Лабораторна робота №2. Елементи мови С++

Мета: засвоїти основні поняття мови програмування С++ та принципи роботи в середовищі Visual Studio.

Теоретичні відомості

Робота з середовищем Microsoft Visual Studio

Інтегроване середовище розробки Microsoft Visual Studio .Net далі IDE (Integrated Development Enviroment) Microsoft Visual Studio .Net є системою, в кій можна досить легко створювати, переглядати, редагувати, зберігати, компілювати та налагоджувати програми, що написані мовами С, С++, С# та ін. Система доступна для завантаження в кількох версіях. Для безкоштовного користування існує версія Community.

Початкове вікно Microsoft Visual Studio виглядає наступним чином (рис. 1).

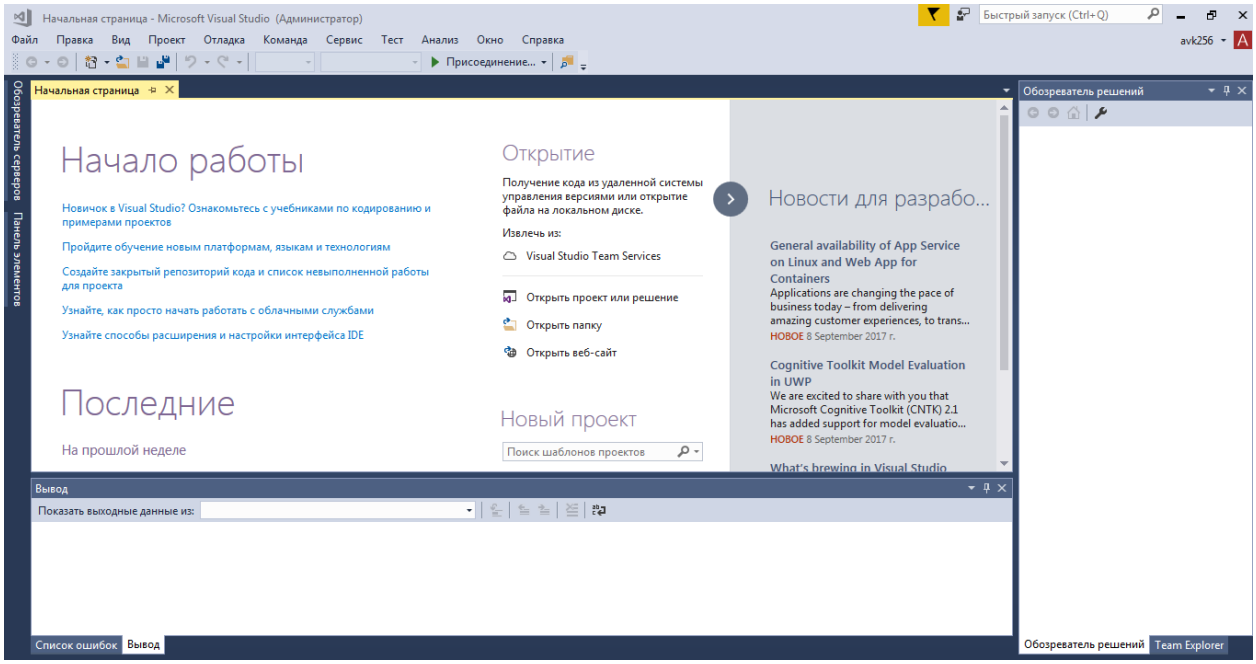

Рис. 1

Для створення консольного застосунку необхідно створити новий проект за допомогою команди

`Файл->Создать->Проект`

Та обрати тип проекту Win32 (рис. 2).

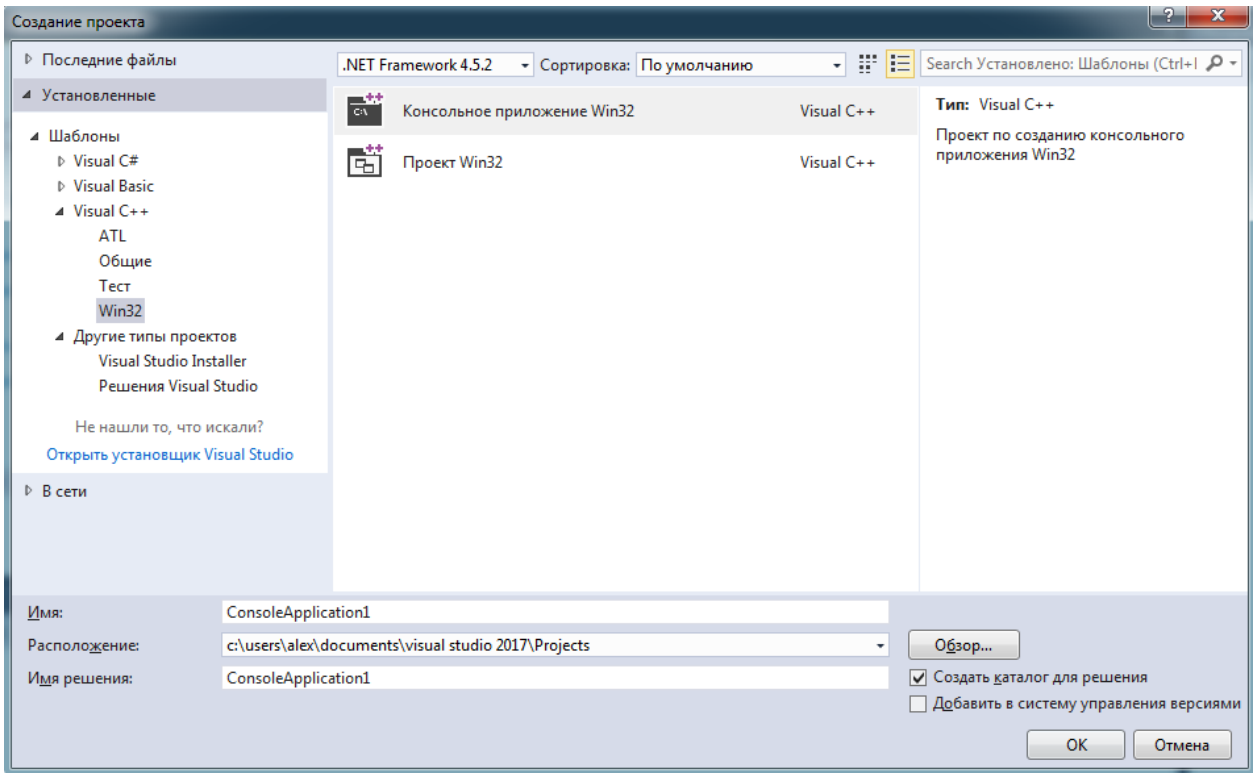

Рис. 2

Приклад 1. Консольна програма виводу привітання на екран (рис. 3).

```
#include "stdafx.h"                    // Включення заголовних файлів
#include <iostream>
using namespace std;                   // Визначення простору імен

int main()
{
	cout << "Hello, world"<<endl;      // Вивести повідомлення
	return 0;
}
```

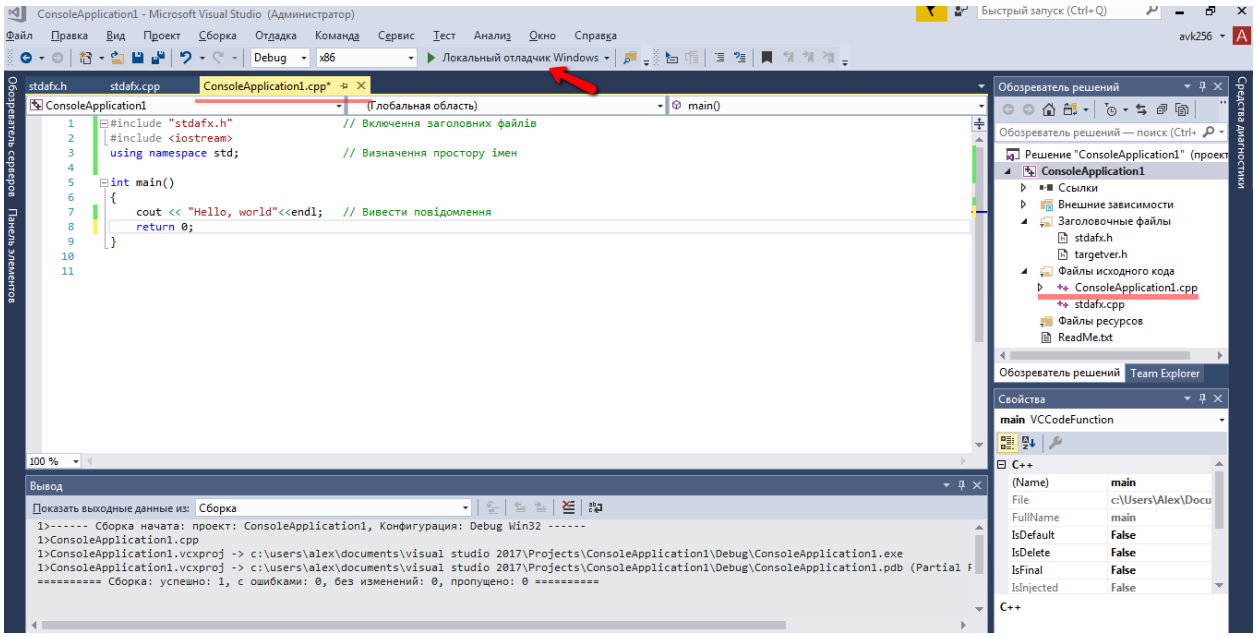

Рис. 3

Програму, в якій відсутні синтаксичні помилки можна запустити на виконання клавішею F5 або комбінацією клавіш Ctrl+F5.

Результати роботи програми виводяться у консольне вікно (рис. 4).

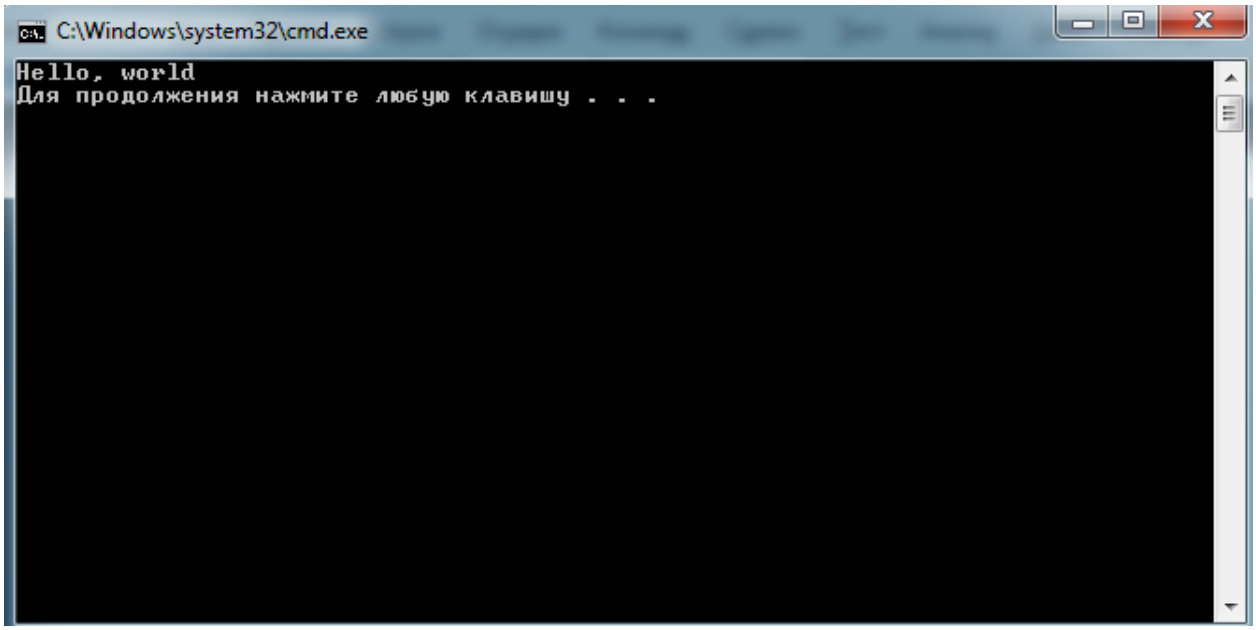

Рис. 4

Повідомлення Press any key to continue (Для продовження натисніть будь-яку клавішу) виводитиметься після результатів, що їх програма обчислила, у разі виконання команди

`Отладка->Запуск без отладки (Ctrl+F5)`

Алфавіт і словник мови програмування

У будь-якій мові програмування програма – це набір зрозумілих компілятору команд. Для створення програм треба знати синтаксис мови, тобто правила запису команд і використання лексичних одиниць мови.

Алфавіт мови програмування – це скінченний набір символів. За допомогою цих символів можуть бути записані ідентифікатори, вирази та оператори мови.

Символи алфавіту мови C\C++ можна поділити на такі категорії:

- символи, що використовуються для складання ідентифікаторів (малі латинські символи, великі латинські символи, десяткові цифри від 0 до 9, символ підкреслення);
- розділовий символ пробілу;
- спеціальні символи, які використовуються в процесі побудови конструкцій мови (+ - * / = > < . , ; : ‘ ( ) [ ] { } ^ @ # $ & | %).

Із символів алфавіту складаються лексичні одиниці мови, або **лексеми** – мінімальні значущі одиниці в текстах програм. Множина всіх допустимих лексем називається **словником** мови програмування.

У мові C\C++ розрізняють такі **види лексем**: спеціальні символи, зарезервовані (ключові) слова, ідентифікатори, неіменовані константи, коментарі та директиви препроцесора.

Лексеми спеціальних символів, окрім спеціальних символів з алфавіту мови, містять ще складені спеціальні символи, що сприймаються компілятором як єдине ціле (<=, >=, +=, *=, ++, ==, -- і т.д.).

Зарезервовані (ключові) слова мають строго визначений зміст. Їх призначення не може змінюватися. Зарезервовані слова використовуються для позначення алгоритмічних конструкцій (циклів, умов, операторів), типів змінних тощо.

Ідентифікатор – це ім’я, значення якого може варіюватися від програми до програми або навіть у межах однієї програми. У мові C\C++ розрізняють стандартні ідентифікатори та ідентифікатори користувача.

Стандартними ідентифікаторами є імена вбудованих у мову функцій (`cin, cout, sin, cos` тощо), типів даних (`int, float, char` тощо) і специфікаторів (`inline, static, virtual` тощо).

Ідентифікатор користувача – це ім’я, яке обирає програміст для позначення (ідентифікації) елементів програми (констант, змінних, типів, функцій). Існують **правила вибору ідентифікаторів**:

- ідентифікатор починається буквою або символом підкреслення;
- ідентифікатор може складатися з букв, цифр, символу підкреслення;
- ідентифікатор може мати довільну довжину, але значущими є тільки перші 63 символи;
- в ідентифікаторі неприпустимо використовувати символи пробілу, крапки та інші символи пунктуації;
- малі та великі літери в ідентифікаторах розрізняються;
- зарезервовані слова не можуть використовуватись як ідентифікатори.

Існують також **рекомендації щодо використання імен у програмах**:

- використовуйте мнемоничні ідентифікатори (такі, що легко запам’ятовуються);
- використовуйте довгі ідентифікатори, що складаються з декількох слів, кожне з яких починається з великої літери, наприклад: CompGraphics, MnemonicCode$;
- замість транслітерації українських і російських слів бажано використовувати їх переклади англійською мовою.

Дотримання цих рекомендацій, нарівні із застосуванням коментарів, зробить текст програми більш зрозумілим.

Коментар – це фрагмент тексту програми, який записується між лексемами /* та */ або після лексеми //. Коментарі ігноруються компілятором та не впливають на роботу програми, але суттєво полегшують її розуміння.

Директива препроцесора – це рядок, що починається символом #. Директиви препроцесора визначають режими компіляції і можуть істотно впливати на зміст згенерованого компілятором машинного коду.

Структура програми

Записана мовами C\C++ програма складається з трьох частин:

- директив препроцесора;
- декларативної (оголошення ідентифікаторів, що використовуються у програмі);
- функцій (іменованих частин програми, які містять запис дій, що виконуються).

Декларативна частина програми, що містить глобальні оголошення, передує функціям, кількість яких визначає розробник.

Самі функції теж містять декларації та оператори мов C\C++. Одна з функцій обов'язково має ім'я `main()` і є точкою входу до програми. Після імені функції обов'язково записують круглі дужки, оскільки в них можуть оголошуватися параметри, що їй передаються. Механізм передачі параметрів у функцію розглядатиметься у подальших лабораторних. Декларації змінних можуть розміщуватися в будь-якому місці програми, де це не заперечено синтаксисом мови. Головне, що слід урахувати розробнику програм, декларації повинні передувати операторам, у яких вони застосовуються. Програма не має оператора, що позначає її ім'я, проте гарним стилем програмування вважається використання коментарів, що описують назву та мету програми.

Приклад 2. Введення двох чисел та обчислення їх суми.

```
// example 2.cpp ввести два числа та визначити їх суму
#include "stdafx.h"                 // Включення заголовних
файлів
#include <iostream>
using namespace std;                // Визначення простору імен
int a;                              // Оголошення змінних
```

```
float b;

void input()                          // Визначення функції
{
     cout << "Enter int a, float b" << endl;        // Вивід
строки з запрошенням вводу даних
     cin >> a >> b;                          // Ввід даних

}

int main()
{
     cout << "Hello, world"<<endl;              // Вивести
повідомлення
     input();                             // Виклик функції
     cout << "a+b= " << a + b << endl;    // Вивід строки з
результатом обчислень
     return 0;
}
```

Пояснімо код програми прикладу 2. На початку програми міститься коментар з назвою та метою створення програми. Він не є обов'язковим, його можна опустити. Він використовується з метою швидкої ідентифікації потрібної програми поміж інших.

Далі записана **директива препроцесора**. Директиви визначають режими роботи компілятора на стадії препроцесорної обробки. У мовах C\C++ означені різні директиви. В прикладах, що розглядатимуться далі, найчастіше застосовується директива `#include`. Директива включення вказує препроцесору, що потрібно додати вміст іншого файла у вихідний файл в точку програми, де директива згадується. Заголовні файли, що використовуються у програмах, написаних мовою C++, не мають розширення, наприклад, файл `iostream`. У директиві `#include` розширення `.h` вказується для тих заголовних файлів, які успадковані від мови C, наприклад, `stdio.h`.

Синтаксис оголошення директиви включення заголовних файлів має вигляд:

```
#include <ім'я заголовного файла>
#include "ім'я заголовного файла"
```

Імена заголовних файлів, що входять до стандартних бібліотек мов C та C++ середовища Visual Studio, записуються у кутових дужках < >, імена заголовних файлів, які створює користувач, записуються у лапках.

Програма має багато ідентифікаторів, які визначені в різних областях дії. Іноді змінна однієї області дії конфліктує із змінною з тим самим ім'ям в іншій області дії. Для запобігання конфлікту імен оголошують простір імен,

який визначає область дії імен. У межах свого простору імен ідентифікатори можуть викликатися саме так, як вони оголошені. Поза свого простору імен ідентифікатори невідомі. Використання імен поза простору імен, в якому вони визначені, можливо із застосуванням імені простору імен і оператора :: розширення області дії.

Для розширення області видимості імен у програмі оголошують директиву `using`. Синтаксис директиви оголошення простору імен такий:

```
using namespace <ім'я простору імен>;
```

Багато імен належать простору `std`, наприклад `cout`, `cin`. В програмах на C++ застосування директиви

```
using namespace <std>;
```

звільняє розробника від дописування імені `std` до імені відповідного оператора, наприклад,

```
std::cout<<"Hello, world";
```

Типи даних

Тип даних визначає:

- множину допустимих значень, яких може набувати змінна або константа зазначеного типу;
- множину допустимих операцій, що застосовуються до даних певного типу;
- спосіб зображення даних у пам'яті комп'ютера.

Цілочислові типи

Цілочислові типи – це типи даних, множини допустимих значень яких є множинами цілих чисел. Ідентифікатори цілочислових типів, множини допустимих значень цих типів та об'єми пам'яті, що потрібні для збереження відповідних даних, наведені у табл. 1.

Табл. 1 – Цілочислові типи

Ідентифікатор типу	Кількість байтів	Діапазон значень
`char`	1	-128..+127 (2^7-1)
`unsigned char`	1	0..255 (2^8-1)
`short`	2	-32768..32768 (2^{15}-1)
`unsigned short`	2	0..65535 (2^{16}-1)
`int`	4	-2147483648.. 2147483647 (2^{31}-1)
`unsigned int`	4	0..4294967295 (2^{32}-1)
`Long`	4	-2147483648.. 2147483647 (2^{31}-1)
`unsigned long`	4	0..4294967295 (2^{32}-1)

Більш детально з типами, що використовуються в системі Visual Studio можна ознайомитися за посиланням

https://msdn.microsoft.com/ru-ru/library/s3f49ktz.aspx

У мовах C\C++ визначені знакові та беззнакові типи даних. У разі, коли нема потреби зберігати від'ємні значення, використовують беззнакові типи даних. Їх позначають з ключовим словом `unsigned`.

Над усіма цілочисловими типами визначено однаковий набір операцій (табл. 2). Ці операції поділяються на унарні (з одним операндом) та бінарні (з двома операндами).

Табл. 2 – Арифметичні операції та операції порівняння

Знак операції	Зміст операції
+	Додавання
-	Віднімання
*	Множення
/	Визначення цілої частини від ділення
%	Визначення остачі від ділення
- (унарний)	Зміна знаку числа
++	Префіксний та постфіксний інкремент (збільшення на 1)
--	Префіксний та постфіксний декремент (зменшення на 1)
<<	Зсув бітів ліворуч (2<<1=4, 1<<5=32)
>>	Зсув бітів праворуч (2>>1=1, 8>>2=2)
==	Дорівнює
!=	Не дорівнює
>	Більше
<	Менше
>=	Більше дорівнює (не менше)
<=	Менше дорівнює (не більше)

Значення типів `char`, `short`, `int`, `long` є знаковими, а типів з ключовим словом `unsigned` – беззнаковими. При додаванні, відніманні та множенні знакових цілих чисел можливе перенесення одиниці зі старшого цифрового розряду в знаковий розряд. Така ситуація називається **переповненням**. Саме це відбувається при додаванні, наприклад, 1 і 32767 для типу `short`. При переповненні комірки оперативної пам'яті формується результат у межах множини значень цілочислового типу, але не обов'язково правильний.

Дійсні типи

Для запису дійсних чисел в оперативній пам'яті використовується формат з **плаваючою комою.**

Для оперування з досить великими числами у прикладних задачах виникає необхідність представити дійсне число у наступному вигляді:

$$Y = \pm M \cdot S^{\pm p}.$$

Тут Y – значення дійсного числа, M – мантиса числа, S – основа системи числення, p – порядок числа.

Числами в формі з плаваючою комою є, наприклад, маса Сонця $2x10^{30}$ кг, або маса електрона $9x10^{-28}$ г.

Кількість цифр у мантисі характеризує точність числа. Чим більше цифр у мантисі, тим вище точність. Порядок визначає місцезнаходження десяткової точки в числі.

У мовах C\C++ означено три дійсних типи (табл. 3): дійсний (`float`), дійсний з подвійною точністю (`double`), дійсний з підвищеною точністю (`long double`). Серед усіх дійсних типів має найширший діапазон і найвищу точність, але потребує при цьому найбільших витрат пам'яті.

Табл. 3 – Дійсні типи даних

Назва	Кількість байтів	Найменше за модулем число	Найбільше за модулем число
`float`	4	$3,4x10^{-38}$	$3,4x10^{38}$
`double`	8	$1,7x10^{-308}$	$1,7x10^{308}$
`long double`	10	$3,4x10^{-4932}$	$1,1x10^{4932}$

Над дійсними числами можна виконувати ті ж самі операції, що і для цілих чисел.

Булів тип

Множина допустимих значень булевого, або логічного, типу містить дві константи: `false` (хибність) і `true` (істина). Назва цього типу даних походить від прізвища видатного англійського математика Джорджа Буля, засновника математичної логіки. Ідентифікатор логічного типу `bool` означено тільки в мові C++. У C\C++ усі значення, що відрізняються від нуля, вважаються істинними.

До булевих значень застосовуються операції «і», «або», «ні», що називаються логічним множенням (кон'юнкцією), логічним додаванням (диз'юнкцією) і запереченням відповідно. Ці операції позначаються лексемами &&, ||, та ! відповідно. Результати застосування цих операцій до булевих значень наведено в табл. 4.

Табл. 4 – Булеві операції

A	B	A&&B	A||B	!A
false	false	false	false	true
Fales	true	false	true	true
true	false	false	true	false
true	true	true	true	false

Оператори умовного переходу

Оператори `if` та `if-else`

Узагальнена форма оператора `if` має вигляд:

```
if (вираз)
     оператор
```

Якщо значення виразу відрізняється від нуля (`true`), тоді `оператор` виконується, якщо вираз дорівнює нулю (`false`) – `оператор` пропускається.

Оператор `if-else` має узагальнену форму:

```
if (вираз)
     оператор 1
else
     оператор 2
```

Якщо вираз відмінний від нуля, тоді виконується `оператор 1`, а `оператор 2` пропускається; якщо вираз – нуль, тоді пропускається `оператор 1` і виконується `оператор 2`.

Оператори циклу

Оператор `while`

Узагальнена форма оператора `while`:

```
while (вираз)
     оператор
```

Спочатку обчислюється `оператор`. Якщо результат відмінний від нуля (`true`), тоді виконується оператор і управління переходить до початку циклу `while`. Тіло цикла, тобто оператор виконується до тих пір, поки вираз не стане рівним нулю (`false`).

Оператор `for`

Оператор `for` – ітераційний оператор, який використовується зі змінною – лічильником циклу. Узагальнена форма оператора `for` має вигляд:

```
for (вираз 1; вираз 2; вираз 3)
     оператор
```

`Вираз 1` представляє собою визначення та ініціалізація лічильника циклу. `Вираз 2` – умова продовження циклу, якщо `вираз 2` відмінно від нуля (`true`) – виконується оператор та обчислюється `вираз 3`. `Вираз 3` – вираз за яким змінюється лічильник циклу (збільшується або зменшується).

Приклад 3. Програма обчислення суми цілих чисел від 0 до 10.

```
sum = 0;
for (i = 1; i <= 0; ++i)
     sum += i;
```

Оператор `do`

Оператор do може розглядатися як варіант оператора while, однак, на відміну від while, цикл do виконує обчислення виразу після виконання оператора. Узагальнена форма має вигляд:

```
do
     оператор
```

```
while (вираз)
```

Таким чином, оператор виконається принаймні один раз.

Спочатку виконується `оператор`, потім обчислюється вираз. Якщо його результат відмінний від нуля (`true`), тоді управління переходить на початок циклу `do`. Якщо значення виразу нуль (`false`), тоді управління переходить до наступного після `do` оператора.

Оператори передачі управління

Оператори `break` і `continue`

Щоб перервати потік управління в циклі використовуються два спеціальних оператори **`break`** і **`continue`.**

Оператор **`break`** виходить з самого вкладеного циклу. На відміну від нього, оператор **`continue`** завершує поточну ітерацію циклу та починає наступну.

Оператор `switch`

Оператор `switch` – умовний оператор, який узагальнює оператор **`if-else`**. Стандартна форма оператора:

```
switch (вираз)
     case вираз 1: оператор 1; break;
     case вираз 2: оператор 2; break;
…
default:
     оператор за замовченням;
```

Завдання до лабораторної роботи

Розробити алгоритм та програму для наведених нижче задач. Передбачити введення даних з клавіатури та виведення результатів на екран консолі.

1. Дано два дійсних числа *a* і *b*. Вивести їх суму, різницю та добуток.
2. Дано довжину ребра куба. Знайти об’єм куба та площу його бокової поверхні.
3. Дано два дійсних додатних числа. Обчислити середнє арифметичне та середнє геометричне цих чисел.
4. Дано катети прямокутного трикутника у вигляді двох дійсних чисел. Обчислити гіпотенузу та площу трикутника.
5. Обчислити силу тяжіння F між двома тілами масами m_1 і m_2, які знаходяться на відстані R.
6. Дана довжина окружності. Знайти площу круга, обмеженого окружністю.

7. Дано два дійсних числа *x* і *y*. Обчислити *min(x,y), max(x,y).*
8. Дано дійсні числа *x, y, z*. Обчислити
   - *max(x+y+z,xyz)*;
   - $min^2(x+y+z/2,xyz)+1$.
9. Дано дійсні числа x і y .Обчислити z:

$$z = \begin{cases} x - y, & якщо\ x > y, \\ y - x + 1, & у\ протилежному\ випадку. \end{cases}$$

10. Обчислити чи є задане число парним.
11. Дано натуральне число *n*. Обчислити:
    - *2n*;
    - *n!*;
    - $(1+\frac{1}{1^2})(1+\frac{1}{2^2})\ldots(1+\frac{1}{n^2})$.
12. Дано натуральне число *n*.
    - Скільки цифр в числі *n*?
    - Чому дорівнює сума його цифр?
    - Знайти першу цифру числа *n*.
13. Дано натуральне число *n*. Знайти добуток всіх його натуральних дільників.

Контрольні запитання

1. Як сформувати складену умову, що об’єднує декілька простих умов?
2. Дайте характеристику умовних операторів.
3. У чому полягає відмінність між операторами циклів `for`, `while`, `do… while`?
4. Що визначає тип даних? Які типи даних підтримує мова C\C++?
5. У чому різниця між типами `char`, `int`, `long`?