Лабораторна робота №1

Основні ключові слова, типи даних та управляючі конструкції C/C++

Короткі теоретичні відомості

Алфавіт мови C++, ідентифікатори та ключові слова

Для програмування будь-яких завдань мовою C++, необхідно знати такі конструкції: ідентифікатори, службові слова, описи даних та функцій, вирази, оператори, вбудовані функції, управляючі конструкції та деякі інші.

Вирази в мові C++ записуються за допомогою 26 рядкових та 26 прописних літер англійського алфавіту: abcdefghijkImnopqrstuvwxyz, ABCDEFGHIJKLMNOPQRSTUVWXYZ; десяти цифр: 0123456789; таких спеціальних символів: + - * / =, . _ : ; ? \ " ’~ | ! # $ & ( ) [ ] { } ^ @. До спеціальних символів відноситься також пропуск. Комбінації деяких символів, не розділених пропусками, інтерпретуються як один значущий символ: ++ — || && << >> >= ⇐ == != += -= = /= **.?: :: /** */ //

Ідентифікаторами називаються імена, що надають змінним, константам, типам даних і функціям, які використовуються в програмах. Після опису ідентифікатора, можна посилатися на об’єкт, що позначається ним.

Ідентифікатор – це послідовність символів довільної довжини, яка містить літери, цифри й символи підкреслення, обов’язково починається з літери, або символу підкреслення. У C++ враховується регістр букв. Компілятор сприймає прописні і рядкові букви, як різні символи. Так, змінні userName та UserName розглядаються як два різні ідентифікатори.

Ключові слова є зарезервованими ідентифікаторами, кожному з яких відповідає певна дія. Змінити призначення ключового слова неможна (директива препроцесора #define дозволяє створити "псевдонім" ключового слова, який дублює його дії, можливо, з деякими змінами). Імена ідентифікаторів, що створюються в програмі, не повинні збігатися з ключовими словами мов C/C++

alignas (починаючи з C++11)	enum	return
alignof (починаючи з C++11)	explicit	short
and	export	signed
and_eq	extern	sizeof
asm	false	static
auto(1)	float	static_assert (починаючи з C++11)
bitand	for	static_cast
bitor	friend	struct
bool	goto	switch
break	if	template
case	inline	this
catch	int	thread_local (починаючи з C++11)
char	long	throw
char16_t (починаючи з C++11)	mutable	true
char32_t (починаючи з C++11)	namespace	try
class	new	typedef
compl	noexcept (починаючи з C++11)	typeid
const	not	typename
constexpr (починаючи з C++11)	not_eq	union
const_cast	nullptr (починаючи з C++11)	unsigned
continue	operator	using(1)
decltype (починаючи з C++11)	or	virtual
default(1)	or_eq	void
delete(1)	private	volatile
do	protected	wchar_t
double	public	while
dynamic_cast	register	xor
else	reinterpret_cast	xor_eq

Стандартні типи даних

Кожна програма обробляє певну інформацію. У C++ дані мають один з базових типів: **char** (текстові дані), **int** (цілі числа), **float** (числа з плаваючою точкою одинарної точності), **double** (числа з плаваючою точкою подвійної точності), **void** (порожні значення), **bool** (логічні значення), перерахування й покажчики.

Текстом (тип даних **char**) є один символ. Зазвичай кожен символ займає 8 біт або один байт з діапазоном значень від 0 до 255.

Цілі числа (тип даних **int**) знаходяться в діапазоні від – 32 768 до 32 767 раніше займали 16 біт, тобто два байти, або одне слово. У Windows NT і Windows XP і сучасніших ОС Windows використовуються 32-розрядні цілі числа, що дозволяє розширити діапазон значень від – 2 147 483 648 до 2 147 483 647. У C++ підтримуються три типи цілих чисел. Разом із стандартним типом **int** існують типи **short int** (коротке ціле) і **long int** (довге ціле). Допускається скорочений запис **short** і **long**.

Числа з плаваючою точкою одинарної точності (тип даних **float**) можуть бути представлені як у фіксованому форматі, так і в експоненціальному. Діапазон значень – від ± 3.4E-38 до ± 3.4E+38, розмірність – 32 біта, тобто 4 байти або 2 слова. Числа з плаваючою точкою подвійної точності (тип даних **double**) мають діапазон значень від ± 1.7E-308 до ± 1.7E+308 і розмірності 64 біт, тобто 8 байтів або 4 слова. Раніше існував тип **long double** з розмірністю

80 біт і діапазоном від ± 1.18E-4932 до ± 1.18E+4932. У нових 32 розрядних версіях компіляторів він еквівалентний типу **double** і підтримується з міркувань зворотної сумісності з написаними раніше програмами.

Перерахування представляються кінцевим набором іменованих констант різних типів.

Тип даних **void**, як правило, застосовується у функціях, що не повертають ніякого значення. Цей тип даних також можна використовувати для створення узагальнених покажчиків.

Покажчики, на відміну від змінних інших типів, містять не дані в звичайному розумінні цього слова, а адреси пам'яті, де зберігаються дані.

Змінні нового логічного типу даних **bool** в C++ можуть містити тільки одну з двох констант: **true** або **false**. Компілятор мови C++ дозволяє при описанні змінних вказувати модифікатор **unsigned**. Він застосовується з чотирма типами даних: **char**, **short**, **int** і **long**. Наявність даного модифікатора вказує на те, що значення змінної повинне інтерпретуватися як беззнакове число, тобто найстарший біт є бітом даних, а не бітом знаку. Існує модифікатор **signed**, який виконує протилежну **unsigned** дію.

Пріоритет операцій

Як і в інших мовах програмування, в мові C++ діють правила пріоритету (порядку виконання) операцій у виразах. Пріоритети операцій наведені в таблиці.

Table 1. Пріоритети операцій

Операція	**Опис**	**Асоціативність**
++	Постфіксний (префіксний) інкремент	Зліва направо
—	Постфіксний (префіксний) декремент	Зліва направо
()	виклик функції	
[]	Доступ до елемента масиву	
->	Непрямий доступ до члена класу	
.	Прямий доступ до члена класу	
!	Логічне НЕ	
~	Побітове НЕ	
-	Унарний мінус	
+	Унарний плюс	
&	Отримання адреси	
*	Розкриття покажчика	
sizeof	Отримання розмірності виразу в байтах	

Операція	Опис	Асоціативність
new	Динамічне створення об’єкта	
delete	Динамічне видалення об’єкта	
(тип даних)	Приведення типу	
.*	Прямий доступ до покажчика на член класу (через об’єкт)	Зліва направо
->*	Непрямий доступ до покажчика на член класу (через покажчик на об’єкт)	
*	Множення	Зліва направо
/	Ділення	Зліва направо
%	Ділення по модулю	Зліва направо
+	Додавання	Зліва направо
-	Віднімання	Зліва направо
<<	Зсув ліворуч	Зліва направо
>>	Зсув праворуч	
<	менше	Зліва направо
>	більше	
<=	менше, або дорівнює	
>=	більше, або дорівнює	
==	дорівнює	Зліва направо
!=	не дорівнює	
&	побітове І	Зліва направо
^	Побітове АБО (що виключає)	Зліва направо
|	Побітове АБО	Зліва направо
&&	Логічне І	Зліва направо
||	Логічне АБО	Зліва направо
?:	Умовний вираз	Справа наліво
=	Просте присвоювання	Справа наліво
*= /= %= += -= <<= >>= &= |= ^=	присвоювання з множенням, діленням …	
,	кома	Зліва направо

У мовах C/C++ усі стандартні функції знаходяться у бібліотеках, які можна підключити за допомогою заголовочних файлів. Так, функції введення-виведення у стилі мови C описані у файлі stdio.h (cstdio). Функції та класи для введення-виведення у стилі C++ описані у файлах iostream (для введення-виведення із використанням стандартного пристрою) та fstream (для

файлового введення-виведення). Обчислення у програмах на C/C++ неможливі без використання математичних функцій, які описані у файлі math.h (cmath)

Розглянемо приклад: Знаходження значення похідної функції в точці.

Постановка завдання: Задана функція y=sin(x). Знайти її похідну в точці x=π /2. Для знаходження похідної в точці використовується відомий вираз:

$$y'(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

```
#include <math.h>
#include <iostream>
using namespace std;
int main() {
    double dx=1.0e-11;      // Вибираємо приріст аргументу
    double x = 3.1415926;  // Вибираємо точку для обчислення похідної
    double f1=sin(x+dx);   // Обчислюване значення функції в точці x+dx
    double f2=sin(x);      // Обчислюване значення функції в точці x
    double pf=(f1-f2) /dx; // Знаходимо значення похідної
    cout << "dsin(x) /dx=" << pf <<" x= "<<x;
    return 0;
}
```

Оператор if

Оператор **if** призначений для виконання команди або блоку команд залежно від того, істинно задана умова, чи ні.

Формат оператора **if**:

```
if (умова) вираз;
```

Якщо в результаті перевірки умови повертається значення **true**, виконується вираз, після чого управління передається наступному рядку програми.

Якщо ж результатом перевірки умови є значення **false**, вираз пропускається. Оператор **if/else** дозволяє вибірково виконувати одну з двох дій залежно від умови. Формат даної інструкції має вигляд:

```
if (умова) вираз 1; else вираз 2;
```

Якщо в результаті перевірки умови повертається значення true, виконується вираз 1, інакше – вираз 2. Якщо операторна частина гілки if або else містить не один вираз, а декілька, необхідно укласти їх у фігурні дужки. Після закриваючої фігурної дужки крапка з комою не ставиться. Оператор if обох форм реалізують алгоритми, представлені на рисунку

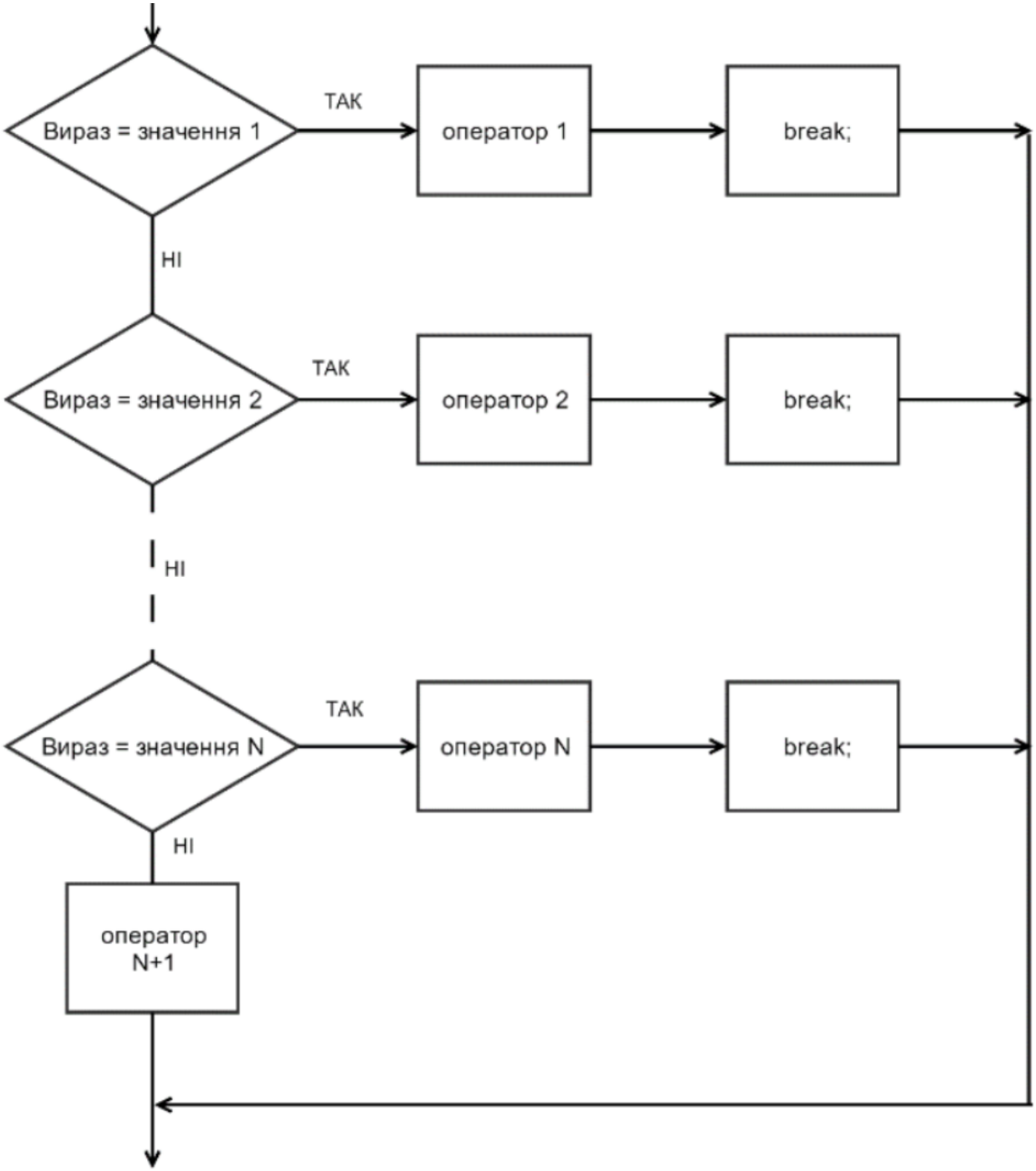
Як аналізований вираз в операторові if найчастіше використовується одна з операцій відношення.

Оператор switch/case

Оператор switch/case дозволяє залежно від значення деякого виразу вибрати один з багатьох варіантів продовження програми. Оператор має такий формат:

```
switch(expr) {
    case val1: op1; break;
    case val2: op2; break;
...
    case valN: opN; break;
    default: opN+1;
}
```

Оператор switch/case реалізує алгоритм, наведений на рисунку



Оператор **switch/case** може бути використаний у варіанті без оператора N+1. Як вираз при операторі **switch** зазвичай використовується змінна типу **int** або **char**, хоча можна використовувати й складніші вирази, в які входять, наприклад, арифметичні або логічні операції над декількома змінними та константами. Як значення при операторі **case** зазвичай використовуються просто константи (у числовій формі або в символьній, якщо вони були заздалегідь визначені за допомогою оператора препроцесора #define), проте можуть використовуватися і вирази над константами. Виконання оператора **switch** починається з обчислення виразу в дужках, який повинен давати цілочисельний результат. Цей результат послідовно порівнюється із значеннями при операторах **case**, і, якщо буде виявлено рівність результатів, то виконується оператор відповідного **case**. Якщо збіги результатів не виявлено, виконується оператор при операторі **default**, якщо оператор

default відсутній, то починають виконуватися оператори, наступні за всією конструкцією **switch/case**.

Оператори break, continue і goto

Оператор break використовується для виходу з оператора while, do…while, for і switch, що безпосередньо його містить. Управління передається на оператор, наступний за оператором, з якого здійснюється вихід. Приклад використання оператора break наведений вище.

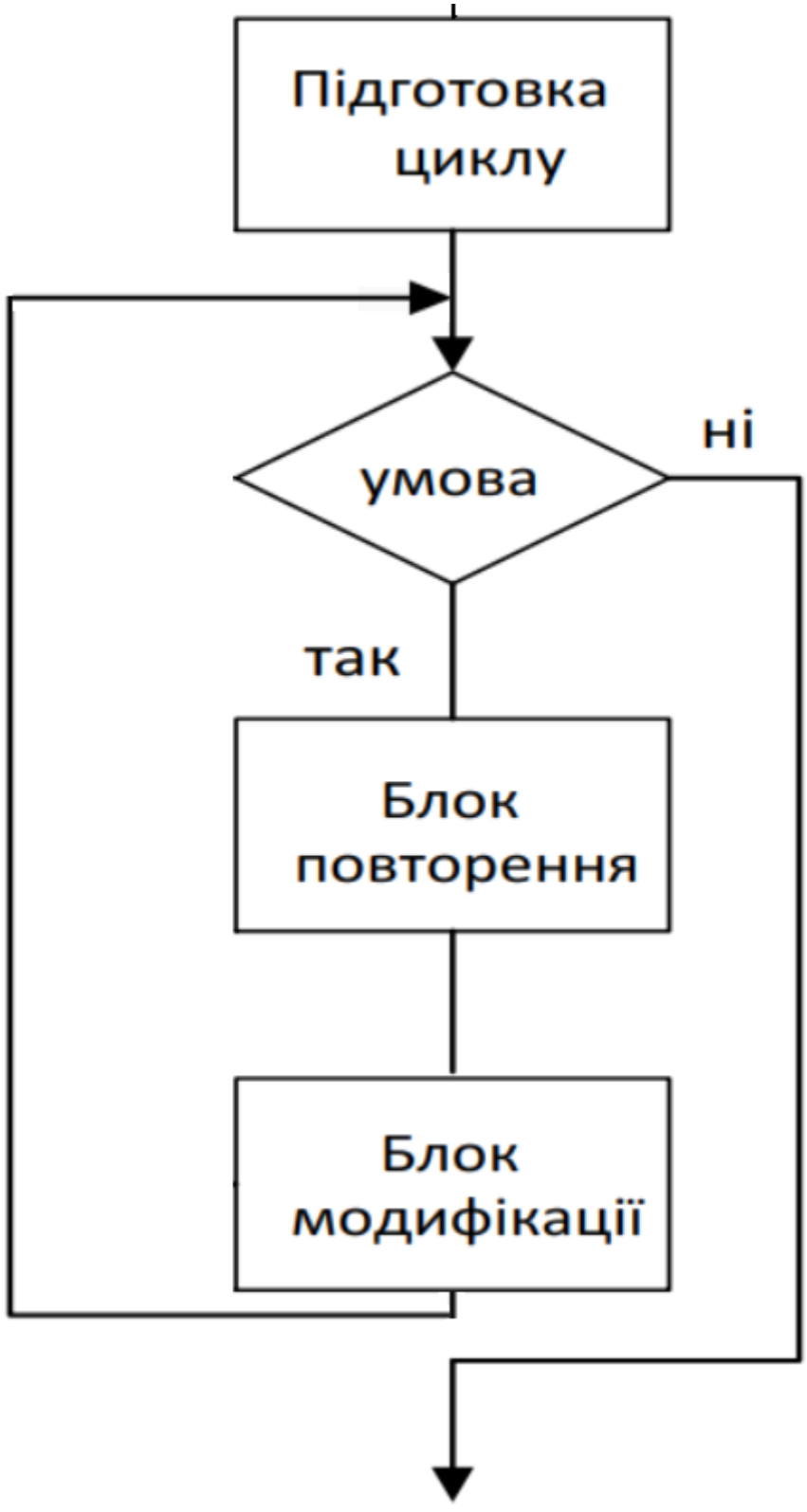
Оператор continue використовується для ігнорування частини виконуваної ітерації циклу, який безпосередньо його містить, що залишилася. Якщо умовами циклу допускається нова ітерація, то вона виконується, інакше цикл завершується.

Оператор goto реалізує безумовний перехід, тобто дозволяє перейти в будь-яку точку програми, як вперед по тексту програми, так і назад. Точка переходу позначається за допомогою мітки, яка є довільним ідентифікатором з двокрапкою в кінці.

Оператори циклів

Оператори циклів використовують для виконання деякого фрагмента програми кілька разів. В окремих випадках фрагмент виконується в кожному послідовному кроці циклу без змін; частіше кожен крок циклу декілька відрізняється від попереднього.

Для грамотної реалізації будь-якого циклічного обчислювального процесу необхідно виконати дії, представлені у вигляді узагальненого алгоритму на рисунку



Підготовка циклу полягає у визначенні початкових значень змінних, що будуть змінюватися у циклі, до початку виконання циклу

Блок повторення - це дії, які повторюються в циклі. Вони завжди однакові, при цьому їх багаторазове повторення здійснюється при різних значеннях змінних циклу.

Модифікація - це зміна значень змінних циклу перед кожним новим повторення циклу.

Блок повторення та Блок модифікації разом складають *тіло циклу*.

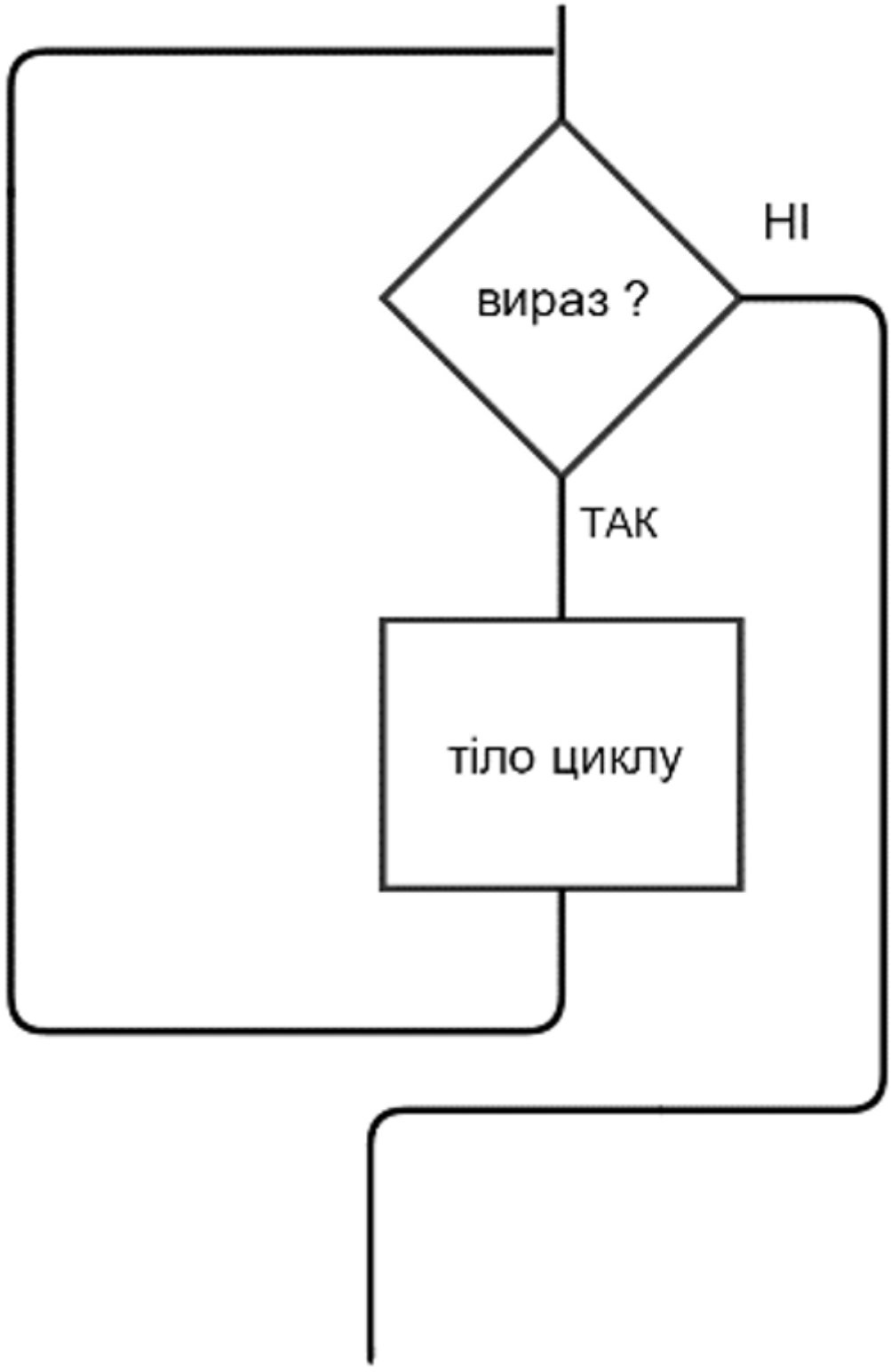
Умова продовження циклу (команда переходу) полягає в перевірці умови продовження або закінчення циклу, тобто визначає скільки разів потрібно повторити тіло циклу.

Існує три види циклів: while, for і do…while.

Оператор циклу while називається циклом з передумовою та має такий формат:

```
while (вираз) тіло циклу;
```

Оператор while реалізує алгоритм, представлений на рисунку



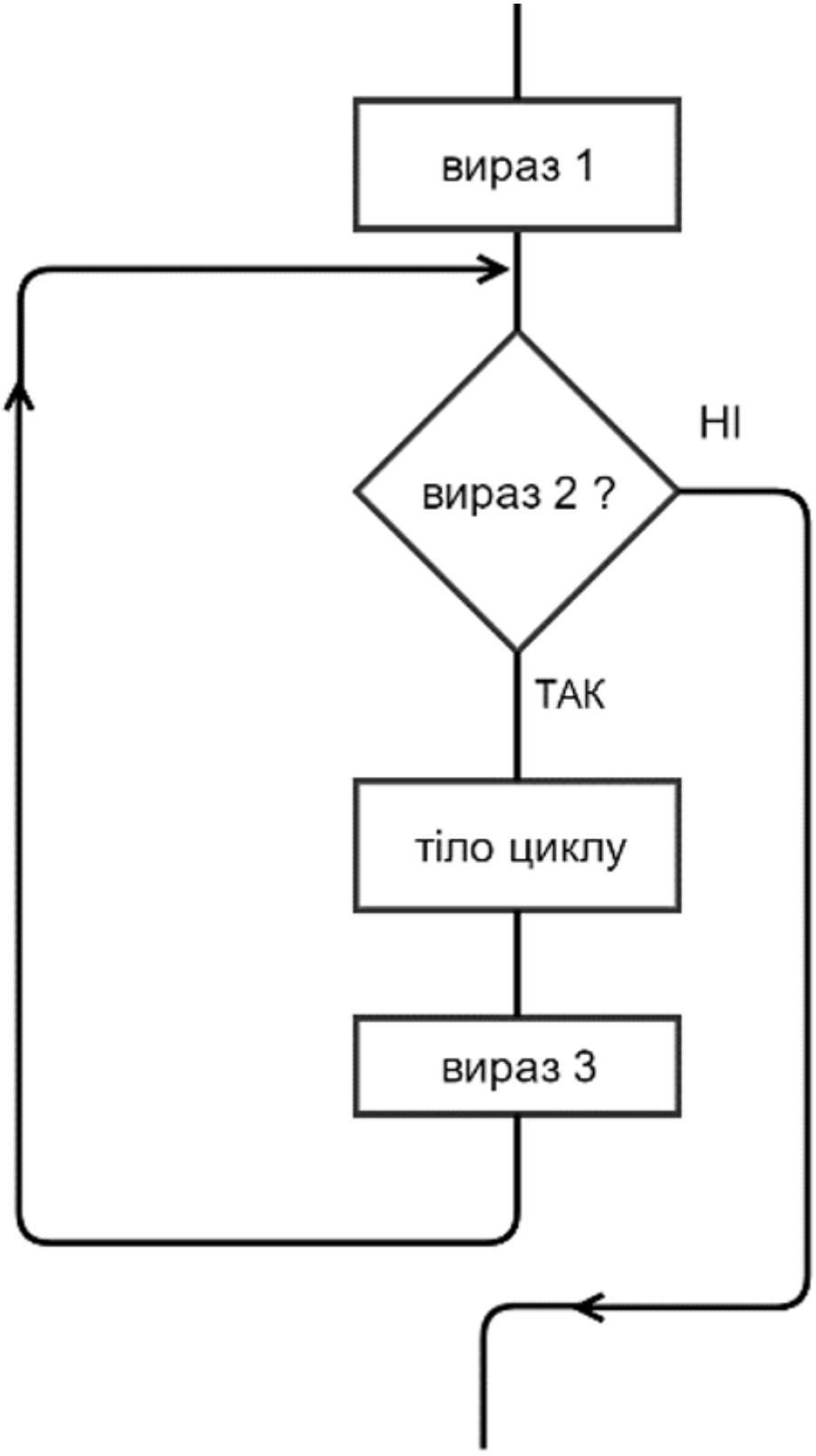
Як вираз, допускається використовувати будь-який вираз мови C++, а як тіло – будь-який оператор, зокрема порожній або складений. Схема виконання оператора while така:

1. Обчислюється вираз.
2. Якщо вираз false, то виконання оператора while закінчується і виконується наступний за порядком оператор. Якщо вираз true, то виконується тіло циклу.
3. Процес повторюється з пункту 1.
4. Тіло циклу виконується доти, поки значення виразу дорівнює true.
5. Вираз обчислюється перед кожним виконанням оператора.

Цикл for має такий формат:

```
for (вираз 1; вираз 2; вираз 3;) тіло циклу;
```

Оператор for реалізує алгоритм, представлений на рисунку



Вираз 1 зазвичай використовується для встановлення початкового значення змінних, які управляють циклом. Вираз 2 – це вираз, що визначає умову, при якій тіло циклу виконуватиметься. Вираз 3 визначає зміну змінних, що управляють циклом після кожного виконання тіла циклу.

Схема виконання оператора for

1. Обчислюється *вираз 1*.

2. Обчислюється *вираз 2*.
3. Якщо значення *виразу 2* відмінно від нуля (true), виконується тіло циклу,
4. Обчислюється *вираз 3* і здійснюється перехід до пункту 2
5. Якщо вираз 2 дорівнює нулю (false), то управління передається до оператора, наступного за оператором for.

Істотне те, що перевірка умови завжди виконується на початку циклу. Це означає, що тіло циклу може жодного разу не виконатися, якщо умова виконання відразу буде хибною.

Цикл for є зручним скороченим записом для циклу while вигляду

```
expr 1;
while (expr 2) {
   body-of-loop;
   expr 3;
}
```

Вираз 1 задає початкові умови виконання циклу, вираз 2 забезпечує перевірку умови виходу з циклу, а вираз 3 модифікує умови, задані виразом 1.

Будь-який з виразів може бути опущений. Якщо опущено вираз 2, то за замовчуванням замість нього підставляється значення true.

Наприклад, цикл:

```
for (;вираз 2;) тіло циклу;
```

з опущеними вираз 1 і вираз 3 еквівалентний циклу

```
while (вираз 2) тіло циклу;
```

Цикл

```
for (;;) тіло циклу;
```

зі всіма опущеними виразами еквівалентний циклу

```
while (true) тіло циклу;
```

тобто еквівалентний нескінченному циклу.

Такий цикл може бути перерваний тільки явним виходом з нього за допомогою операторів **break**, **goto** або **return**, що містяться в тілі циклу.

Оператор циклу **do while** називається оператором циклу з постумовою і використовується в тих випадках, коли необхідно виконати тіло циклу хоч би один раз. Формат оператора має такий формат:

```
do тіло циклу while (вираз);
```

Схема виконання оператора **do while**:

1. Виконується тіло циклу (яке може бути складеним оператором).
2. Обчислюється вираз.
3. Якщо вираз false, то виконання оператора do while закінчується й виконується наступний за порядком оператор. Якщо вираз true, то виконання оператора продовжується з пункту 1.

Щоб перервати виконання циклу до того, як умова стане хибною, можна використовувати оператор break.

Оператор **do while** реалізує алгоритм, наведений на рисунку

На відміну від циклу **while**, в якому перевірка умови закінчення циклу робиться *до виконання* тіла циклу, в циклі **do while** така перевірка має місце *після виконання* тіла циклу. Отже, тіло циклу **do while** буде виконано *хоча б один раз*, навіть якщо вираз має значення false із самого початку.

Завдання 1.1

Представити математичний запис фрагмента програми та обчислити значення змінної x після його виконання, де n – це номер варіанта

№ вар.	Завдання	№ вар.	Завдання

1-3	`x = 1;`
	`for (int j=7; j>=N; j--)`
	`  x *= j;`
	`x *= 2;`
		4-6	`x = 0; j = 1;`
			`do {`
			`  x += j; j += 2;`
			`} while (j<=N);`

7-9	`x = 0; k = 3 * N;`
	`while (k > 0) {`
	`  x = sqrt(k+x);`
	`  k -= 3;`
	`}`
		10-12	`x = N;`
			`for (int k=0; k<6; k++)`
			`{`
			`  if (k<2) continue;`
			`  x++;`
			`}`

13-15	`x = 0;`
	`for (int j=0; j<N; j++)`
	`  x += 2;`
	`x *= 2;`
		16-18	`x = 1;`
			`while (x<=N)`
			`  x++;`
			`x *= 2;`

19-21	`x = 1.0 / N; n = N;`
	`while (n>1) {`
	`  n -= 2;`
	`  x = 1.0 / (n + x);`
	`}`
		22-25	`x = 1.0/N; n = N;k = 1;`
			`while (n>1) {`
			`  n -=2; k = -k;`
			`  x = 1.0/(n + k * x);`
			`}`

Завдання 1.2

Скласти програму табулювання функції f(x) на відрізку [a; b] з кроком h Значення a, b, h вводити з клавіатури. Обов’язково врахувати область визначення функції

Варіант	Функція		Варіант	Функція
1	$y = \frac{tgx}{\ln x}$		9	$y = \frac{\ln(x-0.5)}{\sqrt{x}}$
2	$y = \sqrt[3]{x}$		10	$y = \frac{\sin x^3}{2-x}$
3	$y = tg(\ln x)$		11	$y = \frac{\cos^3 x}{1-\lg x}$
4	$y = \frac{\ln(x-1)}{4-x}$		12	$y = \frac{tgx}{\ln x - 1}$
5	$y = tg^2(\ln x)$		13	$y = \frac{e^2}{\sqrt{1-x^2}}$
6	$y = ctg(\ln x)$		14	$y = \frac{x+1}{\sqrt{x}} - \sqrt[4]{\lvert x-2 \rvert}$
7	$y = x^{0,2}$		15	$y = \frac{\ln\lvert 2x \rvert}{\sin x - \pi}$
8	$y = \frac{x}{1+tgx}$		16	$y = e^{\ln x - 1} + \sin x$

Завдання 1.3

Для заданих *x*, *n*, *eps*, що вводяться з клавіатури:

a. обчислити *n* доданків згідно варіанту

b. обчислити суму тих доданків, які за абсолютним значенням більше *eps*. (Завдання виконати для двох різних *eps*, які відрізняються на порядок, для кожного випадку обчислити кількість доданків)

c. Порівняти результати з "точним" значенням відповідної функції (сума визначає наближене значення) для *x Є (-R,R)*

Примітка 1. У цьому завданні значення *x* повинно належати області допустимих значень,

визначеної у пункті c, значення n повинно бути досить великим (більше 20). *eps* потрібно обирати як маленьке додатне число, яке не більше 1e-9. Виконати обчислення для двох РІЗНИХ *eps*, що відрізняються не менше ніж на порядок (в 10 або більше разів).

Примітка 2. Результат виконання пункту b вважається задовільним, та програма може бути зарахована як правильна, лише за умови, що різниця між "точним" значенням функції та значеннями, обчисленими за наближеними сумами, взята за модулем, є числом, що менше або дорівнює *eps*.

Варіант 1

$$\frac{sinx}{x} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \cdots (R = \infty)$$

Варіант 2

$$e^{-x^2} = 1 - \frac{x^2}{1!} + \frac{x^4}{2!} - \cdots + (-1)^n \frac{x^{2n}}{n!} \quad (R = \infty)$$

Варіант 3

$$\ln\left(x + \sqrt{x^2 + 1}\right) = x - \frac{1}{2} \cdot \frac{x^3}{3} + \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{x^5}{5} - \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdot \frac{x^7}{7} + \cdots \quad (R = 1)$$

Варіант 4

$$arctg\ x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \cdots \quad (R = 1)$$

Варіант 5

$$arcsin\ x = x + \frac{1}{2} \cdot \frac{x^3}{3} + \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{x^5}{5} + \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdot \frac{x^7}{7} + \cdots \quad (R = 1)$$

Варіант 6

$$\frac{1}{\sqrt{1-x^2}} = 1 + \frac{1}{2}\cdot x^2 + \frac{1}{2}\cdot\frac{3}{4}\cdot x^4 + \frac{1}{2}\cdot\frac{3}{4}\cdot\frac{5}{6}\cdot x^6 + \cdots \quad (R=1)$$

Варіант 7

$$\frac{1}{\sqrt{1+x}} = 1 - \frac{1}{2}\cdot x + \frac{1}{2}\cdot\frac{3}{4}\cdot x^2 - \frac{1}{2}\cdot\frac{3}{4}\cdot\frac{5}{6}\cdot x^3 + \cdots \quad (R=1)$$

Варіант 8

$$\sqrt{1+x} = 1 + \frac{1}{2}\cdot x - \frac{1}{2\cdot 4}\cdot x^2 + \frac{1\cdot 3}{2\cdot 4\cdot 6}\cdot x^3 - \cdots \quad (R=1)$$

Варіант 9

$$\frac{1}{(1+x)^3} = 1 - \frac{2\cdot 3}{2}\cdot x + \frac{3\cdot 4}{2}\cdot x^2 - \frac{4\cdot 5}{2}\cdot x^3 + \cdots \quad (R=1)$$

Варіант 10

$$\frac{1}{(1+x)^2} = 1 - 2x + 3x^2 - 4x^3 + 5x^4 - \cdots \quad (R=1)$$

Варіант 11

$$\frac{1}{1+x} = 1 - x + x^2 - x^3 + x^4 - \cdots \quad (R=1)$$

Варіант 12

$$ln\frac{1+x}{1-x} = 2\cdot\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \frac{x^9}{9} + \cdots\right) \quad (R=1)$$

Варіант 13

$$\ln(1-x) = -\frac{x}{1} - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^4}{4} - \cdots \quad (R=1)$$

Варіант 14

$$\ln(1+x) = \frac{x}{1} - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} - \cdots \quad (R=1)$$

Варіант 15

$$ch(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \cdots (R=\infty)$$

Варіант 16

$$sh(x) = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \cdots (R=\infty)$$

Варіант 17

$$cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots (R=\infty)$$

Варіант 18

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots (R=\infty)$$

Варіант 19

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots \quad (R = \infty)$$

Варіант 20

$$e^{-x} = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \cdots \quad (R = \infty)$$