МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

ПРОГРАМУВАННЯ ТА АЛГОРИТМІЧНІ МОВИ

Методичні вказівки
до виконання лабораторних робіт 1–15
для здобувачів першого (бакалаврського) рівня вищої освіти
спеціальностей F3 «Комп'ютерні науки»
та F6 «Інформаційні системи і технології»

Київ 2026

УДК 004.43
П78

Укладачі: О. А. Поплавський д-р техн. наук, доцент;
І. В. Босенко, д-р філ., старш. викладач
І.А. Пороховніченко, асистент

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

Відповідальна за випуск Т. А. Гончаренко, д-р техн. наук, професор, зав. каф. інформаційних технологій

Затверджено на засіданні кафедри інформаційних технологій, протокол № 6 від 16 лютого 2026 року.

В авторській редакції.

Програмування та алгоритмічні мови [електронне видання]: П78 методичні вказівки до виконання лабораторних робіт 1–15 / уклад.: О. А. Поплавський, І. В. Босенко, І.А. Пороховніченко. – Київ: КНУБА, 2026. – 69 с.

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

Призначено для здобувачів першого (бакалаврського) рівня вищої освіти спеціальностей F3 «Комп’ютерні науки» та F6 «Інформаційні системи і технології».

Зміст

Загальні положення

Методичні вказівки до освітньої компоненти «Програмування та алгоритмічні мови» призначені для здобувачів першого (бакалаврського) рівня вищої освіти спеціальностей F3 «Комп'ютерні науки» та F6 «Інформаційні системи і технології». Метою методичних вказівок є забезпечення послідовної підготовки здобувачів до виконання лабораторних робіт, формування практичних навичок алгоритмізації, розроблення, налагодження, тестування та аналізу програм.

Виконання лабораторних робіт є важливою складовою вивчення дисципліни та спрямоване на закріплення теоретичних знань, отриманих під час лекційних занять і самостійної роботи. У процесі виконання лабораторних робіт здобувачі мають оволодіти базовими підходами до побудови алгоритмів, засобами структурного програмування, принципами модульної організації програм, а також набути практичного досвіду розв'язання прикладних задач засобами мови C/C++ у середовищі розробки, визначеному викладачем.

Тематика лабораторних робіт охоплює основні розділи курсу: розроблення та відлагодження програм у середовищі програмування, реалізацію лінійних, розгалужених і циклічних алгоритмів, роботу з одновимірними та багатовимірними масивами, використання функцій, структурованих типів даних, покажчиків, рядків, файлів, динамічних масивів, а також створення багатофайлових проєктів. Окремі завдання можуть передбачати застосування елементів графіки, засобів візуалізації та базових механізмів обробки подій відповідно до змісту освітньої компоненти.

Під час виконання лабораторних робіт здобувач повинен продемонструвати вміння аналізувати умову задачі, будувати алгоритм її розв'язання, обґрунтовувати вибір методів і засобів реалізації, розробляти коректний програмний код, виконувати компіляцію, налагодження та тестування програми, а також аналізувати отримані результати. Особлива увага приділяється якості програмної реалізації, дотриманню вимог до оформлення коду, логічності побудови алгоритму та коректності роботи програми на тестових даних.

Варіанти індивідуальних завдань до лабораторних робіт визначаються викладачем. Перед початком виконання кожної лабораторної роботи здобувач повинен ознайомитися з її метою, теоретичними відомостями, постановкою завдання, вимогами до результатів виконання та порядком

оформлення звіту. У разі потреби здобувач має виконати попередню підготовку: опрацювати рекомендований теоретичний матеріал, повторити відповідні конструкції мови програмування та підготувати початкові варіанти алгоритмів.

Звіт до лабораторної роботи повинен містити тему, мету роботи, завдання, короткі теоретичні відомості, опис алгоритму розв’язання, текст програми з поясненнями, результати тестування та висновки. За потреби до звіту можуть включатися блок-схеми, таблиці вхідних і вихідних даних, ілюстрації результатів роботи програми та інші матеріали, що підтверджують правильність виконання завдання. Оформлення звіту має відповідати встановленим вимогам.

Методичні вказівки орієнтовані на формування у здобувачів алгоритмічного мислення, навичок самостійної роботи, культури програмування та здатності застосовувати здобуті знання для розв’язання навчальних і прикладних задач у галузі інформаційних технологій.

Лабораторна робота №1

РОБОТА З IDE MICROSOFT VISUAL STUDIO. ПРОГРАМУВАННЯ АЛГОРИТМІВ ЛІНІЙНОЇ СТРУКТУРИ, КОМПІЛЯЦІЯ ТА ВІДЛАГОДЖЕННЯ ПРОГРАМИ

Тема. Програмування алгоритмів лінійної структури, компіляція та відлагодження програм.

Мета роботи: Вивчення основних прийомів роботи з інтегрованим середовищем розробки *Microsoft Visual Studio*. Написання найпростішої програми на мові C/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*.

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 C++ при інсталяції *IDE* слід виконати такі налаштування (рис. 1).

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

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.

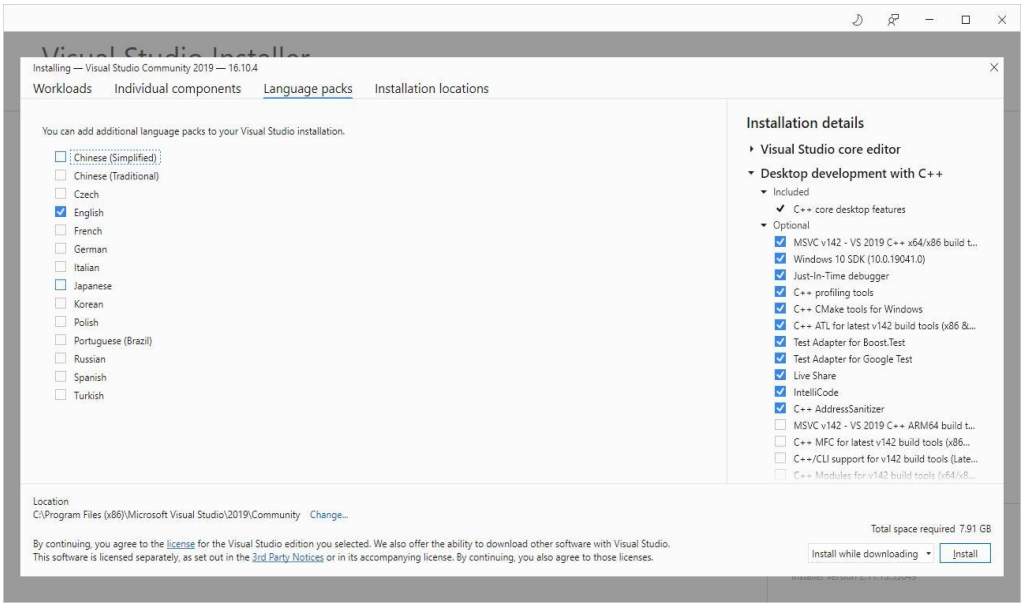

Рис. 1. Налаштування при інсталяції

Рішення та проєкти 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, при побудові 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. 2).

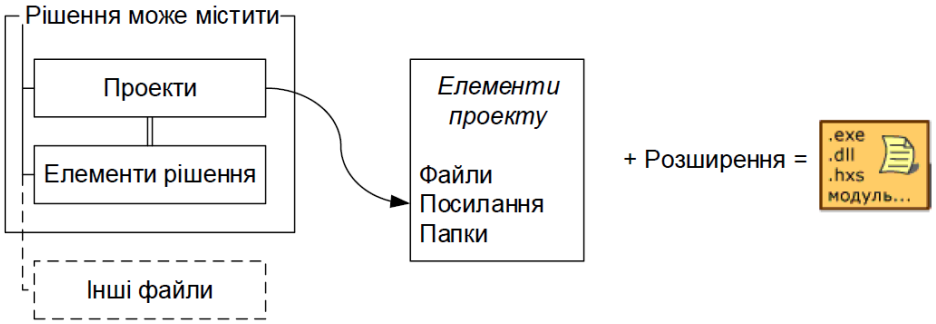

Рис. 2. Схема вмісту рішення

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 їх 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.

При додаванні файлу в проєкт інформація про його фізичне місце розташування заноситься в файл проєкту. При видаленні 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.

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. 3)

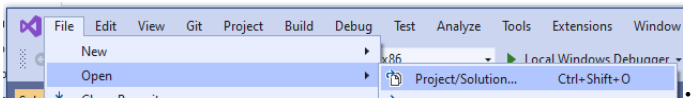

Рис. 3. Вибір існуючого проєкту

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. 4):

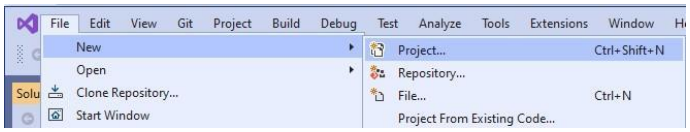

Рис. 4. Створення нового проєкту

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

Щоб створити порожній проєкт, використовуйте шаблон *Empty Project.*

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. 5).

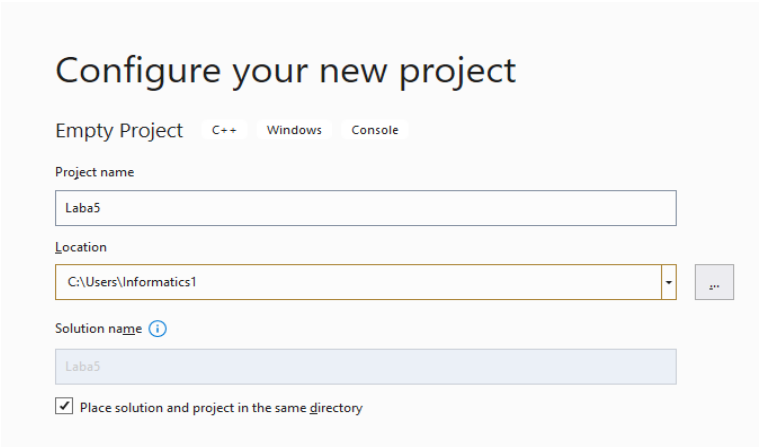

Рис. 5. Налаштування параметрів проєкту

Після цього відкриється робоче вікно *Visual Studio*. В області ліворуч розташований Провідник рішень (Solution Explorer), який відображає поточне рішення, яке може містити один чи більше проєктів. На рис. 6 наведено приклад рішення «*Laba5*» (наведено зеленим). Це рішення містить один проєкт із ім'ям «*Laba5*» (наведено синім). В 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.

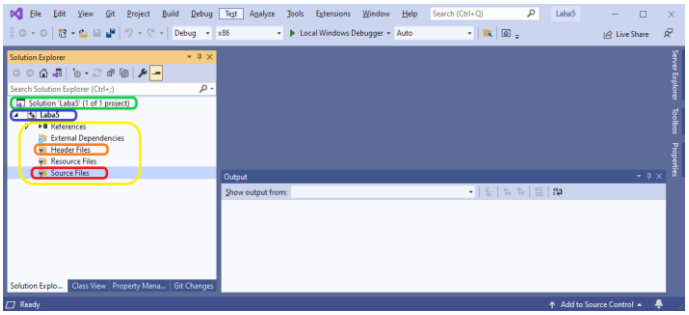

Рис. 6. Приклад рішення «*Laba5*»

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, викликати 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. 7 показано додавання до проєкту «*Laba5*» нового файлу, в якому буде написано код програми:

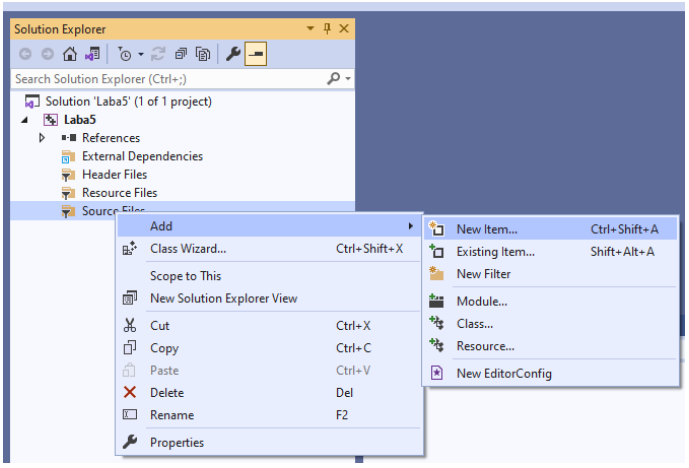

Рис. 7. Додавання нового файлу

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 рисунку нижче 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. 8).

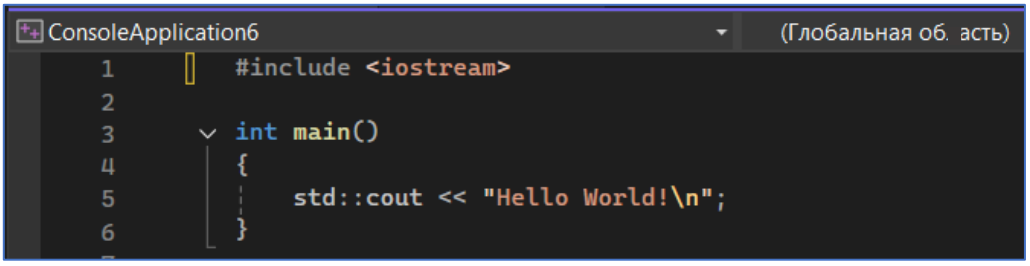

Рис. 8. Вікно для написання коду

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. 9)

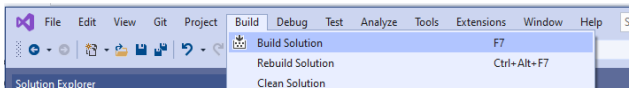

Рис. 9. Збірка програми

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. 10 (наведено синім).

Якщо при побудові 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 в вікні *Output* (рис. 11).

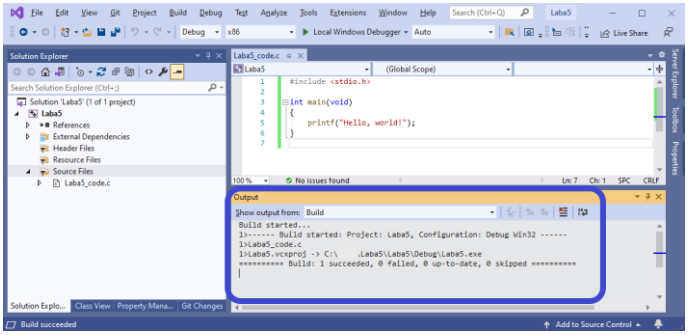

Рис. 10. Вікно *Output*

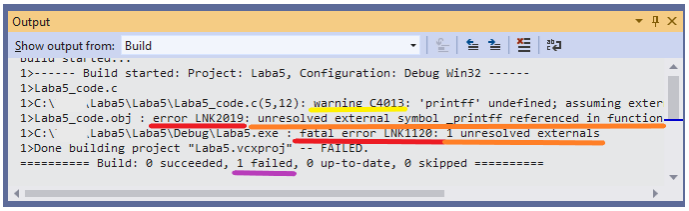

Рис. 11. Помилки при побудові програми

Якщо вікно *Output* було випадково закрито, то знову відкрити його можна обравши опції меню *Debug: Windows: Output* (рис. 12).

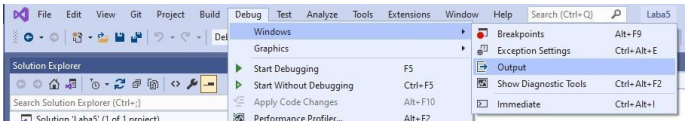

Рис. 12. Відкриття вікна *Output*

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 таких проблем, як логічні та семантичні помилки, які проявляються 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. 13 *Break Point* – це червоний кружечок).

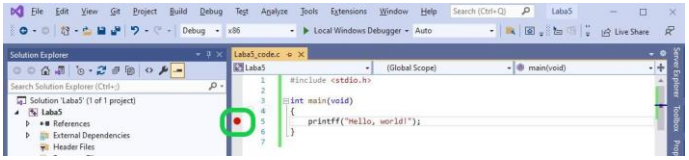

Рис. 13. *Break Point*

Для запуску програми в режимі налагодження потрібно або обрати опції меню

Debug: Start Debugging, або натиснувши гарячу клавішу **F5** (рис. 14).

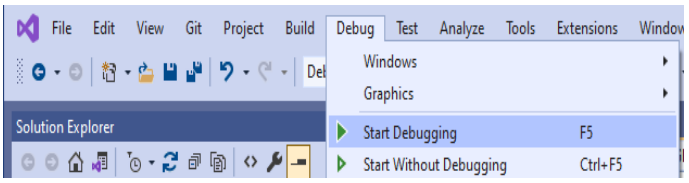

Рис. 14. *Start Debugging*

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 рисунку нижче наведено помаранчевим). Інформацію про значення, які зберігаються у змінних, можна отримати або за допомогою вікна *Autos* (на рис. 15 наведено салатовим), або вікна *Watch* (на рисунку нижче наведено зеленим).

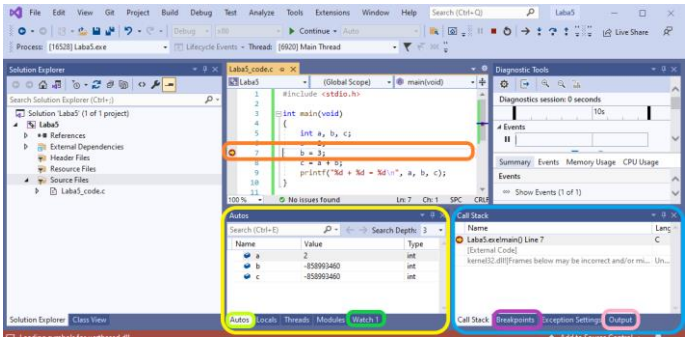

Рис. 15. Вікна *Autos* та *Watch*

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 підказка *Add item to watch*), або клікнути правою кнопкою миші і в контекстному меню обрати опцію *Add Watch* (рис. 16).

Рис. 16. 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. 17).

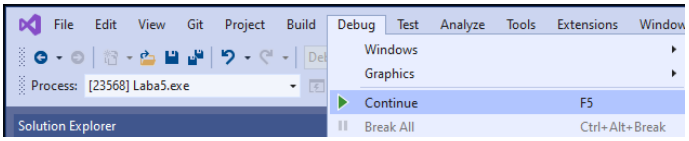

Рис. 17. Функція *Debug: Continue*

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. 18).

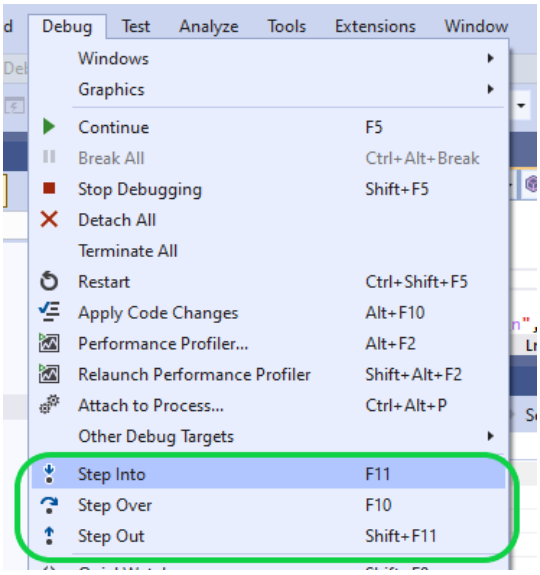

Рис. 18. Покрокове виконання функцій

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 натисніть **F11,** то наступний рядок буде виконано із заходом всередину функцій 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 по мірі виконання програми.

На рис. 19 наведено приклад вмісту змінної *c* до (рисунок ліворуч) та після (рисунок праворуч) виконання її модифікації (рядок 8).

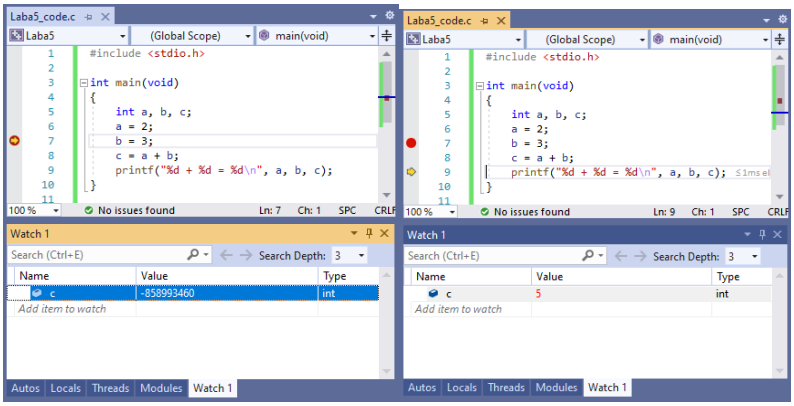

Рис. 19. Вміст змінної *c*

Дострокове завершення роботи програми в налагоджувальному режимі.

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

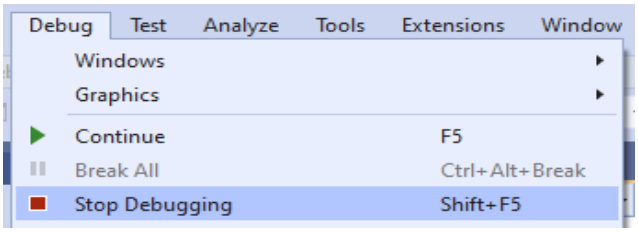

Рис. 20. Функція *Debug*:*Stop Debugging*

Деякі стандартні математичні функції

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

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

fabs – абсолютне значення дробового числа - |x|: *double fabs(double x);*

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

pow – піднесення до степеня: *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 acos(double x);*

asin – арксинус – *arcsin(x)*: *double asin(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.

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 варіанту. 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 рядку 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. Повернутись із математичної функції в функцію *main()* та переконатися за допомогою віконця *Watch* що деякі змінні змінили своє значення.

g. Виконати програми до кінця та продемонструвати результат обчислення задачі згідно варіанту (табл. 2).

Таблиця 2

Варіанти завдань

Варіант	Завдання
1	Обчислити довжину кола, площу кола 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, заданого внутрішнім радіусом r та зовнішнім радіусом R (R > r)
5	Обчислити площу прямокутного трикутника за довжинами його катета *a* і гіпотенузи .
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 акваріум зі стороною ***a*** метрів, наповнений до країв, опустили кулю діаметром ***b*** метрів. Обчислити, який відсоток від початкового об’єму води вилляється
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 наскрізний циліндричний отвір діаметром b. На скільки відсотків зменшився об’єм куба?

11	Обчислити периметр прямокутного трикутника за довжинами 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 внього кіл
17	Обчисліть периметр прямокутного трикутника за довжинами його катета *a* і гіпотенузи *c*
18	Є рівносторонній трикутник зі стороною a. Знайти співвідношення довжин кіл, описаного та вписаного в нього
19	Обчислити довжину кола, описаного навколо рівностороннього трикутника зі стороною .
20	Обчислити об’єм кулі та площу її поверхні за заданим радіусом r
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

Тема. Програмування алгоритмів, що розгалужуються.

Мета: Набути практичних навичок в 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, вкладена 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
}
```

При виконанні оператора switch обчислюється вираз, записаний після *switch*, він має бути цілим числом. Отримане значення послідовно порівнюється з константами, які записані слідом за *case*. При першому 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 C/C++ є умовна операція «? :», яка називається тернарною операцією (тобто тримісна (має три операнди), єдина в C/C++).

Форма запису тернарної операції:

"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 дійсних чисел ***a***, ***b***, ***c*** визначити корені квадратного рівняння $ax^2 + bx + c = 0$, якщо вони є.
3. При заданому ***x*** обчислити значення ***y*** відповідно до формули:

$$y = \begin{cases} x^2 - 1, & x > 0 \\ 5x^2 - x + 3, & x \le 0 \end{cases}$$

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 трьох значного числа однакові цифри.

8. Визначити позицію цифри з мінімальним значенням у трьох-значному 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 **1,2,3,4,5**).

10. Визначити, чи існує трикутник з заданими кутами **a**, **b**, **c**.

11. Знайти максимальне значення серед заданих трьох змінних.

12. Визначити, чи існує ромб з заданими кутами **a**, **b**, **c**, **d**.

13. Визначити, чи існує чотирикутник з 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 довжину кола 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 трьох- значному числі. Нумерація починається з 1. Наприклад, ***x***=453 -> ***y***=позиція [5]=2.

17. Визначити, чи існує трикутник з заданими сторонами ***a***, ***b***, ***c***.

18. Визначити, чи існує ромб з заданими кутами **a**, **b**, **c**, **d**.

19. Визначити століття за заданим роком. Наприклад, ***x*** = 2010 -> ***y*** = 21; ***x*** = 2201 -> ***y*** = 23.

20. За номером дня тижня d у діапазоні 1..7 вивести назву дня тижня та визначити, чи це робочий день чи вихідний. Для інших значень

вивести повідомлення про помилку.

21. За номером місяця *m* у діапазоні 1..12 визначити квартал пору року. Для інших значень вивести повідомлення про помилку.

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. Програмування алгоритмів циклічної структури. Документування коду.

Мета: Набути практичних навичок розроблення програм із використанням циклічних структур та 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 при різних значеннях 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:

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)
{
sum+=a++;
}
```

2. Цикл із післяумовою:

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

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

Наприклад,

```
int a=0, sum = 0;
 do
{
sum+=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, поки вираз-умова не стане дорівнювати 0.

for (int a=0, sum = 0; a <= 10 ; a++) sum+=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. Оператор break перериває виконання найближчого циклу або оператора switch, після чого керування передається оператору, що йде безпосередньо після нього. Використовується, якщо умову продовження ітерацій треба перевіряти в середині циклу;

continue – оператор переходу до наступної ітерації циклу *while, do-while* або *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-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. В 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 відсотковою 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 раз.
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 цикла?
10. Яке призначення операторів *break* і *continue*?

Лабораторна робота №4
ОДНОВИМІРНІ МАСИВИ

Тема. Одновимірні масиви.

Мета: Набути практичних навичок розроблення програм з використанням статичних масивів.

Теми для попереднього опрацювання:

- одновимірні масиви;
- ініціалізація масиву;
- рядки.

Загальні відомості

Масиви є важливою структурою даних у програмуванні. Масив - 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 використовуються 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 знака 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 можуть

використовуватися С-рядки (масиви символів char[]) та об’єкти типу 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 масивцілих чисел за зростанням методом «бульбашки».
3. Відсортувати масивцілих 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 в 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* і *@date*;
- звіт має бути виконаний згідно з вимогами до оформлення лабораторних робіт.

Контрольні питання

1. Як рядки представляються в мові програмування C++?
2. Яке призначення індексів при оголошенні масиву символів, 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 рядків типу char?

Лабораторна робота №5
ФУНКЦІЇ

Тема. Функції. Варіативні функції.

Мета: Набути практичні навички щодо розроблення програм з 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/С++є два типи функцій:

- які не повертають значень;
- що повертають значення.

Функції, що не повертають значення по завершенню 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 по завершенню 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>];

При виконанні *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/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 (*.cpp), у якому оголошуються та визначаються функції;
- створення файлів типу *.c (для визначення) і *.h (*.hpp) (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 при 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. При 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;

- при передачі об’єктів виконується їх копіювання, як наслідок, у випадку великих об’єктів витрачається багато пам’яті.

2) Передача параметрів по посиланню. 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 по посиланню, 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 збільшено, 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 у відповідності 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 більше, 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 більше, 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 всіх своїх додатних дільників, крім самого числа). 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 величину прибутку при вкладі заданої суми грошей x у банк під встановлений процент річних y за n років.

12. Отримати заголовний еквівалент переданого символу, якщо символ – рядкова літера. 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 початкового значення та кроку прогресії.

17. Знайти значення n-го елементу геометричної прогресії, при заданих значеннях начального значення та кроку 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 при 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. Функції. Робота з багатовимірними масивами. Використання функцій стандартної бібліотеки. Функція *rand()*.

Мета: Придбати практичні навички щодо розроблення програм з 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 C/C++ дозволяє створювати багатовимірні масиви. Найпростішим видом багатовимірного масиву є двовимірний масив.

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

тип ім’я_масиву [кількість_рядків][кількість_стовпців];

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

Наприклад, для оголошення двовимірного масиву цілих чисел з 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]**

При оголошенні 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 знака присвоєння 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 у 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 C/C++ 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 при кожному наступному 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 у двовимірний масив, що має *M* рядків і *N* стовпців. Отриманий двовимірний масив обробити відповідно до індивідуального завдання і вивести його на екран;
- одновимірний масив заповнити псевдовипадковими числами (функція *rand ()*) у діапазоні від *5 * K* до *10 * K*, де *K* – номер 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.
3. Дано двовимірний масив з *N*N* цілих чисел. Помножити кожен елемент рядка матриці на значення елементу бічної діагоналі відповідного рядка.
4. Дано двовимірний масив з *N*N* цілих 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).
8. Дано двовимірний масив з N*N цілих чисел. Поміняти елементи головної і бічної діагоналей місцями.
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* цілих 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.
14. Дано двовимірний масив з N*M цілих чисел. Повернути з функції середнє значення максимальних елементів кожного з рядків матриці.
15. Дано двовимірний масив з *N*N* цілих 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 $N*N$ цілих 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.

Контрольні запитання

1. Що таке «вимірність масиву», якою вона може бути?
2. При доступі до елементів двовимірного масиву укажіть, яке призначення першого та другого індексів?
3. Від чого залежить обсяг пам’яті, що необхідний для збереження масива?
4. Чому при роботі 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?

Лабораторна робота №7
СТРУКТУРОВАНІ ТИПИ ДАНИХ

Тема. Робота із структурованими типами даних.

Мета: придбати практичні навички щодо розроблення програм із застосуванням структур.

Теми для попереднього опрацювання:

- структури, доступ до членів структури;
- покажчики на структури;
- об’єднання (union) та перерахування.

Загальні відомості

Структура – це складений тип даних, що створюється програмістом і 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 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 об’єднання 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 (полів), згідно з індивідуальним завданням. Створити 4 структурні зміні і заповнити їх значеннями, при цьому:

- чисельні поля генеруються за допомогою функції *rand()*;
- рядкові поля вводяться з клавіатури;
- числові поля мають бути згенеровані випадковим чином в прийнятному діапазоні (напр. Рік народження не може бути – «12121»)
- після заповнення структурні зміні вивести 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 його посади (рядок).

3. **Завдання з квартирами**: Створіть структуру **Apartment**, яка містить поля для площі (число, яке генерується випадковим чином), номеру квартири (число, яке генерується випадковим чином) 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 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 з 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 (число, яке генерується випадковим чином), 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 (число, яке генерується випадковим чином).

12. **Завдання з меблями**: Створіть структуру **Furniture**, яка містить поля для матеріалу (рядок), номер типу меблів (число, яке генерується випадковим чином) 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), нимера інструменту (число, яке генерується випадковим чином) 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 (число, яке генерується випадковим чином) 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 (число, яке генерується випадковим чином).

18. **Завдання з канцелярськими товарами**: 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 (число, яке генерується випадковим чином).

20. **Завдання з 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 питання

1. Як виконати доступ до окремих елементів структури?
2. Чи можна вміст однієї структури присвоїти 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
МАЛЮВАННЯ В VISUAL STUDIO

Тема. Малювання графічних примітивів.

Мета: набути практичних навичок створення найпростішого графічного Windows-застосунку в середовищі Microsoft Visual Studio та використання засобів Win32 API і GDI для побудови графічних примітивів і виведення тексту у вікні програми.

Загальні відомості

У середовищі Microsoft Visual Studio графічні завдання виконуються у клієнтській області вікна за допомогою функцій Windows API та бібліотеки GDI.

Графічне виведення у Windows-застосунках здійснюється за допомогою контексту пристрою, який описується типом HDC. Контекст пристрою містить параметри малювання: поточне перо, пензель, шрифт, кольори ліній і тексту та інші атрибути, необхідні для побудови графічних об’єктів.

Для коректного відображення графіки малювання слід виконувати в обробнику повідомлення WM_PAINT. Контекст пристрою в такому разі отримують функцією BeginPaint(), а після завершення малювання звільняють функцією EndPaint().

Для побудови найпростіших графічних об’єктів можуть використовуватися такі функції:

- MoveToEx() і LineTo() – для побудови відрізків;
- Rectangle() – для побудови прямокутників;
- Ellipse() – для побудови кіл та еліпсів;
- Polygon() – для побудови багатокутників;
- TextOut() або DrawText() – для виведення тексту;
- SetTextColor() – для задання кольору тексту.

Для зміни параметрів малювання використовують графічні об’єкти HPEN і HBRUSH. Перо визначає колір, стиль і товщину контуру, а пензель – колір заповнення фігури. Перед використанням створені графічні об’єкти вибирають у контекст пристрою, а після завершення роботи їх потрібно коректно знищити.

Створення проєкту

Для виконання лабораторної роботи необхідно створити проєкт типу **Windows Desktop Application** або **Windows Desktop Wizard** у середовищі Microsoft Visual Studio. Під час створення проєкту потрібно обрати мову програмування **C++**, платформу **Windows** і тип проєкту **Desktop**.

Після створення проєкту програма повинна створювати головне вікно застосунку, обробляти повідомлення Windows і виконувати малювання в обробнику WM_PAINT.

Приклад найпростішої графічної програми

Нижче наведено приклад найпростішого Windows-застосунку, створеного в середовищі Microsoft Visual Studio. Програма відкриває вікно, у якому в обробнику WM_PAINT малюється рамка, коло, лінії та текстовий напис червоного кольору. Цей приклад демонструє базовий принцип роботи з функціями Win32 API і GDI та може бути використаний як основа для виконання індивідуального завдання.

Після створення проєкту в середовищі Visual Studio основний код малювання слід розміщувати в обробнику повідомлення WM_PAINT. У шаблоні проєкту потрібно знайти фрагмент:

```
case WM_PAINT:
{
PAINTSTRUCT ps;
HDC hdc = BeginPaint(hWnd, &ps);
// код малювання
EndPaint(hWnd, &ps);
}
break;
```

Саме **всередину цього блоку**, між викликами BeginPaint() і EndPaint(), потрібно додавати команди побудови графічних примітивів.

Нижче наведено приклад найпростішого фрагмента програми для побудови рамки, кола, лінії та текстового напису:

```
case WM_PAINT:
{
PAINTSTRUCT ps;
HDC hdc = BeginPaint(hWnd, &ps);

// Малювання зовнішньої рамки
```

```
Rectangle(hdc, 30, 30, 550, 350);

// Малювання кола
Ellipse(hdc, 100, 100, 200, 200);

// Малювання лінії
MoveToEx(hdc, 250, 120, NULL);
LineTo(hdc, 400, 220);

// Встановлення червоного кольору тексту
SetTextColor(hdc, RGB(255, 0, 0));

// Виведення підпису у вікні
TextOut(hdc, 360, 320, L"Прізвище І.П.", 13);

EndPaint(hWnd, &ps);
}
break;
```

Основне завдання

Створити в середовищі Microsoft Visual Studio графічний Windows-застосунок. У клієнтській області вікна побудувати рамку у вигляді прямокутника. Усередині рамки за допомогою не менше ніж трьох графічних елементів намалювати зображення згідно з індивідуальним завданням. Для побудови рисунка дозволяється використовувати лінії, прямокутники, кола, еліпси, багатокутники та текстові написи.

У правому нижньому куті вікна необхідно вивести прізвище та ініціали студента червоним кольором.

Індивідуальні завдання

1. Сонце. Побудувати зображення сонця жовтого кольору та восьми променів оранжевого кольору.

2. Дерево. Побудувати зображення дерева, використовуючи прямокутник для стовбура та трикутники для крони.

3. Годинник. Побудувати зображення годинника, використовуючи коло для циферблата та 12 коротких ліній для позначення годинних поділок.

4. Квітка. Побудувати зображення квітки, використовуючи коло для серединки та еліпси для пелюсток, розташованих навколо неї.

5. Будинок. Побудувати зображення будинку, використовуючи прямокутник для основної частини, трикутник для даху та прямокутники або квадрати для вікон.

6. Літак. Побудувати зображення літака, використовуючи прямокутники для корпусу та трикутники для крил і хвостової частини.

7. Автомобіль. Побудувати зображення автомобіля, використовуючи прямокутники для кузова та кола для коліс.

8. Корабель. Побудувати зображення корабля, використовуючи прямокутний корпус, щогли та трикутні вітрила.

9. Риба. Побудувати зображення риби, використовуючи еліпс для тулуба та трикутники для плавників і хвоста.

10. Гори. Побудувати пейзаж із зображенням гір, використовуючи кілька трикутників різної висоти.

11. Метелик. Побудувати зображення метелика, використовуючи лінію або вузький еліпс для тулуба та еліпси для крил.

12. Котик. Побудувати зображення котика, використовуючи кола або еліпси для голови й тулуба та трикутники для вух.

13. Яблуко. Побудувати зображення яблука, використовуючи коло або еліпс для плоду та прямокутник для плодоніжки.

14. Книга. Побудувати зображення книги, використовуючи прямокутник для обкладинки та лінії для позначення сторінок.

15. Сніжинка. Побудувати зображення сніжинки, використовуючи кілька ліній, проведених із центра в різних напрямках.

16. Зірка. Побудувати зображення зірки за допомогою ліній або багатокутника.

17. Корона. Побудувати зображення корони, використовуючи комбінацію трикутників і кіл для декоративних елементів.

18. Велосипед. Побудувати зображення велосипеда, використовуючи лінії для рами та два кола для коліс.

19. Серце. Побудувати зображення серця, використовуючи два кола у верхній частині та трикутник або ламані лінії в нижній частині.

20. Вітряк. Побудувати зображення вітряка, використовуючи прямокутник для опори та чотири лопаті, побудовані з ліній або трикутників.

Додаткові вимоги виконання завдання:

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

Контрольні запитання

1. Що таке контекст пристрою HDC і для чого він використовується?
2. У якому обробнику повідомлень слід виконувати малювання у Windows-застосунку?
3. Для чого призначені функції BeginPaint() і EndPaint()?
4. Які функції використовують для побудови лінії, прямокутника та еліпса?
5. Яке призначення мають об’єкти HPEN і HBRUSH?
6. Як виконати виведення тексту у вікні програми?
7. Як задати колір тексту?
8. Чому віконний застосунок є кращим для графічних завдань, ніж консольне вікно?
9. Які дії потрібно виконати для коректного звільнення графічних ресурсів?
10. Що відбувається з графічним вмістом вікна після його перекриття або зміни розміру?

Лабораторна робота №9
ПОКАЖЧИКИ

Тема. Основи роботи з покажчиками. Робота з динамічними масивами.

Мета: набути практичних навичок щодо розроблення програм з динамічно створеними даними.

Теми для попереднього опрацювання:

- адресна арифметика;
- покажчики;
- масиви;
- виділення та звільнення 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 при передачі у функцію параметрів, які мають бути змінені, при роботі з

масивами, при роботі 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*

Для роботи з покажчиками використовують такі основні операції:

- 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*. Функція malloc() використовується для динамічного виділення блоку пам’яті потрібного розміру. 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 розміру в байтах використовується оператор sizeof. Наприклад, *sizeof(int)* поверне кількість байтів, необхідних для зберігання даних типу int.

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

Основне завдання

1. Всі масиви в індивідуальному завданні – динамічні.
2. Розмірність масивів ввести клавіатури
3. Масиви заповнити випадковими числами.
4. В кінці кожної програми вивести в консоль розмір пам’яті яку займає оброблений масив та ім’я автора розробника коду

Індивідуальні завдання

1. Дано масив із N цілих додатних чисел. Визначити, які числа масиву є досконалими (тобто дорівнюють сумі всіх своїх додатних дільників, крім самого числа), і підрахувати їх кількість. Передбачити перевірку коректності вхідних даних: якщо масив містить від'ємні числа, вивести повідомлення про помилку та завершити виконання програми.

2. Дано двовимірний масив розміру N×N із цілих чисел. Мінімальні елементи кожного рядка записати в одновимірний масив.

3. Дано масив із N цілих чисел. Створити другий масив, що міститиме всі елементи початкового масиву, розташовані між першим і другим від'ємними елементами.

4. Дано масив із N цілих чисел. Визначити, чи є в масиві повторювані елементи. Якщо такі елементи є, створити масив, у якому для кожного повторюваного значення вказати кількість його появ.

5. Дано двовимірний масив розміру N×N із цілих чисел. У кожному рядку масиву знайти кількість парних додатних чисел. Отримані результати записати в одновимірний масив.

6. Дано масив із N цілих чисел. Знайти мінімальний і максимальний елементи масиву. Обчислити суму елементів, розташованих між ними. Створити другий масив, що міститиме ці елементи.

7. Дано двовимірний масив розміру N×N із цілих чисел. Максимальні елементи кожного стовпця записати в одновимірний масив.

8. Дано двовимірний масив розміру N×N із цілих чисел. Елементи головної діагоналі записати в одновимірний масив і впорядкувати його за зростанням.

9. Дано два масиви: mas1[N] і mas2[M]. Створити третій масив, до якого спочатку записати елементи масиву mas1, а потім елементи масиву mas2. Отриманий масив упорядкувати за зростанням.

10. Дано масив із N цілих чисел. Знайти неперервну підпослідовність додатних елементів, сума яких є максимальною, і записати цю підпослідовність у вихідний масив.

11. Дано два масиви: mas1[N] і mas2[M]. Створити третій масив, у який почергово записувати по два елементи з вхідних масивів, починаючи з масиву mas2.

12. Дано масив із N цілих чисел. Визначити кількість пар сусідніх елементів з однаковими значеннями. Усі такі пари записати до другого масиву.

13. Дано масив із N цілих чисел. Розмістити всі додатні елементи в лівій частині масиву, усі від'ємні — у правій, а нульові елементи — між ними.

14. Дано двовимірний масив розміру N×N із цілих чисел. Елементи головної діагоналі записати в одновимірний масив і впорядкувати його за спаданням.

15. Дано масив із N цілих чисел. Підрахувати кількість ділянок, що утворюють неперервні послідовності з неспадними значеннями. Найдовшу таку ділянку записати в інший масив.

Додаткові умови виконання завдання:

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

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. Чому треба звільняти пам'ять, яка динамічно виділялась?

Лабораторна робота №10
РЯДКИ ТИПУ CHAR*

Тема. Робота з рядками типу char*.

Мета: Набути практичних навичок щодо розроблення програм з використанням рядків.

Теми для попереднього опрацювання:

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

Загальні відомості

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

Оголосити змінну типу рядок можна трьома способами:

1. Оголосити масив символів фіксованої довжини (із місцем під завершальний нульовий символ ‘\0’). Наприклад, для рядка довжиною до 40 символів:

char s[40 + 1];

2. Оголосити масив і ініціалізувати його рядковим літералом (довжину масиву обчислює компілятор):

char s [] = "Приклад ініціалізації рядка";

Під час ініціалізації в кінець рядка автоматично додається нульовий символ ‘\0’. У цьому варіанті створюється масив, що містить копію рядкового літерала, тому елементи такого рядка можна змінювати.

3. Оголосити вказівник на символ і ініціалізувати його адресою рядкового літерала:

*char *s = "Другий варіант ініціалізації";*

У цьому випадку змінна s є вказівником, який зберігає адресу першого символу рядкового літерала. Рядковий літерал має статичну тривалість зберігання та не призначений для зміни, тому використовується тип const char. Поширена помилка полягає в ототожненні “рядка” з “вказівником”: вказівник лише вказує на послідовність символів і не виділяє пам’ять для змінюваного рядка.

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.

Загальне завдання

1. В кінці кожної програми вивести в консоль ім’я автора розробника коду.

2. Текст задавати рядковою константою, якщо інше не передбачено умовою завдання.

Індивідуальне завдання

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. «Зашифрувати» заданий текст, для чого в кожному рядку тексту поміняти місцями перший символ з останнім, другий – з передостаннім і т. д. Виконати дешифрування.

Додаткові вимоги виконання завдання

- звіт має бути виконаний згідно з вимогами до оформлення робіт;

Контрольні питання.

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?

Лабораторна робота №11
РОБОТА З ФАЙЛАМИ

Тема. Основи роботи з файлами.

Мета: Набути практичних навичок розроблення програм із використанням файлів для зчитування та збереження даних.

Теми для попереднього опрацювання:

- файли;
- рядки;
- покажчики.

Загальні відомості

Файл – це іменована галузь даних на будь-якому носії інформації. Для програміста відкритий файл представляється як послідовність зчитуваних або записуваних даних. При відкритті файлу з ним зв’язується потік введення- виведення. Інформація, що виводиться, записується в потік, а інформація, що вводиться, зчитується з потоку.

Коли потік відкривається для введення-виведення, він зв’язується зі стандартною структурою типу *FILE*, яка визначена у файлі *stdio.h*.

Структура *FILE* містить необхідну інформацію про файл.

Робота з файлами розподіляється на три етапи:

Відкриття файлу. Функція fopen() повертає покажчик на потік типу FILE*. У разі помилки повертається NULL.

FILE *fopen(name, type);

де *name* – ім’я файлу, що відкривається (включаючи шлях),

type – покажчик на рядок символів, що визначають спосіб доступу до файлу:

"r" – відкрити файл для читання (файл повинен існувати);

"w" – відкрити порожній файл для запису; якщо файл існує, то його вміст губиться;

"a" – відкрити файл для запису в кінець (для додавання); файл створюється, якщо він не існує;

"r+" – відкрити файл для читання і запису (файл повинен існувати);

"w+" – відкрити порожній файл для читання і запису; якщо файл існує, то його вміст губиться;

"a+" – відкрити файл для читання і доповнення, якщо файл не існує, то він створюється.

Значення, що вертається, – покажчик на відкритий потік. Якщо виявлена помилка, то вертається значення NULL.

Читання і запис даних.Для запису даних у файл використовують функції fprintf(), fputs(), fputc().

Для читання даних із файлу використовують функції fscanf(), fgets(), fgetc().

Закриття файлу. Після завершення роботи з файлом його потрібно закрити за допомогою функції fclose().

Стандартні функції роботи з файлами містяться в заголовному файлі stdio.h.

Основне завдання

У програму, що розроблена в попередній лабораторній роботі (робота з рядками), виконати наступні зміни:

- читання даних виконувати не з клавіатури, а з файлу.
- видача даних проводиться і у консоль і у окремий файл.

Додаткові вимоги виконання завдання:

- звіт має бути виконаний згідно з вимогами до оформлення робіт;

Контрольні запитання

1. Що таке файл?
2. Які існують функції неформатованого введення даних?
3. Які існують функції неформатованого виведення даних?
4. Як визначити розмір файлу?
5. Як виконувати читання даних з файлу, коли кількість їх невідома?
6. Чи можна форматувати дані при їх записі у файл?
7. Як ввести з файлу та записати у файл рядки?
8. Як ввести з файлу та записати у файл символ?
9. Порівняйте текстові та двійкові файли.

Лабораторна робота №12
РЕКУРСИВНІ ФУНКЦІЇ В МОВІ ПРОГРАМУВАННЯ C/C++

Тема. Основи роботи з рекурсивними функціями в мові програмування C/C++.

Мета: Набути практичних навичок щодо розроблення програм із використанням рекурсивних функцій та їх використанням на мові програмування C/C++. Розвиток навичок аналізу рекурсивних алгоритмів та їх реалізація в програмному коді.

Теми для попереднього опрацювання:

- функції;
- масиви;
- рядки;
- умовні оператори;
- цикли.

Загальні відомості

Рекурсія в програмуванні — це спосіб організації обчислень, за якого функція викликає саму себе прямо або опосередковано. Рекурсивні функції використовують для розв’язання задач, які можна поділити на подібні підзадачі меншого розміру. Важливим поняттям рекурсії є базовий випадок – умова, за якої рекурсивні виклики припиняються, що дає змогу уникнути нескінченного виконання функції.

Принцип роботи. Коли рекурсивна функція викликає сама себе, вона

робить це з новим набором параметрів, приближаючись до базового випадку. Кожен виклик функції зберігається в стеку викликів до тих пір, поки не буде досягнуто базового випадку. Після цього стек починає розгортатись, повертаючи контроль до попередніх рівнів виклику, аж до першого виклику функції.

Переваги рекурсії:

- спрощення програмного коду для окремих класів задач;
- природність розв'язання задач, які мають рекурсивну структуру;
- зручність реалізації деяких алгоритмів обробки дерев, пошуку, перебору та математичних обчислень.

Недоліки рекурсії:

- додаткові витрати пам'яті на зберігання стеку викликів;
- можливість переповнення стеку при великій глибині рекурсії;
- у деяких випадках менша ефективність порівняно з ітеративними алгоритмами.

Основні терміни:

Базовий випадок **–** умова , за якої рекурсія припиняється.

Рекурсивний випадок **–** частина функції, у якій виконується рекурсивний виклик із новими параметрами.

Стек викликів **–** структура даних, у якій зберігається інформація про активні виклики функцій і їх параметри.

Основне завдання

Написати програму на мові C/C++, яка використовує рекурсивну функцію для обчислення індивідуального завдання. Всі дані вводяться користувачем через консольний ввід, окрім елементів масиву які заповняються за допомогою генератора випадкових чисел (якщо це необхідно до індивідуального завдання).

Індивідуальне завдання

1. Рекурсивно обчислити факторіал числа.
2. Рекурсивно знайти максимальний елемент масиву.
3. Рекурсивно обчислити n-те число Фібоначчі.
4. Рекурсивно виконати обернення рядка.
5. Рекурсивно обчислити суму цифр цілого числа.
6. Рекурсивно вивести елементи масиву у зворотному порядку.
7. Рекурсивно обчислити степінь числа.

8. Рекурсивно перевірити, чи є слово паліндромом.
9. Рекурсивно обчислити суму цифр числа, що є степенем двійки.
10. Рекурсивно визначити, чи є масив впорядкованим за зростанням.
11. Рекурсивно визначити кількість нульових елементів у масиві.
12. Рекурсивно знайти найменше спільне кратне двох чисел.
13. Рекурсивно знайти мінімальний елемент масиву.
14. Рекурсивно обчислити суму елементів масиву.

Додаткові вимоги виконання завдання:

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

Контрольні запитання

1. Що таке рекурсія в програмуванні?
2. Як працює рекурсивна функція на прикладі обчислення факторіалу числа?
3. Які основні компоненти має рекурсивна функція?
4. Що таке базовий випадок і чому він є обов’язковим?
5. Як рекурсія впливає на стек викликів програми?
6. Що може статися при надмірній глибині рекурсії?
7. У чому полягають переваги і недоліки рекурсії порівняно з ітерацією?
8. Які підходи можна використати для перетворення рекурсивного алгоритму на ітеративний?

Лабораторна робота №13
ВИКОРИСТАННЯ ДИНАМІЧНИХ ЧИСЛОВИХ МАСИВІВ

Тема. Робота з динамічними числовими масивами в мові програмування C/C++.

Мета: Набути практичних навичок роботи з динамічними одновимірними та двовимірними масивами, використання функцій динамічного виділення пам’яті, звертання до елементів масиву за допомогою індексів і покажчиків, а також звільнення виділеної пам’яті після завершення роботи програми.

Теми для попереднього опрацювання:

- масиви;

- покажчики;
- функції;
- динамічне виділення пам’яті;
- двовимірні масиви.

Загальні відомості

Динамічний масив відрізняється від звичайного тим, що пам’ять під нього виділяється під час виконання програми, а кількість елементів може задаватися змінною. Це дає змогу створювати масиви, розмір яких визначається лише після запуску програми.

Для створення динамічних масивів у мові C/C++ можуть використовуватися функції malloc() і calloc(). Функція malloc() виділяє потрібний обсяг пам’яті без її ініціалізації, а функція calloc() додатково заповнює виділену пам’ять нульовими значеннями.

Синтаксис оголошення динамічного одновимірного масиву за допомогою malloc() має вигляд:

тип *ім’я = (тип*)malloc(sizeof(тип) * кількість_елементів);

Наприклад:

```
int N = 10;
float *a = (float*)malloc(sizeof(float) * N);
```

Оголошення динамічного одновимірного масиву за допомогою calloc():

```
int N = 10;
float *a = (float*)calloc(N, sizeof(float));
```

Після завершення роботи з динамічним масивом виділену пам’ять потрібно звільнити за допомогою функції free():

```
free(a);
```

До елементів динамічного одновимірного масиву можна звертатися як за допомогою індексів, так і за допомогою покажчиків. Наприклад, запис a[i] еквівалентний запису *(a + i).

Двовимірний динамічний масив можна організувати двома способами. Перший спосіб полягає у виділенні пам’яті під один суцільний блок із m*n елементів. У такому разі масив фактично зберігається в пам’яті як одновимірний, а доступ до елемента a[i][j] виконується через адресний вираз.

Наприклад, оголошення такого масиву:

```
float *a = (float*)malloc(m * n * sizeof(float));
```

або

```
float *a = (float*)calloc(m * n, sizeof(float));
```

Другий спосіб полягає у створенні масиву покажчиків на рядки. У цьому випадку спочатку виділяється пам’ять під масив покажчиків, а потім окремо під кожний рядок матриці:

```
float **a = (float**)malloc(m * sizeof(float*));
for (int i = 0; i < m; i++)
a[i] = (float*)malloc(n * sizeof(float));
```

Перевагою такого підходу є можливість звертатися до елементів у звичному вигляді – a[i][j].

Отже, використання динамічних масивів дає змогу гнучко працювати з даними змінного розміру, однак вимагає обов’язкового контролю за коректним виділенням і звільненням пам’яті.

Основне завдання

Розробити програму мовою C/C++, у якій використовується динамічний числовий масив для розв’язання індивідуального завдання.

Під час виконання роботи необхідно:

- організувати динамічне виділення пам’яті під масив або матрицю;
- реалізувати введення, оброблення та виведення даних;
- використати звертання до елементів масиву за індексами або за допомогою покажчиків;
- після завершення роботи програми звільнити всю динамічно виділену пам’ять.

Індивідуальне завдання

1. Заповнити матрицю випадковими числами. Знайти суму всіх елементів матриці.
2. Заповнити матрицю випадковими числами. Знайти середнє арифметичне всіх елементів матриці.
3. Заповнити матрицю випадковими числами. Визначити кількість додатних, від’ємних і нульових елементів.
4. Заповнити матрицю випадковими числами. Знайти найбільший елемент матриці та його індекси.
5. Заповнити матрицю випадковими числами. Знайти найменший елемент матриці та його індекси.
6. Заповнити матрицю випадковими числами. Обчислити суму елементів кожного рядка.
7. Заповнити матрицю випадковими числами. Обчислити суму

елементів кожного стовпця.

8. Заповнити матрицю випадковими числами. Знайти добуток елементів кожного рядка.

9. Заповнити матрицю випадковими числами. Знайти кількість парних елементів у кожному рядку.

10. Заповнити матрицю випадковими числами. Знайти кількість непарних елементів у кожному стовпці.

11. Заповнити матрицю випадковими числами. Обчислити суму елементів головної діагоналі.

12. Заповнити матрицю випадковими числами. Обчислити суму елементів бічної діагоналі.

13. Заповнити матрицю випадковими числами. Знайти найбільший елемент у кожному рядку.

14. Заповнити матрицю випадковими числами. Знайти найменший елемент у кожному стовпці.

15. Заповнити матрицю випадковими числами. Поміняти місцями перший і останній рядки матриці.

16. Заповнити матрицю випадковими числами. Поміняти місцями перший і останній стовпці матриці.

17. Заповнити матрицю випадковими числами. Вивести елементи матриці у зворотному порядку за рядками.

18. Заповнити матрицю випадковими числами. Замінити всі від’ємні елементи на нулі.

19. Заповнити матрицю випадковими числами. Замінити всі парні елементи на їх квадрати.

20. Заповнити матрицю випадковими числами. Для кожного рядка знайти середнє арифметичне його елементів.

21. Заповнити матрицю випадковими числами. Для кожного стовпця знайти середнє арифметичне його елементів.

22. Заповнити матрицю випадковими числами. Транспонувати матрицю.

23. Заповнити матрицю випадковими числами. Відобразити матрицю симетрично відносно вертикальної осі.

24. Заповнити матрицю випадковими числами. Відобразити матрицю симетрично відносно горизонтальної осі.

Додаткові вимоги виконання завдання:

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

Контрольні запитання

1. Для чого використовуються динамічні масиви?
2. За допомогою яких функцій можна створити динамічний масив?
3. За допомогою якої функції звільняється пам’ять, виділена під динамічний масив?
4. Яким чином можна звертатися до елементів динамічного масиву?
5. Що таке адресний вираз під час звертання до елементів масиву?
6. У чому полягає різниця між одновимірним і двовимірним динамічним масивом?
7. Які способи організації двовимірного динамічного масиву існують?
8. У якому порядку потрібно звільняти пам’ять, виділену під двовимірний динамічний масив?

Лабораторна робота №14
РОЗРОБКА БАГАТОФАЙЛОВИХ ПРОЄКТІВ

Тема. Розробка багатофайлових проєктів у мові програмування C/C++.

Мета: Набути практичних навичок розроблення багатофайлових програм, розподілу коду між кількома файлами, використання заголовних файлів, оголошення та визначення функцій у різних модулях, а також організації структури проєкту в середовищі розробки.

Теми для попереднього опрацювання:

- функції;
- масиви;
- рядки;
- покажчики;
- динамічні масиви;
- робота з файлами.

Загальні відомості

У процесі розроблення програм невеликі навчальні приклади часто розміщують в одному файлі. Проте зі збільшенням обсягу програмного коду та кількості функцій такий підхід стає незручним. Великі програми, як правило, поділяють на окремі модулі, кожен з яких відповідає за виконання певної частини задачі. Такий підхід називають **багатофайловою організацією проєкту**.

Під час розроблення багатофайлових проєктів програмний код поділяють на кілька типів файлів:

- **файли з вихідним кодом** (.c, .cpp) — містять визначення функцій і реалізацію алгоритмів;
- **заголовні файли** (.h) — містять оголошення функцій, констант, структур, макросів і типів даних, які використовуються в кількох файлах проєкту.

Поділ програми на кілька файлів дає змогу:

- зробити структуру програми зрозумілішою;
- спростити відлагодження та супровід коду;
- повторно використовувати окремі модулі;
- організувати командну розробку, коли різні частини проєкту створюються окремо.

Заголовний файл підключається до інших файлів за допомогою директиви препроцесора #include. Наприклад:

#include "functions.h"

У заголовному файлі зазвичай записують лише оголошення функцій, а їх реалізацію розміщують в окремому файлі вихідного коду. Наприклад:

Файл functions.h

```
int sum(int a, int b);
int max(int a, int b);
```

Файл functions.cpp

```
int sum(int a, int b)
{
   return a + b;
}

int max(int a, int b)
{
   return (a > b) ? a : b;
}
```

Файл main.cpp

```
#include <iostream>
#include "functions.h"

int main()
{
   int x = 5, y = 8;
   std::cout << "Сума = " << sum(x, y) << std::endl;
   std::cout << "Максимум = " << max(x, y) << std::endl;
   return 0;
}
```

Під час розроблення багатофайлового проєкту важливо дотримуватися таких правил:

- у заголовних файлах розміщувати лише оголошення;
- у файлах реалізації розміщувати визначення функцій;
- імена файлів і функцій мають бути змістовними;
- структура проєкту має бути логічною та зрозумілою.

Основне завдання

Розробити багатофайловий проєкт мовою C/C++, який складається щонайменше з трьох файлів:

- головного файла програми;
- заголовного файла;
- файла реалізації функцій.

Індивідуальне завдання

1. Розробити програму для обчислення суми, різниці, добутку і частки двох чисел. Кожну операцію оформити окремою функцією.

2. Розробити програму для знаходження мінімального, максимального і середнього арифметичного трьох чисел. Кожну дію оформити окремою функцією.

3. Розробити програму для обчислення площі квадрата, прямокутника і кола. Для кожної фігури створити окрему функцію.

4. Розробити програму для перевірки, чи є число парним, додатним і кратним 5. Кожну перевірку оформити окремою функцією.

5. Розробити програму для роботи з одновимірним масивом: знаходження суми елементів, найбільшого елемента та кількості парних елементів.

6. Розробити програму для роботи з одновимірним масивом:

знаходження найменшого елемента, середнього арифметичного та кількості від’ємних елементів.

7. Розробити програму для роботи з рядком: визначення довжини рядка, кількості голосних літер і кількості пробілів.

8. Розробити програму для роботи з рядком: перевірки, чи є рядок паліндромом, та обчислення кількості слів у рядку.

9. Розробити програму для роботи з матрицею: знаходження суми елементів головної діагоналі, найбільшого елемента та кількості нульових елементів.

10. Розробити програму для роботи з матрицею: знаходження суми елементів кожного рядка та кожного стовпця.

11. Розробити програму для переведення температури між шкалами Цельсія, Фаренгейта та Кельвіна. Кожне перетворення оформити окремою функцією.

12. Розробити програму для обчислення факторіалу числа, n-го числа Фібоначчі та суми цифр числа. Кожне завдання оформити окремою функцією.

13. Розробити програму для сортування одновимірного масиву за зростанням і за спаданням. Кожен варіант сортування оформити окремою функцією.

14. Розробити програму для роботи з файлами: запис масиву у файл, зчитування масиву з файлу та виведення його на екран. Кожну дію оформити окремою функцією.

15. Розробити програму для обчислення периметра і площі трикутника за заданими сторонами. Окремо реалізувати перевірку існування трикутника.

Додаткові вимоги виконання завдання

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

Контрольні запитання

1. Що таке багатофайловий проєкт?
2. Для чого програму поділяють на кілька файлів?
3. Яке призначення мають заголовні файли?
4. Яка різниця між оголошенням і визначенням функції?
5. Для чого використовується директива #include?
6. Що таке захист від повторного включення заголовного файла?
7. Які файли зазвичай входять до складу багатофайлового проєкту?
8. Які переваги має багатофайлова організація програми?

Лабораторна робота №15

РОБОТА З ФОРМАМИ І ГРАФІЧНИМ ІНТЕРФЕЙСОМ

Тема. Розроблення програм із використанням форм і елементів графічного інтерфейсу користувача.

Мета: Набути практичних навичок створення віконних програм, роботи з формами, розміщення елементів керування на формі, оброблення подій користувача та організації простого графічного інтерфейсу.

Теми для попереднього опрацювання:

- змінні та типи даних;
- умовні оператори;
- цикли;
- функції;
- основи роботи в середовищі розробки Microsoft Visual Studio.

Загальні відомості

Графічний інтерфейс користувача – це сукупність візуальних елементів, за допомогою яких користувач взаємодіє з програмою. До таких елементів належать вікна, кнопки, текстові поля, написи, списки, прапорці, перемикачі та інші компоненти.

Основним елементом віконної програми є **форма**. Форма являє собою вікно, на якому розміщують елементи керування. Саме через форму користувач вводить дані, запускає обчислення, отримує результати та керує роботою програми.

До основних елементів графічного інтерфейсу належать:

- **Label** — текстовий напис;
- **TextBox** — поле для введення тексту;
- **Button** — кнопка для виконання дії;
- **CheckBox** — прапорець для вибору параметра;
- **RadioButton** — перемикач для вибору одного з кількох варіантів;
- **ListBox** або **ComboBox** — список для вибору значень;
- **PictureBox** — область для відображення зображення;
- **Form** — головне вікно програми.

Кожний елемент графічного інтерфейсу має властивості, методи та події.

Властивості визначають зовнішній вигляд і стан елемента. Наприклад:

- Text — текст, що відображається;
- Name — ім’я елемента;
- Size — розмір;
- Location — розташування;
- Visible — видимість елемента;
- Enabled — доступність елемента для взаємодії.

Методи задають дії, які виконує елемент.

Події виникають у відповідь на дії користувача або зміну стану програми. Наприклад:

- натискання кнопки;
- зміна тексту в полі введення;
- вибір елемента зі списку;
- завантаження форми.

Однією з найпоширеніших є подія натискання кнопки. У програмі для неї створюють спеціальний обробник події, у якому записують код, що має виконуватися після натискання кнопки.

Використання графічного інтерфейсу дає змогу:

- зробити програму зручнішою для користувача;
- спростити введення та виведення даних;
- зробити результати роботи програми більш наочними;
- підвищити зручність керування програмою.

Основне завдання

Розробити віконну програму з графічним інтерфейсом користувача. Програма повинна містити форму та елементи керування, необхідні для введення даних, виконання обчислень або оброблення інформації, а також для виведення результату.

Під час виконання роботи необхідно:

- створити форму;
- розмістити на формі необхідні елементи керування;
- налаштувати їх основні властивості;
- реалізувати оброблення подій користувача;
- забезпечити коректне виведення результатів у графічному інтерфейсі.

Індивідуальне завдання

1. Розробити форму для обчислення суми двох чисел.

2. Розробити форму для обчислення площі прямокутника за довжинами сторін.

3. Розробити форму для обчислення площі круга за заданим радіусом.

4. Розробити форму для переведення температури з градусів Цельсія у градуси Фаренгейта.

5. Розробити форму для перевірки, чи є введене число парним.

6. Розробити форму для знаходження більшого з двох введених чисел.

7. Розробити форму для обчислення факторіалу введеного числа.

8. Розробити форму для визначення кількості символів у введеному рядку.

9. Розробити форму для перевірки, чи є введене слово паліндромом.

10. Розробити форму для обчислення середнього арифметичного трьох чисел.

11. Розробити форму для обчислення периметра і площі квадрата за довжиною сторони.

12. Розробити форму для знаходження суми цифр введеного цілого числа.

13. Розробити форму для переведення довжини з сантиметрів у метри та міліметри.

14. Розробити форму для визначення, чи є введене число додатним, від'ємним або нулем.

15. Розробити форму для обчислення вартості покупки за ціною товару та його кількістю.

16. Розробити форму для знаходження мінімального з трьох введених чисел.

17. Розробити форму для визначення кількості голосних літер у введеному слові.

18. Розробити форму для перевірки правильності введення електронної адреси за найпростішими ознаками.

19. Розробити форму для зміни регістру введеного тексту.

20. Розробити форму для обчислення добутку елементів заданого масиву чисел, введених через пробіл. Розробити програму для обчислення суми, різниці, добутку і частки двох чисел. Кожну операцію оформити окремою функцією.

Додаткові вимоги виконання завдання:

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

Контрольні питання

1. Що таке графічний інтерфейс користувача?
2. Що таке форма у віконній програмі?
3. Які основні елементи керування використовують у графічному інтерфейсі?
4. Що таке властивості елемента керування?
5. Що таке подія в графічному інтерфейсі?
6. Для чого використовується обробник події?
7. Яке призначення мають елементи Label, TextBox і Button?
8. Як відрізняються прапорець і перемикач?
9. Як у програмі можна перевірити правильність введених даних?
10. Які переваги має графічний інтерфейс порівняно з консольним?

Список літератури

1. Stroustrup, B. *Programming: Principles and Practice Using C++*. 3rd ed. Addison-Wesley Professional, 2024.
2. Stroustrup, B. *A Tour of C++*. 3rd ed. Addison-Wesley Professional, 2022.
3. Davidson, J. Guy, Gregory, K. *Beautiful C++: 30 Core Guidelines for Writing Clean, Safe, and Fast Code*. Addison-Wesley Professional, 2021.
4. Roy, P. *C++ Memory Management: Write Leaner and Safer C++ Code Using Proven Memory-Management Techniques*. Packt, 2025. – 442 p.
5. Bolboaca, A., Deák, F.-L. *Debunking C++ Myths: Embark on an Insightful Journey to Uncover the Truths Behind Popular C++ Myths and Misconceptions*. Packt, 2024. – 226 p.
6. Іванов, Є.О., Ліндер, Я.М., Жереб, К.А. *Основи мови програмування С++: навчальний посібник*. – Київ: Логос, 2020. – 90 с.
7. Зеленський, О.С., Лисенко, В.С. *Основи програмування на С++: навчальний посібник*. – Кривий Ріг: ДУЕТ, 2023. – 269 с.
8. 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 ed. Packt Publishing, 2020. – 606 p.
9. Бібліотека КНУБА. URL: http://library.knuba.edu.ua/.
10. Освітній сайт КНУБА. URL: http://org.knuba.edu.ua/.

Навчально-методичне видання

ПРОГРАМУВАННЯ ТА АЛГОРИТМІЧНІ МОВИ

Методичні вказівки
до виконання лабораторних робіт 1–15
для здобувачів першого (бакалаврського) рівня вищої освіти
спеціальностей F3 «Комп'ютерні науки»
та F6 «Інформаційні системи і технології»

Укладачі: ПОПЛАВСЬКИЙ Олександр Анатолійович,
БОСЕНКО Ігор Валерійович
ПОРОХОВНІЧЕНКО Ірина Анатоліївна

Комп'ютерне верстання *А. П. Селівестрової*

Ум. друк. арк. 3,95. Обл.-вид. арк. 4,25
Електронний документ. Вид № 39/V-26.

Виконавець 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