

Федеральное агентство связи Федеральное государственное бюджетное образовательное
учреждения высшего образования «Сибирский государственный университет
телекоммуникаций и информатики»
(СибГУТИ)

Курсовая работа по курсу «Технология разработки программного обеспечения» для
бакалавров. Заочная форма обучения.

Выполнил:

Группа: ЗП-92

Проверил: _____

Новосибирск, 2020

Введение и постановка задачи

Крупнейшим веб-сервисом для хостинга IT-проектов и их совместной разработки является GitHub. Создатели сайта называют GitHub «социальной сетью для разработчиков».

С помощью широких возможностей Git программисты могут объединять свои репозитории — GitHub предлагает удобный интерфейс для этого и может отображать вклад каждого участника в виде дерева.

Цель работы:

освоение курса по предмету «технологии разработки

программного обеспечения», включающего в себя использование системы

контроля версий git, системы сборки приложения make, а также CI и юнит-

тестирование.

Основной задачей является программа «Calculator» для расчета выражений, введенных пользователем с выводом результата на языке программирования C++ с использованием системы контроля версий Git.

Техническое задание

Функционал приложения:

В консоли приложения пользователь попадает в меню выбора, где ему доступен ввод выражений и значений для расчета. Производится расчет и с выводом результата вычислений. Приложение способно выполнять следующие действия:

Сложение “+”;

Вычитание “-”;

Умножение “*”;

Деление “/”.

Возведение в степень “^”

В момент выполнения программы пользователь также будет видеть правильность написания выражения и доступность действий.

Формат входных данных:

Пользователь вводит числовые значения в консоли приложения. Тип данных double.

Интерфейс приложения:

Приложение реализуется в консольном окне, без графического интерфейса.

План работы

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1. разработка программы;
2. разработка программной документации;
3. испытания программы.

Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. постановка задачи;
2. определение и уточнение требований к техническим средствам;
3. определение требований к программе;
4. определение стадий, этапов и сроков разработки программы и документации на неё;
5. согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

Для законченных проектов будет организован процесс защиты. Каждый член команды представляет письменный отчет о проделанной работе. Отчет состоит из следующих обязательных частей:

1. ТЗ проекта и итоговый план работ на команду
2. Описание командной работы и полученного результата
3. Описание личного вклада в результат работы команды.

Описание готового проекта

Командная работа:

Выявленные задачи были распределены между участниками команды. Каждым участником были успешно выполнены поставленные задачи. Каждый принимал активное участие в разработке логики проекта.

Тестирование:

Для тестирования проекта используется библиотека «`stdio.h`».


Пример написания тестов функции:

```
TEST_CLASS(UnitTestcalculator)
{
public:

    TEST_METHOD(TestSum)
    {
        Assert::IsTrue(sum(2, 3) == 5);
        Assert::IsTrue(sum(-2, -3) == -5);
        Assert::IsTrue(sum(0, 0) == 0);
        Assert::IsTrue(sum(-8, 3) == -5);
        Assert::IsTrue(sum(0, -3) == -3);
        Assert::IsTrue(sum(2, -3) == -1);
    }

    TEST_METHOD(TestMultiply)
    {
        Assert::IsTrue(multiply(2, -3) == -6);
        Assert::IsTrue(multiply(0, -3) == 0);
        Assert::IsTrue(multiply(-2, -3) == 6);
        Assert::IsTrue(multiply(2, -3) == -6);
    }
}
```

Пример работы приложения:



```
Введите выражение и нажмите enter
Пример
A+B
A-B
A*B
A/B
A^B
60/3
Ответ = 20.000000
1 - Рассчитать выражение
2 - Выйти
```

Индивидуальный вклад в проект

У каждого участника в команде есть своя роль и свои задачи. В рамках данного проекта была выполнена роль тестировщика.

Главные обязанности тестировщика:

Выявление и анализ ошибок и проблем, возникающих у пользователей при работе с программными продуктами;

Интеграция travis CI;

Подключение библиотеки и подготовка структуры для тестирования.

Приложение. Листинг программы

// main.cpp

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <stdio.h>
#include <cmath>
#include <locale>
#include <stdlib.h>
using namespace std;

double sum(double a, double b) {
    double r;
    r = a + b;
    return r;
}

double mynus (double a, double b) {
    double r;
    r = a - b;
    return r;
}

double multiply (double a, double b) {
    double r;
    r = a * b;
    return r;
}

double share(double a, double b) {
    double r;
    r = a / b;
    return r;
}

double elevate(double a, double b) {
    double r;
    r = pow(a, b);
    return r;
}

int main()
{
    setlocale(LC_ALL, "Russian");
    double a, b ;
    double c = 0;
    char d;
    int p = 1;
    while (p != 2)
    {
        printf("1 - Рассчитать выражение \n2 - Выйти \n");
        scanf("%d", &p);
        if (p != 1) break;
        system("cls");
        printf("Введите выражение и нажмите enter \n Пример \n A+B \n A-B \n A*B \n A/B \n A^B \n");
        scanf("%lf%c%lf", &a, &d, &b);
        while (getchar() != '\n');
        switch (d)
        {
            case '+': {sum(a, b);    printf("Ответ = %lf\n",    sum(a, b)); }; break;
            case '-': {mynus(a, b);  printf("Ответ = %lf\n",    mynus(a, b)); }; break;
            case '*': {multiply(a, b); printf("Ответ = %lf\n", multiply(a, b)); }; break;
            case '/': {share(a, b);  printf("Ответ = %lf\n",    share(a, b)); }; break;
        }
    }
}
```

```

        case '^': {elevate(a, b); printf("Ответ = %lf\n", elevate(a, b)); }; break;
    default:
        printf("Ошибка ввода\n");
        continue;
    }
}
}

// pch.h

// pch.h: это предварительно скомпилированный заголовочный файл.
// Перечисленные ниже файлы компилируются только один раз, что ускоряет последующие
// сборки.
// Это также влияет на работу IntelliSense, включая многие функции просмотра и завершения
// кода.
// Однако изменение любого из приведенных здесь файлов между операциями сборки приведет к
// повторной компиляции всех(!) этих файлов.
// Не добавляйте сюда файлы, которые планируете часто изменять, так как в этом случае
// выигрыша в производительности не будет.

#ifndef PCH_H
#define PCH_H

// Добавьте сюда заголовочные файлы для предварительной компиляции

#endif //PCH_H

// pch.cpp

// pch.cpp: файл исходного кода, соответствующий предварительно скомпилированному
// заголовочному файлу

#include "pch.h"

// При использовании предварительно скомпилированных заголовочных файлов необходим
// следующий файл исходного кода для выполнения сборки.

//Unit_Test_calculator.cpp

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include "pch.h"
#include "CppUnitTest.h"
#include "..\calculator\main.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTestcalculator
{
    TEST_CLASS(UnitTestcalculator)
    {
    public:

        TEST_METHOD(TestSum)
        {

            Assert::IsTrue(sum(2, 3) == 5);
            Assert::IsTrue(sum(-2, -3) == -5);
            Assert::IsTrue(sum(0, 0) == 0);
            Assert::IsTrue(sum(-8, 3) == -5);
            Assert::IsTrue(sum(0, -3) == -3);
            Assert::IsTrue(sum(2, -3) == -1);
        }
    }
}

```



```

    }

    TEST_METHOD(TestMynus)
    {
        Assert::IsTrue(mynus(2, -3) == 5);
        Assert::IsTrue(mynus(-2, -3) == 1);
        Assert::IsTrue(mynus(0, -8) == 8);
        Assert::IsTrue(mynus(2, 9) == -7);
    }

    TEST_METHOD(TestMultiply)
    {
        Assert::IsTrue(multiply(2, -3) == -6);
        Assert::IsTrue(multiply(0, -3) == 0);
        Assert::IsTrue(multiply(-2, -3) == 6);
        Assert::IsTrue(multiply(2, -3) == -6);
    }

    TEST_METHOD(TestShare)
    {
        Assert::IsTrue(share(8, -2) == -4);
        Assert::IsTrue(share(0, -3) == 0);
        Assert::IsTrue(share(-3, -3) == 1);
        Assert::IsTrue(share(-3, -3) == 1);
        Assert::IsTrue(share(27, 3) == 9);
    }

    TEST_METHOD(TestElevate)
    {
        Assert::IsTrue(elevate(2, 2) == 4);
        Assert::IsTrue(elevate(2, 0) == 1);
        Assert::IsTrue(elevate(2, 1) == 2);
        Assert::IsTrue(elevate(2, 5) == 32);
    }

};

}

```