**Практична робота №6.** Розробка простої програми з блоками, операціями порівняння при використанні математичних формул та операціями введення та виведення виразів.

**Мета:** навчитись складати прості програми з Розробка простої програми з блоками, операціями порівняння при використанні математичних формул та операціями введення та виведення виразів.

**Хід роботи**

1. Ознайомитися з теоретичними відомостями лекції №8-9.
2. Запустити в Інтернет-браузері он-лайн компілятор C++Shell за посиланням <http://cpp.sh/> або використати інший доступний вам компілятор С++
3. Продовжуємо працювати з програмою, складеною на минулій ПР.
4. Перед блоком, створеним у п.3 виведіть охайне повідомлення такого змісту: **Цю програму створив студент … групи … у … році**. Замість "**…**" вставте особисті дані (прізвище, ім'я та по-батькові, номер групи й рік написання програми).
5. В цій же програмі далі перед розрахунком розрахунок значень z1, z2, z3 за відповідними формулами провести перевірку а) на ділення на нуль і при нульовому знаменнику присвоїти змінній значення -555, б) на від‘ємне підкорінне значення і при від‘ємному значенні присвоїти змінній значення -777.

Вивести результати таким чином:

* Перший рядок заголовок"Results".
* Другий рядок через табуляцію виводяться номери формул (1, 2 3).
* Третій рядок - через табуляцію результати розрахунку з використанням маніпуляторів **setw(w)** та **setprecision(d**), обравши довжину поля 7, кількість позицій у дробовій частині дійсних чисел – 2.

1. Перший рядок програми повинен містити коментар з номером ЛР, номером групи та прізвищем студента.
2. Значення змінних вводяться з консолі.
3. Програма компілюється, результати перевіряються на правильність. Програму збережіть для подальшої роботи з удосконалення.
4. Результати (код програми, скріншот виконання) оформити, надавши титульний лист, завдання, та результати, а також окремо код програми у вигляді cpp-файлу з іменем у форматі

**<Номер групи><Номер лабораторної><Прізвище>**

Наприклад, 23-04Ivanov.cpp

надсилати на електронну адресу викладача [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)у вигляді cpp-файлу з іменем у форматі

**<Номер групи><Номер лабораторної><Прізвище>**

Наприклад, 23-04Ivanov.cpp.

Іншим рішенням є надсилання поштою посилання на текст програми за URL адреси, яку надає C++Shell, вказавши в темі листа, номер групи прізвище студента та номер ЛР.

**Строк відсилки ЛР для ІПЗ-31 26.09.2024**

**ІПЗ-32 25.09.2024**

**ІПЗ-33 25.09.2024**

Всі запитання, що виникнуть, надсилайте на електронну адресу викладача.

В темі листа вказати, номер групи, прізвище студента та номер ПР як "ПР№6" В темі листа вказати, номер групи, прізвище студента, номер ПР та фразу "Запитання".

**ТЕОРЕТИЧНА ЧАСТИНА**

**Макровизначення.**

***Макровизначення*** (*макропідстановка* або просто *макрос*)– це вираз, який при компіляції файлу з кодом програми підставляється замість символьного імені, що визначає дану макропідстановку. Макроси дозволяють замінити великий текст коротким ім'ям макросу.

Визначаються макроси директивою препроцесора ***#define***. Формат завдання:

***#define ім'я\_макросу текст\_макросу***

Перед компіляцією замість імені макроса в програму підставляється його значення.

Директива **#define** – слугує для заміни у всіх місцях коду програми замість вказаної послідовності символів обрану користувачем. За негласною домовленість слова підстановки записують великими літерами, для візуального виділення у програмі.

Наприклад,

#define MAXNUMBER 100  
У всіх місцях коду замість MAXNUMBER буде підставлено 100.  
Тобто при компіляції рядок  
if (a>MAXNUMBER) { … }  
Буде замінено на:  
if (a>100) { … }

#define SIMPLETEXT “lorem ipsum dolor sit amet”  
Аналогічно замість:  
printf(SIMPLETEXT);  
printf(“lorem ipsum dolor sit amet”);

#define РІ 3.14159

Дана директива вказує замінити всі появи у тексті програми символьної константи РІ на чисельну константу 3.14159. Якщо є необхідність змінити значення такої константи у всій програмі, то достатньо це зробити в одному місці програми, в директиві #define, і після повторної компіляції програми всі включення константи в програму будуть автоматично замінені.

Дозволяються також макроси з параметрами, останні підставляються в текст заміни, після чого макрос розширюється, тобто в програму підставляється текст заміни замість ідентифікатора і списку параметрів. Наприклад, для макровизначення з одним аргументом для знаходження площі круга:

#define CIRCLE\_AREA (x) ((PI)\*(x)\*(x))

Кожен раз, коли в програмному коді з’явиться CIRCLE\_AREA (x), значення х підставляється замість х в тексті заміни, символьна константа РІ автоматично заміняється її значенням (визначеним раніше), і макрос розширюється в програмі. Наприклад, оператор

area=CIRCLE\_AREA (4)

розширюється в

area=((3.14159)\*(4)\*(4)).

Макроси є спадщиною мови С, і в С++, зазвичай, замінюються більш безпечними можливостями мови, такими, як шаблони, inline-функції, перерахування enum, перевизначення типів typedef.

Символьні константи і макроси можна відмінити, використовуючи директиву препроцесора ***#undef***. Ця директива відміняє визначення символьної константи або імені макроса. Область їх дії продовжується від місця визначення в програмному коді до місця відміни визначення за допомогою директиви #undef, чи до кінця файлу, якщо дана директива відсутня. Після дії директиви #undef дозволяється повторно визначати макрос за допомогою директиви #define.

**Директиви умовної компіляції**.

Умовна компіляція дозволяє управляти виконанням директив препроцесора і компіляцією програмного коду. Директиви умовної компіляції: #if, #elif, #else та #endif. Кожна конструкція #if закінчується #endif. Умовні конструкції препроцесора, що перевіряють декілька варіантів, реалізуються за допомогою #elif (еквіваленту else if структури if) і #else (еквіваленту else структури if). Порядок використання директив умовної компіляції:

**#if константний\_вираз\_1**

**…**

**#elif константний\_вираз\_2**

**…**

**#elif константний\_вираз\_3**

**#else**

**…**

**#endif**

Кожна із умовних директив препроцесора оцінює значення цілочисельного виразу. Якщо цілий константний вираз у директиві #if має ненульове значення (TRUE), то при компіляції включаються всі наступні рядки до #elif або #endif або #else (elif діє як гілка else-if). Наприклад, змінюючи константу VERSION, можна керувати включенням файлів:

#define VERSION 3

#if VERSION == 1

#define INCLUDE\_FILE "file\_1.h"

#elif VERSION == 2

#define INCLUDE\_FILE "file\_2.h"

#else

#define INCLUDE\_FILE "file\_3.h"

#endif

#include INCLUDE\_FILE

**Зовнішні оголошення**

*Зовнішні оголошення* – це глобальні описи даних (змінних або констант). Змінна у програмуванні є моделлю (зображенням) об’єкта в пам’яті комп’ютера. На фізичному рівні поняттю змінної відповідає група комірок оперативної пам'яті. Ці адреси ставляться у відповідність ідентифікаторам (іменам) змінних під час їх оголошення. Таким чином, ім'я змінної *вказує* (або *посилається*) на першу комірку з групи, а величину групи визначає тип змінної (рис. 1).

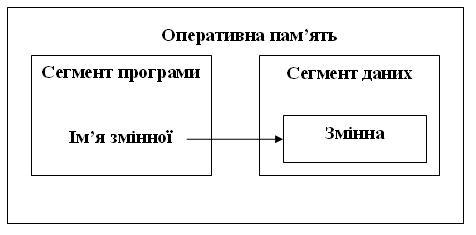


Рис. 1. Ідентифікатор змінної та його асоціація з коміркою пам'яті

**Функція** – це частина програми, що виконує логічно завершений набір дій. В C/С++ вона є єдиним способом представлення підпрограм. Формат оголошення функції:

**тип\_результату ім'я([параметри]) *// заголовок функції***

**{ … } *// тіло функції***

Тіло функції може містити описи констант, змінних і оператори мови, які відокремлюються символом «;». Описи даних можуть зустрічатися в будь-якому місці функції, але до першого звертання до змінної. Якщо параметри у функції відсутні, то в заголовку все рівно записуються круглі дужки. Одна з функцій програми на мові C/С++ повинна мати ім'я main***.*** Саме з неї починається виконання програми. Функція main визначає дії, що виконуються програмою, і викликає інші функції. Якщо програма містить тільки одну функцію, то вона і є головною (має ім'я main).

Програма може містити довільне число директив препроцесора, оголошень і визначень змінних, описів функцій. Порядок появи цих елементів у програмі досить істотний: він впливає на можливість використання змінних, функцій і типів у різних частинах програми.

Логічні операції та операції порівняння в мові Сі оперують поняттями "істина" і "хиба

**Операції порівняння**

У мові С++ можна використовувати операції порівняння:

|  | **операція** | **відповідник** | **приклади** |
| --- | --- | --- | --- |
| Менше  Менше або дорівнює  Більше  Більше або дорівнює | <  <=  >  >= | <  ≤  >  ≥ | a < b  a<=b  a>b  a>=b |
| Дорівнює  Не дорівнює | ==  != | =  ≠ | a==b  a!=b |

Порівнювати можна операнди будь-якого типу, але вони повинні бути того ж самого вбудованого типу даних (порівняння на рівність і нерівність працює для двох величин будь-якого однакового типу даних), або між ними повинна бути визначена відповідна операція порівняння. Результат – логічне значення true або false. Використовуються для порівняння значень і дають відповідь false (0) true (1).

Операнди можуть бути символи, логічні, числові, покажчики.

Умова, яка перевіряється, може містити значення, яке не є логічним. Тоді значення нуль (0) сприймається як false, а значення  ≠0 - як true. Наприклад:

**int a = 10;**

**int b = 4;**

**int c = a == b; // 0**

**int c = 10 <= 4; // 0**

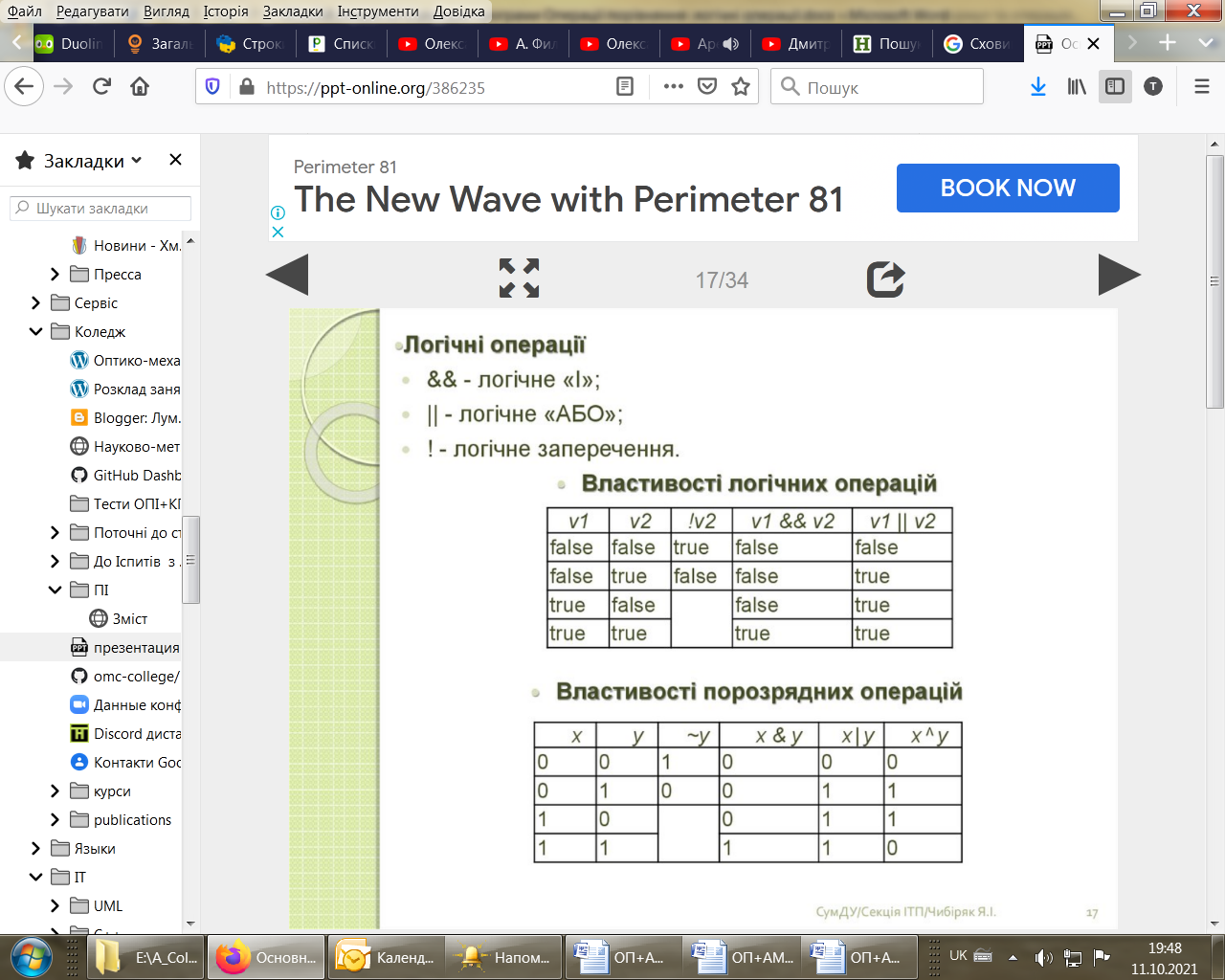
**Логічні операції**

Логічні операції реалізують операції математичної логіки. Серед логічних операцій є унарні й бінарні.. Дають відповідь типу bool:  false (0)  та true (1).

| **Операція** | **Відповідник** | **Приклади** |
| --- | --- | --- |
| ! (унарна) | not | !b, !(c>1 && c<9) |
| &&(бінарна) | and | a && b, a>1 && c<9 |
| || (бінарна) | or | a||b, c<0 || c>10 |

Операнди логічних операцій можуть бути логічні змінні та відношення.

Правила виконання логічних операцій



**Логічні вирази**

Логічні (булівські) вирази складаються з  логічних операцій та операндів - логічних змінних і значень та відношень, з урахуванням пріоритетів. Для зміни порядку обчислень можна використовувати круглі дужки. Результат виконання має тип ***bool***. Правила побудови загальні - лінійна форма написання, явно вказуються знаки операцій та операнди без дублювання.

Приклади запису логічних виразів:

**x<0 || x>11**

**(i!=0 ||i>100) && (j!=i || j>0)**

**x!=0 && x!=10 && x!=100**

Вираз може містити операнд, який не є логічним. Тоді значення нуль (0) сприймається як false, а значення  ≠0 - як true. Приклади:

**// логічні операції**

**bool res;**

**int a, b;**

**// операція && (AND)**

**a = 8;**

**b = 5;**

**res = a && b; // res = True**

**a = 0;**

**res = a && b; // res = False**

**// операція || (OR)**

**a = 0;**

**b = 0;**

**res = a || b; // res = False**

**b = 7;**

**res = a || b; // res = True**

**// операція ! (логічне "НІ")**

**a = 0;**

**res = !a; // res = True**

**a = 15;**

**res = !a; // res = False**