Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Звіт

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 8

з навчальної дисципліни

“Базові методології та технології програмування”

ПРОГРАМНА РЕАЛІЗАЦІЯ ОБРОБЛЕННЯ МАСИВІВ ДАНИХ ТА СИМВОЛЬНОЇ ІНФОРМАЦІЇ

ЗАВДАННЯ ВИДАВ

доцент кафедри кібербезпеки та програмного забезпечення

Доренський О. П.

[https://github.com/odorenskyi/](https://github.com/odorenskyi/Dmytro-Parkhomenko-KB18)

ВИКОНАВ

студент академічної групи КН-23

Чубенко І. К.

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

Поліщук Л. І.

Кропивницький 2024

Тема: Реалізація статичних бібліотек модулів лінійних обчислювальних процесів .

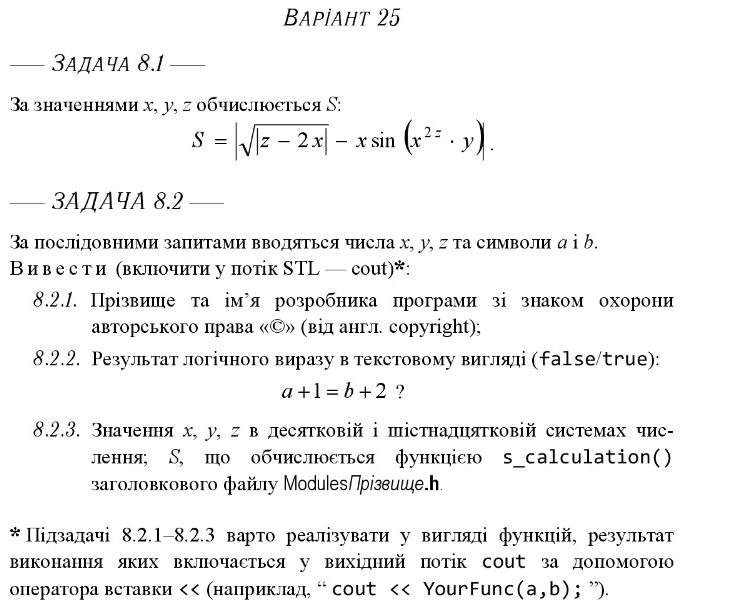
Мета роботи: полягає у набутті ґрунтовних вмінь і практичних навичок застосування теоретичних положень методології модульного програмування, реалізації метода функціональної декомпозиції задач, метода модульного (блочного) тестування, представлення мовою програмування С++ даних скалярних типів, арифметичних і логічних операцій, потокового введення й виведення інформації, розроблення програмних модулів та засобів у кросплатформовому середовищі Code::Blocks (GNU GCC Compiler).

**ЗАВДАННЯ:**

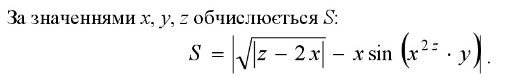
1. Реалізувати статичну бібліотеку модулів libModulesПрізвище C/C++, яка містить функцію розв’язування задачі 8.1.

2. Реалізувати програмне забезпечення розв’язування задачі 8.2 — консольний застосунок.

Постановка задачі за заданими значеннями x,y,z обчислити S.

Варіант 25 

8.1)



Вхідні x,y,z x>y; >0; y!=0

Вивести: S

Алгоритм

1.Вести x,y,z

2.Якщо x>y;

>0; y!=0 то



Інакше

S=” Результат обчислень невизначений або недійсний.”

3.Вивести S

4.Кінець

Лістинг

Модуля

#include <iostream>

#include <cmath>

#include <iomanip>

#include <locale.h>

using namespace std;

void s\_calculation(int x, int y, int z)

{

setlocale(LC\_ALL,"UA");

float S;

if (x > y && y != 0){

S = abs(abs(sqrt(z - 2 \* x)) - x \* sin(pow(x, 2 \* z)) \* y);

cout << "S: " << setprecision(3) << fixed << S << endl;

}

else {

cout << "Результат обчислень невизначений або недійсний." << endl;

}

}

Лістинг тестового драйвера

#include <iostream>

#include <Windows.h>

#include"ModulesChubenko.h"

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "TC\_01: ";

s\_calculation(5, 3, 10);

cout << "TC\_02: ";

s\_calculation(28, 22, 76);

cout << "TC\_03: ";

s\_calculation(8, 2, 16);

cout << "TC\_04: ";

s\_calculation(3, 2, 50);

cout << "TC\_05: ";

s\_calculation(6, 3, 20);

cout << "TC\_06: ";

s\_calculation(9, 5, 22);

cout << "TC\_07: ";

s\_calculation(11, 7, 28);

cout << "TC\_08: ";

s\_calculation(12, 4, 30);

cout << "TC\_09: ";

s\_calculation(15, 10, 36);

cout << "TC\_10: ";

s\_calculation(14, 9, 40);

return 0;

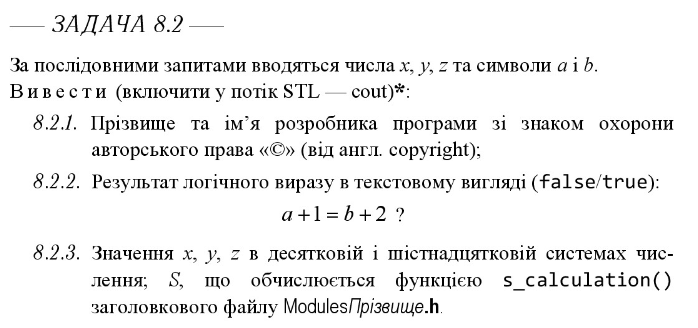
}

Результати виконання тестового драйвера

|  |  |  |  |
| --- | --- | --- | --- |
| ID TS | Вхід | Вихід | Результат тестування /  Test Result |
| TC\_01 | x = 5, y = 3, z = 10 | 9.524 | passed |
| TC\_02 | x = 28, y = 22, z = 76: | 58.853 | passed |
| TC\_03 | x = 8, y = 2, z = 16 | 5.383 | passed |
| TC\_04 | x = 3, y = 2, z = 50 | 12.257 | passed |
| TC\_05 | x = 6, y = 3, z = 20 | 6.067 | passed |
| TC\_06 | x = 9, y = 5, z = 22 | 8.144 | passed |
| TC\_07 | x = 11, y = 7, z = 28 | 70.075 | passed |

|  |  |  |  |
| --- | --- | --- | --- |
| TC\_08 | x = 12, y = 4, z = 30 | 0.980 | passed |
| TC\_09 | x = 1, y =1 , z = 1 | Результат обчислень невизначений або недійсний. | passed |
| TC\_10 | x = 14, y = 9, z = 40 | 44.484 | passed |

8.2)



Вхідні int x,y,z ; -100<=x,y,z<=100

Char a,b

Вивести: Прізвище та ім’я розробника програми зі знаком охорони авторського права «©»;

a + 1 => b+2;



Алгоритм

1.Вести x,y,z,a,b

2.Вивести : Прізвище та ім’я розробника програми зі знаком охорони авторського права «©»

3.Якщо a + 1 => b+2;

То вивести 1

Інакше 0

4.Вивести x,y,z

5. Вивести x,y,z в шістнадцятковий системі

6.Вивести s

7.Кінець

Лістинг

#include <iostream>

#include "ModulesChubenko.h"

#include <Windows.h>

using namespace std;

bool getBoolResult(char a, char b)

{

return (a + 1 >= b + 2);

}

int getHexadecimal\_16(int number)

{

cout << hex;

return number;

}

void clearScreen() {

system("cls");

}

void printCopyright()

{

cout << " -----------------------------------------------\n\

| Chubenko Ihor, CNTU, https://github.com/IgorChubenkoKN-23 |\n\

| Chubenko Ihor, CNTU, https://github.com/IgorChubenkoKN-23 |\n\

------------ © All rights reserved ------------\n" << endl;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("chcp 1251 > nul");

int x;

cout << "Введіть x: ";

cin >> x;

int y;

cout << "Введіть y: ";

cin >> y;

int z;

cout << "Введіть z: ";

cin >> z;

char a;

cout << "Введіть a: ";

cin >> a;

char b;

cout << "Введіть b: ";

cin >> b;

clearScreen();

printCopyright();

cout << "\n" << "Результат логічного виразу: " << getBoolResult(a, b) << endl;

cout << "\nx (в десятковій): " << x << "\ny (в десятковій): " << y << "\nz (в десятковій): " << z << endl;

cout << "\nx (в шістнадцятковій): " << getHexadecimal\_16(x) << "\ny (в шістнадцятковій): " << getHexadecimal\_16(y) << "\nz (в шістнадцятковій): " << getHexadecimal\_16(z) << endl;

s\_calculation(x, y, z);

return 0;

}

|  |  |  |  |
| --- | --- | --- | --- |
| Ід-р тест-кейса / Test Case ID | Дії (кроки) /  Action (Test Steps) | Очікуваний  результат /  Expected Result | Результат тестування /  Test Result |
| TC\_01 | Введіть x: 43  Введіть y: 34  Введіть z: 45  Введіть a: 3  Введіть b: 6 | Результат логічного виразу: 0  x (в десятковій): 43  y (в десятковій): 34  z (в десятковій): 45  x (в шістнадцятковій): 2b  y (в шістнадцятковій): 22  z (в шістнадцятковій): 2d  S: nan | Passed |
| TC\_02 | Введіть x: 12  Введіть y: 34  Введіть z: 45  Введіть a: 34 | Результат логічного виразу: 0  x (в десятковій): 12  y (в десятковій): 34  z (в десятковій): 45  x (в шістнадцятковій): c  y (в шістнадцятковій): 22  z (в шістнадцятковій): 2d  Результат обчислень невизначений або недійсний. | Passed |
| TC\_03 | Введіть x: розробник | Результат логічного виразу: 0  x (в десятковій): 0  y (в десятковій): 62  z (в десятковій): 0  x (в шістнадцятковій): 0  y (в шістнадцятковій): 3e  z (в шістнадцятковій): 0  Результат обчислень невизначений або недійсний. | Passed |
| TC\_04 | Введіть x: 34  Введіть y: 45  Введіть z: 56  Введіть a: 12 | Результат логічного виразу: 0  x (в десятковій): 34  y (в десятковій): 45  z (в десятковій): 56  x (в шістнадцятковій): 22  y (в шістнадцятковій): 2d  z (в шістнадцятковій): 38  Результат обчислень невизначений або недійсний. | Passed |
| TC\_05 | Введіть x: 43  Введіть y: 54  Введіть z: 56  Введіть a: 23 | Результат логічного виразу: 0  x (в десятковій): 43  y (в десятковій): 54  z (в десятковій): 56  x (в шістнадцятковій): 2b  y (в шістнадцятковій): 36  z (в шістнадцятковій): 38  Результат обчислень невизначений або недійсний. | Passed |

**Висновок**

Виконуючи цю лабораторну роботу я набув знань та практичних навичок застосування методології модульного програмування, реалізації метода функціональної декомпозиції задач, метода модульного тестування та роботи з мовою програмування C++.

Лабораторна робота №8 унікальна тим, що потрібно використовувати Git-репозиторій з фіксацією багатьох пунктів ,створення статичної библиотеки з подальшим її використанням ,а так же її провірка на якість за допомогою модульного тестування.

В задачі 8.1 нам потрібно було створити статичну бібліотеку і обчислити



Фактично, ми пройшли практику створення статичної бібліотеки, вивчили її правильне структурування та перевірили якість за допомогою модульних тестів. Почали з аналізу та постановки задачі 8.1, провели детальний аналіз вимог і проектування архітектури, включивши результати у звіт. Створили набір контрольних прикладів для модульного тестування функцій на C++, документували тест-сьют та зберегли його. У середовищі розробки Code::Blocks створили проект статичної бібліотеки ModulesChubenko та відтворили файл вихідного коду як cpp. На основі проектування модуля реалізували функцію s\_calculation мовою програмування C++, яка відповідала поставленій задачі. Скомпілювали проект статичної бібліотеки. У Code::Blocks створили проект заголовкового файлу ModulesChubenko в папці проктекту та описали в ньому прототип функції s\_calculation. Реалізували тестовий драйвер для виконання тестів та провели модульне тестування функції s\_calculation з використанням статичної бібліотеки libModulesChubenko. Результати тестування документували та додали до звіту. Включили вихідний код проектів ModulesChubenko та TestDriver у звіт як додатки.

Мої особисті враження від цього завдання полягають у тому, що ми отримали практичний досвід у створенні статичних бібліотек, і хоча виникали певні труднощі, вони були успішно подолані, і тепер ця задача вже є минулим для нас.

У завданні 8.2 ми розпочали з аналізу та постановки задачі. Виконали аналіз вимог, проектування архітектури та детальне проектування програмного забезпечення для вирішення поставленої задачі. Результати цих кроків були документовані та включені до нашого звіту. Наступним кроком було розроблення тест-сьюта для системного тестування програмного забезпечення, що вирішує задачу 8.2. У середовищі розробки Code::Blocks був створений проект консольного додатка. За допомогою мов програмування С/С++ ми реалізували програмне забезпечення відповідно до результатів проектування та скомпілювали його. Відповідно до вимог міжнародного стандарту ISO/IEC 12207, ми провели системне тестування програмного забезпечення Chubenko\_task.exe. Отриманий тестовий артефакт (тестовий набір у теці Lab8\TestSuite) був включений до звіту як додаток. Консольний додаток Chubenko\_task.exe був скопійований до папки \Software.

Мої особисті враження від цього завдання полягають у використанні раніше створеної бібліотеки, і варто зауважити, що завдання 8.2.1 буде корисним у подальшому, оскільки на його основі ми створимо статичну бібліотеку для полегшення роботи.

**Контрольні запитання**

1.У чому полягають мета й задачі процесів проектування програмного забезпечення відповідно до міжнародного стандарту ISO/IEC 12207:2008?

- Процес проектування архітектури

Визначення структури та організації системи

- Процес детального проектування

Розробка детальних специфікацій компонентів системи

- Процес конструювання

Реалізація компонентів системи на основі детальних специфікацій

- Процес комплексування ПЗ

Об'єднання різних компонентів та підсистем для створення готового програмного продукту.

2. Обґрунтовано поясніть, чим функція мови програмування С/С++ відрізняється від модуля.

**Функція (Function)**:

* Функція - це фрагмент програмного коду, який виконує певну задачу або обчислення.
* Вона може бути викликана з інших частин програми для виконання своїх дій та повернення результату.
* Функція може бути частиною модуля, але вона сама по собі не є модулем.
* У мові програмування C/C++, функції можуть бути оголошені та визначені в будь-якому місці програми, а також можуть бути вкладені одна в одну.

**Модуль (Module):**

* Модуль - це логічно відокремлена частина програми, яка містить одну або декілька функцій, класів, змінних або інших об'єктів.
* Він зазвичай організовується для впорядкування програми та розділення її на логічні частини.
* Модуль може містити функції, але також може містити інші структури даних, константи, класи та інші складові програми.
* Модуль може бути відокремленим файлом або групою файлів.
* У мові програмування C/C++, модуль може бути визначений у вигляді окремого файлу (наприклад, .cpp або .h файл), який містить функції, класи та інші об'єкти.

3.Яка відмінність функції main від решти функцій С/С++?

1. **Початкова точка входу**: Функція **main** є початковою точкою входу у програму. Вона є першою функцією, яка виконується при запуску програми.
2. **Обов'язкова наявність**: Кожна програма на C/C++ повинна містити функцію **main**. Вона є обов'язковою для наявності в програмі.
3. **Параметри командного рядка**: Функція **main** може приймати параметри командного рядка. Це дозволяє передавати аргументи при запуску програми через командний рядок.
4. **Повернення значення**: Функція **main** може повертати значення в операційну систему, що вказує на результат виконання програми. Зазвичай 0 означає успішне завершення, а будь-яке інше значення - помилку.
5. **Специфікація типу повернення**: Тип повернення функції **main** повинен бути **int** в стандарті C, або **int** або **void** в стандарті C++. Однак у стандарті C++ він може повертати значення, що вказує на статус виконання програми.
6. **Необов'язковість повернення**: Хоча функція **main** може повертати значення, у багатьох програмах його вказування не є обов'язковим. Якщо **main** не містить оператор **return**, його повернення 0 вважається автоматичним.

4.Яке призначення маніпуляторів і яким чином вони використовуються під час реалізації ПЗ мовою програмування С++?

Маніпулятори в мові програмування C++ - це функції, які використовуються для форматування виводу та вводу даних через потоки (наприклад, стандартний потік виводу **std::cout** та стандартний потік вводу **std::cin**). Вони дозволяють контролювати розміщення, точність, вирівнювання та інші аспекти виводу та вводу даних. Основне призначення маніпуляторів - забезпечити гнучке та зручне форматування виводу та вводу для користувача.

Наприклад, маніпулятори можуть використовуватися для:

1. **Вирівнювання тексту**: Наприклад, маніпулятор **std::setw()** встановлює ширину поля виводу.
2. **Форматування чисел**: Наприклад, маніпулятори **std::fixed**, **std::scientific**, **std::setprecision()** встановлюють формат виводу для чисел.
3. **Керування відображенням позиції курсора**: Наприклад, маніпулятор **std::endl** виводить символ нового рядка та примусово очищує буфер виводу, а **std::flush** просто очищує буфер.
4. **Форматування дати та часу**: Наприклад, маніпулятор **std::put\_time()** дозволяє форматувати дату та час виводу.
5. **Керування відображенням булевих значень**: Наприклад, маніпулятор **std::boolalpha** дозволяє виводити значення типу bool у вигляді "true" або "false".

5.Як і для чого використовується заголовковий файл в процесі препроцесинга програми?

* 1. Заголовкові файли використовуються в процесі препроцесора програми для інклюду (включення) інших файлів у джерело програми. Основна мета використання заголовкових файлів полягає в організації та структуруванні програмного коду для полегшення розробки, утримання та повторного використання коду. Ось які функції вони виконують:
  2. Інклюзія файлів: Заголовкові файли дозволяють включати (інклудити) зовнішні файли з кодом у вихідний код програми. Це забезпечує можливість використовувати функції, класи, константи та інші складові програми, які визначені в інших файлах.
  3. Декларація інтерфейсів: Заголовкові файли містять декларації функцій, класів та інших об'єктів, але не їх визначення. Це дозволяє іншим частинам програми коректно викликати ці функції або використовувати ці класи без необхідності знати деталі їх внутрішньої реалізації.
  4. Уникнення повторного включення (include guards): Для запобігання проблемам з подвійним включенням файлів, заголовкові файли зазвичай мають директиви умовної компіляції (наприклад, #ifndef, #define, #endif), які унеможливлюють включення файлу більше одного разу.
  5. Розділення коду на логічні модулі: Заголовкові файли дозволяють розділити код на логічні модулі або бібліотеки, що полегшує організацію та управління проектом. Кожен заголовковий файл може відповідати окремому функціональному або тематичному блоку коду.

6.Що під час виконання лабораторної роботи Вами використано зі стандартного заголовкового файлу іostream та задля реалізації яких функцій?

* Введення даних з клавіатури за допомогою **cin**.
* Виведення даних на екран за допомогою **cout**.
* Функції для форматованого виведення даних, такі як **setprecision** та **fixed**.
* Використання потокових маніпуляторів для контролю формату виведення, наприклад, **setw**.
* Використання потокових маніпуляторів для встановлення мови виведення за допомогою **setlocale**.

7.Що розуміють під стандартним простором імен у С++ і якою директивою він визначається?

Стандартний простір імен - це простір імен, який містить об'єкти, функції та класи, які визначені в стандартній бібліотеці C++ (STL) або в самій мові. Цей простір імен містить стандартні функції та класи, такі як **std::cout**, **std::cin**, **std::string** та інші. Стандартний простір імен в C++ визначається за допомогою директиви using namespace std;

8.Наведіть приклади одночасного оголошення й ініціалізації початковим значенням змінної (об’єкта).

int x = 10;

double pi = 3.14159;

string name = "John";

9. Перелічіть символи, використання яких у ідентифікаторах не допускаються відповідно до синтаксису мови С/С++.

Символи, використання яких не допускається у ідентифікаторах згідно зі синтаксисом мови C/C++:

1. Пробіл.
2. Спеціальні символи: **!**, **@**, **#**, **$**, **%**, **^**, **&**, **\***, **(**, **)**, **-**, **+**, **[**, **]**, **{**, **}**, **|**, **\**, **;**, **'**, **"**, **,**, **.**, **<**, **>**, **?**, **/**, **=**, **~**, **:**,
3. Цифри, які стоять на початку ідентифікатора.
4. Символи, які вже зарезервовані для операторів або ключових слів мови C/C++.

Яким чином у С++ можливо вивести десятковий літерал у

шістнадцятковій системі числення?

#include <iostream>

using namespace std;

int main() {

int decimalNumber = 255;

cout << "Десяткове число у шістнадцятковій формі: " << hex << decimalNumber << endl;

return 0;

}

Що необхідно виконати для підключення й використання функцій нестандартної (наприклад, власної) бібліотеки?

1. **Створення бібліотеки:** Спочатку потрібно створити бібліотеку, що містить ваші функції, класи або інші об'єкти. Це може бути зроблено створенням файлів ісходного коду (.cpp) та файлів заголовків (.h або .hpp) для оголошення вашого інтерфейсу.
2. **Компіляція:** Після створення вашої бібліотеки, вам потрібно скомпілювати її у виконуваний код. Це може бути зроблено за допомогою вашого улюбленого компілятора C++, наприклад, g++, clang++, або Visual C++ Compiler.
3. **Створення об'єктних файлів:** Компілятор створить об'єктні файли (.o або .obj) з вашого вихідного коду.
4. **Збірка бібліотеки:** Після створення об'єктних файлів вам потрібно зібрати їх у виконуваний код. Це може бути зроблено за допомогою утиліти ar (для Unix-подібних систем) або lib (для Windows).
5. **Використання у власному коді:** Ваша бібліотека тепер готова до використання. Вам потрібно лише включити файл заголовка вашої бібліотеки у свій вихідний код за допомогою директиви **#include**, а також вказати компілятору шлях до скомпільованої бібліотеки для лінкування.
6. **Лінкування:** Під час компіляції вашої програми вам потрібно буде вказати компілятору шлях до об'єктних файлів вашої бібліотеки для лінкування. Використовуйте опцію компілятора **-l**, щоб вказати, яка бібліотека має бути лінкована (наприклад, **-lmylibrary**).

Чим відрізняється заголовковий файл від об’єктного?

1. **Заголовковий файл:**
   * Розширення файлу: .h або .hpp.
   * Містить декларації функцій, класів, змінних та інших елементів програми.
   * Використовується для оголошення інтерфейсу модуля (бібліотеки).
   * Може містити глобальні константи, макроси, визначення типів тощо.
   * Підключається до вихідного коду за допомогою директиви **#include**.
   * Не містить реалізації функцій або класів.
2. **Об'єктний файл:**
   * Розширення файлу: .o або .obj (після компіляції).
   * Містить скомпільований код (машинний код) для функцій та інших елементів програми, які були описані у заголовкових файлах або в ісходному коді.
   * Створюється компілятором з вихідного коду (.cpp) під час процесу компіляції.
   * Використовується під час лінкування разом з іншими об'єктними файлами та бібліотеками для створення виконуваного файлу програми

В чому полягає відмінність між записом символьного і рядкового константного літерала в С/С++?

1. **Символьні константні літерали:**
   * Представляють один символ у відповідності до коду ASCII або Unicode.
   * Записуються в одинарних лапках (**'**) і можуть містити один символ, наприклад: **'A'**, **'3'**, **'@'**, **'%'**, **'!'**.
   * Символьний літерал є типу **char**.
2. **Рядкові константні літерали:**
   * Представляють послідовність символів.
   * Записуються в подвійних лапках (**"**) і можуть містити один або більше символів, наприклад: **"Hello"**, **"123"**, **"@#$"**.
   * Рядковий літерал є масивом символів (**char[]**) і завершується нульовим символом **\0**, що вказує на кінець рядка.

15.Яким є синтаксис запису прототипа функції у С/С++? Де він записується у програмі та яке його призначення?

int add(int a, int b);

Прототип функції записується зазвичай у заголовковому файлі (header file) або в самому початку вихідного файлу програми.

1. **Оголошення функцій:** Прототип функції дозволяє оголосити функцію до її використання у програмі. Це оголошення включає ім'я функції, її повертаючий тип та типи параметрів, що приймаються функцією. Це дозволяє компілятору правильно інтерпретувати виклики функції та перевіряти правильність типів аргументів та поверненого значення.
2. **Документація коду:** Прототип функції може служити як документація коду, описуючи інтерфейс функції, тобто тип повернення, ім'я та типи параметрів. Це допомагає іншим програмістам легше розуміти, як використовувати функцію.
3. **Лінкування:** Прототип функції також використовується під час лінкування програми. Коли функція використовується у різних частинах програми (наприклад, коли вона визначена у відокремленому файлі), прототип допомагає компілятору та лінкеру правильно зв'язати ці частини разом.

Який оператор С/С++ призначений для повернення функцією значення - результату і який синтаксис його запису?

16.У мовах програмування C/C++, для повернення значення (результату) функцією використовується оператор return. Синтаксис запису оператора return наступний:

int add(int a, int b) {

return a + b;

}

17.Перелічіть ключові символи (ESC-послідовності) мови С, їх

призначення, синтаксис запису та спосіб використання у С++.

|  |  |  |
| --- | --- | --- |
| ESC-послідовність | Код | Дія |
| \a | 0x7 | Звуковий сигнал |
| \n | 0xA | Перехід на новий рядок |
| \f | 0xC | Перегортання сторінки |
| \r | 0xD | Повернення курсора |
| \t | 0x9 | Горизонтальна табуляція |
| \v | 0xB | Вертикальна табуляція |
| \b | 0x8 | Повернення курсора на одну позицію назад |
| \\ | 0x5C | Зворотна коса риса |
| \’ | 0x27 | Апостроф |
| \” | 0x22 | Подвійні лапки |
| \? | 0x77 | Знак питання |

18.Перелічіть відомі Вам функції заголовкового файла cmath та їх призначення? В чому полягає їх відмінність від функцій бібліотеки math.h?

1. **std::sqrt(x):** Повертає квадратний корінь числа **x**.
2. **std::sin(x), std::cos(x), std::tan(x):** Повертають синус, косинус та тангенс кута **x** (в радіанах).
3. **std::log(x), std::log10(x):** Повертають натуральний (з основою е) та десятковий логарифми числа **x**.
4. **std::pow(x, y):** Повертає **x** в ступені **y**.
5. **std::ceil(x), std::floor(x):** Повертають найменше ціле число, яке не менше (для **ceil**) або не більше (для **floor**) від **x**.

Яке призначення тестових драйверів?

Тестові драйвери - це програми або компоненти програм, які використовуються для виклику та виконання тестів на програмному забезпеченні. Вони використовуються для реалізації специфічних тестових сценаріїв та забезпечення взаємодії тестів з основним програмним кодом або компонентами.

Основні призначення тестових драйверів:

* Запуск тестів: Тестові драйвери використовуються для запуску тестів, які включені в програму або бібліотеку, та збору результатів виконання тестів.
* Керування виконанням тестів: Вони забезпечують контроль над тим, які тести виконуються, в якому порядку та з якими параметрами.
* Налагодження тестів: Тестові драйвери можуть надавати можливості для налагодження тестів, щоб виявити та виправити помилки в програмному коді.
* Збір результатів: Вони забезпечують збір результатів тестів, включаючи інформацію про тестові кейси, час виконання, статус (успішно чи невдача), а також додаткові деталі про помилки, якщо такі виникли.
* Автоматизація тестування: Тестові драйвери можуть бути використані для автоматизації процесу тестування, зокрема для автоматичного запуску тестів, виконання різних тестових сценаріїв та автоматичного аналізу результатів тестування.
* Яким чином здійснюється тестування модулів (функцій

бібліотек) і чим означений процес відрізняється від тестування

* 1. програмного засобу (ехе-файла)?  
     Тестування модулів (функцій бібліотек) і тестування програмного засобу (виконуваний файл або екземпляр програми) відрізняються за обсягом, об'єктом тестування та методами виконання. Ось деякі основні відмінності між ними:

**Тестування модулів (функцій бібліотек):**

1. **Обсяг тестування:**
   * Тестування модулів зазвичай зосереджене на окремих функціях або модулях програмного коду.
   * Оцінюються різні входні дані та вихідні результати для кожної функції окремо.
2. **Об'єкт тестування:**
   * Об'єктом тестування є окремі функції або модулі програмного коду, які можуть бути тестовані індивідуально.
3. **Методи тестування:**
   * Для тестування модулів зазвичай використовуються автоматизовані тести, які передають певні вхідні дані до функції та перевіряють, чи повернула вона очікуваний результат.
   * Може використовуватися модульне тестування, функціональне тестування та інші методи.

**Тестування програмного засобу (виконуваний файл або екземпляр програми):**

1. **Обсяг тестування:**
   * Тестування програмного засобу включає в себе всю програму або окремі компоненти програми, щоб переконатися, що вона працює правильно як цілісна система.
2. **Об'єкт тестування:**
   * Об'єктом тестування є виконуваний файл програми або сама програма.
3. **Методи тестування:**
   * Для тестування програмного засобу можуть використовуватися автоматизовані тести, ручні тести, тести з використанням користувацького інтерфейсу тощо.
   * Зазвичай виконуються інтеграційні тести, системні тести, приймальне тестування та інші види тестів.