Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 9

з навчальної дисципліни

“Базові методології та технології програмування”

РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ РОЗГАЛУЖЕНИХ ТА ІТЕРАЦІЙНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

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 КБ-21

Іванов М.Л.

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

Усик П.С.

Кропивницький – 2022

**Лабораторна робота №9**

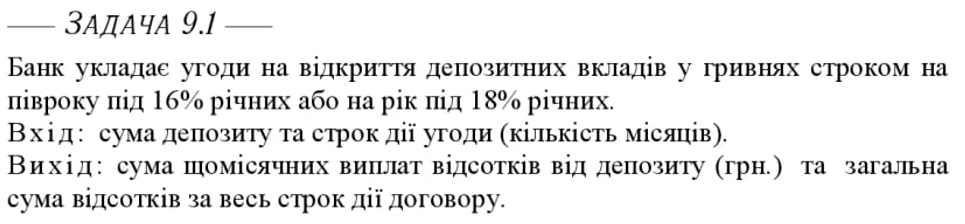
**Тема:** реалізація програмних модулів розгалужених та ітераційних обчислювальних процесів

**Мета роботи:** полягає у набутті ґрунтовних вмінь і практичних навичок реалізації технології модульного програмування, застосування операторів C/С++ арифметичних, логічних, побітових операцій, умови, циклів та вибору під час розроблення статичних бібліотек, заголовкових файлів та програмних засобів у кросплатформовому середовищі Code::Blocks.

**Завдання:**

1. Реалізувати функції розв’язування задач 9.1-9.3 як складових статичної бібліотеки libModulesПрізвище.a (проєкт ModulesПрізвище, створений під час виконання лабораторної роботи №8).
2. Реалізувати програмне забезпечення розв’язування задачі 9.4 на основі функцій статичної бібліотеки libModulesПрізвище.a.

**Варіант** **12**



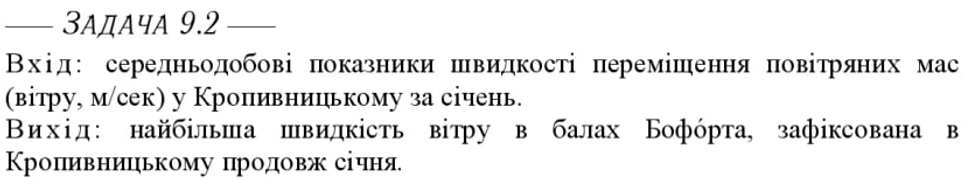
Малюнок 1 – Умова задачі 9.1

**Строга постановка задачі:**

Вхідні дані: dep – дробове число

strok – число 12 або 6.

Вихідні дані: month – дробове число

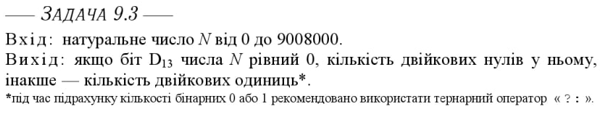


Малюнок 2 – Умова задачі 9.2

**Строга постановка задачі:**

Вхідні дані: jan – масив дійсних чисел (31 елемент)

Вихідні дані: maximum – ціле число (0-12)



Малюнок 3 – Умова задачі 9.3

**Строга постановка задачі:**

Вхідні дані: num – ціле невід’ємне число, 0 <= N <= 9008000.

Вихідні дані: count – ціле невід’ємне число.

**Проектування програмного модуля:**

Модуль ModulesIvanov, що вже містить функцію s\_calculation, розширюється шляхом додавання нових функцій та структур даних, призначених для реалізації повернення декількох змінних функціями.

Функція Belfort приймає ціле число, що позначає швидкість вітру, в якості вхідних даних та повертає відповідний бал Бофорта та опис дії вітру. Для реалізації повернення функцією вихідних даних, потрібно створити складений тип даних Belfort, що складається з цілого числа mark та рядка description. Розрахунок значення відбувається за таблицею.

Функція avg\_temperature приймає масив дійсних чисел, що відповідають середньодобовим показникам температури у градусах Цельсія за січень. Функція сумує елементи масиву, ділить суму на кількість елементів та повертає середню температуру в градусах Цельсія та градусах Фаренгейту. Для цього потрібно створити структуру даних temperature, що складатиметься з змінних дійсного числа Celcium та Fahrenheit. Значення Фаренгейту розраховується за формулою.

Функція bits\_number приймаэ ціле число від 0 до 10008000 та повертає кількість «1» або «0» у двійковому представленні цього числа в залежності від того, яким буде біт D14 (розрахунок починається справа з нуля). Для реалізації функції потрібно представити число у бітовому вигляді, перевірити статус D14 та в залежності від нього повернути кількість одиниць або нулів.

**Контрольні приклади для задачі 9.1**

|  |  |  |
| --- | --- | --- |
| **ID-TC** | **Input** | **Expected result** |
| **9.1.1** | **1000, 6** | **26.666666** |
| **9.1.2** | **1000, 12** | **15** |
| **9.1.3** | **2500, 12** | **37.5** |

**Контрольні приклади для задачі 9.2**

|  |  |  |
| --- | --- | --- |
| **ID-TC** | **Input** | **Expected Result** |
| **9.2.1** | **0.00 0.20 0.60 2.30 0.20 0.60 0.00 0.20 0.60 0.00 0.20 0.60 0.00 0.20 0.60 0.00 0.20 0.60 0.00 0.20 0.60 0.00 0.20 0.60 0.00 0.20 0.60 0.00 0.20 0.50 0.65** | **1** |
| **9.2.2** | **3.00 0.10 2.50 3.00 0.10 2.50 3.00 0.10 2.50 3.00 0.10 2.50 3.00 0.10 2.50 3.00 0.10 2.50 3.00 0.10 2.50 3.00 0.10 2.50 3.00 0.10 2.50 3.00 0.10 3.20 2.56** | **2** |
| **9.3.3** | **1.00 3.00 6.10 1.00 3.00 6.10 1.00 3.00 6.10 1.00 3.00 6.10 1.00 3.00 6.10 1.00 3.00 6.10 1.00 3.00 6.10 1.00 3.00 6.10 1.00 3.00 6.10 1.00 3.00 1.50 6.16** | **4** |

**Контрольні приклади для задачі 9.3**

|  |  |  |
| --- | --- | --- |
| **ID-TC** | **Input** | **Expected result** |
| **9.3.1** | **123456** | **6** |
| **9.3.2** | **654321** | **14** |
| **9.3.3** | **8653752** | **23** |

**Результати тестування модулів для задач 9.1-9.3:**

|Max Ivanov test task 9.1|

TC\_1:|Deposite = 1000 Strok = 6 | - result: 26.666666 - passed

TC\_2:|Deposite = 1000 Strok = 12 | - result: 15.000000 - passed

TC\_3:|Deposite = 2500 Strok = 12 | - result: 37.500000 - passed

|Max Ivanov test task 9.2|

TC\_1:|1 = 1 : passed

TC\_2:|2 = 2 : passed

TC\_3:|4 = 4 : passed

|Max Ivanov test task 9.3|

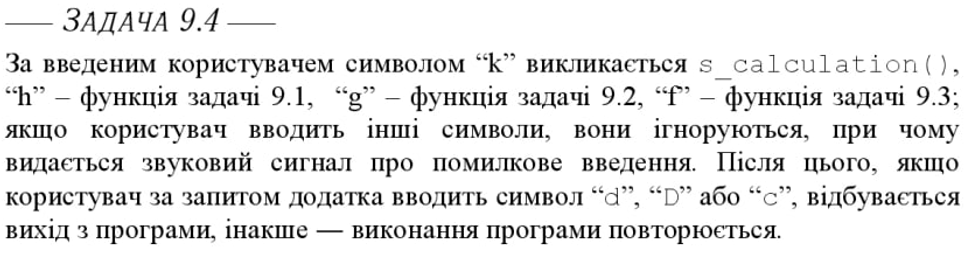
TC\_1:|6 = 6: passed

TC\_2:|14 = 14: passed

TC\_3:|23 = 23: passed

**Process returned 0 (0x0) execution time : 0.023 s**

**Press any key to continue.**



Малюнок 4 – Умова задачі 9.4

**Строга постановка задачі:**

Вхідні дані: symbol – символ;

dep – дробове число

strok – число 12 або 6.

jan – масив дійсних чисел (31 елемент)

num – ціле невід’ємне число, 0 <= N <= 9008000.

Вихідні дані: month – дробове число

maximum – ціле число (0-12)

count – ціле невід’ємне число.

**Проектування програмного модуля:**

Задача 9.4 по своїй суті є драйвером модуля ModulesIvanov. Для запуску функції по вводу символу треба у циклі перевіряти введений символ: якщо він дорівнює «c», «d» або «D», цикл завершується, якщо введено «g», «h», «k» чи «f», запускається зчитування вхідних даних для потрібної функції, інакше – програма повторюється. «m» використовується для виклику контекстного меню. Якщо функція відпрацювала, програма теж повторюється.

**Висновок:** дана лабораторна робота була націлена на набуття навичок

у реалізації технології модульного програмування, застосуванні операторів C/С++ арифметичних, логічних, побітових операцій, умови, циклів та вибору під час розроблення статичних бібліотек, заголовкових файлів та програмних засобів у кросплатформовому середовищі Code::Blocks.

До вихідного коду статичної бібліотеки, створеної під час виконання лабораторної роботи №8, було додано реалізації функцій для розв’язування задач 9.1-9.3 та створено нові структури для реалізації повернення функціями даних складного типу. Після компіляції проекту бібліотеки в теці \obj оновився файл з розширенням .а, що і є файлом статичної бібліотеки. Наступним кроком було додавання до заголовкового файлу бібліотеки прототипів функцій та нових структур, які повертаються функціями.

Особливим є те, що статична бібліотека та заголовковий файл не копіювалися у директорію, а додавалися до проекту в якості відносного шляху до файлів. Варто зазначити, що оновлений модуль залишається сумісним з вихідним кодом проектів у лабораторній роботі №8.

Перша задача потребувала в залежності від вхідного значення за таблицею повертати відповідний бал (ціле число) та опис цього балу (рядок). Вхідне значення береться за модулем через природу показника. Для коректної роботи функції було задекларовано структур даних, яка складається з цілого числа та рядка, що відповідає типу даних, які мають повертатися функцією.

Друга задача складалася в тому, щоб з масиву дійсних чисел, які відображають середньодобові показники температури в певному місяці, знайти середнє арифметичне та виразити його в градусах Цельсія та Фаренгейту (останній розраховується за формулою). Таким чином, новий тип даних, який повертає функція, складається з набору значень у відповідних системах числення (два дійсних числа).

У третій задачі потрібно було у цілому невід’ємному числі, переведенному у двійкову систему, перевірити значення певної за рахунком цифри (у даній реалізації було використано нестандартний тип даних bitset, що може репрезентувати двійковий запис цілого числа; таким чином перевіряється певний біт), і в залежності від цього значення («1» чи «0») потрібно було підрахувати кількість одиниць або нулів у двійковому записі числа (довжина запису бралася фіксована, розрахована для типу даних int – 32 біти). У класичному розв'язку потрібно ітераційно перевіряти ділимість числа на 2 та відповідно ділити його, зберігаючи кількість тих чи інших значень, проте тип даних bitset має відповідний набір функцій, які роблять це автоматично. Так як послідовніть порядків рахується з 0 зправа наліво, операція [ ] відносно bitset знаходить значення елементу у наборі зправа наліво. Також існує функція count, яка для bitset підраховує кількість елементів, які відповідають значенню true («1»), тож щоб підрахувати кількість «0», треба від загальної кількості бітів відняти кількість одиниць.

Далі було створено проект консольного додатка С++ під назвою TestDriver, метою створення якого є реалізація модульного тестування. Для кожної функції було розраховано еталонні вхідні дані та відповідні еталонні результати, які порівнювалися з результатами, що повертаються функціями.

Для першої функції, яка повертає число та рядок з описом, перевірялося тільки число, бо рядок з описов строго прив'язаний до числа (для кожного числа рядок є унікальним).

Для другої функції перевірялося обидва значення складеного типу даних, бо є важливим перевірити функціонування не тільки знаходження середнього арифметичного числа, але й конвертування значень температури з однієї системи числення в іншу.

Для третьої функції у драйвері було виведено двійковий запис числа, яке є еталонним вхідним значенням, та потрібний біт було виділено пробілом, щоб більш наглядно зобразити принцип роботи функції.

Особлива увага приділялася структурованому виводу даних у консоль для більш зручного документування результатів тестування модуля.

Після цього було створено ПЗ з реалізацією функцій з наступного завдання. Важливою умовою було проектування функцій з інтерфейсом, який підходить для виводу у вихідний поток (cout). Для коректного виводу рядка з першої функції довелось перевести консоль у кодування cp1251. Задача потребувала при вводі певного символу запустити потрібну функцію (включаючи функцію s\_calculation(), розроблену раніше) або реалізувати завершення роботи програмного засобу. Для цього була використана функція getch(), яка зчитує введений символ та одразу повертає його значення. Також для кожної з функцій було реалізовано інтерфейс вводу значень для функцій та виведення результату її роботи. Після завершення роботи функції усі попередні дії повторюються.

Для подальшого тестування .exe файла, створеного під час компіляції проекту, довелось в налаштуваннях проекту у вкладці з налаштуваннями компілятора відмітити прапорці з позначками “Static libgcc [-static-libgcc]”, “Static libstdc++ [-static-libstdc++]” та “Static linking [-static]”.

Усі налаштування, перераховані у висновку, були застосовані до проектів, а не до середовища, для того, щоб проект успішно виконувався в IDE Code::Blocks на різних пристроях, незалежно від налаштувань певного середовища.

Репозиторій Git можна розглядати як окрему частину файлової системи, яка має можливість зберігатися дистанційно, синхронізуватися та повертатися до потрібної версії (стану директорії у певний момент часу). Через те, що усі модулі та проекти знаходяться в одному репозиторії, копіювання статичної бібліотеки та заголовкового файлу виявилось зайвим.

**ДОДАТОК А**

**Лістинг вихідного коду проекту ModulesIvanov:**

#include <cmath>

#include <windows.h>

#include <iostream>

#include <bitset>

using namespace std;

double c\_calculation(double x, double y)

{

double S;

S = pow(x, 2) - pow(y, 3) + sqrt(pow(y, x) - M\_PI);

return S;

}

float Deposite(float dep, short strok)

{

float month, a;

if(strok == 6)

{

a = (dep\*16)/100;

month = a/6;

}

if (strok == 12)

{

a = (dep\*18)/100;

month = a/12;

}

return month;

}

short Bolfort(float jan[])

{

short sz = 31;

float bolfort[sz];

short maximum = 0;

for (int i = 0; i < sz; i++)

{

if (jan[i] < 0.3)

{

bolfort[i] = 0;

continue;

}

if ((jan[i] > 0.3) && (jan[i] < 1.5))

{

bolfort[i] = 1;

continue;

}

if ((jan[i] > 1.6) && (jan[i] < 3.4))

{

bolfort[i] = 2;

continue;

}

if ((jan[i] > 3.5) && (jan[i] < 5.4))

{

bolfort[i] = 3;

continue;

}

if ((jan[i] > 5.5) && (jan[i] < 7.9))

{

bolfort[i] = 4;

continue;

}

if ((jan[i] > 8.0) && (jan[i] < 10.7))

{

bolfort[i] = 5;

continue;

}

if ((jan[i] > 10.8) && (jan[i] < 13.8))

{

bolfort[i] = 6;

continue;

}

if ((jan[i] > 13.9) && (jan[i] < 17.1))

{

bolfort[i] = 7;

continue;

}

if ((jan[i] > 17.2) && (jan[i] < 20.7))

{

bolfort[i] = 8;

continue;

}

if ((jan[i] > 20.8) && (jan[i] < 24.4))

{

bolfort[i] = 9;

continue;

}

if ((jan[i] > 24.5) && (jan[i] < 28.4))

{

bolfort[i] = 10;

continue;

}

if ((jan[i] > 28.5) && (jan[i] < 32.6))

{

bolfort[i] = 11;

continue;

}

if (jan[i] >= 32.7)

{

bolfort[i] = 12;

continue;

}

}

for(int i = 0; i < sz; i++)

{

if(maximum < bolfort[i])

maximum = bolfort[i];

}

return maximum;

}

int BitNumber(int num)

{

bitset<32> b\_number(num);

if(b\_number[13])

{

return b\_number.count();

}

return 32 - b\_number.count();

}**ДОДАТОК Б**

**Лістинг вихідного коду проекту TestDriver:**

#include <iostream>

#include <iomanip>

#include <ModulesIvanov.h>

using namespace std;

int main()

{

cout << "\t|Max Ivanov test task 9.1|" << endl;

double input[3][2] = {1000, 6,

1000, 12,

2500, 12};

double result[3] = {26.666666,

15,

37.5};

for (int i = 0; i < 3; i++)

{

cout << resetiosflags(ios::fixed) << "TC\_" << i+1 << ":|" << setfill(' ');

cout << "Deposite = " << setw(3) << left << input[i][0];

cout << " Strok = " << setw(3) << left << input[i][1];

cout << fixed << "| - result: " << setw(10) << Deposite(input[i][0], input[i][1]);

cout << left << " - " << ((round(Deposite(input[i][0], input[i][1])\*1000000)/1000000.0 == result[i]) ? "passed" : "failed") << endl;

}

cout << "\t|Max Ivanov test task 9.2|" << endl;

float in[31][3] = {0, 3, 1,

0.2, 0.1, 3,

0.6, 2.5, 6.1,

2.3, 3, 1,

0.2, 0.1, 3,

0.6, 2.5, 6.1,

0, 3, 1,

0.2, 0.1, 3,

0.6, 2.5, 6.1,

0, 3, 1,

0.2, 0.1, 3,

0.6, 2.5, 6.1,

0, 3, 1,

0.2, 0.1, 3,

0.6, 2.5, 6.1,

0, 3, 1,

0.2, 0.1, 3,

0.6, 2.5, 6.1,

0, 3, 1,

0.2, 0.1, 3,

0.6, 2.5, 6.1,

0, 3, 1,

0.2, 0.1, 3,

0.6, 2.5, 6.1,

0, 3, 1,

0.2, 0.1, 3,

0.6, 2.5, 6.1,

0, 3, 1,

0.2, 0.1, 3,

0.5, 3.2, 1.5,

0.655, 2.5555, 6.1555};

short expected[3] {1, 2, 4};

for(int i = 0; i < 3; i++)

{

float test[31];

for(int t = 0; t < 31; t++)

{

test[i] = in[t][i];

}

if (expected[i] == Bolfort(test))

{

cout << resetiosflags(ios::fixed) << "TC\_" << i+1 << ":|" << setfill(' ');

cout << expected[i] << " = " << Bolfort(test) << " : passed" << endl;

}

else

{

cout << resetiosflags(ios::fixed) << "TC\_" << i+1 << ":|" << setfill(' ');

cout << expected[i] << " = " << Bolfort(test) << " : failed" << endl;

}

}

cout << "\t|Max Ivanov test task 9.3|" << endl;

int inp[3] {123456, 654321, 8653752};

int outp[3] {6, 14, 23};

for (int i = 0; i < 3; i ++)

{

if (BitNumber(inp[i]) == outp[i])

{

cout << resetiosflags(ios::fixed) << "TC\_" << i+1 << ":|" << setfill(' ');

cout << BitNumber(inp[i]) << " = " << outp[i] << ": passed" << endl;

}

else

{

cout << resetiosflags(ios::fixed) << "TC\_" << i+1 << ":|" << setfill(' ');

cout << BitNumber(inp[i]) << " = " << outp[i] << ": failed" << endl;

}

}

return 0;

}**ДОДАТОК В**

|  |  |
| --- | --- |
| Назва тестового набору  Test Suite Description | TS\_lab9\_ \_task |
| Назва проекта / ПЗ  Name of Project / Software | Kozlov\_Ivanovtask.exe |
| Рівень тестування  Level of Testing | системний / System Testing |
| Автор тест-сьюта  Test Suite Author | Max Ivanov |
| Виконавець  Implementer | Max Ivanov |

|  |  |  |  |
| --- | --- | --- | --- |
| Ід-р тест-кейса / Test Case ID | Дії (кроки) /  Action (Test Steps) | Очікуваний  результат /  Expected Result | Результат тестування /  Test Result |
| TC\_01 | 1. Увести «k» 2. Увести 3, 5, 1 3. Увести “c” | ========== Menu ==========  | k - s\_calculation |  | h - task 9.1(Deposite) |  | g - task 9.2(Bolfort) |  | f - task 9.3(BitNumber)|  | c, d(D) - exit |  | m - menu |  ==========================  Press a key: k  =====================================  Function s\_calculation starts running  =====================================  Enter x: 3  Enter y: 5  Enter z: 1  -104.961  =======================================  Function s\_calculation finished running  =======================================  Press a key: c  Process returned 0 (0x0) execution time : 16.290 s  Press any key to continue. | passed |
| TC\_02 | 1. Увести «h» 2. Увести «1000» 3. Увести «12» 4. Увести «j» 5. Увести «d» | ========== Menu ==========  | k - s\_calculation |  | h - task 9.1(Deposite) |  | g - task 9.2(Bolfort) |  | f - task 9.3(BitNumber)|  | c, d(D) - exit |  | m - menu |  ==========================  Press a key: h  ================================  Function Deposite starts running  ================================  Enter a sum of deposite: 1000  Enter a strok of deposite: 12  You will have 15 uan per month  ==================================  Function Deposite finished running  ==================================  Press a key: j  Press a key: d  Process returned 0 (0x0) execution time : 1292.554 s  Press any key to continue. | passed |
| TC\_03 | 1. Увести «f» 2. Увести «123456» 3. Увести «y» 4. Увести «D» | ========== Menu ==========  | k - s\_calculation |  | h - task 9.1(Deposite) |  | g - task 9.2(Bolfort) |  | f - task 9.3(BitNumber)|  | c, d(D) - exit |  | m - menu |  ==========================  Press a key: f  =================================  Function BitNumber starts running  =================================  Enter a number (0 - 9008000):  123456  6  ===================================  Function BitNumber finished running  ===================================  Press a key: y  Press a key: D  Process returned 0 (0x0) execution time : 443.943 s  Press any key to continue. | passed |
| TC\_04 | 1. Увести «m» 2. Увести «c» | ========== Menu ==========  | k - s\_calculation |  | h - task 9.1(Deposite) |  | g - task 9.2(Bolfort) |  | f - task 9.3(BitNumber)|  | c, d(D) - exit |  | m - menu |  ==========================  Press a key: m  ========== Menu ==========  | k - s\_calculation |  | h - task 9.1(Deposite) |  | g - task 9.2(Bolfort) |  | f - task 9.3(BitNumber)|  | c, d(D) - exit |  | m - menu |  ==========================  Press a key: c  Process returned 0 (0x0) execution time : 3.223 s  Press any key to continue. | passed |
| TC\_05 | 1. Увести «k» 2. Увести 2, 6, 3 3. Увести «m» 4. Увести «f» 5. Увести 654321 6. Увести «c» | ========== Menu ==========  | k - s\_calculation |  | h - task 9.1(Deposite) |  | g - task 9.2(Bolfort) |  | f - task 9.3(BitNumber)|  | c, d(D) - exit |  | m - menu |  ==========================  Press a key: k  =====================================  Function s\_calculation starts running  =====================================  Enter x: 2  Enter y: 6  Enter z: 3  -206.268  =======================================  Function s\_calculation finished running  =======================================  Press a key: m  ========== Menu ==========  | k - s\_calculation |  | h - task 9.1(Deposite) |  | g - task 9.2(Bolfort) |  | f - task 9.3(BitNumber)|  | c, d(D) - exit |  | m - menu |  ==========================  Press a key: f  =================================  Function BitNumber starts running  =================================  Enter a number (0 - 9008000):  654321  14  ===================================  Function BitNumber finished running  ===================================  Press a key: | passed |

**ДОДАТОК Г**

**Лістинг вихідного коду проекту Ivanov\_task:**

#include <iostream>

#include "ModulesIvanov.h"

using namespace std;

void menu()

{

cout << "========== Menu ==========" << endl

<< "| k - s\_calculation |" << endl

<< "| h - task 9.1(Deposite) |" << endl

<< "| g - task 9.2(Bolfort) |" << endl

<< "| f - task 9.3(BitNumber)|" << endl

<< "| c, d(D) - exit |" << endl

<< "| m - menu |" << endl

<< "==========================" << endl << endl;

}

void s\_calcFunc()

{

cout << "=====================================" << endl

<< "Function s\_calculation starts running" << endl

<< "=====================================" << endl;

double x, y, z;

cout << "Enter x: ";

cin >> x;

cout << "Enter y: ";

cin >> y;

cout << "Enter z: ";

cin >> z;

cout << c\_calculation(x, y) << endl;

cout << "=======================================" << endl

<< "Function s\_calculation finished running" << endl

<< "=======================================" << endl << endl;

}

void DepFunc()

{

cout << "================================" << endl

<< "Function Deposite starts running" << endl

<< "================================" << endl;

float dep, strok;

cout << "Enter a sum of deposite: ";

cin >> dep;

cout << "Enter a strok of deposite: ";

cin >> strok;

cout << "You will have " << Deposite(dep, strok) << " uan per month" << endl;

cout << "==================================" << endl

<< "Function Deposite finished running" << endl

<< "==================================" << endl << endl;

}

void BolfFinc()

{

cout << "===============================" << endl

<< "Function Bolfort starts running" << endl

<< "===============================" << endl;

cout << "Enter mid-day wind`s speed value for January: " << endl;

float january[31];

for(int i = 0; i < 31; i++)

{

cin >> january[i];

}

cout << Bolfort(january) << endl;

cout << "=================================" << endl

<< "Function Bolfort finished running" << endl

<< "=================================" << endl << endl;

}

void BitFunc()

{

cout << "=================================" << endl

<< "Function BitNumber starts running" << endl

<< "=================================" << endl;

int num;

cout << "Enter a number (0 - 9008000): " << endl;

cin >>num;

cout << BitNumber(num) << endl;

cout << "===================================" << endl

<< "Function BitNumber finished running" << endl

<< "===================================" << endl << endl;

}

int main()

{

char symbol = 'm';

while (1)

{

if (symbol == 'm')

menu();

if (symbol == 'c')

break;

if (symbol == 'd')

break;

if (symbol == 'D')

break;

if (symbol == 'k')

s\_calcFunc();

if (symbol == 'g')

BolfFinc();

if (symbol == 'h')

DepFunc();

if (symbol == 'f')

BitFunc();

cout << "Press a key: ";

cin >> symbol;

}

return 0;

}