Приклад форматованого виведення у файл для варіанту 20.....30
**Список рекомендованої літератури**.....33

Навчальне видання

ЗУЄВ Андрій Олександрович
КАРАМАН Дмитро Григорович
ДЕНИСЕНКО Микола Анатолійович

# ПРОГРАМУВАННЯ НА МОВІ С++

**Методичні вказівки**
до виконання лабораторних робіт

**з курсу «Інформаційні технології та програмування»**

для студентів спеціальностей «Автоматизація та комп'ютерно-інтегровані технології», «Телекомунікація та радіотехніка» усіх форм навчання вищих навчальних закладів

Відповідальний за випуск Зуєв А.О.
Роботу до друку рекомендував Дудник О.В.

План 2022 р., Поз.268
Підписано до друку 05.11.2022. формат 60×84 1/16. Папір друк. № 2.
Друк – різографія. гарнітура Times New Roman. Розум. друк. арк.3,2.
Обл. – вид. арк. 2,7.Наклад 100 прим. Зам. № . Ціна договірна.

Самостійне видання