МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

ПРОГРАМУВАННЯ ТА АЛГОРИТМІЧНІ МОВИ

Методичні вказівки
до виконання лабораторних робіт 1–10
для підготовки здобувачів першого (бакалаврського)
рівня вищої освіти
за спеціальностями 122 «Комп’ютерні науки»,
126 «Інформаційні системи та технології»

Київ 2024

УДК 004.43

П78

Укладачі: О. А. Поплавський, канд. техн. наук, доцент;

І. В. Босенко, аспірант

Рецензент О. О. Терентьєв, д-р техн. наук, професор

Відповідальна за випуск Т. А. Гончаренко, канд. техн. наук, доцент.

Затверджено на засіданні кафедри інформаційних технологій, протокол № 9 від 27 березня 2024 року.

В авторській редакції.

Програмування та алгоритмічні мови : методичні вказівки до

П78 виконання лабораторних робіт / уклад. : Поплавський О. А., Босенко І.В. – Київ : КНУБА, 2024. – 56 с.

Містять зміст, порядок оформлення і вказівки до виконання лабораторних робіт.

Призначено для здобувачів першого (бакалаврського) рівня вищої освіти спеціальностей 122 «Комп’ютерні науки», 126 «Інформаційні системи та технології».

Зміст

Загальні положення 4
Лабораторна робота №1 5
Лабораторна робота №2 19
Лабораторна робота №3 24
Лабораторна робота №4 28
Лабораторна робота №5 31
Лабораторна робота №6 35
Лабораторна робота №7 40
Лабораторна робота №8 44
Лабораторна робота №9 47
Лабораторна робота №10 49
Список літератури 52

Загальні положення

Методичні рекомендації до предмету «Програмування та алгоритмічні мови» мають за мету забезпечити ефективну підготовку студентів до виконання лабораторних робіт, які є невід'ємною частиною даної навчальної дисципліни. Засвоєння та розуміння лекційного матеріалу є передумовою до успішного виконання кожної з лабораторних робіт. Студентам рекомендується приділити особливу увагу загальним положенням, що покладені в основу лабораторних завдань, а також ознайомитися з прикладами програм, що слугують лише як один із варіантів рішення поставленої задачі.

Для виконання лабораторних робіт передбачена єдина конфігурація програмно-апаратних засобів, яка включає персональний комп'ютер та середовище візуальної розробки програм Microsoft Visual Studio.

У процесі виконання лабораторних робіт кожен студент має продемонструвати творчий та індивідуальний підхід до розробки програмних проєктів, вміння грамотно використовувати наявне програмне забезпечення, а також навички програмування на мовах високого рівня C/C++. Важливою є здатність перетворити розроблену програму на готовий програмний продукт, використовувати методи якісного аналізу програми та виконати оцінку отриманих результатів. Не менш значущим є створення зручного інтерфейсу з користувачем та надання пояснень до програми.

Тип завдання до лабораторної роботи визначається відповідно до номера студента в журналі групи. Під час підготовки до виконання лабораторної роботи студенти мають уважно ознайомитися з методичними рекомендаціями, теоретичними відомостями, прикладами програм та формулюванням завдань. На етапі практичної реалізації завдання важливо виправити всі помилки, що виникли під час компіляції початкового тексту програми, а також виконати програму в покроковому режимі для виявлення значення проміжних змінних.

Звіт з лабораторної роботи має включати титульний аркуш, аркуш змісту, опис виконаних завдань з умовою задачі, опис архітектури програми, початковий код програми з коментарями, приклади результатів роботи програми на тестових даних та висновки з урахуванням усіх виконаних завдань. Оформлення звіту має відповідати встановленим вимогам, включаючи форматування сторінок, шрифт, міжрядковий інтервал та нумерацію сторінок.

Методичні рекомендації є ключовим інструментом для забезпечення глибокого розуміння предмету «Програмування та алгоритмічні мови» та успішного виконання лабораторних робіт, що сприяють формуванню професійних компетенцій студентів у галузі програмування.

Лабораторна робота №1.
Робота з IDE Microsoft Visual Studio.
Програмування алгоритмів лінійної структури, компіляція та відлагодження програми

Мета роботи: вивчення основних прийомів роботи з інтегрованим середовищем розробки *Microsoft Visual Studio*. Написання найпростішої програми на мові *C*, отримання навичок компіляції та відлагодження програми.

Короткі теоретичні відомості

Microsoft Visual Studio – лінійка продуктів компанії Майкрософт, що включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів.

Інсталяція IDE Microsoft Visual Studio

Для установки *Microsoft Visual Studio* з належними засобами потрібно близько 8 ГБ дискового простору. Для скачування інсталяційного файлу із *IDE Microsoft Visual Studio* потрібно перейти за посиланням:

https://visualstudio.microsoft.com

У вікні, що відкрилось, потрібно обрати першу опцію *Download Visual Studio* (не *Visual Studio Code*!) (рис. 1):

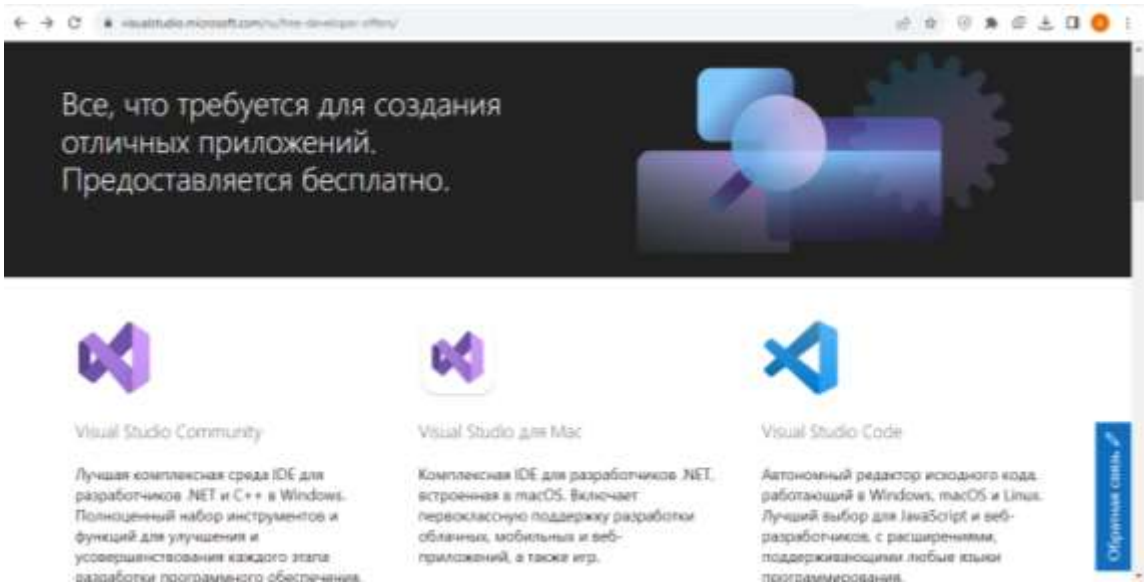

Рис. 1. Вікно вибору

І в меню, що випадає, потрібно обрати *Visual Studio Community* (рік випуску не важливий для наших задач, проте на малюнках показані знімки саме цієї версії).

IDE Microsoft Visual Studio Community – це повнофункціональне та безкоштовне для студентів, ентузіастів інтегроване середовище розробки для створення сучасних додатків для *Android, iOS* та *Windows*, а також веб-додатків та хмарних служб.

Для розробки програм на С при інсталяції *IDE* слід виконати такі налаштування (рис. 2):

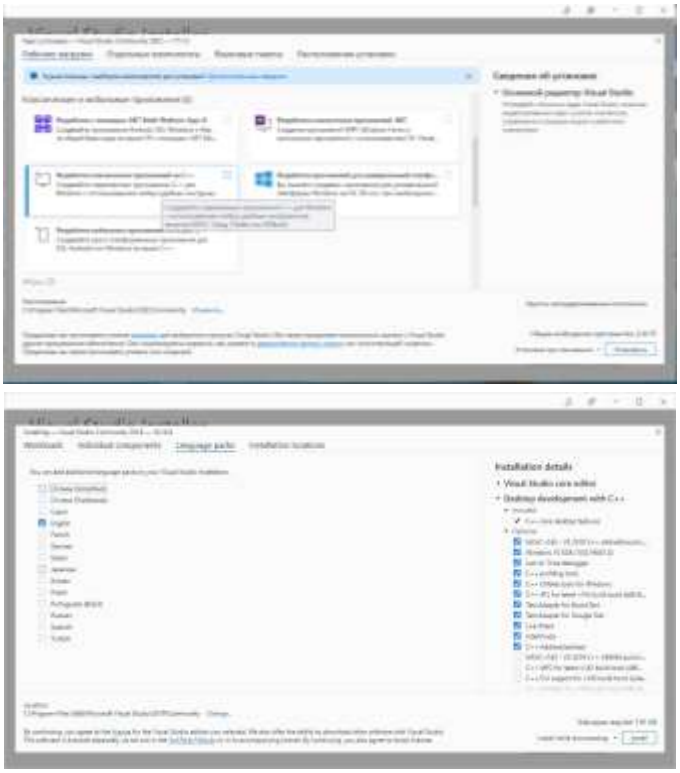

Рис. 2. Налаштування при інсталяції

Після завершення установки цього програмного забезпечення потрібно буде перезавантажити комп'ютер.

Можливості та засоби розробки додатків Microsoft Visual Studio

У сімействі продуктів *Visual Studio* використовується загальне інтегроване середовище розробки, що складається з декількох елементів: панелі інструментів, різних закріплених вікон або вікон, що автоматично приховуються в лівій, нижній або правій областях, а також області редакторів. Набір доступних вікон інструментів, меню і панелей інструментів залежить від типу проєкту або файлу, в якому виконується розробка.

Рішення та проекти як контейнери

У *Visual Studio* реалізовані контейнери: рішення та проєкти, які роблять можливим використання в інтегрованому середовищі розробки (*IDE*) всього діапазону засобів, конструкторів, шаблонів і параметрів.

Також *Visual Studio* надає папки рішень для того, щоб структурувати пов'язані проєкти по групах і потім виконувати дії над цими групами проєктів.

Проєкт – це група файлів і налаштувань, з яких збирається остаточна програма або вихідні файли. Проєкт включає набір файлів вихідних текстів та метаданих, наприклад, посилання на компоненти та інструкції побудови. Як правило, під час побудови проєктів створюється один або кілька файлів із кодом програми. Рішення включає один або декілька проєктів, а також файли і метадані, необхідні для опису рішення в цілому (рис. 3):

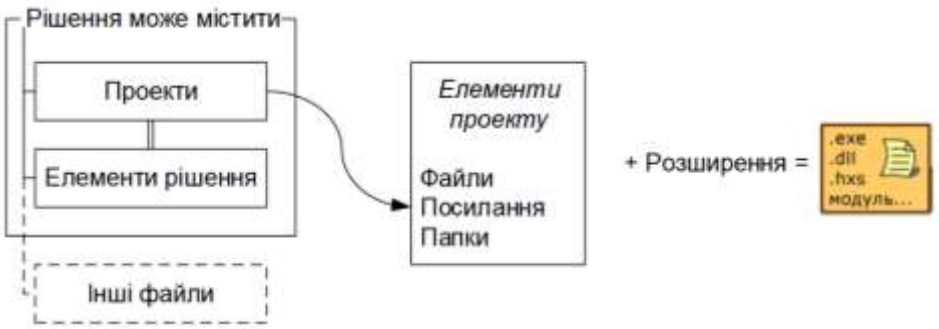

Рис. 3 Схема вмісту рішення

Щоб допомогти користувачам організовувати і виконувати стандартні завдання із застосуванням елементів, що розробляються, проєкти *Visual Studio* використовуються як контейнери в межах рішення. Це дозволяє логічно управляти, виконувати побудову і налагоджувати елементи, що утворюють програму. На виході, як правило, ми отримуємо програму, що виконується (*EXE*), файл бібліотеки динамічного компонування (*DLL*) або модуль.

Проєкт може бути простим або складним, залежно від конкретних вимог. Простий проєкт може містити файли коду програми та файл проєкту. Більш складні проєкти можуть включати ці ж елементи і, крім того, сценарії баз даних, збережені процедури та посилання на існуючі *XML* (веб-служби).

Шаблони проєктів

Visual Studio надає шаблони для проєктів найбільш поширених типів.

Використання проєктів і їхніх шаблонів дозволяє користувачеві зосередитися на реалізації окремої функції, в той час як типовий проєкт обраного типу буде згенерований автоматично.

Файли проєктів

Кожен шаблон створює файл проєкту, в якому містяться метадані поточного проєкту. Файл проєкту містить налаштування проєкту, а також, можливо, список файлів проєкту та їх розташування.

Під час додавання файлу в проєкт інформація про його фізичне місце розташування заноситься в файл проєкту. Під час видалення такого зв'язку, ця інформація видаляється з файлу проєкту. Шаблон проєкту визначає, які команди доступні для кожного його елемента.

Майстер створення програм надає інтерфейс для створення проєкту за шаблоном та створення шаблонів для файлів коду програми. Майстер налаштовує структуру програми, основні меню та панелі інструментів, забезпечує включення деяких файлів із заголовками.

Головна сторінка, відкриття та створення проєктів

Після запуску *Visual Studio* відкривається стартова сторінка, яка дозволяє отримати легкий доступ до наявних проєктів або створити новий проєкт.

Стартова сторінка має приблизно такий вигляд (рис. 4):

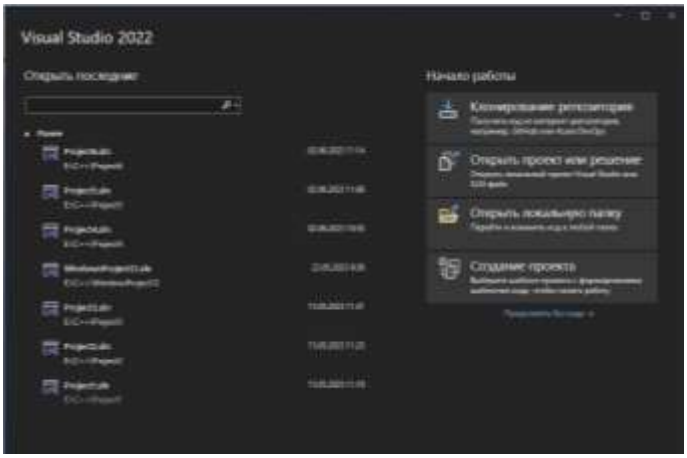

Рис. 4. Стартова сторінка Visual Studio

В області, яка розташована ліворуч, можна обрати та відкрити проєкти, з якими працювали нещодавно (це область – нижче слів *Open recently*). Праворуч, нижче слів *Get started*, розташовані кнопки, які дозволяють відкрити чи створити проєкти та рішення.

Для відкриття вже існуючого проєкту потрібно або натиснути кнопку *Open a project or solution* стартової сторінки, або обрати опції головного меню *File: Open: Project/Solution*…(рис. 5):

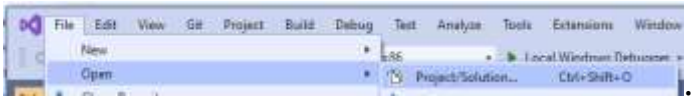

Рис. 5. Вибір існуючого проєкту

Для створення нового проєкту використовується майстер додатків. Для його відкриття потрібно або натиснути кнопку *Create a new project*

стартової сторінки, або обрати опції головного меню *File: New: Project…*(рис. 6):

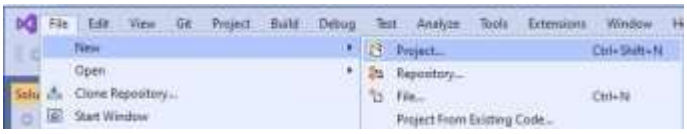

Рис. 6. Створення нового проєкту

В діалоговому вікні можна задати ім’я, місце зберігання і шаблон нового проєкту.

Щоб створити порожній проєкт, використовуйте шаблон *Empty Project* (рис. 7):

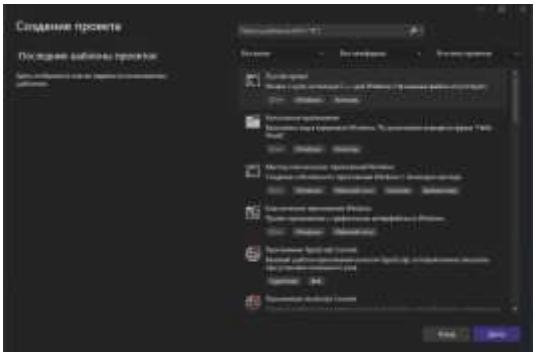

Рис. 7. Створення пустого проєкту

У вікні налаштування параметрів проєкту (*Configure your new project*) потрібно вказати ім’я проєкту (поле *Project name*) та місце його зберігання (поле *Location*). Найчастіше ми будемо створювати рішення, які складаються з одного проєкта, тому для спрощення файлової структури можна вибрати опцію «Place solution and project in the same directory» (рис. 8):

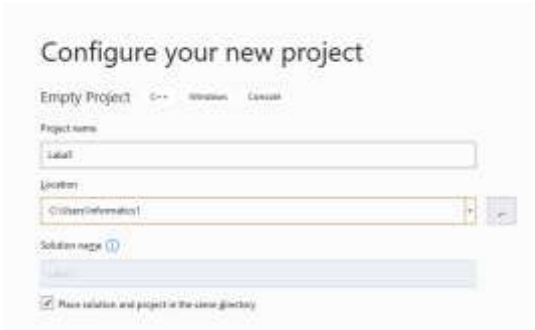

Рис. 8. Налаштування параметрів проєкту

Після цього відкриється робоче вікно *Visual Studio*. В області ліворуч розташований Провідник рішень (Solution Explorer), який відображає

поточне рішення, що може містити один чи більше проєктів. На рис. 9 наведено приклад рішення «*Laba5*» (наведено зеленим). Це рішення містить один проєкт із ім'ям «*Laba5*» (наведено синім). У проєкт можуть входити різні типи файлі (наведено жовтим) – файли з кодом програми (*Source Files*) із розширенням **.c*, файли із заголовками (*Header Files*) із розширенням **.h* та інші. В прикладі на рисунку нижче проєкт ще не містить жодного файлу.

Рис. 9. Приклад рішення «*Laba5*»

Додавання файлів до проєкту

Після створення проєкту до нього можна додавати нові чи вже існуючі файли. Для цього потрібно обрати підкаталог проєкту відповідного типу та, натиснувши правою кнопкою миші, визвати контекстне меню. В цьому контекстному меню потрібно обрати опції *Add: New Item…* для додавання нового файлу чи *Add: Existing Item…* для додавання існуючого файлу. На рис. 10 показано додавання до проєкту «*Laba5*» нового файлу, в якому буде написано код програми:

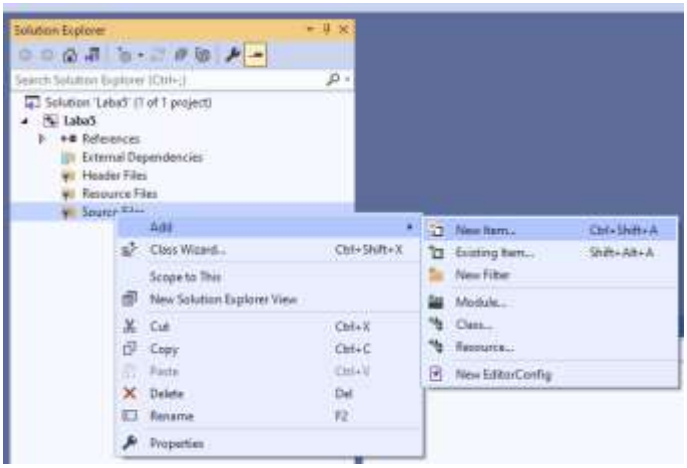

Рис. 10. Додавання нового файлу

Якщо додавати вже існуючий файл, то у провіднику, який відкрився, потрібно просто вказати шлях до потрібного файлу.

Якщо файл потрібно створити (опції *Add: New Item…*), відкривається

вікно *Add New Item*, в якому потрібно вказати тип файлу, його ім’я та місце розташування. За замовчуванням файл буде збережений в поточному каталозі проєкту. На рис. показано додавання файлу, в якому буде написана програма мовою *C++* із ім’ям *Laba5_code.c*. Цей файл буде частиною проєкту *Laba5.*

Після натискання кнопки *Add*, в області ліворуч, над вікном *Output*, у текстовому редакторі відкривається створений файл. Після цього можна писати текст програми (рис. 11):

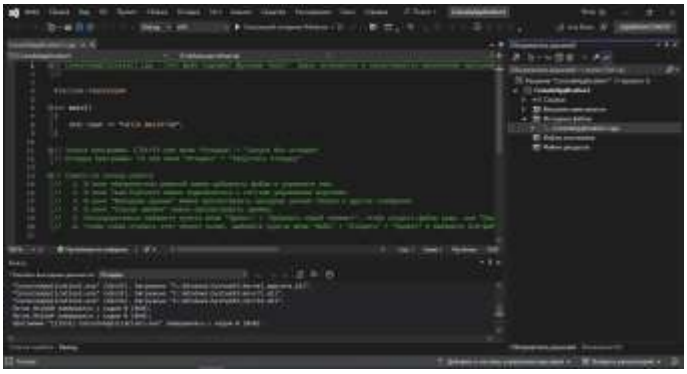

Рис. 11. Вікно для написання коду

Засоби побудови

У середовищі *Visual Studio* передбачений потужний набір засобів побудови та налагодження. За допомогою конфігурацій побудови можна вибирати компоненти для побудови, виключати компоненти, які не потрібно включати в побудову, а також визначати, як будуть побудовані вибрані проєкти і для якої платформи.

Для побудови (збірки) програми потрібно вибрати опції меню *Build*:*Build Solution* (рис. 12):

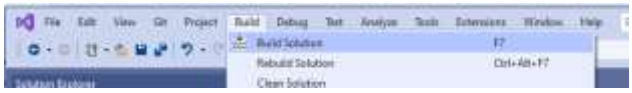

Рис. 12. Збірка програми

У вікні *Output* повинна відображатися інформація про хід збирання, а також про виявлені помилки компіляції. Приклад інформації про результати збирання проєкту можна побачити на рис. 13 (наведено синім):

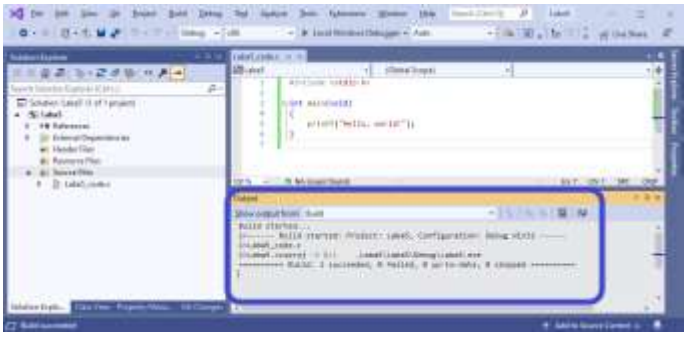

Рис. 13. Вікно *Output*

Якщо під час побудови файлу, що виконується, були виявлені помилки, то, зазвичай, файл, що виконується, побудовано не буде і про це буде повідомлено у вікні *Output* (наведено фіолетовим). Список помилок (наведено червоним) та попереджень (наведено жовтим) із детальним описом (наведено помаранчевим) також можна побачити у вікні *Output* (рис. 14):

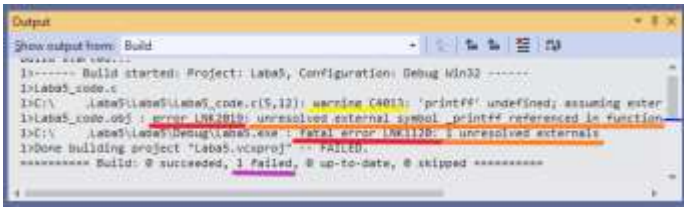

Рис. 14. Помилки під час побудови програми

Якщо вікно *Output* було випадково закрито, то знову відкрити його можна, обравши опції меню *Debug: Windows: Output* (рис. 15):

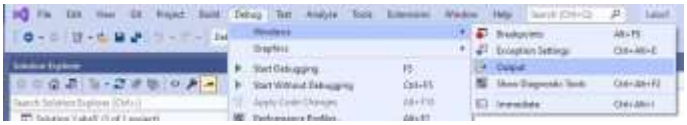

Рис. 15. Відкриття вікна *Output*

Засоби налагодження

Якщо побудова виконуваного файлу пройшла успішно, тобто усі синтаксичні помилки було виправлено, то можна починати процес налагодження.

Процес налагодження, який виконується за допомогою відладчика, дозволяє виявити та усунути такі проблеми як логічні та семантичні помилки, що виявляються вже під час виконання. У режимі зупинки (*Break Point*) можна переглядати локальні змінні та інші дані, використовуючи такі засоби, як Вікна змінних (*Watch*) і Вікно пам’яті (*Memory*).

Для того, щоб зупинити виконання програми до певного рядка, в цьому рядку потрібно встановити *Break Point.* Його можна встановити, клікнувши лівою кнопкою миші на сірій області, яка розташована ліворуч від тексту програми, навпроти обраного рядка (на рис. 16 *Break Point* – це червоний кружечок):

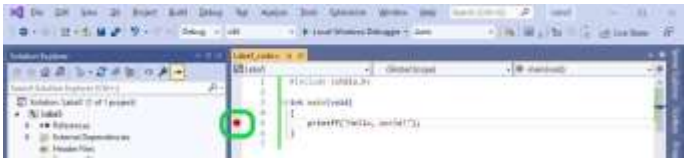

Рис. 16. *Break Point*

Для запуску програми в режимі налагодження потрібно або обрати опції меню *Debug: Start Debugging*, або натиснути гарячу клавішу **F5** (рис. 17):

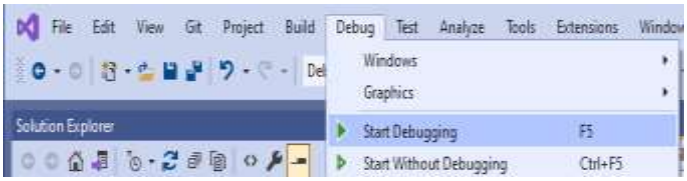

Рис. 17. *Start Debugging*

Якщо заздалегідь були поставлені *Break Point,* то, після запуску програми на виконання в режимі налагодження, програма буде виконана тільки до першого *Break Point*, після чого виконання програми буде призупинено. Про місце зупинки програми інформує жовта стрілка на сірому полі ліворуч від тексту програми (на рис. нижче наведено помаранчевим). Інформацію про значення, які зберігаються у змінних, можна отримати або за допомогою вікна *Autos* (на рис. 18 наведено салатовим), або вікна *Watch* (на рис. нижче наведено зеленим):

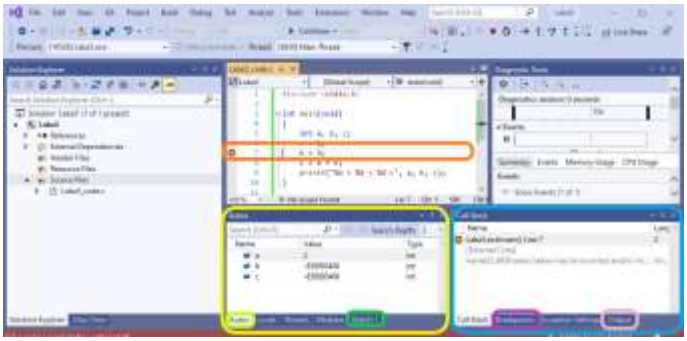

Рис. 18. Вікна *Autos* та *Watch*

До вікна *Autos* автоматично додаються змінні, видимі у межах

поточного блоку, а до вікна *Watch* змінні потрібно додавати вручну. Для цього потрібно або безпосередньо набрати назву змінної в полі *Name* вікна *Watch* (там, де написано підказку *Add item to watch*), або клікнути правою кнопкою миші і в контекстному меню обрати опцію *Add Watch* (рис. 19):

Рис. 19. Функція *Add Watch*

Продовжити виконання програми до наступної *Break Point* або до кінця програми можна, або обравши опції меню *Debug: Continue*, або скориставшись гарячою клавішею *F5* (рис. 20):

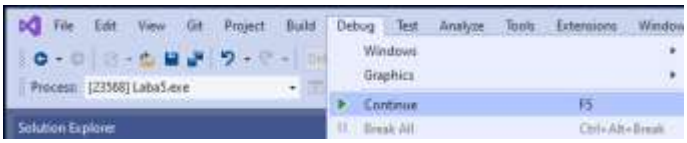

Рис. 20. Функція *Debug: Continue*

Середовище *IDE Visual Studio* також пропонує покрокове виконання програми, коли за один крок буде виконано інструкції певного рядка. Покрокове виконання інструкцій можна виконувати із заходом всередину функцій, що викликаються, або без заходу всередину функцій (рис. 21):

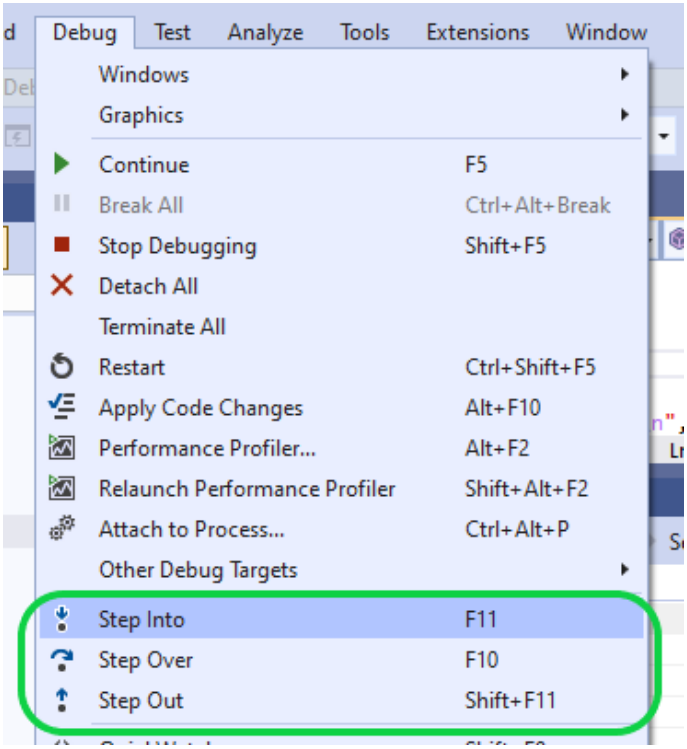

Рис. 21. Покрокове виконання функцій

Якщо обрати опції меню *Debug:Step Over* або натиснути **F10**, то наступний рядок буде виконано без заходу всередину функції.

Якщо обрати опції меню *Debug*:*Step Into* або натиснути **F11,** то наступний рядок буде виконано із заходом всередину функції, яка написана в цьому рядку.

Якщо обрати опцію меню *Debug:Step Out*, то буде виконано вихід із поточної функції в функцію, що її викликала.

Значення змінних у вікні перегляду *Watch* оновлюються у міру виконання програми.

На рис. 22 наведено приклад вмісту змінної *c* до (рис. ліворуч) та після (рис. праворуч) виконання її модифікації (рядок 8):

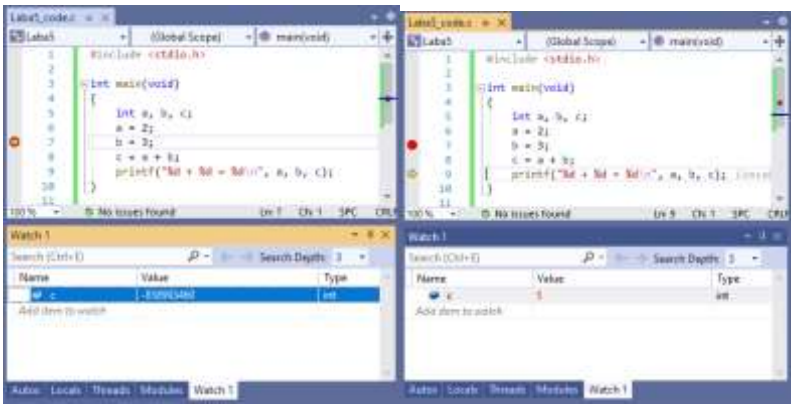

Рис. 22. Вміст змінної *c*

Дострокове завершення роботи програми в налагоджувальному режимі

Для дострокового завершення роботи в програмі, запущеної в налагоджувальному режимі, потрібно обрати опції меню *Debug*:*Stop Debugging* (або натиснути **Shift** + **F5**) (рис. 23) :

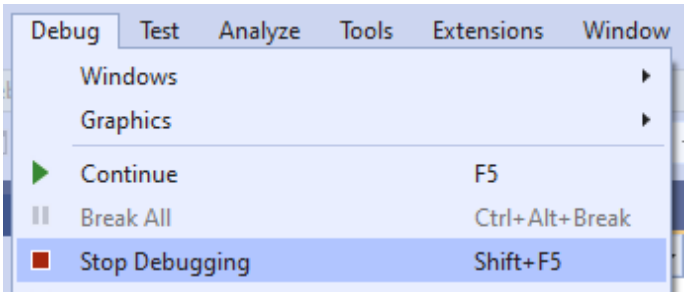

Рис. 23. Функція *Debug*:*Stop Debugging*

Деякі стандартні математичні функції

Стандартні математичні функції оголошено в заголовочному файлі ***math.h***.

abs – абсолютне значення цілого числа – |x|:*int abs(int x);*

fabs – абсолютне значення числа з плаваючою точкою – |x|:*double fabs(double x);*

sqrt – обчислення квадратного кореня: *double sqrt(double x);*

pow – піднесення до ступеня: *double pow(double x, double y);*

cos – косинус – *cos(x)* (тут і далі *x* задається в радіанах): *double cos(double x);*

sin – синус – *sin(x)*:*double sin(double x);*

tan – тангенс – *tg(x)*:*double tan(double x);*

acos – арккосинус – *arccos(x)*: *double cos(double x);*

asin – арксинус – *arcsin(x)*: *double sin(double x);*

atan – арктангенс – *arctg(x)*: *double atan(double x);*

atan2 – арктангенс – *arctg(y/x)*:*double atan2(double y, double x);*

exp – експонента: *double exp(double x);*

log – натуральний логарифм – *ln(x)*: *double log(double x);*

log10 – десятковий логарифм – *log10(x)*:*double log10(double x);*

Для роботи з даними типу *float* більшість перерахованих функцій має еквіваленти, такі як *logf*, *powf* і т.д. (див. довідку).

Деякі математичні константи

Стандартні математичні функції знаходяться в заголовному файлі ***math.h***. Для використання констант **перед** підключенням файлу заголовків ***math.h*** слід додати наступний рядок

#define _USE_MATH_DEFINES

Перелік констант, які визначені в файлі ***math.h*** , наведено в табл. 1.

Таблиця 1

Перелік констант *math.h*

Символ	Вираз	Значення
M_E	e (експонента)	2.71828182845904523536
M_LOG2E	log2(e)	1.44269504088896340736
M_LOG10E	log10(e)	0.434294481903251827651
M_LN2	ln(2)	0.693147180559945309417
M_LN10	ln(10)	2.30258509299404568402
M_PI	Pi	3.14159265358979323846
M_PI_2	pi/2	1.57079632679489661923
M_PI_4	pi/4	0.785398163397448309616
M_1_PI	1/pi	0.318309886183790671538
M_2_PI	2/pi	0.636619772367581343076
M_2_SQRTPI	2/sqrt(pi)	1.12837916709551257390
M_SQRT2	sqrt(2)	1.41421356237309504880
M_SQRT1_2	1/sqrt(2)	0.707106781186547524401

Робоче завдання

1. Скласти алгоритм розв’язання задачі згідно з варіантом. Намалювати блок-схему алгоритму.
2. Створити новий порожній проєкт.
3. Написати програму, що реалізує виконання складеного алгоритму. Вхідні дані (задані символами у кожному варіанті) задати константами, а результат обчислень вивести на екран (використовувати функцію *printf()*).
4. Скомпілювати проєкт.
5. Провести покрокове налагодження:

a. Встановити *Break Point* на певному рядочку всередині програми.
b. Додати до віконця *Watch* частину змінних, які використовуються в програмі.
c. Запустити на виконання програму у режимі налагодження.
d. Подивитися значення змінних за допомогою віконця *Watch*.
e. Виконати декілька рядків коду без заходу всередину функції.
f. Виконати рядок коду, який містить математичну функцію, із заходом в середину функції.
g. Повернутись із математичної функції в функцію *main()* та

переконатися за допомогою віконця *Watch*, що деякі змінні змінили своє значення.

h. Виконати програми до кінця та продемонструвати результат обчислення задачі згідно з варіантом (табл. 2).

Таблиця 2

Варіанти завдань

Варіант	**Завдання**
1.	Обчислити довжину кола, площу круга і об'єм кулі одного і того ж заданого радіусу *r*.
2.	Обчислити периметр прямокутного трикутника за довжинами двох його катетів *a* та *b*.
3.	За координатами трьох вершин ((x_1, y_1), (x_2, y_2) та (x_3, y_3)) трикутника знайти його площу.
4.	Обчислити площу кільця, ширина котрого дорівнює *H*, а відношення радіуса більшого кола до радіусу меншого кола дорівнює *D*.
5.	Обчислити площу прямокутного трикутника за довжинами його катета *a* і гіпотенузи *c*.
6.	Змішали *v1* літрів води з температурою *t1* градусів Цельсія з *v2* літрами води з температурою *t2* градусів Цельсія. Обчислити температуру суміші, що утворилася.
7.	У кубічний, наповнений до країв, акваріум зі стороною *a* метрів опустили кулю, діаметром *b* см. Обчислити, скільки відсотків від початкового об'єму води виллється з акваріума.
8.	Обчислити периметр рівнобедреної трапеції з основами *a* і *b* і висотою *h*.
9.	Обчислити площу рівностороннього трикутника зі стороною *a*.
10.	У кубі зі стороною *a* висвердлили отвір, діаметром *b*. На скільки відсотків зменшилася маса куба?
11.	Обчислити площу прямокутного трикутника за довжинами двох його катетів *a* та *b*.
12.	Обчислити площу рівнобедреної трапеції з основами *a* та *b* і висотою *h*.
13.	Обчислити площу кола, описаного навколо рівностороннього трикутника зі стороною *a*.
14.	За координатами трьох вершин ((x_1, y_1), (x_2, y_2) та (x_3, y_3)) трикутника знайти його периметр.

15.	Обчислити довжину кола, вписаного в рівносторонній трикутник зі стороною *a.*
16.	Є рівносторонній трикутник зі стороною *a*. Знайти, яке співвідношення між площею описаного навколо нього і вписаного в нього кіл.
17.	Обчисліть периметр прямокутного трикутника за довжинами його катета *a* і гіпотенузи *c*.
18.	Є рівносторонній трикутник зі стороною *a*. Знайти співвідношення периметра, описаного навколо нього і вписаного в нього кіл.
19.	Обчислити довжину кола, описаного навколо рівностороннього трикутника зі стороною *a.*
20.	Обчислити площу півкільця, ширина якого дорівнює H, а відношення радіуса більшого кола до радіусу меншої окружності дорівнює *D*.
21.	Обчислити площу кола, вписаного в рівносторонній трикутник зі стороною *a.*

Контрольні запитання

1. Які основні компоненти зазвичай включає в себе сучасне середовище розробки?
2. Які засоби відлагодження в середовищі розробки *Visual Studio* вам відомі?
3. Які додаткові можливості дає запуск програми в режимі налагодження?
4. Які стандартні математичні функції мови C++ вам відомі?

Лабораторна робота №2.
Програма, що розгалужується

Мета роботи: набути практичні навички у розробці програм, що розгалужуються, із використанням операторів *if* та *switch*.

Теми для попереднього опрацювання:

- локальні та глобальні змінні;
- оператори: арифметичні, порівняння, логічні, побітові;
- порядок виконання операторів;
- складені оператори присвоювання;
- умовний оператор *if*;
- оператор вибору *switch*.

Загальні відомості

Алгоритм називається таким, що розгалужується, якщо він містить кілька гілок, які відрізняються одна від одної виконуваними діями. Перехід обчислювального процесу на ту або іншу гілку алгоритму визначається результатом обчислення виразу-умови, яка може мати у своєму складі арифметичні, логічні операції та операції відносин.

Операції відношення:

> – більше;

>= – більше або рівно;

< – менше;

<= – менше або рівно;

== – рівно;

!= – не рівно.

Логічні операції:

&& – логічна І;

|| – логічна АБО.

Оператори керування роботою програми називають керуючими конструкціями програми. До них відносять:

- складені оператори;
- оператори вибору;
- оператори циклів;
- оператори переходу.

Оператор виразу. Будь-який вираз, що закінчується крапкою з комою, розглядається як оператор, виконання якого полягає в обчисленні цього виразу. Окремий випадок – порожній оператор, який позначається як «;».

Приклади:

```
i++; a+=2;
x=a+b;
```

Складені оператори. До складених операторів відносять власно складені оператори і блоки. В обох випадках це послідовність операторів, укладена у фігурні дужки. Блок відрізняється від складеного оператора наявністю визначень у тілі блока. Наприклад:

```
{
    n++; // це складений оператор
    summa+=n;
}
{
```

```
    int n=0;
    n++;  //це блок
    summa+=n;
}
```

Оператори вибору – це умовний оператор і перемикач.

Умовний оператор має повну та скорочену форми.

```
    if ( вираз-умова ) оператор;    //скорочена форма
```

Як вираз-умова можуть використовуватися арифметичний вираз, відношення та логічний вираз. Наприклад:

```
    if (x<y&&x<z) min=x;    //скорочена форма
    if ( вираз-умова ) оператор1;  //повна форма
    else оператор2;
```

Якщо значення виразу-умови відмінні від нуля, то виконується *оператор1*, при нульовому значенні виразу-умови виконується *оператор2*.

Перемикач визначає множинний вибір.

```
switch (вираз)
{
    case константа1 : оператор1 ;
    case константа2 : оператор2 ;
    . . . . . .
    default: оператори;       //може бути відсутнім
}
```

Під час виконання оператора switch обчислюється вираз, записаний після *switch*, він має бути цілим числом. Отримане значення послідовно порівнюється з константами, які записані слідом за *case*. У разі першого же збігу виконуються оператори, позначені даною міткою. Якщо виконані оператори не містять оператора переходу, то далі виконуються оператори всіх наступних варіантів, поки не з’явиться оператор переходу або не закінчиться перемикач. Якщо значення виразу, записаного після *switch*, не збіглося з жодною константою, то виконуються оператори, які ідуть за міткою *default*. Мітка *default* може бути відсутньою.

Оператори переходу виконують безумовну передачу керування.

break — оператор переривання оператора.

Наприклад,

```
    switch (вираз)
{
    case константа1 : оператор1 ; break;
    case константа2 : оператор2 ; break;
```

. . . . . .

default: оператори; //може бути відсутнім

}

У разі, коли отримане значення збіглося з однією із констант, які записані слідом за *case*, виконуються оператори, позначені даною міткою, і оператор *break*, який перериває *case*; керування передається наступному за *case* оператору.

Тернарна операція. В С є умовна операція «? :», яка називається тернарною операцією (тобто тримісна (має три операнди), єдина в С).

Форма запису тернарної операції в С

“умова” ? “вираз 1” : “вираз 2”;

Якщо умова дійсна, то виконується вираз 1, інакше (умова неправильна) виконується вираз 2.

Наприклад:

a > b ? max= a : max = b; // якщо a > b, то виконується max= a,
// інакше виконується max = b

Індивідуальні завдання:

1. Знайти мінімальне значення серед заданих трьох змінних.
2. Для заданих речовинних чисел **a**, **b**, **c** визначити корені квадратного рівняння **a· x2 + b· x + c = 0**, якщо вони є.
3. При заданому x обчислити значення відповідно до формули:

   y={x2-1, x>0 5x2-x+3, x≤0

4. За заданим радіусом **r** та командою (**l**, **s** або **v**) користувача обчислити:

- довжину окружності, якщо команда – **l**;
- площу кола, якщо команда – **s**;
- об’єм кулі, якщо команда – **v**.

5. Визначити, у скільки разів значення дробової частини числа більше за цілу. Організувати перевірку ділення на 0. Наприклад, **x**=123,656 -> **y**=656/123=5,3333
6. Дано три числа **k**, **m**, **n**. Змінити значення змінних таким чином, щоб виконувалась умова **k** < **m** < **n**.
7. Визначити, чи є серед цифр заданого тризначного числа однакові цифри.
8. Визначити позицію цифри з мінімальним значенням у тризначному числі. Нумерація починається з 1. Наприклад, **x**=413 -> **y**=позиція [2]=1.

9. Студенти на іспиті отримують оцінки в системі ЄКТС (літери **A**, **B**, **C**, **D**, **F**). За заданою оцінкою визначити її еквівалент у національній формі (цифровій).
10. Визначити, чи існує трикутник із заданими кутами **a**, **b**, **c**.
11. Знайти максимальне значення серед заданих трьох змінних.
12. Визначити, чи існує ромб із заданими кутами **a**, **b**, **c**, **d**.
13. Визначити, чи існує чотирикутник із заданими кутами **a**, **b**, **c**, **d**.
14. Дано радіус сфери **r** та значення **k**. Використовуючи конструкцію *switch-case*, обчислити значення **y** згідно з умовами:
    а) **k**=1 **y** -> визначити периметр поперечного перерізу;
    б) **k**=2 **y** -> визначити площу поперечного перерізу;
    в) **k**=3 **y** -> визначити об'єм сфери;
    г) інакше -> **y** = -1.
15. Визначити, у скільки разів величина цілої частини числа більше за дробову. Організувати перевірку ділення на 0. Наприклад, **x**=123,656 -> **y** = 656/123 = 0,1875
16. Визначити позицію цифри з максимальним значенням у тризначному числі. Нумерація починається з 1. Наприклад, **x**=453 -> **y**=позиція [5]=2.
17. Визначити, чи існує трикутник із заданими сторонами **a**, **b**, **c**.
18. Визначити, чи існує ромб із заданими кутами **a**, **b**, **c**, **d**.
19. Визначити століття за заданим роком. Наприклад,
    x = 2010 -> **y** = 21; **x** = 2201 -> **y** = 23

Додаткові умови виконання завдання:

- текст програми повинен мати коментарі до коду.

Контрольні запитання

1. Як працює умовний оператор *if*?
2. Який вираз називається складеним логічним? Наведіть приклади.
3. Який оператор називають оператором множинного вибору? Наведіть приклад.
4. Як працює оператор *switch*?
5. Як працює тернарний оператор? Наведіть приклад.
6. Коли умовний оператор називається вкладеним?
7. Навіщо в операторі *switch* використовується оператор *break*?
8. Чи можуть бути вкладеними оператори *switch*?
9. Чи можна замість оператора *if* використовувати тернарний оператор і навпаки, – замість тернарного – оператор *if*?

Лабораторна робота №3.
Циклічні структури

Тема роботи: програмування алгоритмів циклічної структури. Документування коду.

Мета роботи: набути практичні навички із розроблення циклічних програм, використання операторів *for, while, do-while*.

Теми для попереднього опрацювання:

- оператор *sizeof*;
- оператори циклу *for, while, do-while*;
- оператори переходу *break, continue*.

Загальні відомості

Циклічний алгоритм – це алгоритм, який містить такі фрагменти (послідовність операторів), які багаторазово виконуються за різних значень проміжних даних. Група дій, що повторюються в циклі, називається його тілом. Однократне виконання циклу називається його кроком.

Основою таких алгоритмів є оператори циклу. Число повторень цих операторів може бути задане в явній формі (цикл із відомим заздалегідь числом повторень) або в неявній формі (цикл із невідомим заздалегідь числом повторень).

Оператори циклів. Розрізняють:

- ітераційні цикли;
- арифметичні цикли.

В ітераційних циклах умова виконання циклу відома.

1. Цикл з передумовою:

```
while (вираз-умова)
{
    оператор;
}
```

Як <вираз-умова> найчастіше використовується відношення або логічний вираз. Якщо результат виразу відмінний від 0 (*true*), тіло циклу виконується; якщо вираз-умова стане *false* – цикл закінчується.

Наприклад,

```
int a=0, sum = 0;
while (a!= 10)
{
    s+=a++;
```

}

2. Цикл із післяумовою:

do
{
оператор;
}
while (вираз-умова);

Тіло циклу виконується, поки вираз-умова дійсна (*true*).

Наприклад,

```
int a=0, sum = 0;
 do
{
    s+=a++;
}
while (a <= 10);
```

3. Цикл з параметром:

```
for ( вираз_1; вираз-умова; вираз_3)
{
    оператор;
}
```

Вираз_1 та вираз_3 можуть складатися з декількох виразів, розподілених комами. Вираз_1 задає початкові умови для циклу (ініціалізація). Вираз-умова визначає умову виконання циклу: якщо вона відмінна від 0, цикл виконується, а потім обчислюється значення виразу_3. Вираз_3 задає зміну параметра циклу або інших змінних (корекція). Цикл триває доти, доки вираз-умова не стане дорівнювати 0.

```
for ( int a=0, sum = 0; a <= 10 ; a++) s+=a;
```

Будь-який вираз може бути відсутнім, але, поділяючи їх, « ; » повинні залишатися завжди.

Наприклад:

```
int sum = 0; int a = 0;
for ( ; a <= 10 ; a++) s+=a;
```

Оператори переходу. Виконують безумовну передачу керування:

break – оператор, що перериває виконання операторів *while, do-while, for* та *switch*. Управління передається наступному оператору за перерваним. Якщо *break* перериває виконання вкладеного циклу, то управління передається циклу, який охоплює перерваний. Використовується, якщо умову продовження ітерацій треба перевіряти в середині циклу;

continue – перехід до наступної ітерації циклу *while, do-while for*.

У циклах *while, do-while* наступна ітерація починається з обчислення умовного виразу, у циклі *for* – з обчислення виразу для зміни параметра циклу, а потім – умовного виразу.

Текст програми необхідно супроводжувати *doxygen* коментарями.

Основне завдання

Реалізувати програму відповідно до індивідуального завдання за допомогою трьох типів циклів: *for, while-do, do-while* (отримати три однакових результати).

Індивідуальні завдання:

1. Визначити значення 10***n!*** (10 * факторіал числа **n**).
2. Для заданого цілого числа визначити подвійний факторіал. Наприклад, 6!! = 2 * 4* 6 -> 48, 7!! = 1 * 3 * 5 * 7 -> 105.
3. У заданому цілому числі визначити кількість розрядів та суму його цифр. Наприклад, число 123456 має 6 розрядів, сума його цифр – 21.
4. Визначити найбільший спільний дільник для двох заданих чисел.
5. Визначити зворотне число для заданого цілого числа. Кількість розрядів від самого початку не визначено. Наприклад, зворотне число для 12334 –> 43321.
6. Визначити, чи є задане ціле число простим.
7. Для заданого парного числа визначити подвійний факторіал. Наприклад, 6!! = 2 * 4* 6 –> 48.
8. Визначити, чи є ціле 6-значне число «щасливим квитком» («щасливий квиток» – квиток, в якому сума першої половини чисел номера дорівнює сумі другої половини. Наприклад, 102300 –> 1+0+2=3+0+0.
9. Визначити кількість «щасливих квитків», номери яких складаються з 4 цифр.
10. Визначити суму чисел, які кратні 3 та 5 одночасно, із заданого діапазону. Діапазон задається двома змінними (початок діапазону, кінець діапазону включно).
11. Визначити кількість цифр у заданому цілому числі.
12. У заданому діапазоні цілих чисел визначити останнє число, що ділиться на 7 без залишку.
13. Визначити кількість парних (2, 4, 6, 8….) чисел у заданому діапазоні.
14. Обчислити загальну суму вкладу через **N** років із заданою початковою сумою вкладу та заданою процентною ставкою.
15. Знайти добуток всіх чисел із заданого діапазону, що є парними та діляться на 3 без залишку.

16. Знайти найближче просте число, що більше заданого.
17. Для заданого непарного числа визначити подвійний факторіал. Наприклад, 7!! = 1 * 3 * 5 * 7 –> 105.
18. Знайти найближче просте число, що менше за задане.
19. Знайти найбільше 4-значне число, сума цифр якого дорівнює заданому числу.
20. Якою мінімальною кількістю купюр можна набрати потрібну суму S за умови, що в наявності є купюри номіналом 1, 2, 5 та 10 грн ?
21. Число, яке дорівнює сумі своїх дільників, називається досконалим числом. Наприклад, 6 = 1+2+3. Визначити, чи є задане число досконалим.

Додаткові умови виконання завдання:

- звіт має бути виконаний згідно з вимогами до оформлення лабораторних робіт.

Контрольні запитання.

1. Як записується і як працює оператор *for*?
2. У чому відмінність оператора *while* від оператора *do while*?
3. Як програмуються циклічні алгоритми з явно заданою кількістю повторень циклу?
4. Як програмуються циклічні алгоритми із заздалегідь невідомим числом повторень циклу?
5. Напишіть оператор циклу, який не виконується жодного разу.
6. Напишіть оператор циклу, який виконується необмежену кількість разів.
7. Замініть фрагмент програми з оператором *for* рівнозначним фрагментом програми з оператором *while*.
8. Замініть фрагмент програми з оператором *for* рівнозначним фрагментом програми з оператором *do while*.
9. Як можна перервати виконання оператора циклу?
10. Яке призначення операторів *break* і *continue*?

Лабораторна робота №4.
Одновимірні масиви

Тема роботи: одновимірні масиви. Рядки типу *char[]*.

Мета роботи: набути практичні навички з розроблення програм із використанням статичних масивів.

Теми для попереднього опрацювання:

- одновимірні масиви;
- ініціалізація масиву;
- рядки.

Загальні відомості

Масиви – важлива річ у програмуванні; це структура даних, подана у вигляді групи комірок одного типу, об'єднаних одним іменем. Масиви використовуються для обробки великої кількості однотипних даних. Окрема комірка даних масиву називається елементом. Елементами масиву можуть бути дані будь-якого типу.

Масиви можуть мати один або більше одного вимірів. Залежно від кількості вимірів масиви поділяються на одновимірні, двовимірні, тривимірні і так далі до **n**-вимірного масиву.

Масив – це сховище даних, контейнер. Обробляються його елементи кожний окремо. Кожний елемент масиву ідентифікується номером. Компілятор сам нумерує елементи масиву; нумерація в С починається з нуля. Для доступу до елементу масиву використовується ім'я змінної-масиву та його індекс.

Масиви відносять до фундаментальних типів даних, це структуровані (складені) типи. Є два способи роботи з масивами:

1) виділення пам'яті під заздалегідь відоме число елементів;
2) виділення пам'яті в ході роботи програми під кількість елементів, що задається. Цей спосіб вимагає ручного керування пам'яттю.

Бувають задачі, коли заздалегідь відома кількість елементів, що будуть зберігатися в масиві, і ця кількість елементів не змінюється за весь час роботи програми. Кількість елементів таким масивам задається безпосередньо у вихідному коді програми. Змінити кількість елементів масиву у ході роботи програми неможливо. Такі масиви називаються статично-створюваними масивами, або статичними.

Одновимірний масив – масив з одним параметром, що характеризує

кількість елементів у ньому. Фактично одновимірний масив – це масив, у якому може бути тільки один рядок, і n кількість стовпців. Стовпці в одновимірному масиві – це елементи масиву. Приклад оголошення масиву:

int a[16];

Номери елементів будуть мати значення в інтервалі від 0 до 15 включно. Завжди відразу після імені масиву йдуть квадратні дужки, у яких задається розмір одновимірного масиву, цим масив і відрізняється від усіх інших змінних.

Масиви при оголошенні можуть бути проініціалізовані, наприклад,

int a[16] = { 5,-12,-12,9,10,0,-9,-12,-1,23,65,64,11,43,39,-15};

Ініціалізація одновимірного масиву виконується у фігурних дужках після знаку присвоювання; кожний елемент масиву відділяється від попереднього комою.

int a[]={5,-12,-12,9,10,0,-9,-12,-1,23,65,64,11,43,39,-15};

// ініціалізація масиву без визначення його розміру.

У цьому випадку компілятор сам визначить розмір одновимірного масиву. Розмір масиву можна не вказувати тільки при його ініціалізації, при звичайному оголошенні масиву обов'язково потрібно вказувати розмір масиву.

Масиви використовують також для роботи з текстовими даними.

У мові програмування С для роботи з текстом існує два типи рядків:

- масив символів;
- спеціальний клас *string*.

Масив символів закінчується нульовим символом ('\0'). У символьний масив можна ввести відразу весь рядок, використовуючи оператор вводу.

Символьна константа – це один символ, укладений в апострофи.

Рядкова константа – це послідовність символів, укладена у подвійні лапки.

Індивідуальні завдання:

1. Генерація нік-нейму (*nick name*): в заданому рядку замінити всі символи a/A на @, o/O на 0, i/I на 1, s/S на $.
2. Відсортувати масив цілих чисел за зростанням методом «бульбашки».
3. Відсортувати масив цілих чисел за зменшенням методом «бульбашки».
4. Замінити всі числа в масиві цілих чисел на значення максимального елементу масиву.
5. Замінити всі числа в масиві цілих чисел на значення мінімального елементу цього масиву.

6. Знайти відношення максимального елементу масиву до мінімального. Врахувати, що ділити на 0 не можна.
7. Знайти середнє значення елементів масиву цілих чисел.
8. У масиві цілих чисел визначити суму двозначних чисел.
9. У масиві цілих чисел визначити кількість елементів, що більше попереднього елементу та більше наступного.
10. У масиві цілих чисел визначити кількість елементів, що менше попереднього елементу, але більше наступного.
11. У масиві цілих чисел визначити кількість елементів, що менше попереднього елементу та менше наступного.
12. Відсортувати масив символів за абеткою.
13. Заповнити масив цілих чисел заданого розміру числами Фібоначчі.
14. Визначити кількість «щасливих квитків» з 4-розрядними числами (до 9999) та записати їх у масив.
15. Поміняти місцями першу та другу половини заданого рядка. Наприклад, «Іванов» ==> «новІва».
16. Заповнити масив із заданої кількості елементів простими числами, що не повторюються.
17. Знайти перший елемент у масиві, сума цифр якого найбільша в масиві.
18. Визначити кількість голосних букв у заданому слові / реченні.
19. Визначити кількість приголосних букв у заданому слові / реченні.
20. Перетворити число на рядок. Наприклад, 123 – «Сто двадцять три», 4321 – «Чотири тисячі триста двадцять один»).
21. У заданому тексті знайти кількість розповідних, питальних та окличних речень.
22. У заданому тексті знайти кількість слів за умови, що між словами може бути будь-яка кількість пропусків.
23. У заданому тексті знайти слово з найбільшою кількістю букв.

Додаткові умови виконання завдання:

- обов’язково використовувати анотації *@author* и *@date*;
- звіт має бути виконаний згідно з вимогами до оформлення лабораторних робіт.

Контрольні запитання.

1. Як рядки представляються в мові програмування С++?
2. Яке призначення індексів при оголошенні масиву рядків, що завершуються нульовим байтом (С–рядок)?
3. Що таке «мірність масиву»?

4. Від чого залежить об’єм пам’яті, що необхідний для зберігання елементів масиву?
5. Як можна звернутися до елементу масиву?
6. Як оголошуються масиви?
7. Які операції можуть бути застосовані до покажчиків?

Лабораторна робота №5.
Функції

Тема роботи: функції. Варіативні функції.

Мета роботи: набути практичні навички щодо розроблення програм із використанням функцій.

Загальні відомості

Функція – це синтаксично виділений іменований програмний модуль, що виконує певну дію або групу дій. Кожна функція має свій інтерфейс і реалізацію (визначення).

Інтерфейс функції – заголовок функції, у якому вказується назва функції, список її параметрів і тип значення, що повертається.

Прототип функції – це оголошення функції, але не її визначення.

Визначення (опис, реалізація) функції – це програмний код, який реалізує розроблений для даної функції алгоритм.

Не можна визначати будь-яку функцію в тілі іншої функції. У мові С є два типи функцій:

- які не повертають значень;
- що повертають значення.

Функції, що не повертають значення після завершення роботи, мають таке визначення:

void ім’я функції **(параметри функції)** // заголовок функції

```
{
    // тіло функції
}
```

Наприклад:

```
void printArr(*int, int);
```

Якщо потрібно функції передавати якісь дані, то усередині круглих дужок оголошуються параметри функції, які відділяються один від одного комами.

Функції, що повертають значення після завершення своєї роботи, визначають у такий спосіб:

тип даних, що повертаються ім’я функції **(параметри функції)**

{

// тіло функції

return значення, що повертається;

}

Наприклад:

int max (int, int);

float inputMassa();

return – оператор, який закінчує виконання функції. Він повертає виконання у ту функцію, з якої був виклик; управління передається наступному оператору за оператором виклику. Формат оператора такий;

return [<вираз>];

У процесі виконання *return* обчислюється значення виразу, якщо він є, приводиться до типу, який оголошений у функції, і повертається у ту функцію, з якої був виклик. У разі коли вираз відсутній, значення, що повертається функцією, не визначено.

У С існує можливість помістити оголошення функцій в окремий файл; тоді такий файл із функціями необхідно буде підключати, як у випадку з підключенням стандартних заголовних файлів. Є два способи розміщення функцій:

- створення файлу типу *.c, у якому оголошуються та визначаються функції;
- створення файлів типу *.c (для визначення) і *.h (для оголошення функцій).

Переважним стилем програмування вважається другий спосіб.

Функції дозволяють зробити програму модульною, тобто розділити її на кілька функцій, які в сукупності виконують поставлене завдання. Ще один величезний плюс функцій у тому, що їх можна багаторазово використовувати.

Функція виконується в момент її виклику. Після оголошення до функції можна звертатися у програмі по імені. Якщо функція не повертає жодного результату, тобто оголошена як *void*, її виклик не може бути використаний як операнд більш складного виразу (наприклад, значення такої функції не можна привласнити будь-якій змінній). Виклик функції можна використовувати як складову частину більш складного виразу, якщо вона повертає результат.

Прототип вказують у тих випадках, коли функція описується пізніше свого використання. Наприклад, можна оголосити функцію до *main,*

викликати її з *main*, але описати тільки після *main*.

Формальні й фактичні параметри. Формальні параметри існують у прототипі і тілі визначення функції. Вони задаються деякими унікальними іменами і усередині функції доступні як локальні змінні.

Фактичні параметри існують в основній програмі. Вони вказуються під час виклику функції на місці формальних. У момент виклику функції значення фактичних параметрів привласнюються формальним.

Черговість виклику та рекурсія. Одна функція викликається всередині іншої. Зокрема, усередині свого тіла функція може викликати саму себе. Таке явище називається рекурсією, а така функція – рекурсивною.

Способи передачі параметрів у функцію.

1) Передача параметрів за значенням. Під час виклику функції значення фактичного параметра копіюється в локальну змінну, доступну як формальний параметр усередині функції.

   Передача параметрів за значенням має такі обмеження:

- з тіла функції не можна звернутися до будь-якого об’єкта, якщо він не є глобальним або якщо його ім’я перекрите однойменною локальною змінною;
- під час передачі об’єктів виконується їх копіювання, як наслідок, у випадку великих об’єктів, витрачається багато пам’яті.

2) Передача параметрів за посиланням. У цих випадках у функцію передається адреса об’єкта і, відповідно, робота усередині функції відбувається не з копією, а з оригіналом об’єкта.

Щоб параметр передавався за посиланням, досить у прототипі функції поставити знак & після типу параметра.

Наприклад, є функція:

```
void func1(int val, int& ref)
{
        val++;
        ref++;
}
```

В іншій функції є таке:

```
int a = 10, b = 10;
func1(a,b);
```

// **a = 10**, значення буде збільшене, але усередині функції, як локальне

// **b = 11**, буде збільшене значення зовнішньої змінної **b**

При цьому, навіть якщо імена формального і фактичного параметрів

будуть однакові, жодної проблеми не виникне.

Варіативні функції – це функції зі змінною кількістю аргументів. У оголошенні та визначенні такої функції змінне число аргументів задається трьома крапками, обов'язково наприкінці списку формальних параметрів.

При цьому, для коректної роботи функції рекомендується, щоб перший параметр задавав кількість фактично переданих аргументів та/або їх типи (наприклад, як у функції *printf()*).

Для реалізації функцій зі змінною кількістю аргументів у мові програмування С потрібно підключити заголовний файл *stdarg.h*., для С++ – *cstdarg*.

Основне завдання

Створити функцію, яка виконує обчислення відповідно до індивідуального завдання. Продемонструвати роботу функції на декількох наборах вхідних даних (2..5).

Індивідуальні завдання:

1. Визначити факторіал заданого числа.
2. Підрахувати суму чисел у заданому діапазоні. Наприклад, при вхідних даних 50 та 52 повинно бути 50+51+52 = 153. Обробити ситуацію, коли нижня границя більша, ніж верхня.
3. Підрахувати добуток чисел у заданому діапазоні. Наприклад, при вхідних даних 50 та 52 повинно бути 50*51*52 = 132600. Обробити ситуацію, коли нижня границя більша, ніж верхня.
4. Підвести число n у ступінь m.
5. Визначити суму розрядів заданого числа.
6. Визначити, чи є задане число простим.
7. Знайти значення максимального числа з заданих, використовуючи функцію з варіативною кількістю аргументів.
8. Знайти значення мінімального числа з заданих, використовуючи функцію з варіативною кількістю аргументів.
9. Визначити, чи є задане число досконалим (таким, що дорівнює сумі своїх дільників). Наприклад, 6 = 1+2+3 –> досконале число.
10. Визначити, скільки разів зустрічається задана цифра в заданому числі. Наприклад, в числі 1234231 цифра 3 зустрічається 2 рази, цифра 4 – 1 раз, цифра 5 – 0 разів. Обробити ситуацію, коли введений символ не є цифрою (не знаходиться в діапазоні 0...9).
11. Визначити об'єм прибутку під час вкладу заданої суми грошей x у банк під встановлений процент річних y за n років.
12. Отримати заголовний еквівалент переданого символу, якщо символ –

риса літери. Наприклад, ‘a’ -> ‘A’, ‘X’ -> ‘X’.

13. Отримати рядковий еквівалент переданого символу, якщо цей символ – заголовна буква. Наприклад, ‘A’ -> ‘a’, ‘x’ -> ‘X’.
14. Отримати число, що є дзеркальним відображенням заданого. Наприклад, при вхідному числі 1234 має повернутися 4321, а для числа 12305-> 50321.
15. Визначити, до якого століття належить заданий рік.
16. Знайти значення n-го елементу арифметичної прогресії, при заданих значеннях начального значення та шагу прогресії.
17. Знайти значення n-го елементу геометричної прогресії, при заданих значеннях начального значення та шагу прогресії.
18. Визначити подвійний факторіал заданого числа.
19. Визначити, чи є задане число довершеним. (Довершене число таке, що дорівнює сумі своїх дільників).

Контрольні запитання.

1. Чому під яас передачі параметра за значенням усі зміни параметра у функції не відбиваються на значенні аргументу?
2. Скільки операторів *return* може бути в тілі функції?
3. Що таке «прототип функції», яке його призначення?
4. Як визначити список параметрів функції змінної довжини? Наведіть приклад.
5. Як описати функцію, яка не має значень, що повертаються?
6. Як описати, що функція не має аргументів?
7. Наведіть приклад функції, яка не має аргументів та нічого не повертає.
8. Які функції називаються варіативними?
9. Як описати функцію, яка має параметри за замовчуванням?
10. Скільки параметрів за замовчуванням може мати функція?

Лабораторна робота №6.
Багатовимірні масиви

Тема роботи: функції. Робота з багатовимірними масивами. Використання функцій стандартної бібліотеки. Функція *rand()*.

Мета роботи: набути практичні навички щодо розроблення програм із використанням функцій користувача, функції генерації псевдовипадкових чисел, а також передачі масиву як аргументу функції.

Теми для попереднього опрацювання:

- функції, інтерфейс функції;
- передача параметрів;
- одно- та двовимірні масиви.

Загальні відомості

Мова програмування С дозволяє створювати багатовимірні масиви. Найпростішим видом багатовимірного масиву є двовимірний масив.

Двовимірний масив – це масив одновимірних масивів, що декларується так:

тип ім'я_масиву[розмір другого виміру][розмір першого виміру];

У перших квадратних дужках вказується кількість рядків двовимірного масиву, у других – кількість.

Наприклад, для оголошення двовимірного масиву цілих чисел із 10 рядків по 20 елементів у кожному рядку треба записати таке:

int d[10][20];

Для доступу до елементу масиву **d** з індексами **3**, **5** необхідно використовувати такий запис: **d[3][5]**

Під час оголошення двовимірного масиву можна виконувати його ініціалізацію. Наприклад,

```
int a[5][3] = { {4, 7, 8}, {9, 66, -1}, {5, -5, 0},
                {3, -3, 30}, {1, 1, 1} };
```

Після знаку привласнення ставляться загальні фігурні дужки, всередині яких ставиться стільки пар фігурних дужок, скільки має бути рядків у двовимірному масиві, причому ці дужки розділяються комами. У кожній парі фігурних дужок записуються через кому елементи двовимірного масиву. У всіх фігурних дужках кількість елементів має збігатися.

Перетворення одновимірного масиву на двовимірний передбачає наступне. Наприклад, задано одновимірний масив:

[1 2 3 5 6 7 8 9 0]

Необхідно перетворити його у такий масив:

$$\begin{vmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 8 & 9 & 0 \end{vmatrix}$$

Часто на етапі налагодження програм у масиви записують випадкові числа. Випадкові числа в С можуть бути генеровані функцією *rand()* із стандартної бібліотеки <*cstdlib*>. Функція генерує числа в діапазоні від 0 до *RAND_MAX* (константа, яка визначена в бібліотеці <*cstdlib*>).

Для отримання різної послідовності випадкових чисел під час кожного наступного запуску програми треба використовувати функцію *srand( )*.

srand(time(0)); // автоматична рандомізація

Основне завдання:

- розробити функцію, яка на вході приймає одновимірний масив. Одновимірний масив розміром *M*N* елементів перетворити на двовимірний масив, що має *M* рядків і *N* стовпців. Отриманий двовимірний масив обробити відповідно до індивідуального завдання. Далі двовимірний масив перетворити на одновимірний і повернути з функції як результат її роботи;
- одновимірний масив заповнити псевдовипадковими числами (функція *rand ()*) у діапазоні від *5 * K* до *10 * K*, де *K* – номер варіанта.

Додаткове завдання:

- виконати основне завдання;
- додатково розробити функцію, яка виконує обробку переданого одновимірного масиву розміром *M*N* елементів, подаючи його «на льоту» двовимірним (доступ до *[i][j]* елементу масиву) відповідно до варіанта.

Індивідуальні завдання:

1. Дано двовимірний масив з *N*N* цілих чисел. Транспонувати його.
2. Дано двовимірний масив з *N*N* цілих чисел. Помножити кожен елемент рядка матриці на значення елементу головної діагоналі відповідного рядка. Повернути кількість рядків матриці.
3. Дано двовимірний масив з *N*N* речовинних чисел. Помножити кожен елемент рядка матриці на значення елементу бічної діагоналі відповідного рядка. Повернути кількість рядків матриці.
4. Дано двовимірний масив з *N*N* речовинних чисел. Помножити кожен елемент рядка матриці на середнє значення елементів відповідного рядка. Повернути суму середніх значень рядків матриці.
5. Дано двовимірний масив з *N*N* цілих чисел. Помножити кожен елемент стовпця матриці на середнє значення елементів відповідного стовпця. Повернути результат множення середніх значень стовпців матриці.
6. Дано двовимірний масив з *N*N* цілих чисел. Упорядкувати за зростанням елементи кожного рядка окремо.
7. Дано двовимірний масив з N*M цілих чисел. Виконати дзеркальне відображення матриці по горизонталі (необов’язково квадратної матриці).

Наприклад,

$$\begin{vmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 8 & 9 & 0 \end{vmatrix} => \begin{vmatrix} 8 & 9 & 0 \\ 5 & 6 & 7 \\ 1 & 2 & 3 \end{vmatrix}$$

8. Дано двовимірний масив з N*N цілих чисел. Поміняти елементи головної і бічної діагоналей місцями. Наприклад,

$$\begin{vmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 8 & 9 & 0 \end{vmatrix} => \begin{vmatrix} 3 & 2 & 1 \\ 5 & 6 & 7 \\ 0 & 9 & 8 \end{vmatrix}$$

9. Дано двовимірний масив з *N*N* цілих чисел. Додати до кожного елементу рядка матриці елемент відповідного рядка, що має максимальне значення.
10. Дано двовимірний масив з *N*N* цілих чисел. Додати до кожного елементу стовпця матриці елемент відповідного стовпця, що має максимальне значення.
11. Дано двовимірний масив з *N*N* цілих чисел. Помножити матрицю саму на себе (відповідно до правил множення матриць – використовувати квадратні матриці).
12. Дано двовимірний масив з *N*M* речовинних чисел. Кожний елемент рядка матриці помножити на випадкове число (числа-множники для різних рядків мають бути різними).
13. Дано двовимірний масив з *N*M* цілих чисел. Повернути з функції середнє значення мінімальних елементів кожного з рядків матриці. Наприклад,

$$\begin{vmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 8 & 9 & 0 \end{vmatrix}$$ => average (1,5,0) =>(0+1+5)/ 3 = 2,0

14. Дано двовимірний масив з N*M цілих чисел. Повернути з функції середнє значення максимальних елементів кожного з рядків матриці. Наприклад,

$$\begin{vmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 8 & 9 & 0 \end{vmatrix}$$ => average (3,7,9) =>(3+7+9)/ 3 = 6,3

15. Дано двовимірний масив з *N*N* речовинних чисел. Визначити, на

скільки сума елементів, що знаходяться нижче головної діагоналі, більше (менше) суми елементів головної діагоналі.

16. Дано двовимірний масив з *N*N* цілих чисел. Попарно поміняти місцями елементи головної та побічної діагоналей.
17. Дано двовимірний масив з *N*N* цілих чисел. Виконати циклічне зрушення елементів рядків масиву в напрямку зліва направо (останній елемент рядка повинен перемістится в її початок).
18. Дано двовимірний масив з *N*N* цілих чисел. Поміняти місцями максимальний і мінімальний елементи масиву.
19. Дано двовимірний масив з *N*N* цілих чисел. Елементи головної діагоналі записати в одномірний масив, отриманий масив впорядкувати за зростанням.
20. Дано двовимірний масив з *N*N* речовинних чисел. Визначити, на скільки сума елементів, що знаходяться вище головної діагоналі, більше (менше) суми елементів, що знаходяться нижче головної діагоналі.
21. Дано двовимірний масив з *N*N* цілих чисел. Виконати циклічне зрушення елементів рядків масиву в напрямку справа наліво (перший елемент рядка повинен перемістится в її кінець).
22. Дано двовимірний масив з *N*M* цілих чисел. Упорядкувати за зростанням елементи кожного стовпця окремо.

Додаткові умови виконання завдання:

- звіт має бути виконаний згідно з вимогами до оформлення лабораторних робіт;
- програма має бути документована у вигляді *doxygen* документації;
- вхідні дані мають бути провалідовані.

Контрольні запитання

1. Що таке «вимірність масиву», якою вона може бути?
2. Під час доступу до елементів двовимірного масиву вкажіть, яке призначення першого та другого індексів?
3. Від чого залежить обсяг пам’яті, що необхідний для збереження масива?
4. Чому під час роботи з масивами використовують функцію rand()?
5. Як забезпечити генерацію іншої послідовності чисел при кожному наступному виконанні програми?
6. Дайте оцінку наступному оголошенню масиву:

```
int mas[][2] = {{1,2}, {3,4}, {5,6}, {7,8}};
```

7. Як звернутися до елементу масиву? Наведіть приклад.
8. В якому порядку зберігаються у пам'яті комп'ютера елементи двовимірного масиву?
9. Скільки вимірів може мати масив?
10. Чим характеризуються діагональні елементи багатовимірних масивів? Скільки операторів циклу необхідно для їх обробки?

Лабораторна робота №7.
Структуровані типи даних

Тема роботи: робота зі структурованими типами даних.

Мета роботи: набути практичні навички щодо розроблення програм із застосуванням структур.

Теми для попереднього опрацювання:

- структури, доступ до членів структури;
- покажчики на структури;
- масиви структур;
- суміші (об'єднання) та перерахування.

Загальні відомості

Структура – складений тип даних (створюється програмістом); являє собою набір змінних різних типів, які об'єднані загальною назвою. Оголошення структури призводить до утворення шаблону, який використовується для створення об'єктів структури.

Змінні, що утворюють структуру, називаються членами структури. Члени структури також часто називаються елементами або полями. Як правило, члени структури пов'язані один з одним. Наступний фрагмент програмного коду оголошує шаблон структури, що визначає ім'я та адресу. Ключове слово *struct* повідомляє компілятору про оголошення структури.

```
struct addr
{
  char   name[30];   char
  street [40];
};
```

Оголошення обов'язково завершується крапкою з комою, оскільки оголошення структури – це оператор. Ім'я структури (у прикладі – *addr*) ідентифікує структуру даних і є специфікатором типу.

Для оголошення змінної, відповідної до даної структури, слід написати:

```
struct addr addr_info
```

Структура не може містити екземпляри самої себе, але можна визначати покажчики на структуру.

Для доступу до елементу структури необхідно вказати її ім’я, поставити крапку і вказати ім’я потрібного елементу. Для доступу до елементу структури через покажчик на об’єкт замість крапки використовується оператор «стрілка».

Об’єднання (union) – це складений тип даних; являє собою набір змінних різних типів, але які перебувають в одній і тій же області пам’яті. Пам’ять виділяється такого об’єму, щоб її було достатньо для зберігання найбільшого члена.

Для оголошення об’єднань використовується ключове слово *union*.

Приклад шаблону суміші з іменем типу *Telements*:

```
union Telements
{
  int    number;    char
  symbol;           char
  *pointer;
};
```

Приклад оголошення даних типу суміші:

```
union Telements dat, mas[5], *pu;
```

Для доступу до членів об’єднання використовуються ті ж синтаксичні конструкції, що і для структури (крапка і стрілка).

Приклад доступу до елементів об’єднання:

```
dat.number=57;
mas[2].symbol='A';
pu=&dat;
x=pu->number;
```

Основне завдання:

Розробити структуру з трьох елементів, згідно з індивідуальним завданням, при цьому:

- чисельні поля генеруються за допомогою функції *rand()*;
- рядкові поля подані вводяться з клавіатури;
- числові поля мають бути згенеровані випадковим чином у прийнятному діапазоні (напр., рік народження не може бути – «12121»);

- в кінці структура виводиться на екран.

Індивідуальні завдання:

1. **Завдання з автомобілями**. Створіть структуру **Car**, яка містить поля для року випуску, моделі автомобіля та кольору.
2. **Завдання з працівниками**. Створіть структуру **Employee**, яка містить поля для заробітної плати (число, яке генерується випадковим чином), прізвища працівника (рядок) та посади (рядок).
3. **Завдання з квартирами**. Створіть структуру **Apartment**, яка містить поля для площі (число, яке генерується випадковим чином), номера квартири (рядок) та адреси (рядок).
4. **Завдання з продуктами**. Створіть структуру **Product**, яка містить поля для ціни (число, яке генерується випадковим чином), назви продукту (рядок) та категорії (рядок).
5. **Завдання з книгами**. Створіть структуру **Book**, яка містить поля для року видання (число, яке генерується випадковим чином), назви книги (рядок) та імені автора (рядок).
6. **Завдання з користувачами**. Створіть структуру **User**, яка містить поля для віку (число, яке генерується випадковим чином), імені користувача (рядок) та електронної пошти (рядок).
7. **Завдання із замовленнями**. Створіть структуру **Order**, яка містить поля для кількості одиниць (число, яке генерується випадковим чином), назви товару (рядок) та ідентифікатора замовлення (рядок).
8. **Завдання з клієнтами**. Створіть структуру **Client**, яка містить поля для номера телефону (рядок), прізвища клієнта (рядок) та адреси (рядок).
9. **Завдання з рестораном**. Створіть структуру **Dish**, яка містить поля для ціни за порцію (число, яке генерується випадковим чином), назви страви (рядок) та інгредієнтів (рядок).
10. **Завдання з фільмами**. Створіть структуру **Movie**, яка містить поля для тривалості (число, яке генерується випадковим чином), назви фільму (рядок) та режисера (рядок).
11. **Завдання з комп'ютерами**. Створіть структуру **Computer**, яка містить поля для кількості процесорів (число, яке генерується випадковим чином), моделі комп'ютера (рядок) та обсягу оперативної пам'яті (рядок).
12. **Завдання з меблями**. Створіть структуру **Furniture**, яка містить поля для матеріалу (рядок), типу меблів (рядок) та виробника (рядок).
13. **Завдання з фруктами**. Створіть структуру **Fruit**, яка містить поля

для кількості вітамінів (число, яке генерується випадковим чином), назви фрукту (рядок) та кольору шкірки (рядок).

14. **Завдання з музикантами**. Створіть структуру **Musician**, яка містить поля для імені музиканта (рядок), інструменту (рядок) та жанру (рядок).
15. **Завдання з тваринами**. Створіть структуру **Animal**, яка містить поля для кількості ніг (число, яке генерується випадковим чином), назви тварини (рядок) та середовища проживання (рядок).
16. **Завдання з кольорами**. Створіть структуру **Color**, яка містить поля для назви кольору (рядок), коду кольору (рядок) та назви виробника фарби (рядок).
17. **Завдання з напоями**. Створіть структуру **Drink**, яка містить поля для кількості калорій (число, яке генерується випадковим чином), назви напою (рядок) та об'єму (рядок).
18. **Завдання з канцелярськими товариствами**. Створіть структуру **Stationery**, яка містить поля для ціни (число, яке генерується випадковим чином), назви товару (рядок) та виробника (рядок).
19. **Завдання з країнами**. Створіть структуру **Country**, яка містить поля для кількості населення (число, яке генерується випадковим чином), назви країни (рядок) та площі (рядок).
20. **Завдання із спортивними командами**. Створіть структуру **Team**, яка містить поля для кількості перемог (число, яке генерується випадковим чином), назви команди (рядок) та тренера (рядок).

Додаткові вимоги виконання завдання:

- звіт має бути виконаний згідно з вимогами до оформлення лабораторних робіт.

Контрольні запитання

1. Як виконати доступ до окремих елементів структури?
2. Чи можна вміст однієї структури привласнити іншій того ж типу, використовуючи звичайний оператор присвоювання?
3. Коли необхідно використовувати покажчики на структури?
4. Чи може бути членом структури інша структура?
5. Чим відрізняється об'єднання від структури?
6. Що дозволяє визначити оператор *sizeof*? Чим він корисний?
7. Як працює функція *rand*() ?

Лабораторна робота №8.
Покажчики

Темароботи: основи роботи з покажчиками. Робота з динамічними масивами.

Мета роботи: набути практичні навички щодо розроблення програм із динамічно створеними даними.

Теми для попереднього опрацювання:

- адресна арифметика;
- покажчики;
- масиви;
- звільнення та виділення пам’яті.

Загальні відомості

Кожна змінна, яка оголошується у програмі, має адресу – номер комірки пам’яті, в якій вона розташована. Адреса є невід’ємною характеристикою змінної. Можна оголосити іншу змінну, яка буде зберігати цю адресу і яка називається покажчиком. Покажчики застосовуються під час передачі у функцію параметрів, які мають бути змінені, під час роботи з масивами, а також з динамічною пам’яттю та в декількох інших випадках.

Оголошення покажчика має такий синтаксис:

*<тип> * <ідентифікатор> [= <ініціалізатор>];*

Покажчик може вказувати на значення базового типу, структури, об’єднання, функції, покажчика. Наприклад,

```
int * pi; // Покажчик на int
```

Існують дві операції, які стосуються роботи з покажчиками, тобто є:

- операція отримання адреси (адресація) &;
- операція отримання значення за адресою (непряма адресація або розіменування) *.

```
int a, *p;
p = & a; // змінній p присвоюється адреса змінної a
*p = 0; // Значення за адресою, що знаходиться у змінній p (тобто значення змінної a), буде дорівнювати 0
```

Функція *malloc()* визначена у бібліотеці *stdlib.h*. Вона використовується для ініціалізації покажчиків необхідним обсягом пам’яті. Пам’ять виділяється з сектора оперативної пам’яті, що доступний для будь-яких програм, які виконуються на даній машині.

Оскільки різні типи даних мають різні вимоги до пам’яті, треба

навчитися визначати розмір пам’яті в байтах для різного типу даних. Наприклад, пам’ять для масиву елементів типу *int* – це один обсяг пам’яті, а якщо необхідно виділити пам’ять для масиву такого ж розміру, але вже типу char – це буде інший обсяг. Для визначення обсягу пам’яті призначається функція *sizeof()*, яка приймає вираз і повертає його розмір. Наприклад, *sizeof(int)* поверне кількість байтів, необхідних для зберігання даних типу int.

Звільнення пам’яті виконується за допомогою функції *free()*.

Для того, аби уникнути витоку пам’яті, необхідно, щоб уся виділена пам’ять була вручну звільнена.

Контроль витоку пам’яті здійснюється таким чином (працює тільки в режимі *Debug*) (*https://msdn.microsoft.com/en-us/library/x98tx3cf.aspx*):

- на початку файлу з функцією main треба написати таке:
  #define CRTDBG_MAP_ALLOC #include <crtdbg.h>
  #define DEBUG_NEW new(_NORMAL_BLOCK, FILE , LINE) #define new DEBUG_NEW
- в кінці програми (у функції main перед «return 0») треба написати таке: *_CrtDumpMemoryLeaks()*.

Індивідуальні завдання:

1. Дано масив з *N* цілих позитивних чисел. Визначити, які числа в масиві є довершеними (такими, що дорівнюють сумі своїх дільників), вказати їхню кількість. Виконати перевірку на валідність масиву (вивести помилку та зупинити роботу додатка за наявності негативних чисел).
2. Дано двовимірний масив з *N***N* речовинних чисел. Мінімальні елементи кожного рядка переписати в одновимірний масив.
3. Дано масив з *N* речовинних чисел. Створити другий масив, який буде мати у собі всі елементи початкового масиву, що розташовані між першим та другим негативними елементами.
4. Дано масив з *N* цілих чисел. Визначити, чи є в масиві повторювані елементи; якщо такі є, то створити масив, в якому вказати, скільки разів які елементи повторюються.
5. Дано двовимірний масив з *N***N* цілих чисел. У кожному рядку масиву знайти кількість парних додатних чисел. Отримані результати записати в одновимірний масив.
6. Дано масив з *N* цілих чисел. Знайти мінімальний та максимальний елементи масиву. Визначити суму елементів, що розташовані між цими елементами. Створити другий масив, що містить ці елементи.
7. Дано двовимірний масив з *N***N* речовинних чисел. Максимальні

елементи кожного стовпця переписати в одновимірний масив.

8. Дано двовимірний масив з *N*N* цілих чисел. Елементи головної діагоналі записати в одновимірний масив, отриманий масив впорядкувати за зростанням.
9. Дано два масиви: *mas1[N]* і *mas2[M]*. Створити третій масив, у який переписати елементи масиву *mas1*, а потім *mas2*. Отриманий масив впорядкувати за зростанням.
10. Дано масив з *N* цілих чисел. Знайти безперервну послідовність позитивних чисел у вхідному масиві, сума елементів якої максимальна, та переписати їх у вихідний масив.
11. Дано два масиви: *mas1[N]* і *mas2[M]*. Створити третій масив, у який переписати по черзі по два елементи з вхідних масивів; почати з масиву *mas2*.
12. Дано масив з *N* цілих чисел. Визначити кількість пар сусідніх елементів з однаковими значеннями. Ці пари переписати у другий масив.
13. Дано масив з *N* речовинних чисел. Розмістити всі елементи з позитивними значеннями в лівій частині масиву, елементи з негативними значеннями – у правій, а нулі – між ними.
14. Дано двовимірний масив з *N*N* цілих чисел. Елементи головної діагоналі записати в одновимірний масив, отриманий масив впорядкувати за зростанням.
15. Дано масив з *N* речовинних чисел. Підрахувати кількість ділянок, які утворюють безперервні послідовності чисел з не зменшуваними значеннями. Максимальну ділянку переписати у інший масив.

Додаткові умови виконання завдання:

- звіт має бути виконаний згідно з вимогами до оформлення лабораторних робіт;
- проєкт має складатися мінімум з трьох файлів;
- продемонструвати відсутність витоків пам’яті;
- доступ до елементів масиву повинен здійснюватися через розіменування покажчиків, а не через оператор [];

Контрольні запитання

1. Як створити покажчик на масив?
2. Які операції можуть бути застосовані до покажчиків?
3. Як здійснюється звільнення пам’яті?
4. Як здійснюється виділення пам’яті?
5. Як створюється контроль за витоком пам’яті?
6. Чим відрізняється статичний масив від динамічного?

7. Як визначити поточний обсяг пам’яті для динамічного масиву?
8. Що виконується в наступному фрагменті програмного коду?

```
pointMas = &mas[0];
for(i = 0; i < arraySize; *pointMas++ = i++);
```

9. Чому треба звільняти пам’ять, яка динамічно виділялась?

Лабораторна робота №9.
Форматоване введення/виведення даних

Тема роботи: форматоване введення/виведення даних за допомогою функцій *printf / scanf.*

Мета роботи: набути практичні навички щодо розроблення програм із використанням консольного стандартного введення/виведення.

Теми для попереднього опрацювання:

- робота з клавіатурою;
- робота з виведенням на екран;
- бібліотечні функції;
- покажчики та масиви.

Загальні відомості

Основним завданням програмування є обробка інформації, тому будь-яка мова програмування має засоби для введення і виведення інформації.

Введення і виведення інформації здійснюється функціями стандартної бібліотеки. Прототипи функцій знаходяться у файлі *stdio.h.* Ця бібліотека містить, зокрема, функції *printf()* та *scanf().*

Функція *printf()* призначена для форматованого виведення. Вона переводить дані у символьне подання і виводить отримані зображення символів на екран. При цьому у програміста є можливість форматувати дані, тобто впливати на їх подання на екрані.

Загальна форма запису функції *printf ():*

```
printf(«рядок Форматів», об’єкт1, об’єкт 2, ..., об’єкт n);
```

Наприклад:

```
float y;
printf («Введіть y:»); // виводиться повідомлення
scanf («% f», & y); // вводиться значення змінної y
printf («Значення змінної y =% f», y); //виводиться значення y
```

Функція форматованого введення даних з клавіатури *scanf()* виконує читання даних, що вводяться з клавіатури, перетворює їх у внутрішній

формат і передає функції, що її викликає. При цьому програміст задає правила інтерпретації вхідних даних за допомогою специфікацій рядка формату.

Загальна форма запису функції *scanf ():*

scanf («рядок Форматів», адреса1, адреса2, ...);

Для формування адреси змінної використовується символ амперсанд ‘&’:

адреса = & об’єкт

Рядок форматів і список аргументів для функції є обов’язковими, аналогічні функції *printf ()*..

- Для введення рядків із клавіатури призначена функція *gets().*
- Для введення символів із клавіатури призначена функція *getc().*
- Для виведення рядків на екран призначена функція *puts().*
- Для виведення символів на екран призначена функція *putc().*

Основне завдання

Програму, яка була розроблена у попередній лабораторній роботі (робота з покажчиками), змінити так, щоб:

- початкові дані вводилися з клавіатури;
- видача результуючих даних проводилася на консоль.

Додаткові умови виконання завдання:

- звіт має бути виконаний згідно з вимогами до оформлення робіт;
- проєкт має складатися мінімум з трьох файлів;
- продемонструвати відсутність витоків пам’яті;
- дані генерувати за допомогою функції *rand(),* кількість елементів *(N, M...)* вводити з клавіатури;
- доступ до елементів масиву повинен здійснюватися через розіменування покажчиків, а не через оператор [];
- введення / виведення даних виконувати в окремих функціях.

Контрольні запитання

1. Як виводити текст на консоль?
2. Як читати дані з клавіатури?
3. У чому різниця форматованого виведення даних від неформатованого?
4. У якому файлі знаходяться прототипи функцій введення/виведення?
5. Який символ використовується для формування адреси змінної?
6. Як можна форматувати дані під час їх виведення на екран?

7. Які дії виконує функція читання під час введення даних із клавіатури?
8. Які дії виконує функція виведення даних на екран?
9. Як ввести текст, який складається з декількох слів?
10. Як вивести текст, який складається з декількох слів?

Лабораторна робота №10.
Рядки типу char*

Тема роботи: робота з рядками типу char*.

Мета роботи: набути практичні навички щодо розроблення програм із використанням рядків на мові С.

Теми для попереднього опрацювання:

- масиви;
- функції;
- рядки;
- покажчики.

Загальні відомості

Рядок у мові С – це одновимірний масив символів, останнім елементом якого є символ кінця рядка – нуль (рядок, що завершується нулем, тобто NULL *terminated string).*

Оголосити змінну типу рядок у мові С можна трьома способами:

1) оголосити масив, наприклад, з 40 символів:

   char s [40 + 1];

2) присвоїти рядковій змінній початкове значення (при цьому довжину рядка обчислює компілятор):

   char s [] = "Приклад ініціалізації рядка";

Праворуч від знака присвоювання записана рядкова константа. В кінець рядка автоматично додається нуль ('\0'). Константи символьних рядків поміщаються у клас статичної пам'яті;

3) створити неявний вказівник на масив символів, що використовується. У лівій частині від знаку присвоювання вказується покажчик на символ:

   *char *s = «Другий варіант ініціалізації»;*

Змінна *s* буде покажчиком на те місце в оперативній пам'яті, де розташовується рядкова константа. У такій формі запису криється потенційна помилка, яка полягає в тому, що покажчик на символ часто називають рядком.

string.h – заголовок стандартної бібліотеки мови С, що містить функції для роботи з нуль-термінованими рядками.

Для доступу до окремого символу у рядку треба вказати назву рядка та у квадратних дужках – номер символу. Нумерація символів у рядках починається з 0.

Індивідуальне завдання на оцінку «задовільно»:

1. Визначити, скільки разів у тексті зустрічається слово «програма».
2. Визначити, скільки у тексті оповідальних, питальних, окличних речень.
3. Визначити, скільки у тексті слів. Видати всі слова за абеткою.
4. У кожному рядку тексту змінити порядок символів на протилежний.
5. Визначити кількість слів у кожному рядку тексту.
6. Усі скорочення (т. д., т. п., ін.) замінити на повні словосполучення.
7. Усі повні словосполучення (так далі, тому подібне, інше) замінити на їхні загальноприйняті скорочення.
8. Текст – це список студентів. Визначити, скільки серед них мають однакові прізвища.
9. Вирахувати для тексту частотну таблицю: для кожного символу визначити його частоту появи у тексті (число таких символів у тексті ділене на загальне число символів у тексті).
10. Визначити, скільки разів у тексті зустрічається задане слово, яке необхідно ввести з клавіатури.
11. Знайти найдовше та найкоротше слово в заданому тексті.
12. Знайти всі слова – паліндроми (слова, які однаково читаються справа наліво та зліва направо), які зустрічаються в тексті.
13. Визначити % символів, що попарно збіглися, у вхідних текстах (кількість символів, що збіглися, до загальної кількості символів).
14. У кожному рядку тексту записана (без помилок) така послідовність символів: *a # b*, де *a* і *b* – цілі числа, # – одна з арифметичних операцій. Наприклад, 17 + 2. Обчислити значення всіх виразів, які записані у файлі.
15. Знайти всі числа, які зустрічаються в тексті.
16. Визначити, скільки у тексті голосних і скільки приголосних букв.
17. Текст – це програма на мові С. Визначити, скільки в ньому операторів циклу.
18. Текст – це програма на мові С. Визначити, чи є у наведеному тексті всі пари дужок: (), {}, [].
19. Визначити кількість таких слів у тексті, у яких перший і останній символи збігаються між собою.

20. «Зашифрувати» вхідний текст, для чого в кожному рядку тексту виконати циклічну перестановку символів на ***n*** позицій вправо (***i***-й символ стає ***i****+1*-м, а останній – першим). Значення ***n*** ввести з клавіатури.
21. «Зашифрувати» вхідний текст, для чого в кожному рядку тексту поміняти місцями перший символ з другим, третій – з четвертим і т.д. Виконати дешифрування.
22. Для запам'ятовування числа π іноді використовують наступну російську фразу: «это я знаю и помню прекрасно Пи многие знаки мне лишни напрасны». Число букв у кожному слові – це наступна цифра числа: «это» – 3, «я» – 1, «знаю» – 4 і т. д. Розробити програму, яка за зазначеним алгоритмом буде видавати число, використовуючи будь-який текст.
23. «Зашифрувати» заданий текст, для чого в кожному рядку тексту поміняти місцями перший символ з останнім, другий – з передостаннім і т. д. Виконати дешифрування.

Завдання на оцінку «добре» та «відмінно»:

- виконати завдання на «задовільно»;
- розробити функцію, яка реалізує вставку в рядок «*s*» другий рядок «*s2*» в «*i*»-у позицію рядка «*s*». Наприклад, insert(«abrakadabra», «TEXT2», 4) повинна створити рядок «abraTEXT2kadabra»;
- розробити функцію видалення з рядка «*s*» усіх символів з індексами в заданому діапазоні. Наприклад, reduce («abrakadabra», 4, 8) повинна створити рядок «abrara» (без підрядка kadab).

Додаткові вимоги виконання завдання:

- звіт має бути виконаний згідно з вимогами до оформлення робіт;
- проєкт повинен складатися мінімум з трьох файлів;
- продемонструвати відсутність витоків пам'яті.

Контрольні запитання

1. Як «склеїти» два рядки?
2. Як визначити, чи є в заданому рядку заданий підрядок?
3. Чому рядки в мові С закінчуються «\0» ?
4. Які функції використовують для введення С-рядків з клавіатури?
5. Як порівняти два рядки?
6. Як ввести рядок (символ) з клавіатури?
7. Як видати рядок (символ) на екран?
8. Як у заданому рядку видалити заданий підрядок?

Список літератури

1. Іванов Є.О. Основи мови програмування С++ : навчальний посібник/ Є.О. Іванов, Я.М. Ліндер, К.А. Жебер. – Київ : Логос, 2020. – 90 с.
2. Stroustrup B. **A Tour of C++ .**– Addison-Wesley Professional, 2018. – 256p.
3. Зеленський О.С. Основи програмування на С++ : навчальний посібник/ О.С. Зеленський, В.С. Лисенко. – Кривий Ріг : ДУЕТ, 2023.– 269 с.
4. Grigoryan V., Wu S., Expert C++: Become a proficient programmer by learning coding best practices with C++17 and C++20’s latest features, 2nd Edition, Packt Publishing, 2020. – 606p.
5. Davidson J., Gregory K., Beautiful C++: 30 Core Guidelines for Writing Clean, Safe, and Fast Code. – Addison-Wesley Professional, 2021. – 352p.
6. Бібліотека КНУБА. URL: http://library.knuba.edu.ua/
7. Освітній сайт КНУБА. URL: http://org.knuba.edu.ua/

Навчально-методичне видання

ПРОГРАМУВАННЯ ТА АЛГОРИТМІЧНІ МОВИ

Методичні вказівки
до виконання лабораторних робіт 1–10
для підготовки здобувачів першого (бакалаврського)
рівня вищої освіти
за спеціальностями 122 «Комп'ютерні науки»,
126 «Інформаційні системи та технології»

Укладачі: **Поплавський** Олександр Анатолійович,

Босенко Ігор Валерійович

Випусковий редактор Л.С. Тавлуй
Комп'ютерне верстання Л.В. Лабунець

Підписано до друку 24.10.2024. Формат 60 х $84_{1/16}$
Ум. друк. арк. 3,02. Обл.-вид. арк. 3,25.
Електронний документ. Вид. № 129/ІІІ-24.

Видавець і виготовлювач:
Київський національний університет будівництва і архітектури

Проспект Повітряних Сил, 31, Київ, Україна, 03037

Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи ДК № 808 від 13.02.2002