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 .

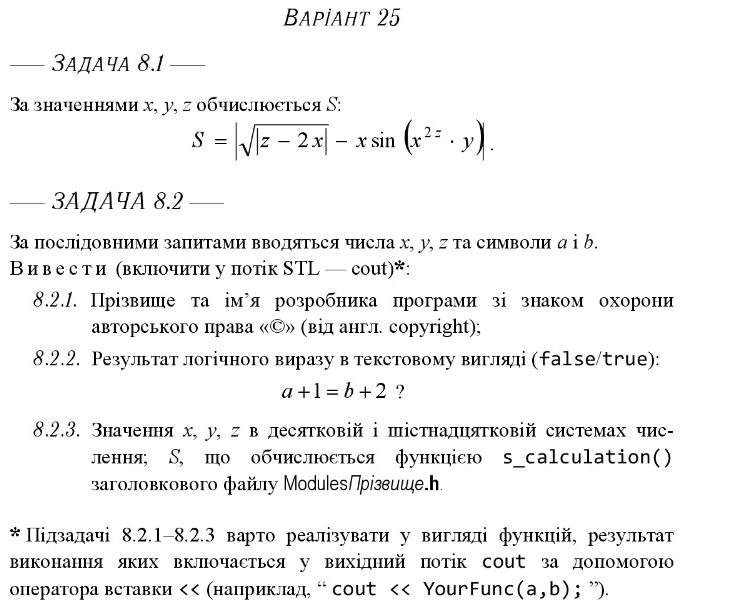
Мета роботи: полягає у набутті ґрунтовних вмінь і практичних навичок застосування теоретичних положень методології модульного програмування, реалізації метода функціональної декомпозиції задач, метода модульного (блочного) тестування, представлення мовою програмування С++ даних скалярних типів, арифметичних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

**ЗАВДАННЯ:**

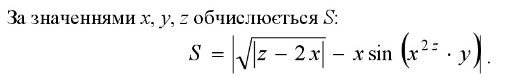
1. Реалізувати статичну бібліотеку модулів libModulesПрізвище C/C++, яка містить функцію розв’язування задачі 8.1.

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;

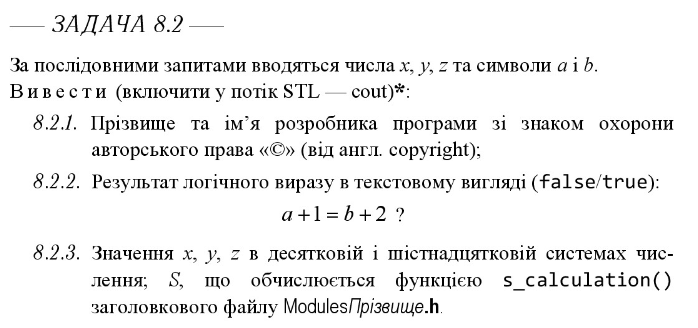
}

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;

}

**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.

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.

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".

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.