Министерство образования и науки Российской Федерации   
Федеральное государственное бюджетное образовательное   
учреждение высшего образования   
«Алтайский государственный технический университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчёт защищён с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель Астахова Е.В.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

Отчёт

по лабораторной работе № 2

«Погрешность машинного вычисления»

по дисциплине «Основы программной инженерии»

Студенты группы ПИ 92:

В.М. Шульпов, И.В. Вильк, А. Н. Гулин, Ю.П. Пирязев

Преподаватель доцент, к.п. н. Астахова Е.В.

Барнаул 2019

**Задание 1** Машинные числа

* Написать программу для определения величины машинного ε.
* Написать программу для определения машинного ε для удвоенной/повышенной точности представления вещественных чисел.
* Написать программу для определения машинного нуля.

**Код программы**

// exercise1.cpp

//основы программной инженерии

//ЛАБОРАТОРНАЯ РАБОТА №2

//ЗАДАНИЕ 1

//программа определяет машинный эпсилон для типов float, double и long double

#include "stdafx.h"

int main()

{

//эпсилон для точности float

float e\_f = (float)1.0;

while ((float)1.0 + e\_f / (float)2.0 > (float)1.0)

e\_f /= (float)2.0;

printf("For float: e = %e\n", e\_f);

//эпсилон для точности double

double e\_d = (double)1.0;

while ((double)1.0 + e\_d / (double)2.0 > (double)1.0)

e\_d /= (double)2.0;

printf("For double: e = %e\n", e\_d);

//вычисление машинного нуля

int i = 0;

float a = (float)1.0;

do

{

a /= (float)2.0;

i = ++i;

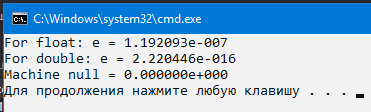
} while (a != 0);

printf("Machine null = %e\n", a);

return 0;

}

**Тест программы**

****

**Задание 2** Точность вычислений

* Написать программу для исследования целочисленного переполнения. Установить, что происходит с целой переменной, когда ее значение превышает правую границу диапазона представления целых чисел.

**Код программы**

// exercise2.cpp: определяет точку входа для консольного приложения.

//основы программной инженерии

//ЛАБОРАТОРНАЯ РАБОТА №2

//ЗАДАНИЕ 2

//Программа для исследования целочисленного переполнения.

#include "stdafx.h"

#include <Windows.h>

#include <locale.h>

int main()

{

setlocale(LC\_ALL, "Rus");

system("color F0");

int a, a1;

a = 2147483647;

printf("Граница значения для int: %d\n", a);

a = a + 1;

printf("Значение, при переходе через границу на один: %d\n", a);

a = a + 1;

a1 = a + 1;

printf("Дальше число изменяется в таком ключе: %d , %d\n", a, a1);

unsigned int b, b1;

b = 4294967295;

printf("Граница значения для unsigned int:%u\n", b);

b = b + 1;

printf("Значение, при переходе через границу на один: %u\n", b);

b = b + 1;

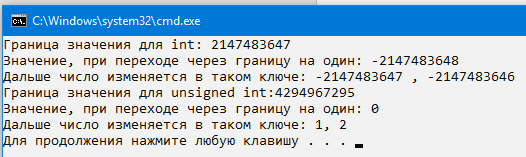
b1 = b + 1;

printf("Дальше число изменяется в таком ключе: %u, %u\n", b, b1);

return 0;

}

**Тест программы**



* Написать программу для вычисления выражений

x2 - y2 и (x - y)\*(x + y). Сравнить полученные результаты.

**Код программы**

// exercise2\_2.cpp: определяет точку входа для консольного приложения.

//основы программной инженерии

//ЛАБОРАТОРНАЯ РАБОТА №2

//ЗАДАНИЕ 2

//Программа для исследования целочисленного переполнения.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdafx.h"

#include <locale.h>

#include <Windows.h>

#include <locale.h>

#include <math.h>

int main()

{

setlocale(LC\_ALL, "Rus");

system("color F0");

double x, y, a, b, a\_b, b\_a;

printf("Сравнение выражений x^2-y^2 и (x-y)(x+y)\n");

printf("Введите значение x: ");

scanf("%lf", &x);

printf("Введите значение y: ");

scanf("%lf", &y);

a = pow(x, 2) - pow(y, 2);

b = (x - y)\*(x + y);

if (a > b){

a\_b = a - b;

printf("Первое выражение больше второго выражения на %lf\n", a\_b);

}

else{

if (b > a){

b\_a = b - a;

printf("Второе выражение больше первого на %lf\n", b\_a);

}

else{

printf("Первое выражение равно второму\n");

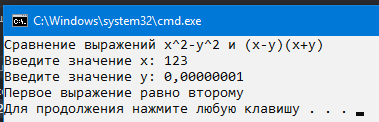
}

}

return 0;

}

**Тест программы**



**Задача на «4»**

Экспериментально доказать, что для вещественной машинной арифметики не всегда выполняются законы:

(х + у) + z = x + (y + z) --[ ассоциативность

(x \* y) \* z = x \* (y \* z) --[ассоциативность

(х + у) \* z = x \* z + y \* z --[ дистрибутивность

**Код программы**

//ОПИ Лаб. 2. Задача на "4". Проверка выполнения законов в вещественной машинной арифметике

#include "stdafx.h"

#include <locale.h>

#include <conio.h>

int \_tmain(int argc, \_TCHAR\* argv[])

{

setlocale(LC\_ALL, "Rus");

int num; // num - аргумент выбора пользователя.

float x, y, z, result1, result2; //x, y, z - переменные.

printf("Проверка выполнения законов в вещественной машинной арифметике.\n");

printf("Программа позволяет экспериментально доказать, что для вещественной\nмашинной арифметики не всегда выполняются законы.\n");

printf("Введите одно из чисел, чтобы проверить законы:\n1 - (x+y)+z=x+(y+z) (ассоциативность),\n");

printf("2 - (x\*y)\*z=x\*(y\*z) (ассоциативность),\n3 - (x+y)\*z=x\*z+y\*z (дистрибутивность),\n");

printf("4 - проверить все законы.\n");

printf("Введите ваш выбор: ");

scanf\_s("%d", &num);

printf("Введите x=");

scanf\_s("%f", &x);

printf("Введите y=");

scanf\_s("%f", &y);

printf("Введите z=");

scanf\_s("%f", &z);

/\*Проверка ассоциативности сложения\*/

if (num==1 || num==4)

{

result1=(x+y)+z;

printf("(x+y)+z=%f\n", result1);

result2=x+(y+z);

printf("x+(y+z)=%f\n", result2);

if (result1==result2)

printf("Свойство ассоциативности выполняется.\n");

else

printf("Свойство ассоциативности не выполняется.\n");

}

/\*Проверка ассоциативности умножения\*/

if (num==2 || num==4)

{

result1=(x\*y)\*z;

printf("(x\*y)\*z=%f\n", result1);

result2=x\*(y\*z);

printf("x\*(y\*z)=%f\n", result2);

if (result1==result2)

printf("Свойство ассоциативности выполняется.\n");

else

printf("Свойство ассоциативности не выполняется.\n");

}

/\*Проверка дистрибутивности\*/

if (num==3 || num==4)

{

result1=(x+y)\*z;

printf("(x+y)\*z=%f\n", result1);

result2=x\*z+y\*z;

printf("x\*z+y\*z=%f\n", result2);

if (result1==result2)

printf("Свойство дистрибутивности выполняется.\n");

else

printf("Свойство дистрибутивности не выполняется.\n");

}

/\*Условный оператор, проверяющий введённый аргумент выбора пользователя на соответствие доступным вариантам\*/

if (num!=1 && num!=2 && num!=3 && num!=4)

{

printf("Ошибка. Вы ввели неизвестный аргумент выбора.\n");

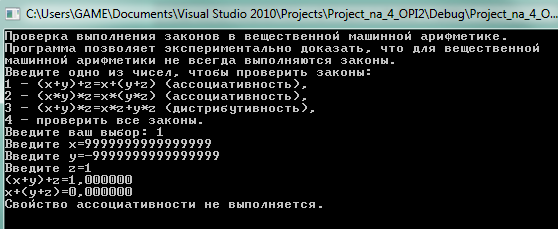
}

\_getch();

return 0;

}

**Тест программы**



**Задача на «5»**

Внутренний таймер системы наблюдения устроен как счетчик количества интервалов времени, прошедшего с момента включения системы. Длина интервала — 0,1 секунды. Чтобы перевести количество этих отрезков в секунды, его нужно разделить на 10, либо умножить на 0,1. В машинной арифметике деление часто подменяется умножением на обратное число для упрощения проектирования вычислительного устройства и ускорения его работы. Написать программу, имитирующую работу таймера в двоичной системе счисления. Определить накопленную ошибку за сутки работы таймера для разных значений точности: 8, 16, 24, 32.

// ecercise\_na5.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <stdlib.h>

#include <stdio.h>

#include <windows.h>

#include <math.h>

#include <locale.h>

int main()

{

setlocale(LC\_ALL, "Russian");

//вычисление погрешности для 8 разрядов

double decate\_num = 0;

double fallibility = 0;

int bin\_8[8] = { 0, 0, 0, 1, 1, 0, 0, 1 };

//перевод в десетичную систему счисления при погрешности 8 разрядов

for (int i = 8; i > 0; i--)

{

decate\_num = bin\_8[i - 1] \* pow(2.0, -i) + decate\_num;

}

printf(" (Значение точности 8) Погрешность за сутки: %g\n", 86400 \* 10 \* (0.1 - decate\_num));

//вычисление погрешности для 16 разрядов

decate\_num = 0;

int bin\_16[16] = { 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1 };

//перевод в десетичную систему счисления при погрешности 16 разрядов

for (int i = 16; i > 0; i--)

{

decate\_num = bin\_16[i - 1] \* pow(2.0, -i) + decate\_num;

}

printf(" (Значение точности 16) Погрешность за сутки : %g\n", 86400 \* 10 \* (0.1 - decate\_num));

//вычисление погрешности для 24 разрядов

decate\_num = 0;

int bin\_24[24] = { 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1 };

//перевод в десетичную систему счисления при погрешности 24 разрядов

for (int i = 24; i > 0; i--)

{

decate\_num = bin\_24[i - 1] \* pow(2.0, -i) + decate\_num;

}

printf(" (Значение точности 24) Погрешность за сутки: %g\n", 86400\* 10 \* (0.1 - decate\_num));

//вычисление погрешности для 32 разрядов

decate\_num = 0;

int bin\_32[32] = { 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1 };

//перевод в десетичную систему счисления при погрешности 24 разрядов

for (int i = 32; i > 0; i--)

{

decate\_num = bin\_32[i - 1] \* pow(2.0, -i) + decate\_num;

}

printf(" (Значение точности 32) Погрешность за сутки: %g\n", 86400\* 10 \* (0.1 - decate\_num));

}

