**Практична робота 08. Розробка програми з використанням вкладених блоків лінійної, розгалуженої та циклічної структури.**

**Мета**: навчитися писати програми лінійної, розгалуженої та циклічної структури з вкладеними циклами та використанням математичних функцій та виконанням операцій введення - виведення **на мові С++** в **консольному режимі.**

**Методичні вказівки щодо організації самостійної роботи студентів**

1. Запустити середовище програмування С++ .
2. Продовжуємо працювати з кодом програми, розробленим в попередній ЛР. Додати до програми ще 3 завдання з пп.3.1-3.3. В першому рядку цієї програми дописати номер цієї ЛР в коментарі:

*// Група № Прізвище Номер ЛР*

де вказаний номер групи, прізвище та номер ЛР.

Програма повинна запитати подальші номери завдання (число 3, 4 для 3.1, та 3.2, відповідно, або 0 для закінчення) і в залежності від введеного значення виконувати відповідне завдання. Якщо введений 0 – програма припиняє роботу.

Вхідні дані ввести, а результати вивести, використовуючи потокове введення-виведення даних. Виводяться (змінні ***a*** або ***N*** перед початком розрахунку*, а* ***х, y*** та відповідні повідомлення на кожній ітерації/кроку циклу***).***

1. Завдання:
   1. Написати код, що виконує розрахунок ***y*** підсумку ***N*** перших членів з точністю ε=10-3:



Підсумок позначити як Y, значення ***N*** ввести перед початком роботи, Х можна задати ініціалізацією, але X>0.

Значення ***N*** вводиться з консолі і не повинне мати обмеження.

Проаналізуйте вираз підсумку та складіть компактну формулу розрахунку в циклі.

* 1. Написати код, що виконує розрахунок за формулами, наданими в ОП+АМ.ПР 08.Завдання..docx:

Код обчислює та виводить на екран у табличному вигляді значення функції *f(x)* на заданому інтервалі зміни значень аргументу *х* від *xпоч* до *xкін* з кроком *h.*

Коефіцієнти *a, b, c –* дійсні числа. Значення *a, b, c* задаєте ініціалізацією, *xпоч*, *= -5.0, xкін = 5.0*, *h=1.0*. Передбачити перевірку допустимості введених значень і вивести відповідне повідомлення щодо можливого ділення на нуль, або випадку, коли підкорінний вираз квадратного кореня від‘ємний.

Вхідні дані ввести, а результати вивести, використовуючи потокове введення-виведення даних.

1. Запустити програму на виконання. Перевірити всі варіанти виконання завдання. При відсутності помилок текст програми надіслати викладачу.
2. При необхідності проведення консультації з питань виконання лабораторних робіт відправте відповідне повідомлення із запитаннями на електронну пошту викладача.

Результати 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 англійською>**

Наприклад, 21-01Ivanov.cpp.

В темі листа вказати, номер групи, прізвище студента та номер ПР як "ЛР№7".

**Строк відсилки ЛР для ІПЗ-21 12.10.2024**

**ІПЗ-22 15.10.2024**

**ІПЗ-23 11.10.2024**

Всі запитання, що виникнуть, надсилайте на електронну адресу викладача, тему в заголовку листа записати

**ОП+АМ-Запитання-<Номер групи>-<Прізвище >**.

**Запитання**

1. Які оператори розгалуження ви знаєте?
2. Які оператори циклу ви знаєте?
3. Яка бібліотека С++ використовується для програмування математичних формул?
4. Як підключити математичні константи до програми С++?
5. Що включає в себе алфавіт мови С++?
6. Що таке лексеми, як вони складаються, що до них відносять?
7. Дайте визначення ідентифікатору, яким чином він утворюється?
8. Що називають ключовими словами?
9. Які символи можуть бути роздільниками лексем?
10. Які коментарі використовує мова C++? Надайте поради до їх створення, наведіть приклади.
11. Що являє собою структура програми на C++? Наведіть основні вимоги, які слід ураховувати при створенні програм мовою C++.
12. Які основні та складені типи даних має мова С++?
13. Що таке змінна і як здійснюється її опис та визначення?
14. Що таке «область дії ідентифікатора» та «клас пам’яті»?
15. Які константи налічує С++? Охарактеризуйте їх застосування.
16. Що таке пріоритет операції? Наведіть приклади арифметичних та логічних операцій.
17. Які операції присвоювання та операції відношення налічує С++?
18. Що реалізують логічні операції та операції обробки окремих бітів?
19. Які операції над покажчиками та додаткові операції має С++?
20. Що таке змінна і як здійснюється її опис та визначення?
21. Що таке «область дії ідентифікатора» та «клас пам’яті»?
22. Які константи налічує С++? Охарактеризуйте їх застосування.
23. Що таке пріоритет операції? Наведіть приклади арифметичних та логічних операцій.
24. Які операції присвоювання та операції відношення налічує С++?
25. Як забезпечується організація введення даних на С++?
26. Як здійснюється виведення результатів на С++?
27. Як реалізується потокове введення даних «**сіn>>**»? Наведiть приклади.
28. Охарактеризуйте основні аспекти використання потокового ви ведення даних «**соut<<**».
29. Що таке форматоване введення-виведення даних?
30. Поясніть правила застосування функції **scanf**.
31. Як здійснює роботу функція **printf**?
32. Для чого слугують заголовні файли?
33. Як компілятор шукає розташування заголовних файлів?
34. Які маніпулятори використовуються для завдання довжини поля, дробової частини числа?
35. Який заголовний файл використовується для маніпуляторів завдання довжини поля, дробової частини числа?
36. Визначте основні частини типової структури програми на С++.
37. Що таке макровизначення? Наведіть приклади.
38. Для чого використовується умовна компіляція?
39. Що може міститися в зовнішніх оголошеннях?
40. Надайте визначення функції.
41. Які значення може містити умова, що перевіряється?
42. Як визначається результат операція **not?**
43. Як визначається результат операція **and?**
44. Як визначається результат операція **or?**
45. Поясніть, що представляє собою логічний вираз.
46. Як визначається тернарна операція?
47. Які оператори реалізують розгалуження у програмі?
48. Як діє умовний оператор if?
49. Як працює оператор-перемикач switch?
50. Які оператори циклу використовуються у C++?
51. Як працює оператор циклу for?
52. Пояснити на прикладах використання циклу з передумовою і циклу з післяумовою.
53. Які оператори з розглянутих є операторами керування у мові C++?

**Представлення даних в пам’яті комп'ютера**

<https://studopedia.net/1_905_predstavlennya-danih-v-pamyati-kompyutera.html> (посилання не працює)

Внутрішнє представлення змінної цілого типу — ціле число у двійковому коді. Згідно формату IEEE всі додатні цілі числа зберігаються в пам'яті комп'ютера в прямому коді, а всі від'ємні – в доповняльному коді. Цілі числа зберігаються в пам'яті комп'ютера у зворотньому порядку розміщення байт числа.

Приклад 1.

Додатнє число 25 типу **int** в пам'яті комп’ютера зберігається в прямому двійковому коді і займає 4 байти: 2510, 1916, 110012

0000 0000 0000 0000 0000 0000 0001 1001

В пам’яті комп’ютера *цілі числа зберігаються у зворотному порядку* розміщення байт числа:

0001 10001 0000 0000 0000 0000 0000 0000

Результат в 16-ковій системі числення: 19 00 00 00.

Приклад 4.

Від'ємне число -3500 типу **long int** в пам'яті комп’ютера зберігається в доповняльному двійковому коді і займає 4 байти: - 350010 = - DAC16 = - 1101 1010 11002

0000 0000 0000 0000 0000 1101 1010 1100 - прямий код;

1111 1111 1111 1111 1111 0010 0101 0011 - обернений код;

         +                                                             1

             1111 1111 1111 1111 1111 0010 0101 0100 - доповняльний код;

                F  F  F F  F  2   5  4 - в 16- ковій системі числення

 В пам’яті комп’ютера зберігається у зворотному порядку розміщення байт числа:

0101 0100 1111 0010 1111 1111 1111 1111

 Результат в 16- ковій системі числення: 54 F2 FF FF.

**Дійсні типи даних**

Дійсні константи записуються у двох формах – з фіксованою десятковою крапкою або в експонентному виді. В першому випадку крапка використовується для поділу цілої і дробової частин константи. Як ціла, так і дробова частини можуть бути відсутніми (наприклад 1.2, 0.725, 1., .35, 0.). В трьох останніх випадках відсутня або дробова, або ціла частина. Десяткова крапка повинна обов'язково бути присутньою, інакше константа буде вважатись цілою.

Експонентна форма запису дійсної константи містить знак, мантису і десятковий порядок (експоненту). Мантиса – це будь-яка додатня дійсна константа у формі з фіксованою крапкою або цілою константою. Порядок вказує степінь числа 10, на яку домножується мантиса. Порядок відокремлюється від мантиси буквою 'E' або 'e' (від слова exponent). Порядок може мати знак плюс або мінус, у випадку додатнього порядку знак плюс можна опускати. Наприклад:

1.5e+6 - константа еквівалентна 1500000.0

1e-4  - константа еквівалентна 0.0001

-.75E3 - константа еквівалентна -750.0

У мові С++ дійсні типи або типи з рухомою комою представляються трьома розмірами, що характеризують точність представлення дійсних чисел:

**float** – одиничної точності;

**double** - подвійної точності;

**long double** – розширеної точності (у деяких реалізаціях тип long double може бути відсутній)

Константи з рухомою комою мають за замовчуванням тип double. Саме він є найбільш природнім для комп'ютера. У програмуванні треба по можливості уникати типу float, тому що його точність недостатня, а процесор однаково при виконанні операцій перетворить його в тип double. Один з випадків, де застосування типу float виправдане – тривимірна комп'ютерна графіка.

Можна явно вказати тип константи за допомогою суфіксів F, f (float) і L, l (long). Наприклад, константа 2E+6L буде мати тип long double.

В  пам'яті  комп'ютера  змінна  типу float займає 4 байти, в яких один біт виділяється під знак, 8 – під порядок, 23 – під мантису.

Розряди мантиси включають один розряд цілої частини, що завжди дорівнює одиниці, і фіксовану кількість розрядів дробової частини. Оскільки старший двійковий розряд мантиси завжди дорівнює одиниці, зберігати його необов'язково, і у двійковому коді він відсутній. Фактично двійковий код зберігає тільки розряди дробової частини мантиси. Отже, насправді, у типу float мантиса містить 24 розряди, але старший розряд завжди дорівнює одиниці, тому зберігати його не потрібно.

Тип double займає 8 байт, у яких один розряд виділяється під знак, 11 – під порядок, 52 – під мантису. Насправді в мантисі 53 розряди, але старший завжди дорівнює одиниці і тому не зберігається.

Тип long double займає 10 байт (або 8 байт), в яких один розряд виділяється під знак, 15 – під порядок, інші 64 – під мантису. Записуються всі 64 розряди мантиси разом зі старшою одиницею.

Оскільки порядок може бути додатній і від'ємний, у двійковому коді він зберігається в **зміщеному виді**: до нього додається константа, яка рівна абсолютній величині максимального по модулю від'ємного порядку. У випадку типу float вона дорівнює 127, у випадку double – 1023, long double – 16383. Таким чином, максимальний по модулю від'ємний порядок представляється нульовим кодом.

Дійсні числа зберігаються в пам'яті комп'ютера у зворотньому порядку розміщення байт числа.

Таблиця 4.7

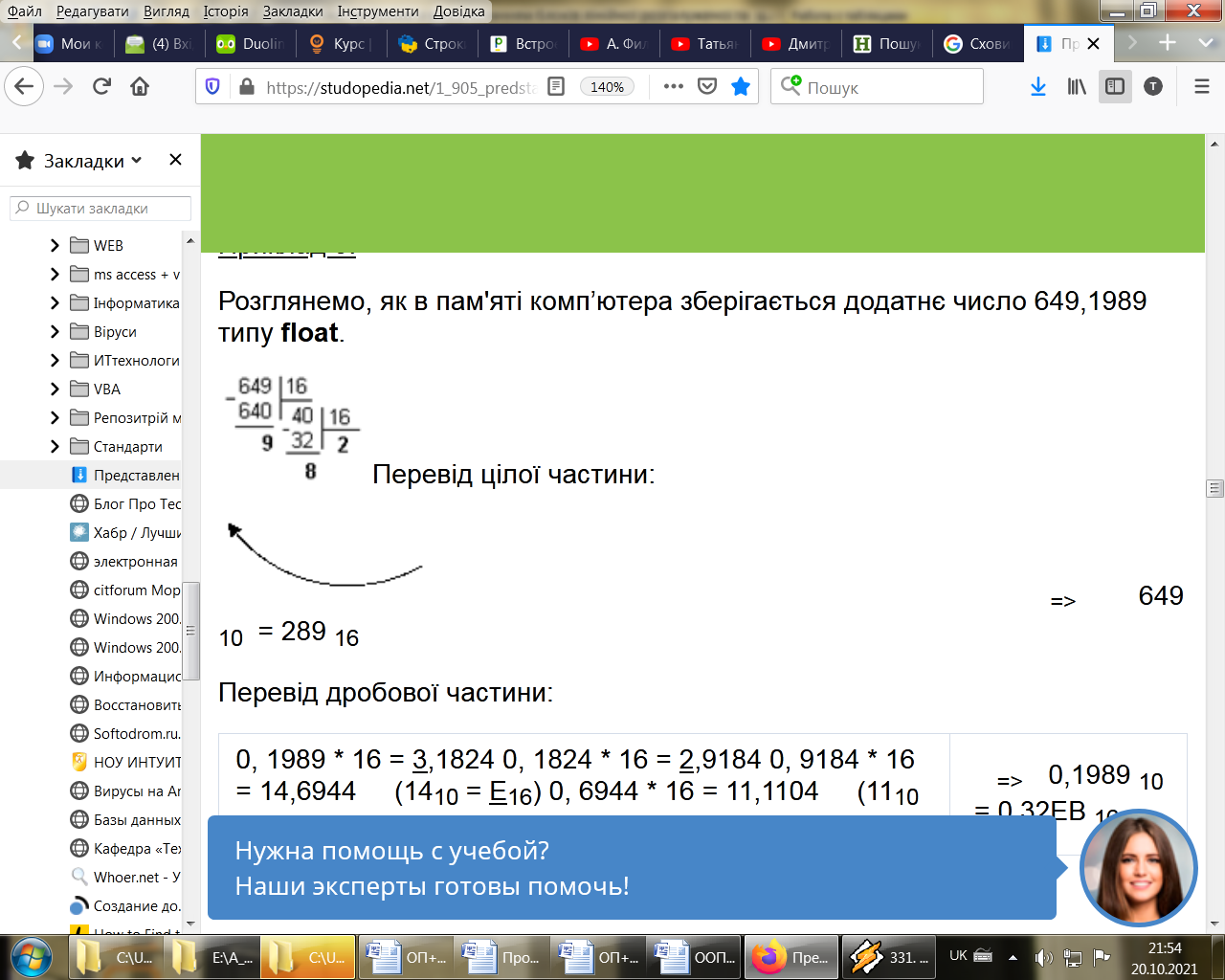
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва типу | Іденти-фікатор | Діапазон значень | Внутрішній формат: s–знак,e–експонента,m–мантиса | Значення числа | Розмір пам’яті в байтах |
| Дійсне одинарної точності | float | від 3.4 10-38 до 3.4 \*1038 | 1 біт 8 біт        23 біта   |  |  |  |  |  | | --- | --- | --- | --- | --- | | s | e |  | m |  | | (-1)S \*1,m \*2e -127 | 4 |
| Дійсне подвійної точності | double | від 1.7 10-308 до 1.7 \*10308 | 1 біт 11 біт       52 біта   |  |  |  |  |  | | --- | --- | --- | --- | --- | | s | e |  | m |  | | (-1)S \*1,m \*2e –1023 | 8 |
| Дійсне підвищеної точності | long double | від 3.4 10-4932 до 3.4 \*104932 | 1 біт 15 біт 1біт 63 біта   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | s | e | 1 |  | m |  | | (-1)S \*1,m \*2e -16383 | 10 (8)\* |

\* Розмір типу long double для різних компіляторів може відрізнятися, наприклад для BCB 6.0 буде займати 10 байт, для VC++ 6.0 буде займати 8 байт (в цьому випадку формат збереження типу long double співпадає з форматом типу double).

Приклад 5.

Розглянемо, як в пам'яті комп’ютера зберігається додатнє число 649,1989 типу **float**.

Перевід цілої частини:



Перевід дробової частини:

|  |  |
| --- | --- |
| 0, 1989 \* 16 = 3,1824 0, 1824 \* 16 = 2,9184 0, 9184 \* 16 = 14,6944     (1410 = E16) 0, 6944 \* 16 = 11,1104     (1110 = B16) | => 0,1989 10  = 0,32EB 16 |

 Отже: 649,1989 10 = 289,32EB 16  = 0010 1000 1001 , 0011 0010 1110 1011 2

Нормалізація: 001 , 0 1000 1001 0011 0010 1110 1011 2 \* 101001

Заокруглення:

1 , 0 1000 1001 0011 0010 1110 10│11

+                                                           1

1 , 0 1000 1001 0011 0010 1110 11

Визначення мантиси: m=0 1000 1001 0011 0010 1110 11

Визначення зміщеного порядку: е = 12710 + 910  = 136 10  =  88 16  = 1000 1000 2

Інший спосіб: 12710   =    111  1111 2

                                   +           1001

                                      1000  1000 2

Визначення знакового розряду: s=0 (бо число додатнє).

Схема внутрішнього представлення:

|  |  |  |
| --- | --- | --- |
| s | e | m |
| 1 біт | 8 біт | 23 біт |

Зборка за схемою:

|  |  |  |
| --- | --- | --- |
| s | e | m |
| 0 | 1000 1000 | 0 1000 1001 0011 0010  1110 11 |

В 16- ковій системі числення: 0100 0100 0010 0010 0100 1100 1011 1011 2= 44 22 4С BB 16

В пам’яті комп’ютера буде зберігатися у зворотному порядку розміщення байт числа:

                                      1011  1011  0100  1100 0010 0010 0100 0100

Результат в 16- ковій системі числення: BB 4С 22 44

Приклад 6.

Розглянемо, як в пам'яті комп’ютера зберігається від'ємне число – 649,1989 типу **float**.

Різниця з попереднім прикладом буде полягати тільки у представленні знакового розряду.

Визначення знакового розряду: s=1 (бо число від'ємне).

Зборка за схемою (відміну від попереднього прикладу будемо виділяти жирним шріфтом):

|  |  |  |
| --- | --- | --- |
| s | e | m |
| 1 | 1000 1000 | 0 1000 1001 0011 0010 1110 11 |

В 16- ковій системі числення:  **1**100 0100 0010 0010 0100 1100 1011 10112 = **С**4 22 4С BB 16

В пам’яті комп’ютера буде зберігатися у зворотному порядку розміщення байт числа:

1011 1011 0100 1100 0010 0010 **1**100 0100

Результат в 16- ковій системі числення: BB 4С 22 **С**4