

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з дисципліни

“Алгоритмізація та програмування”

для студентів спеціальності

122 “Комп’ютерні науки та інформаційні технології”
(денної форми навчання)

2016

Методичні вказівки до виконання лабораторних робіт з дисципліни “Алгоритмізація та програмування” для студентів спеціальності 122 “Комп’ютерні науки та інформаційні технології” (денної форми навчання) / В.М. Льовкін. – Запоріжжя : ЗНТУ, 2016. – 64 с.

Автор: Валерій Миколайович Льовкін, к.т.н., доцент

Рецензент: А. О. Олійник, к.т.н., доцент

Відповідальний
за випуск: С. О. Субботін, д.т.н., професор

Затверджено
на засіданні кафедри
програмних засобів

Протокол № 1
від “16” серпня 2016 р.

ЗМІСТ

Вступ	6
1 Лабораторна робота № 1 Програмування на основі лінійних алгоритмів та алгоритмів з розгалуженою структурою	7
1.1 Мета роботи	7
1.2 Основні теоретичні відомості.....	7
1.2.1 Створення виконуваного коду програми	8
1.2.2 Розроблення програм в інтегрованому середовищі Dev-C++	8
1.2.3 Розроблення програм в інтегрованому середовищі Eclipse CDT	12
1.2.4 Оголошення змінних	20
1.2.5 Базові конструкції мови C++	22
1.2.5.1 Умовний оператор if-else	22
1.2.5.2 Оператор множинного вибору switch	22
1.2.6 Введення/виведення даних	23
1.2.7 Математичні функції мови C++	24
1.3 Завдання до роботи.....	24
1.4 Зміст звіту.....	25
1.5 Контрольні запитання	25
2 Лабораторна робота № 2 Програмування на основі алгоритмів з циклічною структурою.....	26
2.1 Мета роботи	26
2.2 Основні теоретичні відомості.....	26
2.2.1 Цикл з передумовою while.....	26
2.2.2 Цикл з передумовою for.....	27
2.2.3 Цикл з постумовою do-while	28
2.2.4 Перетворення з однієї системи числення в іншу.....	29
2.2.5 Знаходження найбільшого спільного дільника та найменшого спільного кратного двох чисел.....	30
2.3 Завдання до роботи.....	30
2.4 Зміст звіту.....	30
2.5 Контрольні запитання	31
3 Лабораторна робота № 3 Програмування з використанням масивів та покажчиків	32
3.1 Мета роботи	32

3.2 Основні теоретичні відомості.....	32
3.2.1 Масиви.....	32
3.2.2 Показчики.....	33
3.2.3 Динамічні масиви.....	35
3.3 Завдання до роботи.....	36
3.4 Зміст звіту.....	36
3.5 Контрольні запитання.....	36
4 Лабораторна робота № 4 Робота з символьними рядками.....	38
4.1 Мета роботи.....	38
4.2 Основні теоретичні відомості.....	38
4.3 Завдання до роботи.....	40
4.4 Зміст звіту.....	41
4.5 Контрольні запитання.....	41
5 Лабораторна робота № 5 Програмування з використанням функцій.....	43
5.1 Мета роботи.....	43
5.2 Основні теоретичні відомості.....	43
5.2.1 Визначення та виклик функцій у програмі.....	43
5.2.2 Рекурсивні функції.....	45
5.3 Завдання до роботи.....	46
5.4 Зміст звіту.....	46
5.5 Контрольні запитання.....	46
6 Лабораторна робота № 6 Структури. Перелічення. Об'єднання.....	48
6.1 Мета роботи.....	48
6.2 Основні теоретичні відомості.....	48
6.3 Завдання до роботи.....	50
6.4 Зміст звіту.....	50
6.5 Контрольні запитання.....	51
7 Лабораторна робота № 7 Програмування з використанням алгоритмів сортування.....	52
7.1 Мета роботи.....	52
7.2 Основні теоретичні відомості.....	52
7.2.1 Сортування вибором.....	52
7.2.2 Сортування обміном (бульбашкове).....	53
7.2.3 Сортування вставками.....	53

7.2.4 Сортування Шелла	54
7.2.5 Сортування злиттям	54
7.2.6 Сортування підрахунком	54
7.3 Завдання до роботи.....	55
7.4 Зміст звіту.....	55
7.5 Контрольні запитання	56
8 Лабораторна робота № 8 Робота з файлами	57
8.1 Мета роботи	57
8.2 Основні теоретичні відомості.....	57
8.3 Завдання до роботи.....	60
8.4 Зміст звіту.....	61
8.5 Контрольні запитання	61
Література.....	62
Додаток А Приклад оформлення титульного листа звіту з лабораторної роботи.....	64

ВСТУП

Дане видання призначене для вивчення та практичного освоєння студентами денної форми навчання алгоритмізації та програмування – базової дисципліни для подальшого вивчення більшості спеціальних дисциплін.

Відповідно до графіка студенти перед виконанням лабораторної роботи повинні ознайомитися з конспектом лекцій та рекомендованою літературою. Звичайно, в дані методичні вказівки неможливо було внести весь матеріал, необхідний для виконання та захисту лабораторних робіт. Тому тут містяться основні, базові теоретичні відомості, необхідні для виконання лабораторних робіт. Таким чином для виконання лабораторної роботи та при підготовці до її захисту необхідно ознайомитись з конспектом лекцій та проробити весь матеріал, наведений в переліку рекомендованої літературі. При цьому не варто обмежуватись лише наведеним списком.

Для одержання заліку з кожної роботи студент здає викладачу повністю оформлений звіт, а також демонструє на екрані монітору комп'ютера результати виконання лабораторної роботи.

Звіт має містити:

- титульний аркуш (на ньому вказують назву міністерства, назву університету, назву кафедри, номер, вид і тему роботи, виконавця та особу, що приймає звіт, рік; приклад наведено в додатку А);
- мету роботи;
- завдання до роботи;
- лаконічний опис теоретичних відомостей;
- результати виконання лабораторної роботи;
- змістовний аналіз отриманих результатів та висновки.

Звіт виконують на білому папері формату А4 (210 x 297 мм). Текст розміщують тільки з однієї сторони аркуша. Поля сторінки з усіх боків – 20 мм. Аркуші скріплюють за допомогою канцелярських скріпок або вміщують у канцелярський файл.

Під час співбесіди при захисті лабораторної роботи студент повинен виявити знання про мету виконання роботи, за теоретичним матеріалом, методами виконання кожного етапу роботи, змістом основних розділів оформленого звіту з демонстрацією результатів на конкретних прикладах. Студент повинен вміти правильно аналізувати отримані результати. Для самоперевірки при підготовці до виконання і захисту роботи студент повинен відповісти на контрольні запитання, наведені наприкінці опису відповідної роботи.

1 ЛАБОРАТОРНА РОБОТА № 1

ПРОГРАМУВАННЯ НА ОСНОВІ ЛІНІЙНИХ АЛГОРИТМІВ ТА АЛГОРИТМІВ З РОЗГАЛУЖЕНОЮ СТРУКТУРОЮ

1.1 Мета роботи

Отримати знання і навички, необхідні для програмування з використанням лінійних алгоритмів і алгоритмів з розгалуженою структурою, та навчитися використовувати їх на практиці в процесі розроблення програм мовою програмування C++.

1.2 Основні теоретичні відомості

До складу кожної програми мовою C++ повинна входити головна функція `main()`. Дана функція є початковою точкою входу в програму.

Основну структуру програми мовою C++ наведено на рис. 1.1.

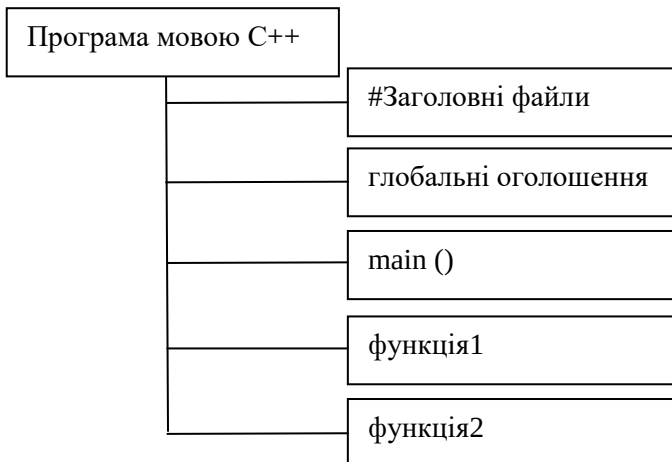


Рисунок 1.1 – Структура програми мовою C++

1.2.1 Створення виконуваного коду програми

Для того, щоб забезпечити перетворення програмного коду мовою C++ у виконуваний файл, потрібно виконати наступні дії:

1. **Написати** програму і зберегти її у файлі на диску. Даний файл містить код програми.

2. **Скомпілювати** код програми.

3. Зв'язати об'єктний код основної програми (отриманий в процесі компіляції) з додатковим об'єктним кодом бібліотечних та/або користувацьких функцій і таким чином **скомпонувати** єдину програму.

Деякі реалізації мови C++, наприклад, Microsoft Visual C++, Borland C++, реалізовані у вигляді інтегрованих середовищ розробки, що дозволяють виконувати всі вищеописані етапи створення виконуваного файлу в автоматичному режимі.

Розглянемо два вільні інтегровані середовища розробки (Integrated development environment, IDE), які можуть використовуватися для розроблення програм мовами C/C++:

- **Dev-C++**: у склад дистрибутива входить компілятор MinGW. Оригінальну версію було розроблено компанією Bloodshed Software. На даний момент продовження розробки виконується компанією Orwell. Завантажити останню версію середовища можна за посиланням <http://sourceforge.net/projects/orwelldevcpp/>;

- **Eclipse CDT** – інтегроване середовище на базі платформи Eclipse. Завантажити останню версію середовища можна за посиланням <https://eclipse.org/cdt/downloads.php>.

1.2.2 Розроблення програм в інтегрованому середовищі Dev-C++

Після запуску IDE Dev-C++ необхідно створити проект, для чого обрати пункт меню Файл → Створити → Проект. Далі на екрані з'явиться вікно Новий проект (рис. 1.2), в якому можна обрати тип проекту (наприклад, Console Application для створення консольного додатку), мову програмування (C-проект або C++-проект) та задати ім'я проекту, після чого необхідно натиснути на кнопку Гаразд для підтвердження створення проекту.

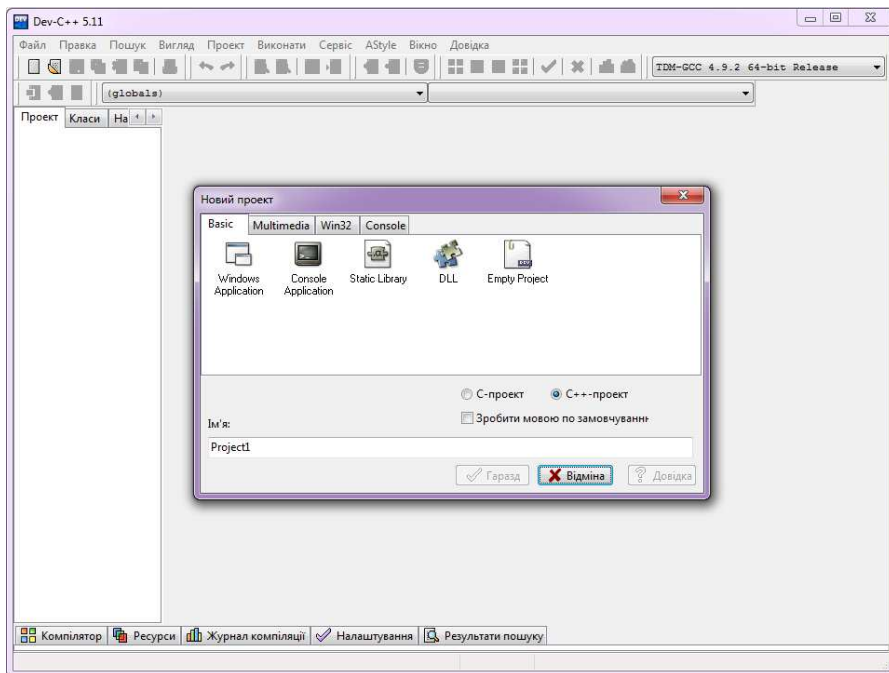


Рисунок 1.2 – Створення нового проекту в Dev-C++

Після виконання всіх перелічених дій буде створено проект, який складається з одного програмного файлу `main.cpp`. Для додавання нових файлів до проекту призначений розділ меню **Файл** → **Створити**. Для новоствореного проекту в браузері Проект в лівій частині вікна відображається структура проекту (рис. 1.3).

У правій частині вікна відображається текст файлу проекту, обраного в браузері Проект. Після того, як у текст програми внесено всі необхідні зміни, необхідно виконати компіляцію проекту, для чого призначений пункт меню **Виконати** → **Скомпільовати** (або клавіша F9).

Після того, як проект скомпільовано, можна запустити програму на виконання, для чого призначений пункт меню **Виконати** → **Виконати** (або клавіша F10).

У випадку новоствореного проекту можна в перший раз його одразу запустити на виконання, після чого буде запропоновано виконати компіляцію проекту.

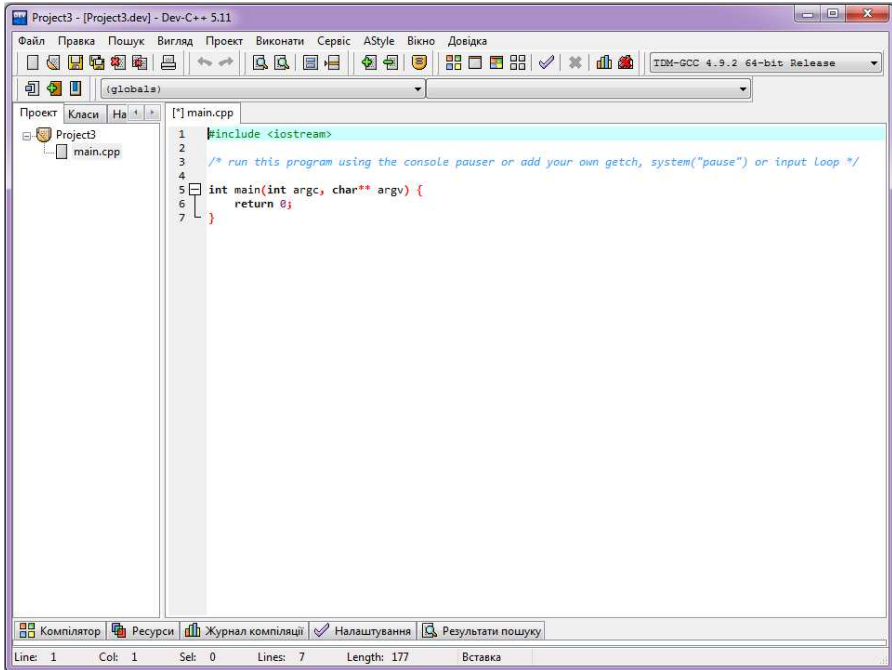


Рисунок 1.3 – Розроблення C++-проекту в Dev-C++

Якщо в процесі роботи над програмою виникли помилки під час її виконання і необхідно виконати зневадження, то потрібно в перший раз вибрати з меню пункт Виконати → Налаштування. Після цього необхідно дозволити зневадження у вікні підтвердження.

Для виконання зневадження потрібно встановити точки зупину на необхідних рядках за допомогою клавіши F4, натискаючи лівою кнопкою миші на номері відповідного рядка програми зліва або за допомогою пункту меню Виконати → Встановити/зняти точку зупину.

Після того, як всі точки зупину визначено, у панелі Налаштування в нижній частині вікна необхідно натиснути на кнопку Налаштування, а далі переміщатися за текстом програми між точками зупину (рис. 1.4) необхідним чином, для цього призначені кнопки Наступний крок (дозволяє перейти до наступного оператора поточного блоку програми), Продовжити виконання (дозволяє перейти до наступної точки зупину), Крок всередину (дозволяє перейти до наступного оператора з переміщенням за блоками

програми: наприклад, переходячи всередину функції, виклик якої відбувається на поточному рядку програми) тощо.

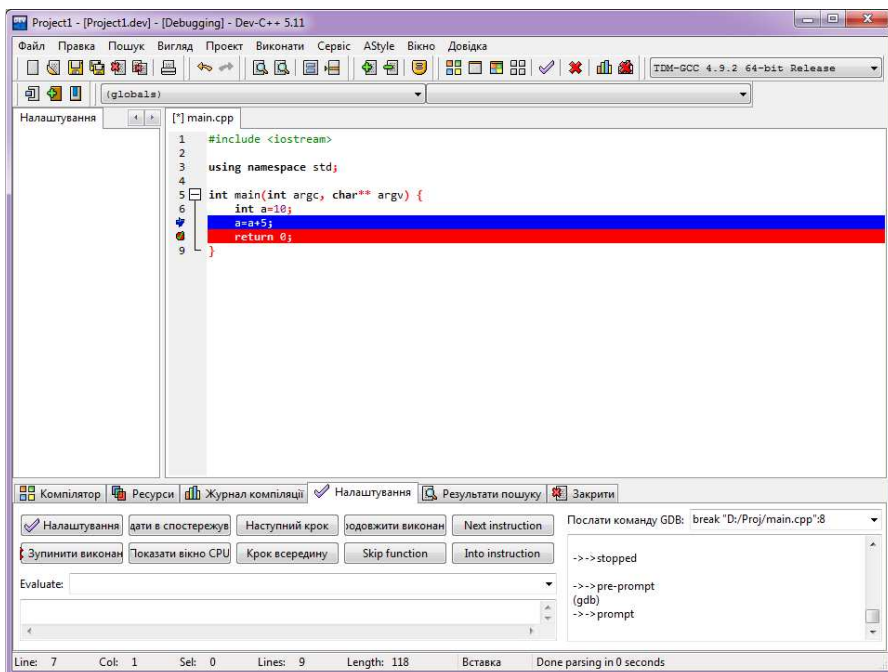


Рисунок 1.4 – Зневадження проекту в Dev-C++

У той момент, коли в процесі зневадження програму зупинено на відповідній точці зупину, можна навести курсор на деяку змінну і буде відображено поточне її значення (рис. 1.5).

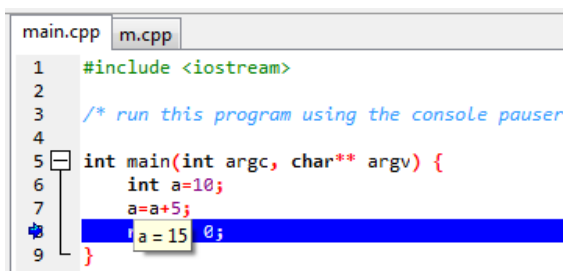


Рисунок 1.5 – Відображення значень змінних у процесі зневадження

1.2.3 Розроблення програм в інтегрованому середовищі Eclipse CDT

На відміну від Dev-C++ у Eclipse CDT в дистрибутив не входить компілятор, а тому інсталиувати та налаштувати компілятор доведеться самостійно. Окрім того використання платформи Eclipse призводить до того, що необхідно інсталиувати Java (<http://java.com/ru/download/>).

Після того, як інсталиовано Java, необхідно завантажити середовище та розпакувати його в будь-яку теку.

Для запуску середовища необхідно з теки з середовищем запустити на виконання файл eclipse.exe, після чого обрати робочий простір на диску (будь-яке місце на диску, де Ви плануєте зберігати Ваші програми). Далі на екрані з'явиться вікно з вітанням (рис. 1.6), з якого за допомогою лівої панелі необхідно розгорнути редактор з браузером.

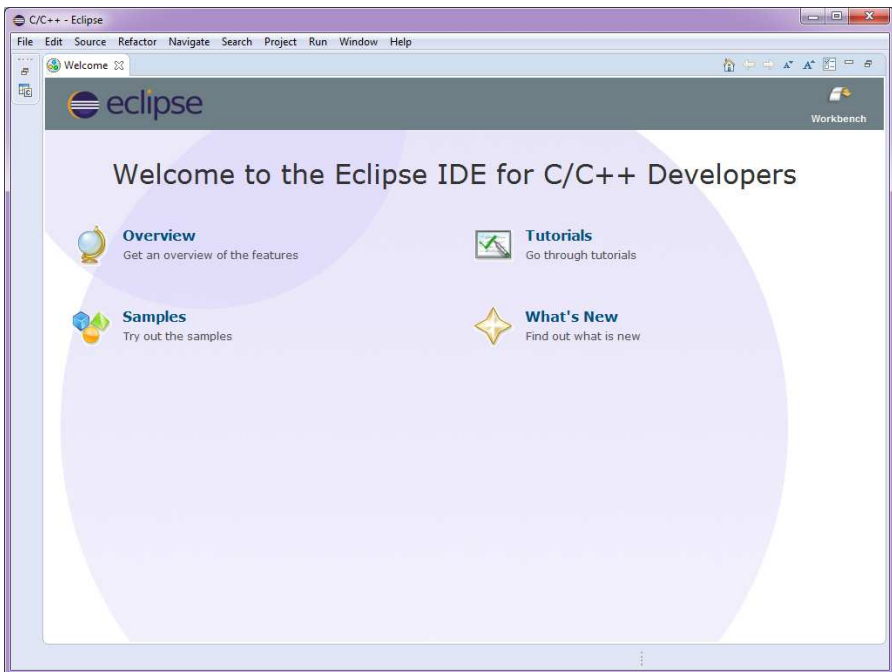


Рисунок 1.6 – Вікно ініціалізації середовища Eclipse CDT

Для створення виконуваного коду знадобиться компілятор, наприклад, MinGW, інстальатор якого можна завантажити за посиланням <http://sourceforge.net/projects/mingw/files/>. Далі необхідно запустити інстальатор MinGW Installation Manager та вибрати для інсталяції пакети mingw-32-base і mingw-gcc-g++, для чого з контекстного меню обрати пункт Mark for Installation, а далі обрати з меню Installation -> Apply Changes. Після інсталяції даних пакетів потрібно інсталювати пакет msys-base. Далі в системну змінну Path необхідно додати шлях до теки bin MinGW та до теки bin msys (Панель управління -> Система -> Додаткові параметри системи -> Змінні середовища), а далі з переліку Системні змінні обрати змінну Path і натиснути кнопку Змінити, після чого додати до шляху через крапку з комою вказані вище параметри (наприклад; ;C:\MinGW\bin;C:\MinGW\msys\1.0\bin).

Після того, як середовище запущено, необхідно за допомогою меню File -> New -> C++ Project створити новий проект (рис. 1.7).

У вікні створення нового проекту необхідно виконати наступні дії:

- задати ім'я проекту в полі Project name;
- обрати місце розташування проекту: або використати розташування за замовчуванням, тобто в обраному раніше робочому просторі (натиснути на прапорець Use default location), або обрати будь-яке інше місце на диску (зняти прапорець Use default location та обрати в полі Location місце розташування проекту, ввівши шлях вручну або натиснувши на кнопку Browser);

- визначити тип проекту за допомогою двох вікон Project type та Toolchains: тип проекту Executable (виконуваний) дозволяє створити виконуваний додаток; для виконуваних проектів передбачено шаблони Empty Project (створює теку вихідних кодів проекту, що не містить жодних файлів) та Hello World C++ Project (створює простий додаток типу Hello World з функцією main); якщо Ви інстальували компілятор MinGW, то у правому вікні слід обрати MinGW GCC.

Після того, як всі дані параметри було вірно задано, необхідно натиснути на кнопку Next > внизу вікна.

У випадку, якщо було обрано тип проекту Hello World C++ Project, то у наступному вікні доведеться задати такі параметри проекту як його автора (Author), повідомлення про авторські права (Copyright notice), текст вітання (Hello world greeting) та теку

розташування вихідних кодів проекту (Source). Дані параметри можна залишити за замовчуванням. Після того, як всі необхідні дії виконано, потрібно знов натиснути на кнопку Next > внизу вікна. Для Empty Project даний крок пропускається і відбувається одразу перехід до наступного кроку.

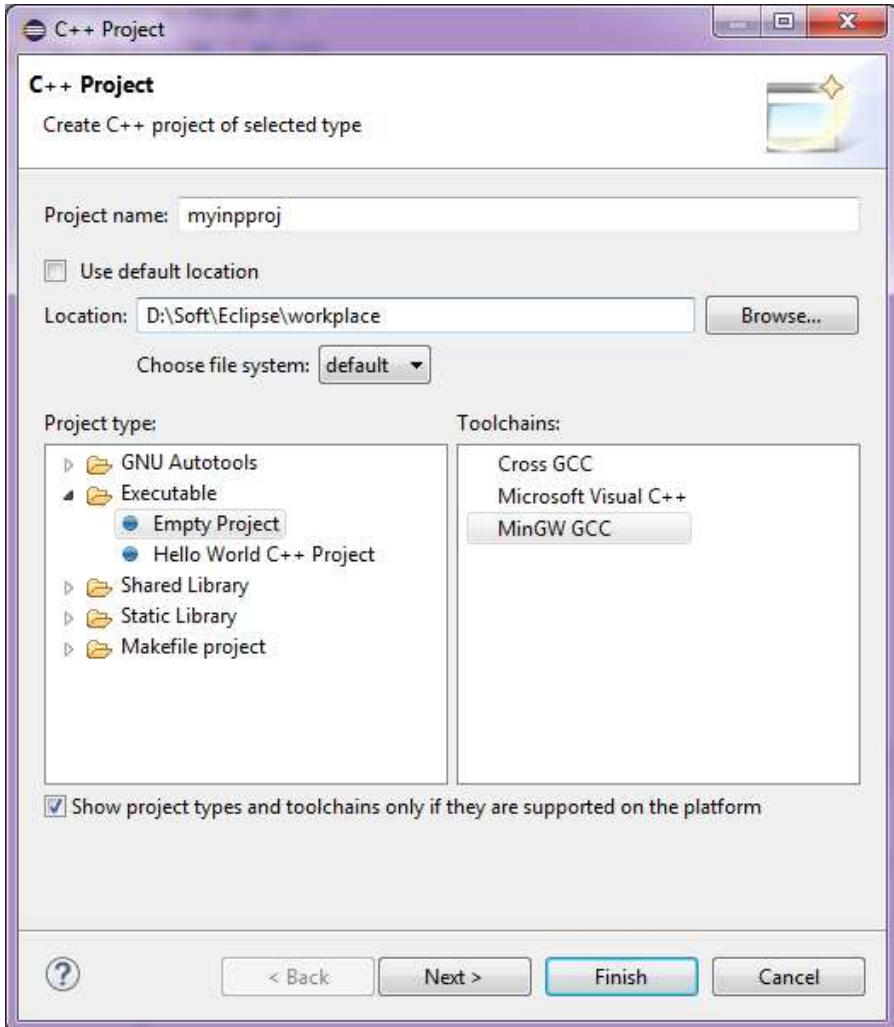


Рисунок 1.7 – Створення нового проекту

У наступному вікні можна задати тип конфігурації, який буде використовуватися: реліз (Release) або зневадження (Debug). Параметри можна залишити за замовчуванням. Для налаштування додаткових параметрів призначена кнопка Advanced settings... Для завершення процесу створення проекту необхідно натиснути на кнопку Finish.

Якщо було обрано порожній проект Empty Project, то в результаті середовище буде виглядати так, як представлено на рис. 1.8.

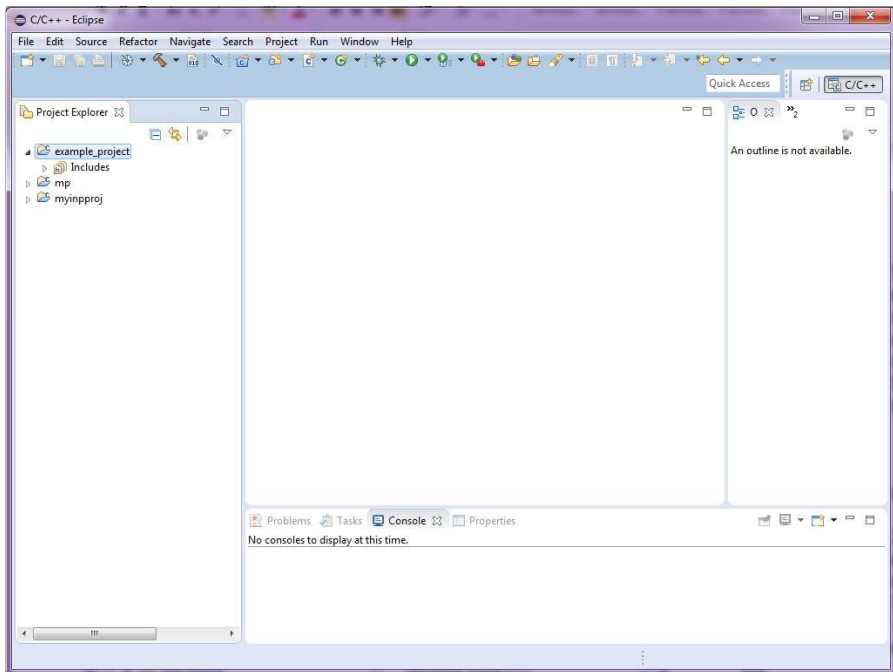


Рисунок 1.8 – Результат створення порожнього проекту в Eclipse CDT

У лівій частині вікна відображається браузер проектів, звідки можна отримати доступ до всіх проектів даного робочого простору. Для кожного проекту можна переглянути перелік файлів і відкрити їх у редакторі в центральній частині вікна, натиснувши двічі на назву відповідного файлу.

У нижній частині вікна відображаються закладки Problems та Console, через які відповідно виконується виведення помилок, які можуть виникнути в процесі роботи, та результатів роботи програми, які повинні виводитись на консоль.

У випадку створення порожнього проекту необхідно додати файл програми, у якому буде виконуватись реалізація необхідних алгоритмів для розв'язання поставлених завдань. Для цього необхідно обрати з меню File → New → Source File.

У вікні, що відкриється (рис. 1.9) необхідно вибрати проект, у якому потрібно створити даний файл, (Source folder), задати ім'я файлу (Source file), а також обрати шаблон (Template) і натиснути на кнопку Finish для завершення даного процесу.

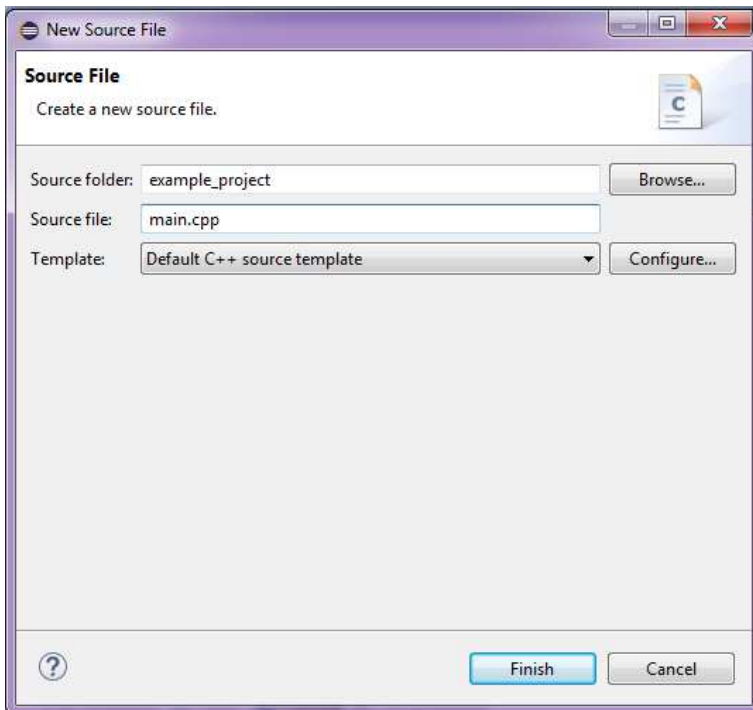


Рисунок 1.9 – Додавання нового файлу до проекту

У файлі, який буде додано до проекту та відображено у вікні редактора, потрібно додати відповідний текст програми (рис. 1.10).

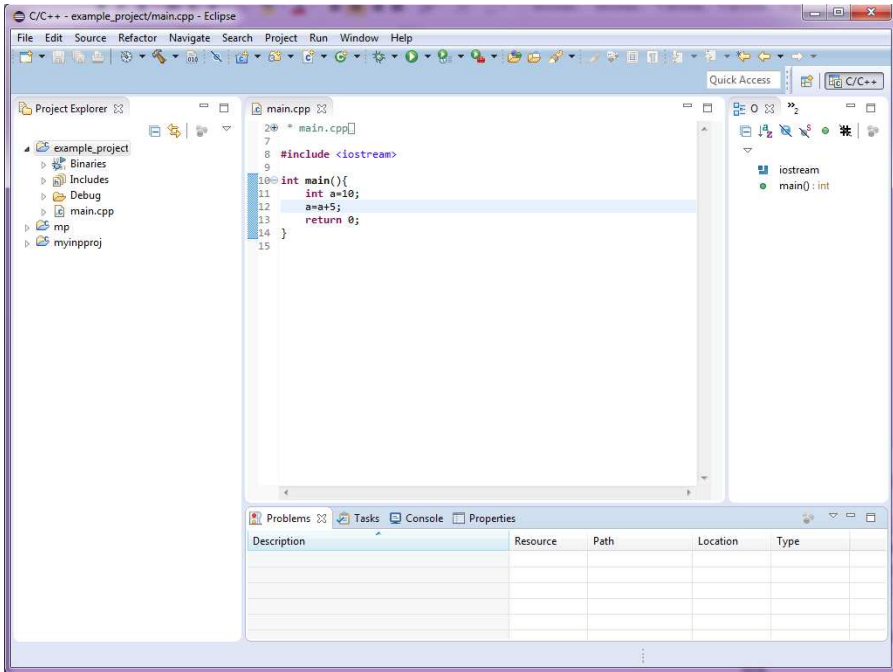


Рисунок 1.10 – Розроблення програми в Eclipse CDT

Після того, як текст програми сформовано, необхідно виконати компіляцію за допомогою пунктів меню Project → Build All, що дозволить створити відповідні виконувані файли.

Для запуску проекту необхідно обрати з меню пункти Run → Run Configurations..., після чого у вікні, що відкриється, (рис. 1.11) необхідно визначити потрібні параметри (у випадку нового проекту в лівій частині вікна з контекстного меню обрати пункт New та в новій конфігурації вибрати потрібний проект в браузері Project за допомогою кнопки Browse..., додаток в полі C/C++ Application за допомогою кнопки Search Project...) та натиснути на кнопку Run.

У результаті проект буде виконано, а у нижній частині вікна на закладці Console будуть виведені всі результати роботи програми, призначені для виведення на консоль.

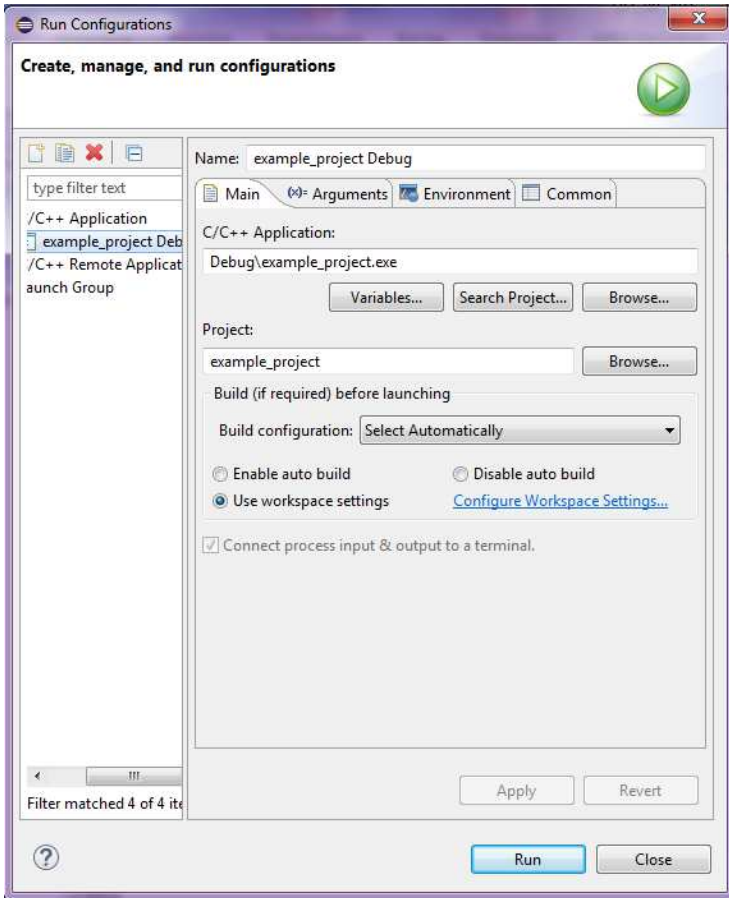


Рисунок 1.11 – Налаштування параметрів виконання проекту

Для зневадження проекту необхідно з контекстного меню вибрати пункт **Debug As -> Debug Configurations...**, після чого відкриється вікно, відображене на рис. 1.12. У даному вікні аналогічно вікну налаштування параметрів виконання проекту необхідно задати всі потрібні параметри, після чого натиснути на кнопку **Debug** для запуску процесу зневадження проекту.

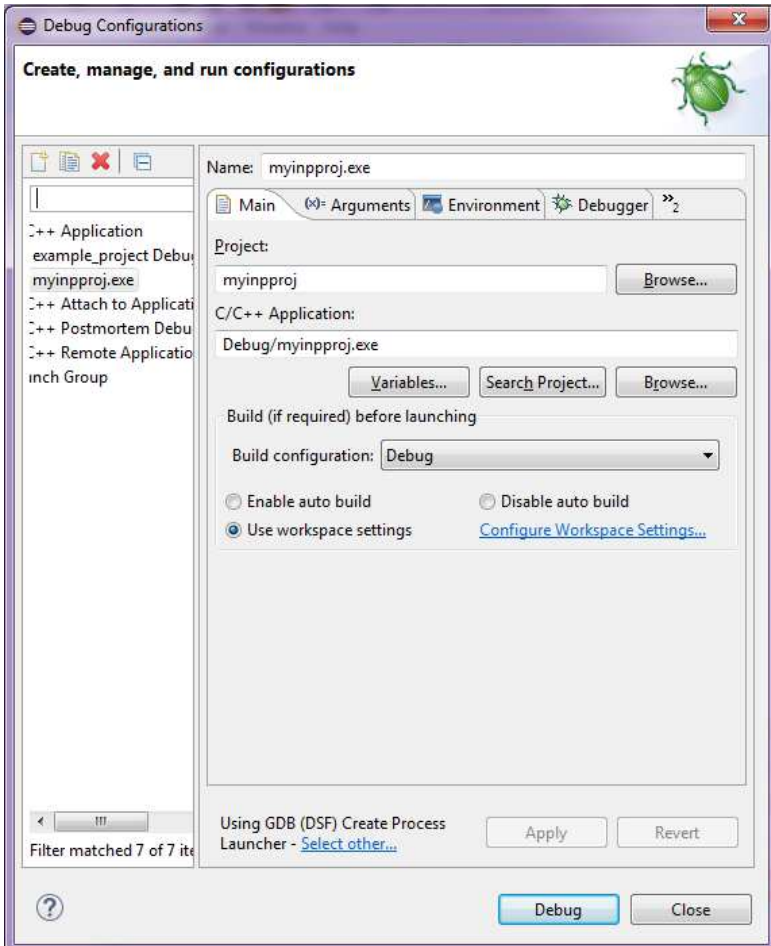


Рисунок 1.12 – Налаштування параметрів зневадження проекту

Після того, як процес зневадження проекту розпочнеться, вигляд середовища зміниться на вид, зображений на рис. 1.13.

Для того щоб встановити точки зупину, необхідно двічі натиснути на номер відповідного рядка. За допомогою інструментів меню можна переходити між точками зупину (або використовуючи клавішу F8 для продовження виконання проекту: в тому числі клавіші F5 для кроку всередину блоку і F6 для кроку повз блок коду).

У правій верхній частині середовища відображаються закладки, призначені для виведення поточних значень змінних (Variables), заданих виразів (Expressions).

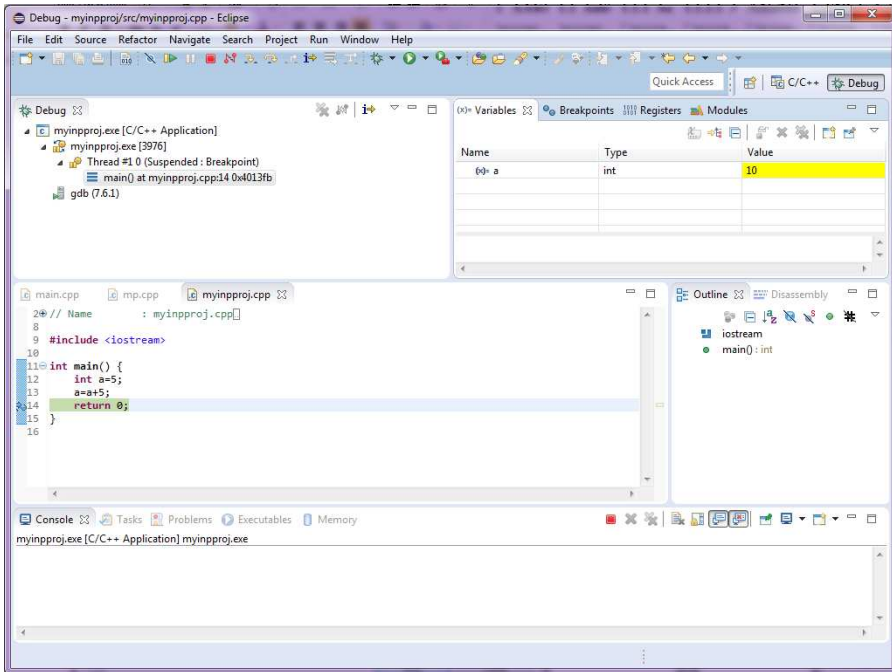


Рисунок 1.13 – Зневадження проекту

1.2.4 Оголошення змінних

Дані в програмі можна розділити на змінні і константи. Перед використанням змінні і константи повинні бути оголошені за допомогою оператора оголошення.

Змінна – це іменована область пам'яті, в яку записуються значення відповідно до оголошеного типу під час виконання програми. Оголошення змінної відбувається наступним чином:

EXP

тип ім'я_змінної;

Під типом даних розуміють множину допустимих значень цих даних і множину дозволених операцій над ними. Водночас тип даних визначає і розмір пам'яті, що займають змінні і константи даного типу. Типи даних, що використовуються у мовах C та C++ наведені у таблиці 1.1.

Приклад оголошення змінних та константи:

C++

```
int a=1, b;
const float g = 8.1;
```

Таблиця 1.1 – Типи даних у мові C++

Тип	Розмір пам'яті, байт	Діапазон значень
[signed] char	1	-128...127
unsigned char	1	0...255
[signed] short [int]	2	-32768...32767
unsigned short [int]	2	0...65535
[signed] int	4	-2147483648...2147483647
unsigned int	4	0... 4294967295
[signed] long [int]	4	-2147483648...2147483647
unsigned long [int]	4	0...4294967295
float	4	3.4e-38...3.4e38
double	8	1.7e-308...1.7e308

Порядок обчислення виразу визначається розташуванням знаків операцій, круглих дужок і пріоритетами виконання операцій. Вирази із найвищим пріоритетом обчислюються першими.

Символ «=» означає бінарну операцію простого присвоювання, у результаті виконання якої значення правого операнду присвоюється лівому операнду:

EXP

```
ім'я_змінної = вираз;
```

1.2.5 Базові конструкції мови C++**1.2.5.1 Умовний оператор if-else**

Умовний оператор належить до категорії операторів керування та забезпечує виконання або невиконання деякого оператора або групи операторів в залежності від заданої умови.

Оператор if-else може бути використаний у такому вигляді:

EXP

```
if (умовний вираз) оператор1;  
else оператор2;  
//або  
if (умовний вираз) {група операторів}  
else if (умовний вираз) {група операторів}  
    else {група операторів}
```

Умовний вираз може містити логічні оператори (&&, ||) та оператори порівняння (==, !=, >, <, >=, <=) тощо.

1.2.5.2 Оператор множинного вибору switch

Оператор **switch** дозволяє виконувати вибір з множини наявних варіантів і має наступну форму запису:

EXP

```
switch (вираз) {  
    case константа1:  
        послідовність операторів;  
    break;
```

```

case константа2:
    послідовність операторів;
    break;

.....
case константаN:
    послідовність операторів;
    break;
default:
    послідовність операторів;
}

```

Оператор вибору *switch* працює наступним чином: спочатку обчислюється *вираз*, який стоїть у дужках після *switch*, далі виконується перехід на одну з міток (*константа1*, ..., *константаN*), позначену ключовим словом *case*, значення константи після якої дорівнює значенню обчисленого виразу.

Зазначені константи повинні **обов'язково** бути цілочисельного типу.

Якщо вираз в дужках не дорівнює жодній з наявних констант, то виконується перехід на мітку *default*.

1.2.6 Введення/виведення даних

При запуску програми мовою C++ автоматично створюється декілька стандартних потоків, зокрема: *cin* (стандартний потік вводу з клавіатури) та *cout* (стандартний потік виводу на екран). Для того щоб використовувати ці потоки, достатньо підключити заголовний файл `<iostream>` та вказати стандартний простір імен.

Наприклад, введення даних виконується наступним чином:

C++

```

int i;
cin >> i;

```

Приклад виведення даних:

C++**cout**<<"i = " <<i;

1.2.7 Математичні функції мови C++

Для використання математичних функцій у програмах мовами C та C++ необхідно підключити заголовний файл <math.h>. Основні функції даної бібліотеки:

- cos – косинус;
- acos – арккосинус;
- exp – експонента;
- log – натуральний логарифм;
- round – повертає значення, округлене до цілого (значення, що повертається, є значенням з плаваючою комою);
- floor – округлення до найближчого меншого цілого числа;
- ceil – округлення до найближчого більшого цілого числа;
- pow (x,y) – піднесення x у ступінь y;
- sin – синус;
- asin – арксинус;
- tan – тангенс;
- atan – арктангенс;
- sqrt – квадратний корінь;
- fabs – абсолютна величина для чисел з плаваючою крапкою;
- random – виводить випадкове число від 0 до значення аргумента.

1.3 Завдання до роботи

1.3.1 Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи дані методичні вказівки, лекції, а також рекомендовану літературу.

1.3.2 Роз'язати за допомогою програми мовою C++ задачі індивідуального завдання, отриманого у викладача. Результат виконання завдання повинен виводитись програмою на екран.

1.3.3 Оформити звіт з роботи.

1.3.4 Відповісти на контрольні запитання.

1.4 Зміст звіту

1.4.1 Мета роботи.

1.4.2 Завдання до роботи.

1.4.3 Короткі теоретичні відомості.

1.4.4 Текст програми.

1.4.5 Результати роботи програми.

1.4.6 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

1.5 Контрольні запитання

1.5.1 Яка основна структура програми мовою C++?

1.5.2 Для чого необхідна функція main?

1.5.3 Як створити виконуваний код програми?

1.5.4 Що таке змінна?

1.5.5 Яким чином відбувається оголошення змінної?

1.5.6 Що таке константа?

1.5.7 Які типи даних вам відомі?

1.5.8 Який розмір пам'яті відповідає відомим вам типам даних?

1.5.9 Які існують модифікатори типів?

1.5.10 Яким чином визначається порядок обчислення виразу?

1.5.11 Наведіть операції за пріоритетом.

1.5.12 Коли використовують оператори вибору та множинного вибору?

1.5.13 Яким чином можна використати умовний оператор if-else?

1.5.14 З чого складається умовний вираз?

1.5.15 Яким чином можна використати оператор множинного вибору switch?

1.5.16 Яким чином організувати введення та виведення даних?

1.5.17 Як додати коментарі до програми?

1.5.18 Наведіть основні математичні функції.

1.5.19 Що таке IDE? Які існують сучасні IDE?

1.5.20 Яким чином виконується зневадження в IDE?

2 ЛАБОРАТОРНА РОБОТА № 2 ПРОГРАМУВАННЯ НА ОСНОВІ АЛГОРИТМІВ З ЦИКЛІЧНОЮ СТРУКТУРОЮ

2.1 Мета роботи

Отримати знання і навички, необхідні для програмування з використанням алгоритмів з циклічною структурою, та навчитися використовувати їх на практиці в процесі розроблення програм мовою програмування C++.

2.2 Основні теоретичні відомості

У процесі розроблення алгоритмів, які дозволяють розв'язати ту чи іншу задачу, часто виникає необхідність неодноразового повторення однотипних обчислень над різними даними. Для досягнення таких цілей призначені оператори циклів.

Цикл представляє собою частину програми (алгоритму), у якій одні й ті самі обчислення виконуються неодноразово над різними значеннями одних і тих самих змінних.

У C++ існують три оператори, які використовуються для створення циклів: `while`, `for` та `do-while`.

2.2.1 Цикл з передумовою `while`

Цикл `while` належить до циклів з передумовою. Цикл `while` використовується тоді, коли точно невідома кількість ітерацій і під час побудови алгоритму з циклічною структурою не виникає необхідність виконання циклу хоча б один раз.

Цикл `while` має наступну форму запису:

EXP

`while (вираз) {група операторів}`

Якщо *вираз* має істинне значення, то тіло циклу, представлене групою операторів, виконується один раз, після чого істинність *виразу* перевіряється знову. Ітерації, що включають виконання тіла цикла, повторюються доти, доки *вираз* не отримає хибне значення.

Наприклад:

C++

```
int i=1, rez=0;
while (i<=10&&rez!=10)
{ i++;
  cout<<"\nВведіть число:" ;
  cin>>rez;
}
```

2.2.2 Цикл з передумовою for

Цикл *for* також є циклом з передумовою, але на відміну від циклу *while* зазвичай використовується у випадку, коли відома точна кількість разів, яку необхідно виконати тіло циклу. У циклі *for* виконуються три операції: ініціалізація лічильників циклів, порівняння значення лічильників з деяким граничним значенням і змінювання значення лічильників при кожному виконанні тіла циклу.

Цикл *for* в загальному вигляді можна представити наступним чином:

EXP

```
for (вираз1; вираз2; вираз3) {тіло циклу}
```

У процесі виконання циклу *for* *вираз1* обчислюється першим. Зазвичай в ньому виконується ініціалізація лічильників циклів і змінних. Даний вираз обчислюється один раз, коли цикл *for* починає виконуватися.

Далі реалізується друга операція, описана вище для циклу *for*: обчислюється *вираз2*. Даний вираз дозволяє виконати перевірку умови. Якщо значення *виразу2* істинне, то виконується тіло циклу. Якщо значення *виразу2* хибне, то виконання циклу завершується.

Значення *виразу3* обчислюється в кінці кожного виконання тіла циклу.

Наприклад:

C++

```
int i, rez = 1, N=10;
for ( i = 1; i <= N; i++ ) rez = rez + i;
cout<<"rez="<<rez;
```

2.2.3 Цикл з постумовою *do-while*

На відміну від циклів *for* і *while* цикл *do-while* є циклом з постумовою і використовується у тих випадках, коли невідома точна кількість ітерацій, але цикл має бути виконано не менше ніж один раз. Цикл *do-while* в загальному вигляді можна представити наступним чином:

EXP

do оператор while (вираз);

Проілюструємо його виконання таким прикладом:

C++

```
int rez;
do
{ cout<<"\nВведіть число:" ;
  cin>>rez;
} while (rez != 10);
```

Обчислення суми – типова циклічна задача. Особливістю *задачі обчислення нескінченної суми* є те, що кількість доданків невідома попередньо. Тому виконання циклу повинно завершуватись у момент досягнення необхідної точності.

Для виконання обчислення нескінченної суми необхідно на кожній ітерації розрахувати доданок, накопичуючи суму, і якщо доданок виявився меншим за вказану точність, то потрібно завершити цикл, а якщо ні – то продовжити.

2.2.4 Перетворення з однієї системи числення в іншу

Кожна система числення характеризується своїм набором знаків для представлення чисел. У двійковій системі числення користуються лише двома цифрами: 0 і 1. У вісімковій – цифрами від 0 до 7, а в шістнадцятковій – цифрами від 0 до 9 і літерами від 'A' до 'F', які відповідають десятковим числам від 10 до 15.

Для переведення числового значення з однієї системи числення в іншу використовують ділення у стовпчик.

Розглянемо приклад перетворення числа з десятикової системи числення у двійкову. При діленні числа n на 2 під числом n записуємо остачу від ділення n на 2, під двійкою – частку. Процес ділення закінчується тоді, коли частка стане рівною 1. Далі слід записати останню частку (одиницю) і всі остачі від ділення у зворотному порядку. Наприклад, знайдемо двійкове представлення числа 20:

20	2			
0	10	2		
	0	5	2	
		1	2	2
			0	1

Записавши залишки у зворотному порядку, отримаємо: $20_{10} = 10100_2$.

2.2.5 Знаходження найбільшого спільного дільника та найменшого спільного кратного двох чисел

Найбільшим спільним дільником (НСД) двох цілих чисел a та b , які одночасно не дорівнюють нулю, називається таке найбільше ціле число d , на яке a та b діляться без залишку. Тобто $d = \text{НСД}(a, b)$.

Найменшим спільним кратним (НСК) двох цілих чисел a та b називається найменше додатне ціле число, кратне як a , так і b .

Для пошуку НСД використовується алгоритм Евкліда, який є ітеративним. Для того щоб знайти $\text{НСД}(a, b)$ на спочатку знаходять залишок c_1 від ділення a на b . Потім знаходять залишок від ділення b на c_1 . Дані ітерації повторюють до тих пір, поки не буде отримано залишок, що дорівнює 0. НСД є останній не нульовий залишок c_{n-1} (якщо було виконано n кроків).

Виходячи з відомого НСД двох чисел, можна обчислити $\text{НСК}(a, b) = |ab| / \text{НСД}(a, b)$.

2.3 Завдання до роботи

2.3.1 Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи дані методичні вказівки, лекції, а також рекомендовану літературу.

2.3.2 Роз'язати за допомогою програми мовою C++ задачі індивідуального завдання, отриманого у викладача.

2.3.3 Оформити звіт з роботи.

2.3.4 Відповісти на контрольні запитання.

2.4 Зміст звіту

2.4.1 Мета роботи.

2.4.2 Завдання до роботи.

2.4.3 Короткі теоретичні відомості.

2.4.4 Текст програми.

2.4.5 Результати роботи програми.

2.4.6 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

2.5 Контрольні запитання

2.5.1 Для чого призначені та яким чином реалізуються алгоритми з циклічною структурою?

2.5.2 Які існують види циклів?

2.5.3 З яких складових побудований цикл у C++?

2.5.4 Наведіть приклади задач, для розв'язання яких використовуються цикли.

2.5.5 Наведіть блок-схему циклу з передумовою та поясніть принцип дії алгоритму.

2.5.6 Наведіть блок-схему циклу з постумовою та поясніть принцип дії алгоритму.

2.5.7 Наведіть блок-схему ітераційного циклу та поясніть принцип дії алгоритму.

2.5.8 Який цикл мови C++ реалізує цикл с передумовою? Наведіть форму його запису.

2.5.9 Який цикл мови C++ реалізує цикл с постумовою? Наведіть форму його запису.

2.5.10 Який цикл мови C++ реалізує ітераційний цикл? Наведіть форму його запису.

2.5.11 У яких випадках необхідно застосовувати цикл while?

2.5.12 У яких випадках необхідно застосовувати цикл for?

2.5.13 У яких випадках необхідно застосовувати цикл do-while?

2.5.14 Чим відрізняється цикл while від циклу for?

2.5.15 Якими способами можна реалізувати зчитування послідовності чисел деякої довжини з клавіатури?

2.5.16 Яким чином виконати розрахунок нескінчених сум?

2.5.17 Для чого використовуються цикли з постумовою.

2.5.18 Яким чином виконати перетворення з однієї системи числення в іншу?

2.5.19 У чому полягає алгоритм Евкліда для знаходження найбільшого спільного дільника?

2.5.20 Яким чином визначити найбільше спільне кратне двох чисел?

3 ЛАБОРАТОРНА РОБОТА № 3 ПРОГРАМУВАННЯ З ВИКОРИСТАННЯМ МАСИВІВ ТА ПОКАЖЧИКІВ

3.1 Мета роботи

Отримати знання і навички, необхідні для програмування з використанням масивів та покажчиків, та навчитися використовувати їх на практиці в процесі розроблення програм мовою програмування C++.

3.2 Основні теоретичні відомості

3.2.1 Масиви

Масив – це набір однотипних елементів, які мають спільне ім'я, а доступ до них виконується за індексом у довільному порядку. Елементи масиву займають одну неперервну область оперативної пам'яті і розміщені в ній послідовно безпосередньо один за одним, починаючи з базової адреси.

Наприклад:

C++

```
int a[10];
```

Квадратні дужки означають, що ідентифікатори, після яких вони стоять, є іменами масивів. Число в дужках, задане під час оголошення масиву, визначає кількість елементів масиву. Доступ до окремого елементу масиву здійснюється з використанням номера цього елементу (індексу). Нумерація елементів масиву починається з нуля і закінчується $n - 1$, де n – кількість елементів масиву.

Ініціалізація масиву відбувається при оголошенні. Масиви можна ініціалізувати списком значень або списком відокремлених комою виразів, розташованих у фігурних дужках. Наприклад:

C++

```
int a[3]={1,2,3};
```

Матриця – двомірний масив. Перший індекс матриці задає номер рядка, а другий – стовпця.

3.2.2 Показчики

Операція визначення адреси «&» дозволяє визначити адресу комірки пам'яті, що містить задану змінну. Наприклад, якщо *a* – ім'я змінної, то *&a* – адреса змінної *a*.

Показчик – це похідний тип даних, який використовується для зберігання адрес змінних. Тобто значенням змінної-показчика є адреса змінної. Опис таких змінних у загальному вигляді можна представити наступним чином:

EXP

```
<тип> *<ім'я показчика>;
```

Приклади опису показчика:

C++

```
int *pt; //показчик на змінну цілого типу
char *pc; //показчик на змінну символьного типу
float *pf; //показчик на змінну дійсного типу
```

Змінні різних типів займають різну кількість комірок пам'яті. При цьому для деяких операцій з показниками необхідно знати об'єм відведеної пам'яті. Однак самі змінні типу показчик мають однаковий розмір.

Виходячи з описаної вище операції визначення адреси, якщо змінна-показчик має ім'я *ptr* (оголошена як *int* ptr*), тоді в якості значення їй можна присвоїти адресу наступним чином:

C++

```
ptr=&vr;
```

Операція розіменування показчика (непрямої адресації) «*» дозволяє звертатися до змінної не напряму, а через показчик, який містить адресу цієї змінної.

Якщо *ptr* – показчик, то **ptr* – це значення змінної, на яку вказує *ptr*. Для прикладу:

C++

```
int *ptr;  
*ptr=1;
```

Для отримання значення елементу масиву через показчик необхідно використати операцію розіменування.

У наступному прикладі представлено два способи знаходження суми елементів деякого масиву:

C++

```
const int N = 10;  
int arr[N];  
int* ptr = arr;  
  
// 1 варіант: через показчик  
int sum1 = 0;  
for (ptr = arr; ptr < &arr[N]; ++ptr)  
    sum1 += *ptr;
```

```
// 2 варіант: з використанням індексів
int sum2 = 0;
for (int i = 0; i < N; ++i)
    sum2 += arr[i];
```

3.2.3 Динамічні масиви

Під динамічним масивом розуміють масив, розмірність якого стає відомою в процесі виконання програми. У програмах мовою C++ для роботи з динамічними масивами використовують оператори керування вільною пам'яттю *new* та *delete*.

Вільна пам'ять (heap) – це область пам'яті, яка виділяється системою для розміщення об'єктів, час життя яких напряду керується програмістом. За допомогою оператора *new* виділяється пам'ять під динамічний масив, а за допомогою оператора *delete* створений масив видаляється з пам'яті.

Оператор *new* має наступний формат:

EXP

```
new ім'я_типу[вираз];
```

У результаті виконання оператора *new* в пам'яті виділяється об'єм пам'яті, необхідний для зберігання даних вказаного типу, і повертається базова адреса.

Оператор *delete* має наступний формат:

EXP

```
delete вираз;  
delete[] вираз;
```

Розглянемо виділення пам'яті під динамічний масив:

C++

```
int* mas = new int[n];  
delete[] mas;
```

3.3 Завдання до роботи

3.3.1 Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи дані методичні вказівки, лекції, а також рекомендовану літературу.

3.3.2 Роз'язати за допомогою програми мовою C++ задачі індивідуального завдання, отриманого у викладача.

При цьому одне з завдань необхідно розв'язати таким чином, щоб передбачалися варіанти введення даних масиву як з клавіатури, так і завдання їх програмно.

Всі завдання розв'язати двома способами: за допомогою операцій для роботи з масивами та за допомогою покажчиків.

Під час ініціалізації масивів з клавіатури передбачити динамічне виділення пам'яті.

3.3.3 Оформити звіт з роботи.

3.3.4 Відповісти на контрольні запитання.

3.4 Зміст звіту

3.4.1 Мета роботи.

3.4.2 Завдання до роботи.

3.4.3 Короткі теоретичні відомості.

3.4.4 Текст програми.

3.4.5 Результати роботи програми.

3.4.6 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

3.5 Контрольні запитання

3.5.1 Що розуміють під масивом?

3.5.2 Яким чином можна оголосити масив?

- 3.5.3 Як можна ініціалізувати масив?
- 3.5.4 Значення яких типів можуть приймати елементи масиву?
- 3.5.5 Яким чином отримати доступ до елементів масиву?
- 3.5.6 Яким чином розташовані елементи масиву в пам'яті?
- 3.5.7 Чим відрізняється матриця від масиву?
- 3.5.8 Що таке операція визначення адреси?
- 3.5.9 Що таке покажчик?
- 3.5.10 Яким чином оголосити покажчик?
- 3.5.11 Що таке операція розіменування покажчика?
- 3.5.12 Яким чином отримати доступ до елементів масиву через покажчик?
- 3.5.13 Які масиви називаються динамічними?
- 3.5.14 У якому випадку слід використовувати динамічні масиви, а у якому статичні?
- 3.5.15 Який синтаксис має оператор new?
- 3.5.16 Які дії виконує оператор new?
- 3.5.17 Який синтаксис має оператор delete?
- 3.5.18 Яким чином виділити пам'ять під динамічний масив?
- 3.5.19 Що таке heap?
- 3.5.20 Які операції можна виконувати над покажчиками?

4 ЛАБОРАТОРНА РОБОТА № 4 РОБОТА З СИМВОЛЬНИМИ РЯДКАМИ

4.1 Мета роботи

Отримати знання і навички, необхідні для програмування з використанням текстових даних (символьних рядків), та навчитися використовувати їх на практиці в процесі розроблення програм мовою програмування C++.

4.2 Основні теоретичні відомості

Рядок – це послідовність символів, яка закінчується нульовим байтом `'\0'`. Дана послідовність символів записується в подвійних лапках.

Послідовності в C++ представляються масивами або покажчиками, тому стандартно програми мовою C++ зберігають рядки як масиви типу `char`. Для оголошення строкової змінної необхідно всередині програми оголосити масив типу `char` з кількістю елементів, достатньою для зберігання необхідного рядка символів.

Головна відмінність між символьними рядками та іншими типами масивів полягає в тому, що C++ визначає останній елемент масиву. Програми мовою C++ представляють кінець символьного рядка за допомогою символу `NULL`, який в C++ зображується як спеціальний символ `'\0'`, або нульовий байт.

Створення рядків, наприклад, відбувається наступним чином:

C++

```
char str2[10]="Hello";
```

Для стандартної роботи з символьними рядками в стилі C призначена бібліотека `string.h`, яка містить наступні основні функції:

– `char *strcat(char *dest, const char *src);` – дописує рядок `src` в кінець `dest`;

– `char *strncat(char *dest, const char *src, size_t n)`; – дописує не більше n початкових символів рядка `src` (або весь `src`, якщо його довжина менше) у кінець `dest`;

– `char *strchr(const char *str, int ch)`; – шукає символ `ch` у рядку `str`, починаючи з голови, і повертає його адресу, або `NULL` якщо символ не знайдено;

– `char *strrchr(const char *str, int ch)`; – шукає символ `ch` у рядку `str`, починаючи з хвоста і повертає його адресу, або `NULL` якщо символ не знайдено;

– `int strcmp(const char *str1, const char *str2)`; – лексикографічне порівняння рядків;

– `int strncmp(const char *str1, const char *str2, size_t n)`; – лексикографічне порівняння перших n байтів рядків;

– `char *strcpy(char *dest, const char *src)`; – копіює рядок з `src` у `dest`;

– `char *strncpy(char *dest, const char *src, size_t n)`; – копіює до n байт рядку з `src` у `dest`;

– `size_t strlen(const char *str)`; – повертає довжину рядка;

– `size_t strspn(const char *str, const char *alp)`; – визначає максимальну довжину початкового підрядка з `str`, що складається виключно з байтів, перерахованих в `alp`;

– `char *strpbrk(const char *str, const char *alp)`; – знаходить перше входження будь-якого символу, перерахованого в `alp`, у `str`;

– `char *strstr(const char *str, const char *pstr)`; – знаходить перше входження рядка `pstr` у `str`;

– `char *strtok(char *src, const char *delim)`; – перетворює рядок у послідовність токенів.

Однак окрім розглянутого вище стандартного способу роботи з символьними рядками в бібліотеці STL C++ є ще одна бібліотека для роботи з рядками. Вона дозволяє легко вирішувати наступні задачі з високою ефективністю:

– створювати, присвоювати, копіювати і видаляти рядки;

– виконувати перетворення типів символьних змінних;

– порівнювати рядки;

– поєднувати рядки;

– визначати довжину рядка;

– знаходити і заміщати потрібний фрагмент у рядку.

Для використання даної бібліотеки необхідно підключити заголовний файл <string>. Після цього операція створення нового рядка виявиться настільки ж простою, як і створення змінної будь-якого базового типу.

C++

```
string hi("hello");  
string lo="greetings";  
string es="";
```

Рядковим змінним можна присвоювати значення, як і змінним будь-яких інших типів:

C++

```
string name("Fred");  
name = "Paul";
```

При цьому всі операції розподілу пам'яті будуть виконані коректно.

Операції + (конкатенація), операція порівняння (==, > тощо) дозволяють легко виконувати відповідні дії над рядками.

4.3 Завдання до роботи

4.3.1 Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи дані методичні вказівки, лекції, а також рекомендовану літературу.

4.3.2 Розв'язати за допомогою програми мовою C++ задачі індивідуального завдання, отриманого у викладача.

Завдання а) розв'язати двома способами: виконуючи дії над масивом символів та за допомогою функцій бібліотеки string.h.

Завдання б) розв'язати за допомогою функцій бібліотеки string.

Завдання в) розв'язати одним зі способом на вибір: за допомогою функцій бібліотек string.h або string.

У завданні в) передбачити, що в тексті можуть зустрічатися будь-які розділові знаки.

4.3.3 Оформити звіт з роботи.

4.3.4 Відповісти на контрольні запитання.

4.4 Зміст звіту

4.4.1 Мета роботи.

4.4.2 Завдання до роботи.

4.4.3 Короткі теоретичні відомості.

4.4.4 Текст програми.

4.4.5 Результати роботи програми.

4.4.6 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

4.5 Контрольні запитання

4.5.1 Що розуміють під символьним рядком?

4.5.2 Чим відрізняються рядки від масивів символів?

4.5.3 Чим відрізняється запис рядків від запису окремих символів?

4.5.4 Яким типом даних представляються рядки в мові C++?

4.5.5 Яким чином оголосити рядок у C++?

4.5.6 Якою може бути максимальна довжина рядка в C++?

4.5.7 Які бібліотеки призначені для роботи з рядками?

4.5.8 Які основні функції надає бібліотека string.h?

4.5.9 Яким чином дозволяє працювати з рядками бібліотека string?

4.5.10 Чим робота з рядками в бібліотеці string відрізняється від бібліотеки string.h?

4.5.11 Які основні функції роботи з рядками надає бібліотека string?

4.5.12 Чим відрізняється робота з рядками як з масивами символів та за допомогою бібліотечних функцій? Який зі способів у яких випадках є ефективнішим?

4.5.13 Яким чином визначити довжину рядка?

4.5.14 Яким чином виділити слова з рядка?

4.5.15 Що таке конкатенація рядків та яким чином вона виконується?

4.5.16 Яким чином порівняти значення двох рядків?

4.5.17 Яким чином знайти рядок у рядку?

4.5.18 Яким чином знайти символ у рядку?

4.5.19 Яким чином перетворити рядок на значення іншого типу?

4.5.20 Яким чином перетворити значення деяких типів у символічний тип?

5 ЛАБОРАТОРНА РОБОТА № 5 ПРОГРАМУВАННЯ З ВИКОРИСТАННЯМ ФУНКЦІЙ

5.1 Мета роботи

Отримати знання і навички, необхідні для програмування на основі створення і використання користувацьких функцій, та навчитися використовувати їх на практиці в процесі розроблення програм мовою програмування C++.

5.2 Основні теоретичні відомості

5.2.1 Визначення та виклик функцій у програмі

Функція – це іменована частина програмного коду, призначена для виконання визначених дій в межах програми.

З поняттям функції у мові C++ пов'язані:

- опис функції;
- прототип функції;
- виклик функції.

Опис функції складається з двох частин:

- заголовка;
- тіла.

Опис функції має наступну загальну форму запису:

EXP

```
[тип_результату] <ім'я_функції>([список_параметрів])  
{  
    тіло_функції  
}
```

Тип результату – це тип значення, яке може повертати функція. Специфікатор `void` використовується для визначення функцій, які не повертають ніякого значення.

У списку параметрів для кожного параметра необхідно зазначити тип параметра та його ім'я. Функція може не мати параметрів.

Тіло функції – це послідовність оголошень і операторів, які описують визначений алгоритм.

Оператор повернення з функції в точку виклику `return` [вираз] має подвійне призначення: він забезпечує негайне повернення у зовнішню функцію і може використовуватися для передачі у відповідну точку отриманого у функції значення.

Прототип функції може вказуватися до виклику функції замість опису функції для того, щоб компілятор міг виконати перевірку відповідності типів аргументів і параметрів. Прототип функції за синтаксисом подібний заголовку функції. Наприкінці прототипу ставиться крапка з комою. Параметри функції в прототипі можуть мати імена, але це не обов'язково.

Компілятор використовує прототип функції для порівняння типів аргументів з типами параметрів. Мова C++ не передбачає автоматичного перетворення типів у випадках, коли аргументи не співпадають за типами з відповідними їм параметрами.

Виклик функції можна представити за допомогою наступного загального вигляду:

EXP

ім'я_функції(список_аргументів);

Для функції *fun* з фактичним параметром *x*, який передається під час виклику, виклик може виконуватись, використовуючи змінну *res*, якій буде присвоєно значення, отримане з функції за допомогою оператора `return`, наступним чином:

– *fun(x)*; – якщо функція не повертає значення або якщо значення непотрібно зберігати;

– *res = fun(x)*; – якщо функція повертає значення.

Кількість і типи формальних аргументів повинні співпадати з кількістю і типом фактичних параметрів функції. При виклику функції фактичні параметри підставляються замість формальних аргументів.

Значення формальних параметрів можуть бути задані за замовчуванням. При цьому при описі функції перелік параметрів за замовчуванням повинен міститися в кінці списку формальних параметрів функції. Наприклад:

C++

```
void fun(int i, int j = 7);  
void myfun(int i, int j = 7, int k = 8);
```

Приклад функції, яка використовується для обчислення суми двох чисел:

C++

```
int add_values (int a, int b)  
{ int result;  
  result = a+b;  
  return result;  
}
```

У тексті програми виклик даної функції можна виконати наступним чином:

C++

```
int result=add_values (1, 2);
```

5.2.2 Рекурсивні функції

Рекурсивна функція – це функція, у тілі якої здійснюється виклик цієї ж функції.

Приклад рекурсивної функції:

C++

```
int sum (int n) {  
    if (n == 1)  
        return 1;  
    else  
        return n + sum(n-1);  
}
```

5.3 Завдання до роботи

5.3.1 Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи дані методичні вказівки, лекції, а також рекомендовану літературу.

5.3.2 Роз'язати за допомогою програми мовою C++ задачі індивідуального завдання, отриманого у викладача. Визначити відповідні програмні функції та виконати їх виклик для заданих з клавіатури значень.

5.3.3 Оформити звіт з роботи.

5.3.4 Відповісти на контрольні запитання.

5.4 Зміст звіту

5.4.1 Мета роботи.

5.4.2 Завдання до роботи.

5.4.3 Короткі теоретичні відомості.

5.4.4 Текст програми.

5.4.5 Результати роботи програми.

5.4.6 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

5.5 Контрольні запитання

5.5.1 Що таке функція?

5.5.2 Для чого використовуються функції?

5.5.3 Яким чином виконується опис функції?

5.5.4 Що таке прототип функції?

- 5.5.5 Яким чином можна визначити прототип функції?
- 5.5.6 Яким чином виконується виклик функції?
- 5.5.7 Значення яких типів може повертати функція?
- 5.5.8 Яким чином повернути значення з функції?
- 5.5.9 Яким чином задати значення параметрів функції за замовчуванням?
- 5.5.10 Які типи можуть мати параметри функції?
- 5.5.11 Яким чином можна передати у функцію в якості аргументу масив?
- 5.5.12 Для чого використовуються посилання?
- 5.5.13 Які існують варіанти передачі аргументів у функцію?
- 5.5.14 Що таке рекурсивна функція?
- 5.5.15 Для виконання яких дій можуть використовуватися рекурсивні алгоритми?

6 ЛАБОРАТОРНА РОБОТА № 6 СТРУКТУРИ. ПЕРЕЛІЧЕННЯ. ОБ'ЄДНАННЯ

6.1 Мета роботи

Отримати знання і навички, необхідні для розроблення складних типів даних: структур, перелічень, об'єднань та навчитися застосовувати їх на практиці в процесі розроблення програм мовою програмування C++.

6.2 Основні теоретичні відомості

Структури дозволяють групувати у одній змінній пов'язані дані, типи яких можуть різнитися.

Для визначення структури використовується наступна загальна форма:

EXP

```
struct ім'я_структури {  
    тип_1 поле_1;  
    тип_2 поле_2;  
    ...  
    тип_N поле_N;  
} список_змінних;
```

Розглянемо приклад визначення структури:

C++

```
struct building  
{ char *owner;  
  char *city;  
  int amountRooms;  
  float price;  
};
```

Для визначення змінних заданого типу та встановлення значень може використовуватися наступний спосіб:

C++

```
building myb;
myb.price=300;
myb.amountRooms=3;
myb.city="Zaporizhzhia";
myb.owner="P. Petrenko";
```

Перелічення використовуються, коли необхідно оголосити деяку змінну, яка приймає значення з деякого набору.

Приклад визначення перелічення:

C++

```
enum open_modes{ input = 1, output, append };
```

Під об'єднанням розуміють тип, у якому всі змінні розділяють одну ділянку пам'яті.

Для визначення об'єднання використовується наступна загальна форма:

EXP

```
union ім'я_типу {
    тип_1 поле_1;
    тип_2 поле_2;
    ...
    тип_N поле_N;
} перелік_змінних;
```

Об'єднання є структурами даних, які дозволяють зберігати пов'язані частини інформації всередині однієї змінної. На відміну від структури об'єднання зберігає значення тільки одного елементу в кожний момент часу.

Наприклад:

C++

```
union distance
{
    int miles;
    long meters;
} walk;
```

6.3 Завдання до роботи

6.3.1 Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи дані методичні вказівки, лекції, а також рекомендовану літературу.

6.3.2 Роз'язати за допомогою програми мовою C++ задачу індивідуального завдання, отриманого у викладача.

Для кожного завдання передбачити реалізацію функцій для роботи зі структурами: введення, виведення та виконання дій у відповідності з індивідуальним завданням.

Використовувати структури та бітові поля, перелічення, об'єднання, де це необхідно, для розв'язання задач оптимальним чином.

При введенні даних з клавіатури виділяти пам'ять динамічно.

6.3.3 Оформити звіт з роботи.

6.3.4 Відповісти на контрольні запитання.

6.4 Зміст звіту

6.4.1 Мета роботи.

6.4.2 Завдання до роботи.

6.4.3 Короткі теоретичні відомості.

6.4.4 Текст програми.

6.4.5 Результати роботи програми.

6.4.6 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

6.5 Контрольні запитання

6.5.1 Для чого необхідні структури?

6.5.2 Що таке об'єднання?

6.5.3 Яким чином оголосити структуру в програмі мовою C++?

6.5.4 Яким чином оголосити об'єднання в програмі мовою C++?

6.5.5 Яким чином виконується ініціалізація структури?

6.5.6 Яким чином можна звернутися до елементів структури?

6.5.7 Чи можна передавати структуру в якості аргументу в функцію та повертати з функції значення типу структура?

6.5.8 Що таке бітове поле?

6.5.9 Які дії можна виконувати над бітовими полями?

6.5.10 Чи може структура містити в собі структури або об'єднання?

6.5.11 Яким чином розподіляється пам'ять для структури?

6.5.12 Яким чином розподіляється пам'ять для об'єднання?

6.5.13 Що таке перелічення?

6.5.14 Яким чином оголосити перелічення в програмі мовою C++?

6.5.15 Чим відрізняються об'єднання від структур?

7 ЛАБОРАТОРНА РОБОТА № 7 ПРОГРАМУВАННЯ З ВИКОРИСТАННЯМ АЛГОРИТМІВ СОРТУВАННЯ

7.1 Мета роботи

Отримати знання і навички, необхідні для виконання сортування даних, ознайомитись з основними алгоритмами сортування даних та навчитися використовувати їх на практиці в процесі розроблення програм мовою програмування C++.

7.2 Основні теоретичні відомості

Сортуванням називають впорядковування за ключами елементів деякої структури даних, на якій визначено відношення порядку.

Існує багато алгоритмів сортування, серед яких зокрема можна розглянути сортування вибором, обміном, вставками, Шелла, злиттям, підрахунком.

7.2.1 Сортування вибором

Ідея методу полягає в тому, щоб створювати відсортовану послідовність шляхом приєднання до неї у правильному порядку одного елемента за іншим.

Побудова відсортованої послідовності починається з лівого кінця масиву.

Алгоритм складається з n послідовних кроків, починаючи від нульового і закінчуючи $(n-1)$ -м.

На i -му кроці вибирається найменший з елементів $a[i]...a[n-1]$ і виконується заміна його місцями з $a[i]$.

Незалежно від номера поточного кроку i , в результаті його виконання послідовність $a[0]...a[i]$ є впорядкованою. Таким чином, на $(n-1)$ -му кроці вся послідовність, окрім $a[n]$, виявляється відсортованою, а $a[n]$ стоїть справедливо на останньому місці: всі менші елементи вже розташовані зліва.

7.2.2 Сортування обміном (бульбашкове)

Під час виконання сортування обміном масив представляється розташуванням згори вниз: від нульового елемента до останнього.

Ідея методу: крок сортування полягає в проході знизу вгору масивом. У процесі просування по масиву порівнюються пари сусідніх елементів. Якщо елементи деякої пари знаходяться в неправильному порядку, то необхідно поміняти їх місцями.

Після нульового проходу по масиву "вгорі" виявляється "найлегший елемент".

Наступний прохід робиться до другого згори елемента, таким чином другий за величиною елемент підіймається на правильну позицію.

На кожній ітерації виконуються проходи нижньою частиною масиву, яка з кожним проходом зменшується доти, доки в ній залишиться тільки один елемент. На цьому сортування закінчується, оскільки послідовність впорядкована за зростанням.

7.2.3 Сортування вставками

Розглянемо дії алгоритму на i -му кроці. Послідовність до цього моменту розділена на дві частини: готову $a[0]...a[i]$ і неврегульовану $a[i+1]...a[n]$.

На наступному, $(i+1)$ -у кроці алгоритму беремо $a[i+1]$ і вставляємо на потрібне місце в готову частину масиву.

Пошук відповідного місця для чергового елемента вхідної послідовності здійснюється шляхом послідовних порівнянь з елементом, що стоїть перед ним.

Залежно від результату порівняння елемент або залишається на поточному місці (вставка завершена), або вони міняються місцями і процес повторюється.

Таким чином, у процесі вставки просується елемент x до початку масиву, зупиняючись у разі, коли знайдено елемент, менший x або досягнуто початок послідовності.

7.2.4 Сортування Шелла

Сортування Шелла є модифікацією алгоритму сортування простими вставками.

Розглянемо алгоритм сортування на прикладі масиву $a[0]..a[15]$:

- а) виконати сортування простими вставками кожних 8 груп з 2-х елементів ($a[0]$, $a[8]$), ($a[1]$, $a[9]$) ... ($a[7]$, $a[15]$);
- б) виконати сортування кожної з чотирьох груп по 4 елементи ($a[0]$, $a[4]$, $a[8]$, $a[12]$) ... ($a[3]$, $a[7]$, $a[11]$, $a[15]$);
- в) виконати сортування 2 груп по 8 елементів, починаючи з ($a[0]$, $a[2]$, $a[4]$, $a[6]$, $a[8]$, $a[10]$, $a[12]$, $a[14]$);
- г) виконати сортування вставками всіх 16 елементів.

7.2.5 Сортування злиттям

При сортуванні злиттям на i -му кроці масив a розбивається на впорядковані підмасиви (послідовності елементів масиву, що розташовані поспіль) довжини $Size$.

Пари сусідніх підмасивів зливаються у впорядковані підмасиви довжини $Size * 2$ у допоміжному масиві b .

Після присвоєння a значення b та збільшення вдвічі значення $Size$ злиття повторюється для підмасивів більшого розміру.

Процес завершується, коли $Size$ стає більшим або рівним n . Оскільки при $i = 1$ підмасиви з 1 елемента впорядковані за означенням, у результаті буде отримано впорядкований масив a .

7.2.6 Сортування підрахунком

В сортуванні підрахунком (counting sort) передбачається, що всі n елементів – цілі числа, які лежать в інтервалі від 0 до k , де k – деяка ціла стала.

Основна ідея сортування підрахунком полягає в тому, щоб для кожного вхідного елементу x визначити кількість елементів, які менші за x . За допомогою даної інформації елемент x можна розташувати на тій позиції вихідного масиву, де він повинен знаходитись. Наприклад, якщо всього є 17 елементів, які менші за x , то у вихідній послідовності елемент x повинен займати 18-у позицію.

7.3 Завдання до роботи

7.3.1 Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи дані методичні вказівки, лекції, а також рекомендовану літературу.

7.3.2 Роз'язати за допомогою програми мовою C++ задачі індивідуального завдання, отриманого у викладача.

7.3.3 Оформити звіт з роботи.

7.3.4 Відповісти на контрольні запитання.

7.4 Зміст звіту

7.4.1 Мета роботи.

7.4.2 Завдання до роботи.

7.4.3 Короткі теоретичні відомості.

7.4.4 Текст програм.

7.4.5 Результати роботи програми.

7.4.6 Таблиця порівняння результатів застосування алгоритмів сортування (за швидкістю).

Алгоритм сортування	Ітерація	Масив тип ₁		Масив тип ₂	
		Розмір-ність ₁	Розмір-ність ₂	Розмір-ність ₁	Розмір-ність ₂
Алгоритм сортування ₁	1				
	2				
	3				
	4				
	5				
	Середній результат				
Алгоритм сортування ₂	1				
	2				
	3				
	4				
	5				
	Середній результат				

7.4.7 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

7.5 Контрольні запитання

7.5.1 Що таке сортування даних?

7.5.2 Для яких даних може виконуватись сортування?

7.5.3 Які алгоритми сортування відомі?

7.5.4 Які алгоритми сортування даних вважаються більш ефективними та чому?

7.5.5 У чому полягає алгоритм сортування вибором?

7.5.6 У чому полягає алгоритм сортування вставками?

7.5.7 У чому полягає алгоритм сортування обміном?

7.5.8 У чому полягає алгоритм сортування Шелла?

7.5.9 У чому полягає алгоритм сортування злиттям?

7.5.10 У чому полягає алгоритм сортування підрахунком?

7.5.11 Для чого необхідне сортування випадковим чином?

7.5.12 Які характеристики використовуються для порівняння алгоритмів сортування?

7.5.13 Для чого використовується перевантаження функцій?

7.5.14 Яким чином можна поліпшити ефективність алгоритму сортування обміном?

7.5.15 Яким чином реалізується сортування за декількома параметрами одночасно?

8 ЛАБОРАТОРНА РОБОТА № 8 РОБОТА З ФАЙЛАМИ

8.1 Мета роботи

Отримати знання і навички, необхідні для роботи з даними на основі використання файлів за допомогою засобів мови C++, та навчитися використовувати їх на практиці в процесі розроблення програм мовою програмування C++.

8.2 Основні теоретичні відомості

У програмах, написаних мовою C/C++, всі операції введення/виведення реалізуються за допомогою бібліотечних функцій, що входять до складу конкретної системи програмування. Під час роботи з файлами дані можуть передаватися у бінарному або в текстовому форматі.

Бібліотека C/C++ підтримує два основні способи роботи з файлами:

- форматоване введення/виведення за допомогою функцій (заголовний файл `stdio.h`);
- потокове введення/виведення (заголовний файл `fstream`).

За першого способу роботи з файлами (за допомогою бібліотечних функцій `stdio.h`) застосовується спеціальна керуюча структура, що містить інформацію про файл та надає тимчасовий буфер для зберігання даних. Дана структура має тип `FILE`. Крім тимчасового буферу у керуючій структурі міститься інформація про ідентифікатор файлу, його розташування на диску та покажчик поточної позиції у файлі.

Типовий сценарій роботи з файлами:

- відкриття файлу: вказується ім'я файлу, визначається режим доступу (читання, запис, додавання) та тип файлу (текстовий або двійковий);
- читання або запис даних: після того, як відкриття файлу успішно виконано, з нього можна прочитати або записати в нього дані у визначеному форматі (форматоване введення/виведення);

– закриття файлу: для завершення роботи з файлом його необхідно закрити.

Для реалізації даного сценарію в бібліотеці `stdio.h` призначені наступні функції:

- відкриття файлу – функція `fopen()`;
- форматоване виведення даних – сімейство функцій `printf()` (`fprintf()`);
- форматоване введення даних – сімейство функцій `scanf()` (`fscanf()`);
- закриття файлу – `fclose()`.

Функція `fopen()` має наступний синтаксис:

EXP

```
FILE* fopen (const char * ім'я_файлу, const char * режим);
```

У стандарті мови C/C++ серед інших визначено наступні режими доступу до файлів:

- `r` – доступ тільки для читання (застосовний тільки для існуючого файлу);
- `w` – доступ для запису: якщо файл існує, його вміст очищується, а якщо файл не існує, то в такому випадку він створюється;
- `a` – доступ для додавання нової інформації: якщо файл вже існує, дані додаються в кінець, а якщо файл не існує, то в такому випадку він створюється.

При визначенні режиму доступу для двійкового файлу використовується тип файлу `b`, а для текстового файлу – `t`.

Функція виведення даних у файл `fprintf()` має наступний синтаксис:

EXP

```
int fprintf (FILE * номік, const char * формат, ...);
```

В якості параметра *формат* може використовуватися довільний рядок, що може містити специфікатори формату наступного вигляду:

EXP

%[прапори][ширина][точність][довжина]специфікатор

Поле *специфікатор* визначає тип та формат даних, що буде записано у файл.

Функція `fscanf()` має наступний синтаксис:

EXP

```
int fscanf (FILE * stream, const char * format, ...);
```

Для читання з файлу або запису в файл структури або масиву використовують наступні функції:

EXP

```
size_t fread (void * ptr, size_t size, size_t count, FILE * stream);  
size_t fwrite (const void * ptr, size_t size, size_t count, FILE *  
stream);
```

Дані функції мають наступні параметри:

- ptr – покажчик на масив, що буде прочитано з файлу (або записано в файл);
- size – розмір елементу масиву в байтах;
- count – кількість елементів у масиві;
- stream – покажчик на структуру FILE.

За другого способу роботи з файлами використовуються потоки введення/виведення `ifstream` та `ofstream` відповідно.

У такому випадку для відкриття файлу можна використати наступні способи на основі створення потоку та застосування відповідних методів, представлений у загальному вигляді:

EXP

```
ifstream ( const char * filename, mode);  
ofstream ( const char * filename, mode);  
void open ( const char * filename, mode);  
void open ( const char * filename, mode);
```

При цьому використовуються наступні параметри:

- filename – ім'я файлу;
- mode – режим доступу до файлу:
 - а) app – доступ для додавання нової інформації;
 - б) ate – перемістити вказівник файлу в кінець;
 - в) binary – режим доступу до бінарного файлу;
 - г) in – доступ для читання;
 - д) out – доступ для запису;
 - е) trunc – створити порожній файл для читання та запису.

Приклад відкриття файлів за допомогою потоків:

C++

```
ifstream in1("test.in");  
ofstream out1("test.out");
```

8.3 Завдання до роботи

8.3.1 Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи дані методичні вказівки, лекції, а також рекомендовану літературу.

8.3.2 Роз'язати за допомогою програми мовою C++ задачі індивідуального завдання, отриманого у викладача.

8.3.3 Оформити звіт з роботи.

8.3.4 Відповісти на контрольні запитання.

8.4 Зміст звіту

8.4.1 Мета роботи.

8.4.2 Завдання до роботи.

8.4.3 Короткі теоретичні відомості.

8.4.4 Текст програми.

8.4.5 Результати роботи програми.

8.4.6 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

8.5 Контрольні запитання

8.5.1 Які основні операції роботи з файлами є в мові C++?

8.5.2 Яким чином реалізується робота з файлами в бібліотеці `stdio.h`?

8.5.3 Які існують переваги бібліотеки `stdio.h` над бібліотекою `fstream`?

8.5.4 Яким чином реалізується робота з файлами в бібліотеці `fstream`?

8.5.5 Які існують переваги бібліотеки `fstream` над бібліотекою `stdio.h`?

8.5.6 Які типи файлів розрізняються в програмах мовою C++?

8.5.7 Які існують методи відкриття файлів?

8.5.8 У яких режимах можна відкрити файл у C++?

8.5.9 Яким чином записати дані у файл?

8.5.10 Яким чином закрити файл?

8.5.11 Яким чином отримати доступ у файлі за заданою позицією?

8.5.12 Яким чином задати формат даних, що виводяться у файл?

8.5.13 Яким чином можна прочитати дані з файлу?

8.5.14 Яким чином можна прочитати дані з файлу, визначаючи форматування?

8.5.15 Яким чином записати у файл структуру?

ЛІТЕРАТУРА

1. Прата, С. Язык программирования C++. Лекции и упражнения, 6-е изд. [Текст] : Пер. с англ. / С. Прата. – М. : ООО “И.Д. Вильямс”, 2012. – 1248 с.
2. Страуструп, Б. Программирование : принципы и практика использования C++ [Текст] / Б. Страуструп. – М. : Вильямс, 2011. – 1248 с.
3. Керниган, Б. Язык программирования C [Текст] : Пер. с англ. / Б. Керниган, Д. Ритчи. – М. : Вильямс, 2009. – 304 с.
4. Павловская, Т.А. C/C++. Программирование на языке высокого уровня [Текст] / Т.А. Павловская. – СПб. : Питер, 2003. – 461 с.
5. Павловская, Т.А. Структурное программирование [Текст] : Учеб. Пособие / Т.А. Павловская, Ю.А. Щупак. – СПб. : Питер, 2003. – 240 с.
6. Крячков, А.В. Программирование на C и C++ [Текст] : Учеб. Пособие для вузов / А.В. Крячков, И.В. Сухинина, В.К. Томшин ; пер. ред. В.К. Томшина. – 2-е изд., испр. – М. : Горячая линия – Телеком, 2006. – 344 с.
7. Демидович, Е.М. Основы алгоритмизации и программирования. Язык C [Текст] : Учеб. Пособие / Е.М. Демидович. – СПб. : БХВ – Петербург, 2006. – 440 с.
8. Шилдт, Г. Искусство программирования на C++ [Текст] / Г. Шилдт. – СПб. : БХВ – Петербург, 2006. – 496 с.
9. Шилдт, Г. Справочник программиста по C/C++ [Текст] : Пер с англ. / Г. Шилдт. – 2-е изд. – М. : ВШ, 2000. – 447 с.
10. Дейтел, Х.М. Как программировать на C++ [Текст] : Пер с англ. / Х.М. Дейтел, П.Дж. Дейтел. – М. : Бином, 1999. – 1022 с.
11. Хортон, А. Visual C++ 2010 [Текст] : полный курс / А. Хортон. – М. : Диалектика, 2010. – 1216 с.
12. Страуструп, Б. Язык программирования C++. Специальное издание [Текст] : Пер. с англ. / Б. Страуструп. – М. : Издательство Бином, 2011. – 1136 с.
13. Ишкова, Э.А. C++. Начала программирования [Текст] / Э.А. Ишкова. – 2-е изд. перераб. и доп. – М. : ЗАО «Изд-во Бином»,

2001. – 479 с.

14. Крупник, А. Изучаем С. [Текст] / А. Крупник. – СПб. : Питер, 2001. – 256 с.

15. Культин, М.Б. С/С++ в задачах и примерах [Текст] / М.Б. Культин. – СПб. : БХВ – Петербург, 2005. – 288 с.

Додаток А
Приклад оформлення титульного листа звіту
з лабораторної роботи

Міністерство освіти і науки України
Запорізький національний технічний університет

кафедра програмних засобів

ЗВІТ
з лабораторної роботи № 1
з дисципліни "Алгоритмізація та програмування" на тему:
"ПРОГРАМУВАННЯ НА ОСНОВІ ЛІНІЙНИХ АЛГОРИТМІВ ТА
АЛГОРИТМІВ З РОЗГАЛУЖЕНОЮ СТРУКТУРОЮ"
Варіант № 1

Виконав:
студент групи КНТ-716

А. Б. Іваненко

Прийняв:
к.т.н., доцент

В. М. Льовкін