МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування
імені адмірала Макарова

С. Б. ПРИХОДЬКО, Л. М. МАКАРОВА

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни "Основи програмування"

У двох частинах

Частина 1

Рекомендовано Методичною радою НУК

Електронне видання
комбінованого використання на DVD-ROM

МИКОЛАЇВ • НУК • 2016

УДК 004.4(076)
ББК 32.973.2я73
П 77

Автори:
С. Б. Приходько, д-р техн. наук, професор;
Л. М. Макарова, канд. техн. наук

Рецензент І. І. Коваленко, д-р техн. наук, професор

Приходько С. Б.

П 77 Методичні вказівки до виконання лабораторних робіт з дисципліни "Основи програмування" : у 2 ч. Частина 1 / С. Б. Приходько, Л. М. Макарова. – Миколаїв : НУК, 2016. – 42 с.

Уміщено завдання для лабораторних робіт, стислий виклад теоретичних відомостей, етапи виконання робіт і контрольні завдання.

Призначено для студентів першого курсу спеціальності 121 "Інженерія програмного забезпечення" (попередня назва "Програмне забезпечення систем"), а також для усіх, хто вивчає основи програмування мовою С++.

УДК 004.4(076)
ББК 32.973.2я73

Навчальне видання

ПРИХОДЬКО Сергій Борисович
МАКАРОВА Лідія Миколаївна

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни "Основи програмування"

У двох частинах

Частина 1

Комп'ютерне складання та верстання *А. Й. Лихіна*
Коректор *М. О. Паненко*

Формат 60×84/16. Ум. друк. арк. 2,4. Об'єм даних 6986 кб. Тираж 15 прим.
Вид. № 3. Зам. № 35.

Видавець і виготівник Національний університет кораблебудування імені адмірала Макарова
просп. Героїв Сталінграда, 9, м. Миколаїв, 54025, e-mail : publishing@nuos.edu.ua

Свідоцтво суб'єкта видавничої справи ДК № 2506 від 25.05.2006 р.

ВСТУП

У Національному університеті кораблебудування імені адмірала Макарова (НУК) дисципліна "Основи програмування" вивчається студентами спеціальності 121 "Інженерія програмного забезпечення" (попередня назва "Програмне забезпечення систем") у першому – третьому семестрах. Вона відноситься до циклу професійної та практичної підготовки бакалаврів.

Ці методичні вказівки мають допомогти студентам першого курсу спеціальності 121 "Інженерія програмного забезпечення" у підготовці та виконанні лабораторних робіт з дисципліни "Основи програмування". Вони містять завдання для лабораторних робіт, стислий виклад теоретичних відомостей, етапи виконання робіт та контрольні запитання.

При виконанні кожної лабораторної роботи рекомендується:

- засвоїти теоретичні відомості;
- розібратися з завданням та етапами виконання роботи;
- виконати запропоновані завдання та оформити звіт з роботи;
- перевірити повноту розуміння теми за допомогою контрольних запитань, розташованих у кінці кожної роботи.

Лабораторну роботу необхідно виконувати відповідно до наведених етапів виконання роботи. Звіт про лабораторну роботу оформлюється кожним студентом індивідуально. Звіт повинен містити: назву і мету роботи; завдання (постановку задачі); методику й алгоритм розв'язання задачі; текст програми; результати роботи; аналіз результатів та висновки.

На вступному занятті студенти знайомляться з роботою в NetBeans IDE – інтегрованому середовищі розробки (Integrated Development Environment – IDE) додатків, з наступного заняття починається виконання лабораторних робіт.

Робота № 1

Розробка та реалізація програми з лінійною структурою

Мета роботи: закріплення знань алфавіту мови програмування С++; набуття навичок запису його констант, змінних, виразів, операторів присвоєння; оволодіння навичками складання програми з лінійною структурою та виконання її в NetBeans IDE.

Завдання

Завдання 1.1. Записати мовою С++ математичні вирази за варіантами, які наведені в таблиці 1.1.

Таблиця 1.1 – Варіанти завдання 1.1

№	Математичні вирази	№	Математичні вирази
1–3	а) $3(z+1)^2+2,1\cdot10^6$ б) $\|x+z\|>0 \wedge 0<b<1$	4–6	а) $10^{-4}\, e^{-2f}+\|z^3\|$ б) $x+z<0 \vee 0<f<2$
7–9	а) $3(z+3)^{0,5}+10^{-3}$ б) $\|x+z\|>1 \wedge 1<b<2$	10–12	а) $\ln(z+1)^{0,5}+4,2\cdot10^4$ б) $\|x\|>2 \vee 0<b<3$
13–15	а) $5(z+1)^5+10^6$ б) $0<b<1 \vee 0<f<0,5$	16–18	а) $6(z+1)^6+1,6\cdot10^6$ б) $\cos\|x+z\|>0 \vee 0<b<6$
19–21	а) $7(z+1)^{0,5}+2,7\cdot10^6$ б) $\|x+z\|>0 \wedge 0<b<7$	22–24	а) $10^{-7}\sin\|3z\|+b^{0,5}$ б) $\ln\|x+z\|>0 \vee 0<b<1$
25–27	а) $\|z+1\|^3+2\cdot10^5$ б) $\ln\|x+z\|>0 \wedge b>9$	28–30	а) $(z+1)^{0,5}+10^6$ б) $\|x+z\|>0 \vee 0<b<1$

Прийняті позначення: $\wedge$ – об'єднання множин, $\vee$ – переріз множин.

Завдання 1.2. Представити математичний запис виразу за варіантами, які наведені в таблиці 1.2, і показати порядок дій.

Таблиця 1.2 – Варіанти завдання 1.2

№	Вираз
1	`x+2./3./x/a+sin(x)/2*sqrt(x)+1.0e-6*pow(x,1./7.)`
2	`(x+7)/3.*x+3*sqrt(x)/2./x+1.0e7-1./3.*pow(x,5)`
3	`x+2./3.*x/a+cos(x)/2./sqrt(x)+1.0e-5*pow(x,7)`
4	`(x+4)/3./x+sqrt(abs(x))/2.*x+1.0e-6*pow(x,1./3.)`
5	`x+2./3./x/a+sqrt(x)/2./sin(x)+1.0e5*pow(x/3.,2./7.)`
6	`1.4e-4*pow(2*x,3)+sqrt(sin(x))/2.+cos(x)/2./x`
7	`abs(cos(x))/2./x-5./7.*x/a/1.0e-6*pow(x/2.,1./7.)`

Продовж. табл. 1.2

№	Вираз
8	`x+2./3./x*a+sqrt(sin(x))/2/x+1.0e-3*pow(x/7.,2./3.)`
9	`(x+7)/3.*x+3*sqrt(x)/2./x+1.0e7-4*pow(x,b)`
10	`sqrt(cos(x*x))/2./x-5./7.*x/a/1.0e-6*pow(x,1./8.)`
11	`x+9./(3*x/a)+cos(x)/2./sqrt(x)+1.0e-5*pow(x,9)`
12	`x+4./3./x+exp(abs(x))/2.*x+1.0e-4*pow(x,1./3.)`
13	`sqrt(x)/2./(x-5./7.)*x/a/1.0e-6*pow(x/2.,1./9.)/11`
14	`x+2*3./x*a+sin(x)/(2*x+1.0e-3*pow(x,5./7.)`
15	`x+4./3./(x+abs(x))/2.*x+1.0e-5*pow(x,5./3.)`

Завдання 1.3. Скласти програму обчислення наступних величин за варіантами, які наведені в таблиці 1.3, та виконати її в NetBeans IDE.

Таблиця 1.3 – Варіанти завдання 1.3

№	Величини, які треба обчислити
1	Модуль вектора $5\boldsymbol{a}+10\boldsymbol{b}$, якщо $\boldsymbol{a}=\{3; 2\}$ і $\boldsymbol{b}=\{0; -1\}$
2	Проекція вектора $\boldsymbol{a}=\{5; 2; 5\}$ на вісь вектора $\boldsymbol{b}=\{2; -1; 2\}$
3	Периметр трикутника з вершинами A(1; 1), B(4; 1), C(4; 5)
4	Модуль вектора $-2\boldsymbol{a}+4\boldsymbol{b}$, якщо $\boldsymbol{a}=\{3; 2\}$ і $\boldsymbol{b}=\{0; -1\}$
5	Кути трикутника з вершинами A(0; 1,7), B(2; 1,7), C(1,5; 0,85)
6	Площа чотирикутника з вершинами A(0;0), B(−1;3), C(2;4), D(3;1)
7	Модуль вектора $\boldsymbol{a}+\boldsymbol{b}$, якщо $\lvert\boldsymbol{a}\rvert=11$, $\lvert\boldsymbol{b}\rvert=23$, $\lvert\boldsymbol{a}-\boldsymbol{b}\rvert=30$
8	Модуль вектора $\boldsymbol{a}-\boldsymbol{b}$, якщо $\lvert\boldsymbol{a}\rvert=3$, $\lvert\boldsymbol{b}\rvert=5$, та $\boldsymbol{a}$ і $\boldsymbol{b}$ утворюють кут 120°
9	Кут між векторами $\boldsymbol{a}=\{2; -4; 4\}$ і $\boldsymbol{b}=\{-3; 2; 6\}$
10	Модуль вектора $\boldsymbol{a}+\boldsymbol{b}$, якщо $\boldsymbol{a}=\{1; 2; 3\}$ і $\boldsymbol{b}=\{4; -2; 9\}$

Короткі теоретичні відомості

Алфавіт мови C++ складається з сукупності символів, які наведені у додатку А. За допомогою символів алфавіту можна складати різноманітні конструкції: константи, змінні, оператори, тощо. Управляючі символи мови C++, або escape-послідовності, наведені у додатку Б.

Стала величина (константа) не змінюється в процесі роботи програми. Є декілька видів констант: числові (цілі та з плаваючою точкою), символьні, логічні.

Цілі десяткові константи складаються з послідовності десяткових цифр, перед якою немає цифри `0`, а може стояти знак "+" або "–". Приклад: `-15`.

Цілі вісімкові константи складаються з послідовності вісімкових цифр, перед якою є цифра `0` та може стояти знак "+" або "–". Приклад: `-015`.

У мові С++ цілі константи можна зображати і в шістнадцятковій системі числення. В цьому випадку константи починаються з `0x` або `0X`, за якими йдуть шістнадцяткові цифри. Перед знаком 0x або 0X може бути знак "+" або " ". Приклад: `0x2A`.

Цілі десяткова, вісімкова та шістнадцяткова константа, значення якої перевищує найбільше машинне ціле зі знаком, є довгою константою (**`long`**); в інших випадках цілі константи вважаються **`int`**.

Цілі десяткова, вісімкова та шістнадцяткова константа, за якою безпосередньо стоїть буква `l` або `L`, є довгою константою (**`long`**).

Константа з плаваючою крапкою складається з цілої частини, десяткової точки, дробової частини, букви `e` або `E` та цілого показника степеня. Ціла або дробова частина (але не обидві одразу) може бути відсутньою. Десяткова точка, або `e` (`E`), сумісно з цілим показником степеня (але не обидві частини одночасно) може бути відсутньою. Наприклад, число 0,015 можна записати як `0.015` та `0.15E-01`, або `1.5E-02`. Константа з плаваючою точкою має тип **`double`**.

Символьна константа – це символ в лапках. Приклад: `'f'`. Символьні константи вважаються даними типу **`int`**.

Логічні константи приймають тільки два значення: **`true`** (істина) та **`false`** (хибність).

Коментар – це текст, обмежений символами `/*` і `*/`, або текст, який починається символами `//` та закінчується наприкінці рядка. Коментарі `/* */` не можуть бути вкладеними. Символи `//` можна використовувати для того, щоб закоментувати символи `/*` або `*/`, а символами `/*` можна закоментувати `//`.

Змінна визначається ім'ям (ідентифікатором), типом та значенням.

Ідентифікатор є послідовністю букв і цифр, причому перший символ повинен бути буквою. Символ підкреслювання "_" вважається буквою. В ідентифікаторах великі та малі букви розрізнюються, букви кирилиці використовувати не можна. Не можна використовувати як ідентифікатори службові слова, список яких наведений у додатку В. Кожна змінна повинна бути описана, тобто необхідно визначити її тип.

Обробка даних виконується у виразах. Вираз складається з операцій і операндів. Операндом може бути константа, змінна або ім'я функції (математичні функції наведені у додатку Г). Обчислення

виконуються зліва направо з урахуванням пріоритету операцій. Операції за пріоритетом з описом їх позначень, назв та синтаксису наведені у додатку Д.

Відзначимо, що при виконанні операції ділення "/" цілого на ціле у результаті набуваємо цілу частину. Наприклад, `5/2` дорівнює `2`.

Префіксні операції розташовані зліва від операнда і виконують дію з ним до того, як його значення буде використане, а постфіксні операції виконуються після усіх обчислень.

Логічні операції `!`, `&&`, та `||` здійснюють дії над операндами відповідно до таблиці істинності (див. табл. 1.4). Наприклад, при k=1 результатом виразу `k > 0 && k < 4 є true`.

Таблиця 1.4 – Таблиця істинності логічних операцій

Операція	Дія	Вираз	A	B	Результат
!	Логічне НІ ¬	!A	true false		false true
&&	Логічне І ∧	A && B	true true false false	true false true false	true false false false
||	Логічне АБО ∨	A || B	true true false false	true false true false	true true true false

Програма мовою С++ складається з наступних блоків: блоку заголовків програми; блоку з оголошенням класів, прототипів і функцій; головного методу програми; блоку з описом функцій.

У блоці заголовків програми підключаються зовнішні файли, задається область імен, наводяться директиви препроцесору і вказівки компілятору.

Для підключення до програми зовнішніх файлів (стандартних бібліотек або користувацьких файлів) необхідно використовувати директиву `#include` із зазначенням імені файлу, що підключається. Директива `#include` включає в себе вміст файлу, шлях до якого заданий, в компільований файл замість рядка з директивою. Якщо шлях поміщений в кутові дужки, то пошук файлу здійснюється в стандартних директоріях. Якщо шлях поміщений в лапки і заданий повністю, то пошук файлу здійснюється в заданій директорії, а якщо шлях повністю не заданий – в поточній директорії.

Деякі стандартні бібліотеки наведені нижче:

`iostream` – бібліотека введення-виведення;

`cmath` – математична бібліотека;

`cstring` – бібліотека роботи з рядками.

Для завдання області імен використовується директива `using namespace` із зазначенням конкретної області імен. Для використання стандартної області імен необхідно використовувати конструкцію `using namespace std`;

Заголовки можуть змінюватися в залежності від компілятора, що використовується.

Головний метод програми має стандартну назву `main()`. Виконання програми починається з першого оператора цієї функції. Хоча `main()` і не є ключовим словом, ставитися до нього слід як до ключового, наприклад, не слід використовувати його як ім'я змінної.

Блок з описом функцій містить опис тих функцій, прототипи яких вказані в другому блоці.

Програма може складатися з декількох модулів (вхідних файлів).

Організація виведення даних в консоль виконується за допомогою команди `cout<<`, при цьому необхідно підключити бібліотеку `iostream` та простір імен, до якого належить команда `cout<<`: `using namespace std`. Рядок виводу записують у подвійні лапки. Організація введення даних з консолі виконується за допомогою команди `cin>>`.

Приклад виконання роботи

Завдання 1.1. Дані математичні вирази записати мовою С++:

а) $10^{-3}\cdot e^{-2x}+5(z+1)^3$ б) $|b|<0 \wedge f>1$.

Розв'язання

а) `1.0e-3*exp(-2*x)+5*pow(z+1, 3)`

б) `abs(b)<0 && f>1`

Завдання 1.2. Представити математичний запис виразу `0.51e 6*sqrt(3*x)+2*b*b` і показати порядок дій.

Розв'язання

$0{,}51\cdot 10^{-6}\cdot(3x)0{,}5+2\cdot b^2$ `0.51e 6*sqrt(3*x)+2*b*b`

Завдання 1.3. Скласти програму обчислення скалярного добутку двох векторів $\boldsymbol{a}$={4; 2; 4} і $\boldsymbol{b}$={6; 3; 2} та виконати її в NetBeans IDE.

Розв'язання

1. Постановка задачі

Скласти програму обчислення скалярного добутку двох векторів $\boldsymbol{a}$={4; 2; 4} і $\boldsymbol{b}$={6; 3; 2} на мові С++.

2. Методика розв'язання задачі

Скалярний добуток двох векторів обчислюється за формулою:

$$\boldsymbol{Ab} = A_1 \cdot B_1 + A_2 \cdot B_2 + A_3 \cdot B_3, \qquad (1.1)$$

де $A_1, B_1, A_2, B_2, A_3, B_3$ – відповідні координати векторів $\boldsymbol{a}$ і $\boldsymbol{b}$.

3. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Ввести координати векторів $\boldsymbol{a}$ і $\boldsymbol{b}$.

Дія 2. Обчислити скалярний добуток за формулою (1.1).

Дія 3. Вивести значення скалярного добутку двох векторів.

Представимо алгоритм розв'язання задачі на мові С++, позначивши змінні $A_1, B_1, A_2, B_2, A_3, B_3$ і $\boldsymbol{ab}$ відповідно як `a1, b1, a2, b2, a3, b3` і `ab` (усі типу **`double`**).

4. Текст програми

```
//Обчислення скалярного добутку двох векторів
#include <iostream>
#include <cmath>
using namespace std;
int main(){
   double a1, a2, a3, b1, b2, b3, ab;
   cout<<"Введіть                координати
a1,a2,a3,b1,b2,b3"<<endl;
   cin>>a1>>a2>>a3>>b1>>b2>>b3;
   ab=a1*b1+a2*b2+a3*b3;
   cout<<"Скалярний добуток ab="<<ab<<endl;
   }
```

5. Результати роботи програми

```
Введіть координати a1,a2,a3,b1,b2,b3
4
-2
-4
6
-3
2
Скалярний добуток ab=22
```

Контрольні питання

1. Які дані можна вживати в мові С++?
2. З якою метою використовують директиву `#include`?
3. Назвіть порядок виконання операцій у виразі.
4. Яка структура програми на мові С++?
5. Як працює оператор присвоєння?
6. Як у С++ здійснюється введення і виведення значень змінних?

Робота № 2

Розробка та реалізація програми з розгалуженою структурою

Мета роботи: оволодіння навичками складання програми з розгалуженою структурою за допомогою умовного оператора **`if`** або оператора вибору **`switch`** та виконання її в NetBeans IDE.

Завдання

Завдання 2.1. Представити математичний запис фрагмента програми за варіантами, які наведені в таблиці 2.1, і обчислити значення змінної `x` після його виконання. Вказівка: замість `n` підставити номер варіанта.

Таблиця 2.1 – Варіанти завдання 2.1

№	Фрагмент програми	№	Фрагмент програми
1–5	`t=n; x=t;` `if(t>1 && t<3)x=3;` `if(t<=1) x=0;`	6–10	`t=n; x=0;` `if(t<0)x=-t;` `else x=t;`
11–15	`a=n; b=13; c=12; x=a;` `if(x<b)x=b;` `if(x<c)x=c;`	16–20	`a=n; b=17; c=18; x=a;` `if(b<x)x=b;` `if(c<x)x=c;`
21–25	`t=n;` `switch(t){` `case 0: x=1; break;` `default: x=0;};`	26–30	`t=n;` `if(t>0 && t<28)x=10;` `else if(t>=28)x=20;` `else x=0;`

Завдання 2.2. Скласти програму обчислення значень функції за варіантами, які наведені в таблиці 2.2, та виконати її в NetBeans IDE.

Таблиця 2.2 – Варіанти завдання 2.2

№	Функція	№	Функція	№	Функція	№	Функція
1	$y = \mathrm{tg}x/\ln x$	2	$y = \ln x/\mathrm{tg}x$	3	$y = \arcsin(1/x)$	4	$y = \mathrm{ctg}\ \ln x$
5	$y = \mathrm{ctg}\,x^{1/3}$	6	$y = \mathrm{tg}x/(1-x)$	7	$y = \mathrm{tg}x/\ln^{2/3}x$	8	$y = x^{0,2}\mathrm{tg}x$
9	$y = \ln \mathrm{tg}\,x$	10	$y = \mathrm{tg}^{1/3}x/(x+1)$	11	$y = \mathrm{arcctg}^{1/3}x$	12	$y = \arccos x$
13	$y = \arcsin x$	14	$y = \mathrm{tg}x/(x-1)$	15	$y = \ln x/(1-x^2)$	16	$y = x/(1+\mathrm{tg}x)$
17	$y = \ln x/(1-x)$	18	$y = \mathrm{arcctg}\,x$	19	$y = \mathrm{tg}x/\ln x$	20	$y = x^{1/7}\mathrm{ctg}x$
21	$y = \arccos(1/x)$	22	$y = x^{1/3}/(1-x^2)$	23	$y = \arcsin^{1/3}x$	24	$y = \mathrm{arcctg}\,x^{1/3}$
25	$y = \mathrm{tg}^2\ln x$	26	$y = \mathrm{tg}^{1/3}x$	27	$y = \mathrm{tg}x/(1-x^2)$	28	$y = \ln \mathrm{tg}x^{1/3}$

Короткі теоретичні відомості

Умовний оператор **if** призначений для виконання або невиконання деякого оператора (простого або складеного) залежно від значення виразу.

Загальні вигляди умовного оператора **if**:

if(*вираз*) *оператор1*;

if(*вираз*) *оператор1* **else** *оператор2*;

Виконання умовного оператора **if** полягає в обчисленні логічного *виразу*. Якщо його значення **true** (не дорівнює нулю), то виконується *оператор1*. Якщо значення логічного виразу **false** (дорівнює нулю) і умовний оператор не містить слова **else**, то його виконання завершується. Якщо слово **else** є, то виконується *оператор2*.

Будьте уважні при записі логічного *виразу*: часто замість операції перевірки на рівність == вказується операція присвоєння =, що призводить до некоректної роботи програми. Службові слова **if** та **else** неподільні, при спробі вставити між ними будь-яку команду виникає помилка на етапі компіляції.

Якщо в якій-небудь гілці умовного оператора треба виконати кілька операторів, то їх слід об'єднати в складений оператор за допомогою операторних дужок {}. Один умовний оператор може входити в інший умовний оператор. При цьому кожне слово **else** відповідає останньому перед ним **if**. Так, після виконання наступного фрагмента програми:

```
int y, x=3;
if (x>0 && x<=1)y=1;
else if (x>1)y=10;
else y=0;
```

змінна y має значення 10.

Оператор вибору **switch** призначений для виконання одного з кількох можливих операторів у залежності від збігу значення виразу-селектора та значення константного виразу. Загальний вигляд оператора **switch**:

```
switch (вираз-селектор) {
case константний вираз 1 : оператор1; break;
        . . .
case константний вираз N : операторN; break;
default: оператор;
}
```

Вираз-селектор повинен бути цілого типу, типу **char** або типу вказівника. *Константний вираз* повинен мати такий самий тип, що

і *вираз-селектор*. В одному операторі **switch** ніякі *константні вирази* не можуть мати однакових значень. Може також бути не більше одного префіксу **default** або він може бути відсутній.

Виконання оператора **switch** починається з обчислення значення *виразу-селектора*. При першому збігові цього значення із значенням *константного виразу* (*1*,..., *N*) виконується відповідний *оператор*. Якщо жодного збігу не зафіксовано, а є слово **default**, то виконується оператор, наступний за **default**. У противному разі виконання оператора **switch** завершується.

Слід зазначити, що префікси **case** та **default** самі не змінюють потік управління. Для виходу з оператора **switch** застосовується оператор **break**.

Оператор **break** припиняє виконання оператора **switch**, в якому був виконан цей оператор. Управління передається на оператор, який стоїть одразу за цим оператором **switch**.

У разі, коли необхідно виконати однакові дії для різних значень *константних виразів*, можна записувати декілька виразів поспіль.

Приклад. Після виконання наступного фрагмента програми:

```
char letter= 'A';
switch (letter){
case 'A': cout<<"Case A"; break;
case 'B':
case 'C': cout<<"Case B or C"; break;
default: cout<<"Not A, B or C";
}
```

буде надруковано `Case A`

Приклад виконання роботи

Завдання 2.1. Представити математичний запис фрагмента програми

```
t=-8.;
if(t>0)x=pow(t,1./3.);
else if(t<0) x=-pow(abs(t),1./3.);
     else x=0;
```

і обчислити значення змінної x після його виконання.

Розв'язання

Цей фрагмент програми реалізує обчислення функції $x=t^{1/3}$. Після виконання цього фрагмента x=−2.

Завдання 2.2. Скласти програму обчислення значень функції y=ctg$x$ і виконати її в NetBeans IDE.

Розв'язання

1. Постановка задачі

Скласти програму обчислення значень функції y=ctg$x$ на мові C++.

2. Методика розв'язання задачі

Функція y=ctg$x$ обчислюється за формулою

$$y = \cos x/\sin x, \tag{2.1}$$

якщо

$$\sin x \neq 0. \tag{2.2}$$

3. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Ввести значення x.

Дія 2. Перевірити умову (2.2). Якщо умова істинна, то обчислити значення функції за формулою (2.1) і вивести його, інакше вивести повідомлення: "Функція не існує".

Остаточно представимо алгоритм розв'язання задачі на мові C++, позначивши змінні x і y відповідно як `x` і `y` (обидві типу **`double`**).

4. Текст програми

```
//програма обчислення функції y=ctg(x)
#include <iostream>
#include <cmath>
using namespace std;
int main(){
   double x,y;
   cout<<"Введіть x="; cin>>x;
   if(sin(x)!=0){
        y=cos(x)/sin(x);
        cout<<"y="<<y<<endl;
        }
   else cout<<"Функція не існує"<<endl;
 }
```

5. Результати роботи програми

```
Введіть x=1.57
y=0.000796327
```

Контрольні питання

1. Як працюють оператори **if** та **switch**?
2. Коли в операторі **if** застосовують складений оператор?
3. Як виконується умовний оператор **if**, якщо до нього входить інший умовний оператор **if**?
4. Для чого призначений оператор **switch**?
5. Якого типу може бути вираз-селектор в операторі **switch**?
6. Що робить оператор **break** у разі його виконання в операторі **switch**?

Робота № 3

Розробка та реалізація програми з циклічною структурою

Мета роботи: оволодіння навичками складання програми з циклічною структурою за допомогою операторів циклу **while**, **do while** і **for** та виконання її в NetBeans IDE.

Завдання

Завдання 3.1. Представити математичний запис фрагмента програми за варіантами, які наведені в таблиці 3.1, і обчислити значення змінної x після його виконання. Вказівка: замість n підставити номер варіанта.

Таблиця 3.1 – Варіанти завдання 3.1

№	Фрагмент програми	№	Фрагмент програми
1–5	`x=1;` `for(j=n; j>n; j--)` `  x=x*j;` `x=2*x;`	6–10	`x=0; j=1;` `do` `  x=x+j; j=j+2;` `while(j<=n);`
11–15	`x=0;` `for(j=1; j<=n; j++)` `  x=x+2;` `x=2*x;`	16–20	`x=1;` `while(x<=n)` `  x=x+1;` `x=2*x;`
21–25	`x=1; j=1; x1=n%5;` `do` `  x=x*x1; j=j+1;` `  if(j==4) break;` `while(j<=5);`	26–30	`x=n;` `for(k=1; k<=6; k++){` `  if(k<3) continue;` *x=x+1;* `};`

Завдання 3.2. Скласти програму табулювання функції з завдання 2.2 при зміні значення *x* від –1 до 1 з кроком 0,2 та виконати її в NetBeans IDE.

Короткі теоретичні відомості

Оператор циклу з передумовою **while** складається з ключового слова **while**, за яким йдуть *вираз* логічного типу у круглих дужках та виконуваний у циклі *оператор* (простий чи складений).

Загальний вигляд оператора циклу з передумовою:

while (*вираз*) *оператор – тіло циклу*;

Виконання оператора циклу з передумовою починається з обчислення значення *виразу*. Якщо це значення **false,** то *оператор*

не виконується, управління передається на оператор, який стоїть одразу за циклом. Якщо значення *виразу* **true**, *оператор* виконується, після чого знову обчислюється *вираз*. Щоб запобігти зациклюванню, слід передбачити зміну значення *виразу* всередині *тіла циклу*. Наприклад, після виконання наступного фрагмента програми:

```
bool a=true; int x=5;
while(a || x<9){  //цикл завершиться, як тільки
a=!a; x=x+2;  //вираз набуде значення false
}
```

змінна x має значення 11, а змінна a – 0 (**false**).

Оператор циклу з післяумовою **do while** складається з ключового слова **do**, за яким йде виконуваний у циклі *оператор* (простий чи складений); ключового слова **while** і *виразу* логічного типу.

Загальний вигляд оператора циклу з післяумовою:

do *оператор – тіло циклу* **while** (*вираз*);

Виконання цього оператора циклу відбувається так. Спочатку виконується *оператор*, а потім визначається значення *виразу* логічного типу. Якщо значення *виразу* **false**, то виконання циклу припиняється. Якщо це значення **true**, то виконується *оператор*, а потім знову обчислюється *вираз*. Наприклад, після виконання наступного фрагмента програми:

```
bool a=true; int x=5;
do {
a=!a; x=x+2;} //цикл завершиться, як тільки
while(a || x<9);  //вираз набуде значення false
```

змінна x має значення 11, а змінна a – 0 (**false**).

Треба підкреслити, що на відміну від циклу **while**, тіло циклу з післяумовою **do while** завжди виконується хоча б один раз.

Оператор циклу з параметром **for** складається з ключового слова **for**, за яким йдуть у круглих дужках *вираз1*, крапка з комою (**;**), *вираз2*, крапка з комою (**;**), *вираз3* і виконуваний у циклі *оператор* (простий або складений).

Загальний вигляд оператора циклу **for**:

for (*вираз1*; *вираз2*; *вираз3*) *оператор – тіло циклу*;

Вираз1 – це вираз, який описує ініціалізацію циклу; *вираз2* – перевірка умови завершення циклу; *вираз3* – це вираз, який обчислюється після кожної ітерації циклу. Кожний з *виразів1-3* може складатися з декількох виразів, які об'єднуються оператором кома (**,**). Жоден з *виразів1-3* не є обов'язковим.

Виконання оператора циклу **for** починається з обчислення *виразу1*. Потім обчислюється *вираз2*. Якщо значення *виразу2* **false**, то виконання циклу припиняється і управління передається на оператор, який стоїть одразу за циклом. Якщо це значення **true**, то послідовно виконуються *оператор* і *вираз3*, а потім знову обчислюється *вираз2*.

Оператора циклу **for** еквівалентний наступній послідовності операторів:

вираз1;
while (*вираз2*) {
оператор; вираз3;
}

Якщо немає *виразу2*, то вважається, що він має значення **true**. Оператор **for(;;);** представляє собою нескінченний цикл, який еквівалентний оператору **while(true);**.

Приклад. Після виконання наступного фрагмента програми:

```
bool a; int x;
for(a=true, x=5; a || x<9; a=!a, x+=2);
```

змінна x має значення 11, а змінна a – 0 (**false**).

Оператор **break** припиняє виконання оператора циклу (**while**, **do while** або **for**), в якому був виконаний цей оператор. Управління передається на оператор, який стоїть одразу за оператором циклу.

Оператор **continue** припиняє виконання поточної ітерації оператора циклу (**while**, **do while** або **for**), в якому був виконаний цей оператор, і здійснює перехід до виконання наступної ітерації циклу. Один оператор циклу може входити в інший оператор циклу.

Приклад виконання роботи

Завдання 3.1. Представити математичний запис фрагмента програми

```
for(x=1,j=1; j<5; j++, x*=j);
```

и обчислити значення змінної x після його виконання.

Розв'язання

Цей фрагмент програми реалізує обчислення x!=1 2 ... 5. Після виконання цього фрагмента x=120.

Завдання 3.2. Скласти програму табулювання функції $y = \mathrm{ctg}x$ при зміні значення x від $a=-1$ до $b=1$ з кроком $h=0{,}5$ і виконати її в NetBeans DE.

Розв'язання

1. Постановка задачі

Скласти програму табулювання функції $y = \mathrm{ctg}x$ при зміні значення x від $a=-1$ до $b=1$ з кроком $h=0{,}5$ на мові С++.

2. Методика розв'язання задачі

Методика розв'язання задачі збігається з методикою з прикладу завдання 2.2 (при оформленні роботи треба навести методику наново).

3. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Ввести значення a, b, h.

Дія 2. Присвоїти x значення a.

Дія 3. Повторювати наступні дії, поки $x \le b$:

Дія 3.1. Перевірити умову (2.2). Якщо умова істинна, то обчислити значення функції y за формулою (2.1) і вивести x та y, інакше вивести x та повідомлення: 'не існує';

Дія 3.2. Присвоїти x нове значення, яке дорівнює старому значенню x плюс крок h.

Запишемо алгоритм розв'язання задачі мовою С++, позначивши змінні x, y, a, b, h відповідно як x, y, a, b, h (усі типу **double**).

4. Текст програми

```
#include <iostream.h>
#include <cmath.h>
using namespace std;
void main(){
   double a,b,h,x,y;
   cout<<"Введіть a,b,h: "; cin>>a>>b>>h;
   for(x=a; x<=b; x+=h)
     if(sin(x)!=0){
     y=cos(x)/sin(x); cout<<"x="<<x<<" y="<<y<<endl;
   }
     else cout<<"x="<<x<<" y="<<" не існує"<<endl;
   }
```

5. Результати роботи програми

```
Введіть a,b,h: -1
1
0.5
x=-1 y=-0.642093
x=-0.5 y=-1.83049
x=0 y= не існує
x=0.5 y=1.83049
x=1 y=0.642093
```

Контрольні питання

1. Як працюють оператори циклу **while**, **do while** і **for**?
2. Чим цикл **do while** відрізняється від циклу **while**?
3. Коли у циклах **while** та **for** застосовують складений оператор?
4. Які обов'язкові елементи присутні у циклі **for**?
5. Якій послідовності операторів еквівалентний оператор циклу **for**?
6. Як працюють оператори **break** та **continue**?

Робота № 4

Розробка та реалізація програми з масивами

Мета роботи: оволодіння навичками складання програми з масивами та виконання її в NetBeans IDE.

Завдання

Завдання 4.1. Визначити дію фрагмента програми за варіантами, які наведені в таблиці 4.1, якщо `char sName[20]="Hello World!"; char *p=sName; int n,i;` (Вказівка: n дорівнює номеру варіанта).

Таблиця 4.1 – Варіанти завдання 4.1

№	Фрагмент програми	№	Фрагмент програми
1–5	`for(p+=n; *p; p++) cout<<*p<<" ";`	6–10	`for(p+=n; *p; p--) cout<<*p<<" ";`
11–15	`for(p+=n-9; *p; p++) cout<<*p<<endl;`	16–20	`for(p+=n-11; *p; p--) cout<<*p<<endl;`
21–25	`for(i=n-21; i<5; i++) cout<<sName[i];`	26–30	`for(i=n-21; i>2; i--) cout<<sName[i];`

Завдання 4.2. Скласти програму обчислення наступних величин за варіантами, які наведені в таблиці 4.2, та виконати її в NetBeans IDE, якщо елементи масиву визначаються за формулою $a_{i+1}=(37\cdot a_{i+3}) \bmod 64$. Значення a_0 дорівнює номеру варіанта; i змінюється від 0 до 18.

Таблиця 4.2 – Варіанти завдання 4.2

№	Величини, які потрібно обчислити
1–3	Найбільший елемент масиву ***a*** і його порядковий номер
4–6	Суми елементів масиву ***a***, значення яких кратні номеру варіанта
7–9	Суми елементів масиву ***a***, значення яких парні числа
10–12	Середнє арифметичне додатних елементів масиву ***a***
13–15	Суми елементів масиву ***a***, значення яких непарні числа
16–18	Середнє геометричне додатних елементів масиву ***a***
19–21	Суми елементів масиву ***a***, значення яких двозначні парні числа
22–24	Добуток найбільшого і найменшого елементів масиву ***a***
25–27	Суми елементів масиву ***a***, значення яких двозначні непарні числа
28–30	Модуль вектора ***a***/3

Короткі теоретичні відомості

Масив – це упорядкована послідовність однойменних елементів, кожен з яких має один і той самий тип. Елементи масиву можуть мати будь-який стандартний тип або тип, введений користувачем. Елементи

масиву розміщені впорядковано, кожен має свій номер, який називається індексом. Доступ до елементів масиву відбувається шляхом вказування імені масиву та порядкового номера елемента (індексу).

Масив визначається за модифікатором типу `[]`. Загальний вигляд опису масиву:

тип_елементів ім'я_масиву`[`*кількість_елементів*`];`

Наприклад:

```
char sName[20];
```

визначає масив з двадцяти елементів типу **`char`**. Ім'я масиву `sName` містить адресу першого елементу масиву. Це ім'я може бути використане в операціях арифметики вказівників для доступу до елементів масиву.

Оператор `[]` дає більш короткий вираз для доступу до елементів масиву. Вираз `sName[k]` є еквівалентним `*(sName+k)`. Більш детально робота з оператором `*` (розіменування) буде описана у частині 2 Методичних вказівок.

Масиви в С++, як і в С, нумеруються з нуля. Тому найбільший елемент – це *кількість_елементів* мінус одиниця. В масиві з 20 елементів індекси змінюються від 0 до 19. Перехід через максимум призводить до використання чужої області пам'яті та помилки на етапі виконання програми.

Елементи масиву розміщуються у пам'яті послідовно. Елементи з меншими значеннями індексу зберігаються в більш низьких адресах пам'яті. Елементи багатовимірних масивів розміщуються таким чином, що самий правий індекс зростає самим першим. Так, для двовимірного масиву **`int`** `d[2][2]` елементи масиву розміщуються у пам'яті за зростанням адресів: `d[0][0], d[0][1], d[1][0], d[1][1]`.

Операції з масивами зручніше проводити за допомогою циклів. Неможливо присвоїти один масив іншому цілком, тільки поелементно.

Приклад виконання роботи

Завдання 4.1. Визначити дію фрагмента програми

```
char sName[20]="Hello!"; char *p=sName; int n,i;
for(i=1; i<5; i++) cout<<sName[i];
```

Розв'язання

Цей фрагмент програми виводить на екран рядок: `ello`

Завдання 4.2. Скласти програму перестановки елементів масиву ***a*** в зворотному порядку і виконати її в NetBeans IDE, якщо елементи

масиву визначаються за формулою $a_{i+1}=(37\cdot a_i+3) \bmod 64$. Значення $a_0=40$; i змінюється від 0 до 16.

Розв'язання

1. Постановка задачі

Скласти програму перестановки елементів масиву ***a*** в зворотному порядку на мові С++, якщо елементи масиву визначаються за формулою $a_{i+1}=(37\cdot a_i+3) \bmod 64$. Значення $a_0=40$; i змінюється від 0 до 16.

2. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Ввести елементи масиву ***a***.

Дія 2. Вивести елементи масиву ***a***.

Дія 3. Визначити m (кількість перестановок елементів масиву ***a***) як результат цілочислового ділення числа елементів масиву n на 2.

Дія 4. Присвоїти j (номеру поточного елемента масиву, який переставляється на місце елемента з меншим номером) значення n–1.

Дія 5. Повторювати m разів наступні дії (i= 0, 1, ..., m–1):

Дія 5.1. Присвоїти x (змінній для тимчасового зберігання елемента з меншим номером) значення i-го елемента масиву ***a***;

Дія 5.2. Присвоїти i-му елементу масиву ***a*** значення j-го елемента;

Дія 5.3. Присвоїти j-му елементу масиву ***a*** значення x;

Дія 5.4. Зменшити значення j на 1;

Дія 6. Вивести елементи масиву ***a*** після перестановки.

Запишемо алгоритм розв'язання задачі мовою С++, позначивши масив ***a*** через `a`, елементи якого мають тип **`int`**. Змінні i, j, n, m, x позначимо відповідно як `i`, `j`, `n`, `m`, `x` (усі мають тип int).

3. Текст програми

```
//програма перестановки елементів масиву a[n]
#include <iostream>
#include <cmath>
using namespace std;
int main(){
   int i,j,m,x,n=18;
   int a[18];
   cout<<"Вводимо масив a["<<n<<"]"<<endl;
   a[0]=40;
```

```
   for(i=0; i<=n-2; i++)a[i+1]=(37*a[i]+3)%64;
   for(i=0; i<=n-1; i++)cout<<a[i]<<" ";
   cout<<endl;
   m=n/2; j=n-1;
   for(i=0; i<=m-1; i++){
   x=a[i]; a[i]=a[j]; a[j]=x; j-=1;
   };
   cout<<"Масив a["<<n<<"] після
перестановки"<<endl;
   for(i=0; i<=n-1; i++)cout<<a[i]<<" ";
   cout<<endl;
   }
```

4. Результати роботи програми

```
Вводимо масив a[18]
40 11 26 5 60 47 14 9 16 19 2 13 36 55 54 17 56 27
Масив a[18] після перестановки
27 56 17 54 55 36 13 2 19 16 9 14 47 60 5 26 11 40
```

Контрольні питання

1. Який тип можуть мати елементи масиву?
2. Як відбувається доступ до елементів масиву?
3. Який індекс мають найменший та найбільший елементи масиву?
4. Як розміщуються в пам'яті елементи масиву?
5. Як зростають індекси багатовимірних масивів?
6. Як можна присвоїти значення елементів одного масиву іншому?

Робота № 5

Розробка та реалізація програми з використанням функцій

Мета роботи: оволодіння навичками складання програми з використанням функцій та виконання її в NetBeans IDE.

Завдання

Завдання 5.1. Обчислити значення змінних, які будуть виведені на дісплей, після виконання фрагмента програми за варіантами, наведеними в таблиці 5.1. Вказівка: n дорівнює номеру варіанта.

Таблиця 5.1 – Варіанти завдання 5.1

№	Фрагмент програми	№	Фрагмент програми
1–5	`void d(int &x, int &y);` `void main(){` `  int x,y,n;` `  cin>>n; x=3; y=4;` `  d(y,x); y=n*x;` `  cout<<x<<" "<<y;` `}` `void d(int &x, int &y){` `  x*=2; y=x+2;` `}`	6–10	`int d(int x, y);` `void main(){` `  int x,y,n;` `  cin>>n; x=3; y=4;` `  y=n*x+d(y,x);` `  cout<<x<<" "<<y;` `}` `int d(int x, y){` `  return 2*x+y;` `}`
11–15	`int d(int x, y);` `void main(){` `  int x,y,n;` `  cin>>n; x=3; y=4;` `  y=n*x+d(n,y);` `  cout<<x<<" "<<y;` `}` `int d(int x, y){` `  return 2*x-y;` `}`	16–20	`void d(int &x, int &y);` `void main(){` `  int x,y,n;` `  cin>>n; n-=9; x=3;` `y=4;` `  d(y,x); y=n*x+y;` `  cout<<x<<" "<<y;` `}` `void d(int &x, int &y){` `  x*=2; y=x+1;` `}`
21–25	`int fp(int x);` `void main(){` `  int n,y; cin>>n; n-` `=18;` `  y=fp(n); cout<<y;` `}` `int fp(int x){` `if(x>2)return x*fp(x-` `1);` `else return x;` `}`	26–30	`int fs(int x);` `void main(){` `  int n,y; cin>>n; n-` `=21;` `  y=fp(n); cout<<y;` `}` `int fs(int x){` `if(x>1)return x+fp(x-` `1);` `else return x;` `}`

Завдання 5.2. Скласти програму обчислення величин із завдання 4.2 з використанням функцій і виконати її в NetBeans IDE.

Короткі теоретичні відомості

Функція – спеціальна конструкція, за допомогою якої фрагмент коду, що повторюється в програмі кілька разів, виноситься за тіло основної програми.

Існує два способи оголошення функції: до функції **main**; за допомогою прототипу (тіло оголошеної функції описується після функції **main**). При другому способі необхідно до функції **main** вказати тип значення, що повертається, ім'я функції та її аргументи.

Загальний синтаксис оголошення функції:

тип_значення ім'я_функції (*параметри*) {
тіло функції;
}

Значення, що повертається – результат роботи функції, може бути будь-яким з базових або користувацьких типів. Якщо функція нічого не повертає, на місце значення, що повертається, підставляється **void** (пусто).

Параметри (аргументи) – вхідні дані, необхідні для роботи функції, для кожного параметра вказують тип даних. Параметри можуть бути відсутніми.

Параметри, які вказуються при визначенні функції, називаються формальними, вони створюються в момент виклику функції в оперативній пам'яті. При виході з функції такі параметри будуть знищені. Параметри, які передаються в функцію при її виклику, називаються фактичними.

Формальному параметру функції може бути задане значення за замовчуванням, параметрами за замовчуванням можуть бути параметри, починаючи з правого кінця списку без перерв:

тип_значення ім'я_функції (*тип_параметра ім'я _параметра = значення_за_замовчуванням*);

Не можна створювати одну функцію всередині іншої і викликати функцію до її оголошення.

Для повернення значення з функції в програму використовується оператор **return**:

```
return res;
```

Якщо функція не повертає жодних значень, оператор **return** можна використовувати для зупинки функції, в даному випадку **return**

відпрацює для функції, як **break** для циклу. Операторів **return** у функції може бути декілька, але спрацює тільки один з них. Якщо тип, що повертається функцією, не **void**, то необхідно завжди використовувати форму: **return** *значення*;

Виклик функції складається з зазначення імені функції, передачі аргументів (при необхідності) і отримання значення, що повертається (при необхідності). При виклику функції необхідно вказувати таку кількість параметрів, яка була визначена при оголошенні функції.

При використанні масиву в якості аргументу функції ім'я масиву перетворюється до вказівника на його перший елемент, тобто при передачі масиву в якості аргументу функції відбувається передача вказівника. При передачі одновимірного масиву досить вказати порожні квадратні дужки:

```
int sum (int array[], int size);
```

При передачі двовимірного масиву обов'язково необхідно вказати кількість стовпців, кількість рядків вказувати не обов'язково:

```
int sum (int array[][5], int size_row, int size_col);
```

Будь-які операторні дужки в програмному коді утворюють так звану область видимості. Змінні, оголошені всередині цих дужок, буде видно тільки в цій області видимості. Згідно з правилами області видимості, змінні діляться на два види – локальні і глобальні.

Локальні змінні створюються всередині будь-якої області видимості. Глобальні змінні створюються поза всяких областей видимості, переважно до функції `main()`. Глобальна змінна видна в будь-якому місці програми. За замовчуванням глобальні змінні на відміну від локальних ініціалізуються 0. Ті зміни, які відбуваються з глобальною змінною всередині функції, при виході з останньої зберігаються.

Якщо є глобальна і локальна змінні з однаковими іменами, то всередині будь-якої області видимості буде використовуватися локальна змінна. Тому слід уникати використання в програмі однакових імен змінних.

Якщо функція викликає сама себе, то вона називається рекурсивною. Глибина рекурсії, тобто кількість викликів, у C++ не обмежується. Реально вона залежить від ресурсів пам'яті (розміру стека).

У загальному випадку рекурсивність – це не властивість самої функції, а властивість її опису. Рекурсивна функція, як правило, коротша і наочніша за нерекурсивну, але при виконанні вимагає більше часу і пам'яті за рахунок повторних звертань до самої себе і дублювання локаль-

них змінних. Необхідно добре розуміти, що кожний черговий рекурсивний виклик приводить до утворення нової копії локальних об'єктів функції і усі ці копії, що відповідають ланцюжкові активізованих і незавершених рекурсивних викликів, існують незалежно один від одного та зберігаються у стеку. Це може привести до так званого "вибуху" стека. "Вибух" стека означає, що для запису локальних змінних функції не вистачає пам'яті, яка відводиться під стек.

Приклад виконання роботи

Завдання 5.1. Обчислити значення змінних, які будуть виведені на дісплей, після виконання наступного фрагмента програми:

```
long fact(long x);
void main(){
  long n,x,y; n=40; n-=36;
  y=fact(n); x=n; cout<<x<<" "<<y;
}
long fact(long x){
  return x>2 ? x*fact(x-1):x;
}
```

Розв'язання

Ця програма обчислює $x!=4*3*2*1$ з використанням рекурсивної функції. У результаті її виконання x=4, а y=24.

Завдання 5.2. Скласти програму перестановки елементів масиву ***a*** в зворотному порядку з використанням функції і виконати її в NetBeans IDE. Елементи масиву ***a*** визначаються за формулою $a_{i+1}=(37\cdot a_i+3) \bmod 64$. Значення $a_0=40$; i змінюється від 0 до 16.

Розв'язання

1. Постановка задачі

Скласти програму перестановки елементів масиву ***a*** в зворотному порядку на мові С++ з використанням функції, якщо елементи масиву визначаються за формулою $a_{i+1}=(37\cdot a_i+3) \bmod 64$. Значення $a_0=40$; i змінюється від 0 до 16. Виконати програму в NetBeans IDE.

2. Алгоритм розв'язання задачі

Алгоритм наведений у прикладі розв'язання завдання 4.2 (при оформленні роботи алгоритм необхідно навести наново).

3. Текст програми

```
//програма перестановки елементів масиву a[n]
#include <iostream>
using namespace std;
void rev(int n, int a[]);
void main(){
   int i,j,m,x,n=18;
   int a[18];
   cout<<"Вводимо масив a["<<n<<"]"<<endl;
   a[0]=40;
   for(i=0; i<=n-2; i++)a[i+1]=(37*a[i]+3)%64;
   for(i=0; i<=n-1; i++)cout<<a[i]<<" ";
   cout<<endl;
   rev(n, a);
   cout<<"Масив a["<<n<<"] після
перестановки"<<endl;
   for(i=0; i<=n-1; i++)cout<<a[i]<<" ";
   cout<<endl;
}
//функція перестановки елементів масиву int a[n]
   void rev(int n, int a[]){
     int m,i,j,x;
     m=n/2; j=n-1;
     for(i=0; i<=m-1; i++){
     x=a[i]; a[i]=a[j]; a[j]=x; j-=1;
   }
}
```

4. Результати роботи програми

```
Вводимо масив a[18]
40 11 26 5 60 47 14 9 16 19 2 13 36 55 54 17 56 27
Масив a[18] після перестановки
27 56 17 54 55 36 13 2 19 16 9 14 47 60 5 26 11 40
```

Контрольні питання

1. Які існують способи оголошення функції?
2. Яка різниця між фактичними та формальними параметрами?
3. Як працює оператор **return**?
4. Що таке глобальна та локальна змінні?
5. Як здійснюється виклик функції?
6. Яка функція називається рекурсивною і яку особливість вона має?

Робота № 6

Розробка та реалізація програми з використанням рядків

Мета роботи: оволодіння навичками складання програми з використанням рядків та виконання її в NetBeans IDE.

Завдання

Завдання 6.1. Визначити дію фрагмента програми за варіантами, які наведені в таблиці 6.1, якщо `char str1[20]="C++ language"; char *str2="12345"; int n;` (Вказівка: `n` дорівнює номеру варіанта).

Таблиця 6.1 – Варіанти завдання 6.1

№	Фрагмент програми	№	Фрагмент програми
1–5	`strcat(str1,str2[n-1]);` `cout<<str1;`	6–10	`strupr(str1);` `cout<<str1<<str2[n%5];`
11–15	`strlwr(str1);` `cout<<str1<<str2[n%11];`	16–20	`strcpy(str1,str2);` `cout<<str1<<str2[n%20];`
21–25	`strncpy(str1,str2,n%20);` `cout<<str1;`	26–30	`cout<<atoi(str2)/n;`

Завдання 6.2. Скласти програму знаходження наступних величин за варіантами, які наведені в таблиці 6.2, та виконати її в NetBeans IDE, якщо даний рядок "We study C++ programming language first semester.".

Таблиця 6.2 – Варіанти завдання 6.2

№	Величини, які потрібно знайти
1–3	Кількість слів у рядку
4–6	Кількість букв е у рядку
7–9	Довжина найкоротшого слова у рядку
10–12	Довжина найдовшого слова у рядку
13–15	Кількість заголовних букв у рядку
16–18	Перші букви кожного слова
19–21	Слова, які не містять букву е
22–24	Які букви і скільки раз зустрічаються у рядку
25–27	Яких букв більше у рядку: голосних або приголосних
28–30	Позиції символів, які не є буквами

Короткі теоретичні відомості

У мові С ++, як і в мові С, немає вбудованої підтримки строкового типу даних, для роботи з рядками використовуються символьні масиви, в кожному елементі зберігається один символ і займає один байт, тому що кожен елемент символьного масиву має тип **char**. Ознакою кінця рядка є нуль-термінатор `\0` (нульовий символ). Нуль-символ – це не цифра 0, в таблиці кодів ASCII він має номер 0 та не виводиться на друк. Наявність нуль-символу означає, що кількість елементів масиву має бути хоча б на один більше, ніж кількість символів у рядку. При використанні у виразах рядок укладається в подвійні лапки. Лапки не є частиною рядка, вони відзначають його початок і кінець.

Робота з рядками з використанням класу `string` буде описана у частині 2 Методичних вказівок.

Варіанти ініціалізації рядків:

оголосити масив типу **char**, за допомогою оператора присвоювання = в подвійних лапках вказати необхідний текст, нуль-термінатор \0 додасться автоматично, розмір масиву при цьому вказувати не обов'язково (визначається компілятором): **char** `str[]="Hello world";` Використовуючи порожні лапки при ініціалізації, ми присвоюємо кожному елементу масиву значення `\0`. Таким чином рядок буде очищений від "сміття" інших програм;

оголосити масив типу **char**, текст не присвоювати, при цьому вказати розмір масиву: **char** `str[20];`

Класичний спосіб оголошення масиву **char** `str[100] = {'H','e','l','l','o',' ','w','o','r','l','d','\0'};` при роботі з рядками не використовується.

Основні функції для роботи з рядками наведені у додатку Е. Для виведення рядка на екран достатньо звернутися до нього на ім'я: `cout<<str<<endl; cout` буде виводити на екран символ за символом, поки не зустріне в одному з елементів масиву символ кінця рядка `\0` і вивід перерветься. Таке звернення для звичайного символьного масиву (масиву без нуль-термінатора `\0`) неприпустиме. Також для виведення рядка на екран використовується функція `puts()`.

Введення рядка з клавіатури можна здійснити за допомогою команди `cin`, проте з усього введеного зчитається послідовність символів до першого пробілу. Більш універсальною є функція `gets()`, яка зчитує всі введені символи з пробілами, поки не буде натиснута клавіша Enter.

Також для роботи з рядками використовуються вказівники. В цьому випадку до кожного рядка можна звернутися за допомогою вказівника на його перший символ. Більш детально робота із вказівниками буде описана у частині 2 Методичних вказівок.

Приклад виконання роботи

Завдання 6.1. Визначити дію фрагмента програми

```
char *str="12345";
cout<<atof(str)/5;
```

Розв'язання

Цей фрагмент програми виводить на екран число 2469

Завдання 6.2. Скласти програму, яка видаляє всі пробіли з рядка, та виконати її в NetBeans IDE, якщо даний рядок "We study C++.".

Розв'язання

1. Постановка задачі

Скласти програму, яка видаляє всі пробіли з рядка " We study C++.". Виконати програму в NetBeans IDE.

2. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Ініціалізувати рядок str.

Дія 2. Вивести рядок str.

Дія 3. Визначити *m* (довжину рядка str).

Дія 4. Повторювати *m* разів наступні дії (*i*= 0, 1, ..., *m*-1):

Дія 4.1. *j*-му символу рядка str присвоїти значення *i*-го символу рядка str;

Дія 4.2. Якщо *i*-й символ рядка str не пробіл, то збільшити значення лічильника *j* на 1;

Дія 4.3. Збільшити значення лічильника *i* на 1.

Дія 5. Якщо *j*<*i*, то *j*-му символу рядка str присвоїти значення кінця рядка `\0`.

Дія 6. Вивести рядок str після перетворення.

Запишемо алгоритм розв'язання задачі мовою C++, позначивши рядок через `str`, символи якого мають тип **`char`**. Змінні *i* та *m* позначимо відповідно як `i`, `m`, які мають тип **`int`**.

3. Текст програми

```
//програма видалення пробілів з рядка str
#include <iostream>
#include <cstring>
using namespace std;
int main() {
   char str[]="We study C++.";
   int m, i=0, j=0;
   cout<<"Рядок: "<<str<<endl;
   m=strlen(str);
   while (i<m) {
     str[j]=str[i];
     if (str[i]!=' ') j+=1;
     i+=1;
   }
   if (j<i) str[j] = '\0';
   cout<<"Рядок після перетворення: "<<str<<endl;
}
```

4. Результати роботи програми

```
Рядок: We study C++.
Рядок після перетворення: WestudyC++.
```

Контрольні питання

1. Що використовується для роботи з рядками?
2. Що означає нуль-термінатор `\0`?
3. Які існують способи ініціалізації рядків?
4. Як можна вивести рядок на екран?
5. Як можна ввести рядок з клавіатури?
6. Наведіть основні функції роботи з рядками.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. **Буч, Г.** Объектно-ориентированное проектирование с примерами применения [Текст] : пер. с англ. / Г. Буч. – М. : Конкорд, 1992. – 519 с.

2. **Дьюхарт, С.** Программирование на Си++ [Текст] / С. Дьюхарт, К. Старк. – К. : НИПФ "ДиаСофт", 1993. – 272 с.

3. **Страуструп, Б.** Программирование на Си++ [Текст] : пер. с англ. / Б. Страуструп. – М. : Радио и связь, 1991. – 352 с.

4. **Страуструп, Б.** Язык программирования С++ [Текст] / Б. Страуструп. – М. : Бином, 2004. – 369 с.

5. **Страуструп, Б.** Язык программирования С++. Специальное издание [Текст] / Б. Страуструп. – М. : Бином, 2011. – 1136 с.

6. **Шилдт, Г.** Самоучитель С++, 3-е издание [Текст] : пер. с англ. / Г. Шилдт. – СПб. : BHV – Санкт-Петербург, 1998. – 688 с.

7. **Шилдт, Г.** С++: базовый курс, 3-е издание [Текст] : пер. с англ. / Г. Шилдт. – М. : Издательский дом "Вильямс", 2010. – 624 с.

8. Borland С++ [Текст] : пер. с англ. / П. Киммел [и др.]. – СПб. : BHV – Санкт-Петербург, 1997. – 976 с.

ДОДАТКИ

Додаток А

Алфавіт мови С++

Алфавіт мови С++ складають:

– букви латинського алфавіту: `A = Z, a = z` та символ підкреслення _ (ASCII-код 95);

– арабські цифри: `0 – 9`;

– спеціальні символи: `+ = * / % = < > . , : ; ' " ( ) [ ] { } # $ ^ ~ ! ? & |`;

– символи-розділителі: символ пропуску (ASCII-код 32) та управляючі символи (наведені у додатку Б);

– службові (зарезервовані) слова (наведені у додатку В).

Додаток Б

Управляючі символи мови С++

Послідовності символів, що починаються зі зворотної косої риски (\), називають керуючими, або escape-послідовностями – символи, які виштовхуються в потік виводу, з метою форматування виводу або друку деяких керуючих знаків С++. Основний список керуючих символів мови С++ представлений у таблиці Б.1.

Керуючі послідовності використовуються для опису певних спеціальних символів усередині строкових літералів і розглядаються як один символ.

Таблиця Б.1 – Управляючі символи мови С++

Керуюча послідовність	Опис
`\'`	одинарна лапка
`\"`	подвійна лапка
`\?`	літерал знаку питання
`\\`	зворотний слеш
`\0`	нульовий символ
`\a`	звуковий сигнал
`\b`	видалення попереднього символу
`\f`	нова сторінка
`\n`	новий рядок
`\r`	повернення каретки
`\t`	горизонтальна табуляція

Продовж. табл. Б.1

Керуюча послідовність	Опис
\v	вертикальна табуляція
\nnn	символ ASCII у вісімковій нотації
\xnn	символ ASCII в шістнадцятковій нотації
\unnnn	символ юнікода в шістнадцятковій нотації, якщо ця escape-послідовність використовується в багатобайтовій знаковій константі або строковому літералі юнікода

Додаток В

Службові слова мови С++

Службові (зарезервовані) слова – це обмежена група слів, сенс яких зафіксовано у мові. Службові слова **не можна** вживати як ідентифікатори змінних, констант і т.п.

```
asm             auto            bool            break
case            catch           char            class
const           const_cast      continue        default
delete          do              double          dynamic_cast
else            enum            explicit        extern
false           float           for             friend
goto            if              inline          int
long            mutable         namespace       new
operator        overload        private         protected
public          register        reinterpret_cast return
short           signed          sizeof          static
static_cast     struct          switch          template
this            throw           true            try
typedef         typeid          typename        union
unsigned        using           virtual         void
volatile        wchar_t         while
```

Ідентифікатори `signed` та `volatile` зарезервовані для застосування в майбутньому.

Додаток Г

Математичні функції мови С++

Базові математичні функції мови програмування С++ містяться в заголовочному файлі cmath.h (аналог файлу math.h для мови С). Первісно в мовах С та С++ підтримувався один і той же набір з 22 математичних функцій. По мірі розвитку мови С ++ до нього додалися перевантажені версії цих оригінальних функцій, які явно призначені для прийому значень типу float та long double (на відміну від типу double у мові С). Дії, що виконуються функціями, залишилися тими ж. Усі кути задаються в радіанах. У таблиці Г.1 наведено базові математичні функції мови С++.

Таблиця Г.1 – Базові математичні функції мови С++

Функція	Опис
acos(a)	арккосинус a, де –1.0 < a < 1.0
asin(a)	арксинус a, де –1.0 < a < 1.0
atan(a)	арктангенс a
atan2(a, b)	арктангенс b/a
ceil(a)	найменше ціле, яке перевищує або дорівнює a
cos(a)	косинус a
cosh(a)	гіперболічний косинус a
exp(a)	значення експоненти (e^a)
fabs(a)	абсолютне значення (модуль) a
floor(a)	найбільше ціле, яке менше або дорівнює a
fmod(a, b)	залишок від ділення a/b
frexp(a, int *exp)	розбиває число a на мантиссу і експоненту
ldexp(a, int exp)	повертає значення виразу $a*2^{exp}$
log(a)	натуральний логарифм a
log10(a)	десятковий логарифм a
modf(a, *i)	розбиває a на цілу і дробову частини
pow(a, b)	зведення a в ступінь b
sin(a)	синус a
sinh(a)	гіперболічний синус a
sqrt(a)	корінь квадратний з a, де a більше або дорівнює 0
tan(a)	тангенс a
tanh(a)	гіперболічний тангенс a

Операції мови С++ за пріоритетом

У таблиці Д.1 наведені операції за пріоритетом з описом їх позначень, назв та синтаксису. Найвищий пріоритет мають операції у верхній частині таблиці.

Таблиця Д.1 – Операції мови С++ за пріоритетом

Позначення	Назва	Синтаксис
`::` `::`	Дозвіл області видимості Глобальне ім'я	*ім'я_класу* `::` *елемент* `::` *ім'я*
`.` `->` `[]` `()`	Вибір елемента через об'єкт Вибір елемента через вказівник Елемент масиву Виклик функції	*об'єкт* `.` *елемент* *вказівник* `->` *елемент* *вказівник* `[`*вираз*`]` *вираз* (*аргументи*)
`()` `sizeof` `sizeof`	Задання значення Розмір об'єкта Розмір типа	*тип* (*аргументи*) `sizeof` *вираз* `sizeof` (*тип*)
`++` `++` `--` `--`	Постфіксний інкремент Префіксний інкремент Постфіксний декремент Префіксний декремент	*ідентифікатор*`++` `++`*ідентифікатор* *ідентифікатор*`--` `--`*ідентифікатор*
`~` `!` `-` `+`	Доповнення Заперечення Унарний мінус Унарний плюс	`~`*вираз* `!` *вираз* `-`*вираз* `+`*вираз*
`&` `*`	Отримання адреси Разіменування	`&` *ідентифікатор* `*`*вираз*
`new` `new[]` `delete` `delete[]`	Створити Створити масив Вилучити Вилучити масив	`new` *тип* `new` *тип*`[]` `delete` *вказівник* `delete[]` *вказівник*
`()`	Приведення типу	(*тип*) *вираз*
`.*` `->*`	Вибір вказівника через об'єкт Вибір вказівника через вказівник	*об'єкт* `.*` *вказівник-елемента* *вказівник*`->*` *вказівник-елемента*
`*` `/` `%`	Множення Ділення Остача від ділення	*вираз* `*` *вираз* *вираз* `/` *вираз* *вираз* `%` *вираз*
`+` `-`	Додавання Вилучення	*вираз* `+` *вираз* *вираз* `-` *вираз*
`<<` `>>`	Зсув вліво Зсув вправо	*вираз* `<<` *вираз* *вираз* `>>` *вираз*

Продовж. табл. Д.1

Позначення	Назва	Синтаксис
`<` `<=` `>` `>=`	Менше ніж Менше ніж або дорівнює Більше ніж Більше ніж або дорівнює	*вираз* `<` *вираз* *вираз* `<=` *вираз* *вираз* `>` *вираз* *вираз* `>=` *вираз*
`==` `!=`	Дорівнює Не дорівнює	*вираз* `==` *вираз* *вираз* `!=` *вираз*
`&`	Побітне І	*вираз* `&` *вираз*
`^`	Побітне вилучаюче АБО	*вираз* `^` *вираз*
`|`	Побітне АБО	*вираз* `|` *вираз*
`&&`	Логічне І	*вираз* `&&` *вираз*
`||`	Логічне АБО	*вираз* `||` *вираз*
`?:`	Умовний вираз	*вираз* `?` *вираз* `:` *вираз*
`=` `*=` `/=` `%=` `+=` `-=` `<<=` `>>=` `&=` `|=` `^=`	Просте присвоєння Множення та присвоєння Ділення та присвоєння Остача від ділення та присвоєння Додавання та присвоєння Вилучення та присвоєння Зсув вліво та присвоєння Зсув вправо та присвоєння Побітне І та присвоєння Побітне АБО та присвоєння Побітне вилучающе АБО та присвоєння	*ідентифікатор* `=` *вираз* *ідентифікатор* `*=` *вираз* *ідентифікатор* `/=` *вираз* *ідентифікатор* `%=` *вираз* *ідентифікатор* `+=` *вираз* *ідентифікатор* `-=` *вираз* *ідентифікатор* `<<=` *вираз* *ідентифікатор* `>>=` *вираз* *ідентифікатор* `&=` *вираз* *ідентифікатор* `|=` *вираз* *ідентифікатор* `^=` *вираз*
`,`	Кома	*вираз*`,` *вираз*

Функції для роботи з рядками мови С++

Основні функції для роботи з рядками мови програмування С++ містяться у декількох заголовочних файлах: cstring (аналог файлу string.h для мови С), наведені у таблиці Е.1; cstdio (аналог файлу stdio.h для мови С), наведені у таблиці Е.2; cstdlib (аналог файлу stdlib.h для мови С), наведені у таблиці Е.3.

Таблиця Е.1 – Основні функції для роботи з рядками файлу cstring

Функція	Опис
`int strlen(char* str)`	підраховує довжину рядка (кількість символів без урахування \0)
`char *strcat(char *dest, const char *scr)`	приєднує рядок src в кінець рядка dest, одержаний рядок повертається в якості результату
`char *strncat(char *dest, const char *scr, int n)`	приєднує n символів рядка src в кінець рядка dest і повертає рядок dest
`char *strcpy(char *dest, const char *scr)`	виконує копіювання рядка src в рядок dest і повертає рядок dest
`char *strncpy(char *dest, const char *scr, int n)`	виконує копіювання n символів рядка src в рядок dest і повертає рядок dest
`int ctrcmp(const char *s1, const char *s2)`	порівнює два рядки в лексикографічному порядку з урахуванням відмінності великих і малих літер; функція повертає 0, якщо рядки збігаються, повертає –1, якщо s1 розташовується в упорядкованому за алфавітом порядку раніше, ніж s2, і 1 – в протилежному випадку
`int ctrncmp(const char *s1, const char *s2, int n)`	порівнює n символів двох рядків в лексикографічному порядку з урахуванням відмінності великих і малих літер; функція повертає 0, якщо рядки збігаються, повертає –1, якщо s1 розташовується в упорядкованому за алфавітом порядку раніше, ніж s2, і 1 – в протилежному випадку
`getline(char *s, int n)`	призначена для введення з клавіатури рядка s з пробілами, в рядку не повинно бути більше n символів
`char *strchr(const char *str, int c);`	повертає вказівник на перше входження символу c в рядок, на який вказує str, якщо символ c не знайдений, повертає NULL
`char *strupr(char *str)`	перетворює символи рядка, на який вказує str, в символи верхнього регістру, після чого повертає його
`char *strlwr(char *str)`	перетворює символи рядка, на який вказує str, в символи нижнього регістра, після чого повертає його

Таблиця Е.2 – Основні функції для роботи з рядками файлу cstdio

Функція	Опис
`int sprintf(char*, const char*, ...)`	вивід результату в рядок
`int sscanf(const char*, const char*, ...)`	введення значень з рядка
`char* gets(char*)`	зчитує потік символів зі стандартного пристрою вводу в рядок до тих пір, поки не буде натиснута клавіша ENTER
`int puts(const char*)`	вивід рядка на екран з переводом курсору на наступний рядок

Функція	Опис
`int atoi(const char *str)`	перетворює рядок у ціле число, в разі невдалого перетворення повертається 0
`long atol(const char *str)`	перетворює рядок у довге ціле число, в разі невдалого перетворення повертається 0
`double atof(const char *str)`	перетворює рядок у дійсне число, в разі невдалого перетворення повертається число 0

ЗМІСТ