Assignment_28-08-2024

Author: ThanhTH10
Date: 29/08/2024

1. Explore make file. Write makefile for mathmatical functions in C sqrt, pow, factorial, square,cube...include header file

Step1: Mylib.h

```
double sqrt(double x);
double my_pow(double base, int exp);
unsigned long long factorial(int x);
double square(double x);
double cube(double x);
```

Step2: Logic implementation

- cube.c

```
#include "mylib.h"

double cube(double x)
{
    return x * x;
}
```

- factorial.c

```
#include "mylib.h"
unsigned long long factorial(int n)
{
    if (n < 0)
        return 0; // Factorial is not defined for negative numbers
    unsigned long long result = 1;
    for (int i = 1; i <= n; ++i)
    {
        result *= i;
    }
    return result;
}
```

- my_pow.c

```
#include "mylib.h"
double my_pow(double base, int exp)
{
    if (exp < 0)
        return 1.0 / my_pow(base, -exp); // Handle negative exponents
    double result = 1;
    while (exp > 0)
    {
        if (exp % 2 == 1)
            result *= base;
        base *= base;
        exp /= 2;
    }
    return result;
}
```

- sqrt.c

```
#include "Mylib.h"
double sqrt(double x)
{
    if (x < 0)
        return -1; // Return -1 for negative inputs as square root is not defined
    double tolerance = 1e-10;
    double guess = x;
    while ((guess * guess - x) > tolerance || (x - guess * guess) > tolerance)
    {
        guess = (guess + x / guess) / 2;
    }
    return guess;
}
```

- square.c

```
#include "mylib.h"

double square(double x)
{
    return x * x;
}
```

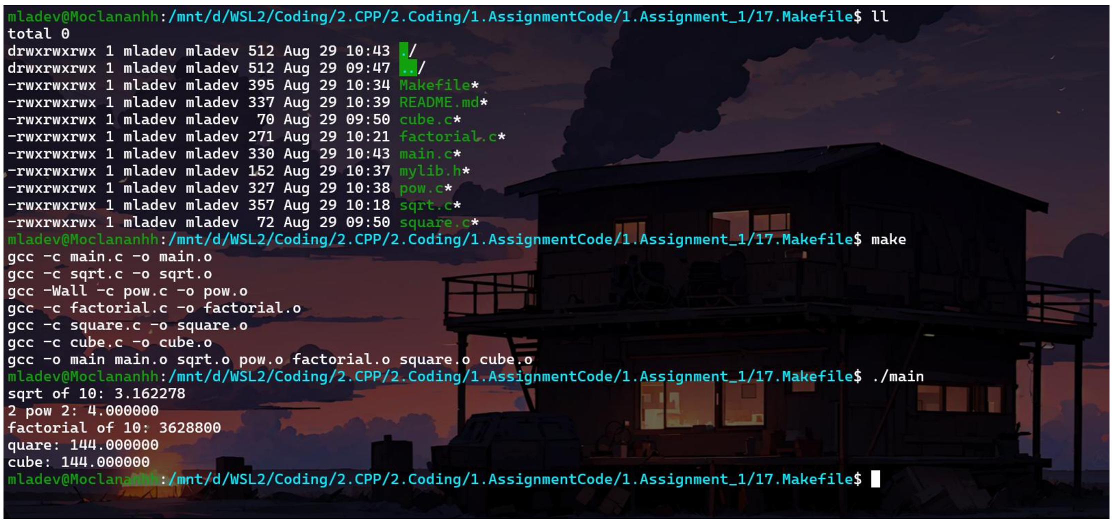
Create main.cpp

```
#include <stdio.h>
#include "mylib.h"

int main(int argc, char const *argv[])
{
    printf("sqrt of 10: %lf\n", sqrt(10));
    printf("2 pow 2: %lf\n", my_pow(2, 2));
    printf("factorial of 10: %lld\n", factorial(10));
    printf("quare: %lf\n", square(12.0));
    printf("cube: %lf\n", cube(12.0));
    return 0;
}
```

Step 3: create Makefile

```
main: main.o sqrt.o pow.o factorial.o square.o cube.o
    gcc -o main main.o sqrt.o pow.o factorial.o square.o cube.o

main.o: main.c
    gcc -c main.c -o main.o

sqrt.o: sqrt.c
    gcc -c sqrt.c -o sqrt.o

pow.o: pow.c
    gcc -Wall -c pow.c -o pow.o

factorial.o: factorial.c
    gcc -c factorial.c -o factorial.o

square.o: square.c
    gcc -c square.c -o square.o

cube.o: cube.c
    gcc -c cube.c -o cube.o
```

Step 4: run program



```
mladev@Moclananhh:/mnt/d/WSL2/Coding/2.CPP/2.Coding/1.AssignmentCode/1.Assignment_1/17.Makefile$ ll
total 0
drwxrwxrwx 1 mladev mladev 512 Aug 29 10:43 ./
drwxrwxrwx 1 mladev mladev 512 Aug 29 09:47 ../
-rwxrwxrwx 1 mladev mladev 395 Aug 29 10:34 Makefile*
-rwxrwxrwx 1 mladev mladev 337 Aug 29 10:39 README.md*
-rwxrwxrwx 1 mladev mladev  70 Aug 29 09:50 cube.c*
-rwxrwxrwx 1 mladev mladev 271 Aug 29 10:21 factorial.c*
-rwxrwxrwx 1 mladev mladev 330 Aug 29 10:43 main.c*
-rwxrwxrwx 1 mladev mladev 152 Aug 29 10:37 mylib.h*
-rwxrwxrwx 1 mladev mladev 327 Aug 29 10:38 pow.c*
-rwxrwxrwx 1 mladev mladev 357 Aug 29 10:18 sqrt.c*
-rwxrwxrwx 1 mladev mladev  72 Aug 29 09:50 square.c*
mladev@Moclananhh:/mnt/d/WSL2/Coding/2.CPP/2.Coding/1.AssignmentCode/1.Assignment_1/17.Makefile$ make
gcc -c main.c -o main.o
gcc -c sqrt.c -o sqrt.o
gcc -Wall -c pow.c -o pow.o
gcc -c factorial.c -o factorial.o
gcc -c square.c -o square.o
gcc -c cube.c -o cube.o
gcc -o main main.o sqrt.o pow.o factorial.o square.o cube.o
mladev@Moclananhh:/mnt/d/WSL2/Coding/2.CPP/2.Coding/1.AssignmentCode/1.Assignment_1/17.Makefile$ ./main
sqrt of 10: 3.162278
2 pow 2: 4.000000
factorial of 10: 3628800
quare: 144.000000
cube: 144.000000
mladev@Moclananhh:/mnt/d/WSL2/Coding/2.CPP/2.Coding/1.AssignmentCode/1.Assignment_1/17.Makefile$ 
```

2. Write makefile for C++ files, for add, subtract, mul,divide..main.cpp
g++ queue.cpp -fprofile-arcs -ftest-coverage
./a.out
gcov xxx.cpp

Step 1: Create header file

```
double add(double a, double b);
double subtract(double a, double b);
double multiply(double a, double b);
double divide(double a, double b);
```

Step 2: Logic implementation
- add.cpp

```
#include "mylib.h"

double add(double a, double b)
{
    return a + b;
}
```

- subtract.cpp

```
#include "mylib.h"

double subtract(double a, double b)
{
    return a - b;
}
```

- multiply.cpp

```
#include "mylib.h"

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

- divide.cpp

```cpp
#include "mylib.h"
#include <iostream>
double divide(double a, double b)
{
    if (b == 0)
    {
        throw std::invalid_argument("Division by zero");
    }
    return a / b;
}
```

Create main.cpp

```cpp
#include <iostream>
#include "mylib.h"

int main()
{
    double a = 10.0, b = 5.0;
    std::cout << "Add: " << add(a, b) << std::endl;
    std::cout << "Subtract: " << subtract(a, b) << std::endl;
    std::cout << "Multiply: " << multiply(a, b) << std::endl;
    std::cout << "Divide: " << divide(a, b) << std::endl;
    return 0;
}
```

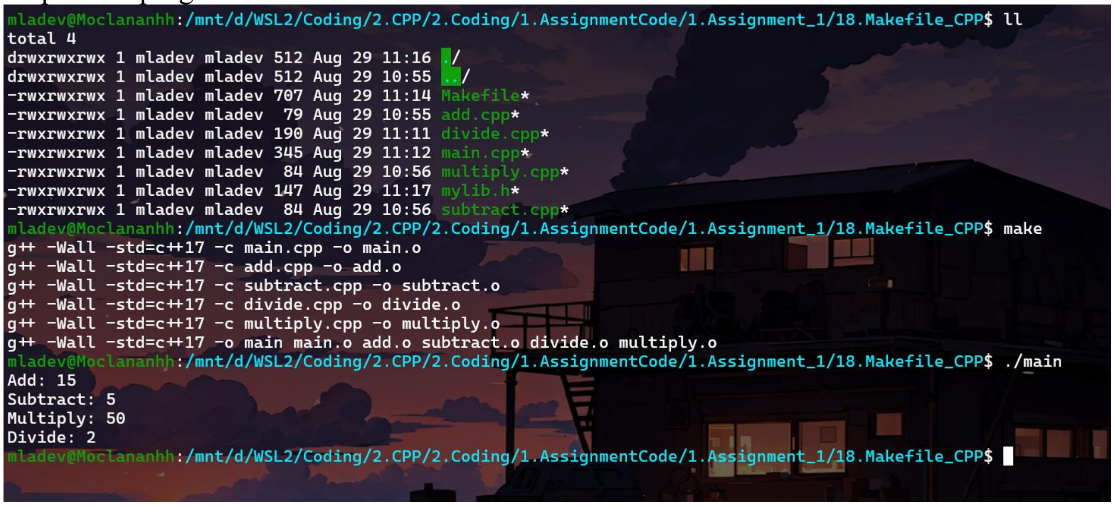
Step 3: Create Makefile

```makefile
# Compiler
CXX = g++

# Compiler flags
CXXFLAGS = -Wall -std=c++17
# Target executable
TARGET = main
# Source files
SRCS = main.cpp _add.cpp _subtract.cpp _divide.cpp _multiply.cpp
# Test source files
TEST_SRCS = _add.cpp _subtract.cpp _divide.cpp _multiply.cpp test.cpp test_main.cpp
# Object files
OBJS = $(SRCS:.cpp=.o)
TEST_OBJS = $(TEST_SRCS:.cpp=.o)
# Default rule
all: $(TARGET)
# Linking rule
$(TARGET): $(OBJS)
    $(CXX) $(CXXFLAGS) -o $@ $^ -fprofile-arcs -ftest-coverage -L/usr/lib/x86_64-linux-
gnu/CppUTest -lCppUTest
# Test executable
test: $(TEST_OBJS)
    $(CXX) $(CXXFLAGS) $(TEST_OBJS) -fprofile-arcs -ftest-coverage -o test -L/usr/lib/x86_64-
linux-gnu/CppUTest -lCppUTest
```

```makefile
# Compilation rule
%.o: %.cpp
    $(CXX) $(CXXFLAGS) -fprofile-arcs -ftest-coverage -c $< -o $@
# Clean rule
clean:
    rm -f $(TARGET) test $(OBJS) $(TEST_OBJS) *.gcda *.gcno *.gcov
# Run rule
```

```
run: $(TARGET)
    ./$(TARGET)
# Coverage rule
coverage: $(TEST_OBJS)
    $(CXX) $(CXXFLAGS) -fprofile-arcs -ftest-coverage -o $(TARGET) $(TEST_OBJS) -
L/usr/lib/x86_64-linux-gnu/CppUTest -lCppUTest
    ./$(TARGET)
    gcov _*.cpp
.PHONY: all clean run coverage test
```

Step 4: run program



```
mladev@Moclananhh:/mnt/d/WSL2/Coding/2.CPP/2.Coding/1.AssignmentCode/1.Assignment_1/18.Makefile_CPP$ ll
total 4
drwxrwxrwx 1 mladev mladev 512 Aug 29 11:16 ./
drwxrwxrwx 1 mladev mladev 512 Aug 29 10:55 ../
-rwxrwxrwx 1 mladev mladev 707 Aug 29 11:14 Makefile*
-rwxrwxrwx 1 mladev mladev  79 Aug 29 10:55 add.cpp*
-rwxrwxrwx 1 mladev mladev 190 Aug 29 11:11 divide.cpp*
-rwxrwxrwx 1 mladev mladev 345 Aug 29 11:12 main.cpp*
-rwxrwxrwx 1 mladev mladev  84 Aug 29 10:56 multiply.cpp*
-rwxrwxrwx 1 mladev mladev 147 Aug 29 11:17 mylib.h*
-rwxrwxrwx 1 mladev mladev  84 Aug 29 10:56 subtract.cpp*
mladev@Moclananhh:/mnt/d/WSL2/Coding/2.CPP/2.Coding/1.AssignmentCode/1.Assignment_1/18.Makefile_CPP$ make
g++ -Wall -std=c++17 -c main.cpp -o main.o
g++ -Wall -std=c++17 -c add.cpp -o add.o
g++ -Wall -std=c++17 -c subtract.cpp -o subtract.o
g++ -Wall -std=c++17 -c divide.cpp -o divide.o
g++ -Wall -std=c++17 -c multiply.cpp -o multiply.o
g++ -Wall -std=c++17 -o main main.o add.o subtract.o divide.o multiply.o
mladev@Moclananhh:/mnt/d/WSL2/Coding/2.CPP/2.Coding/1.AssignmentCode/1.Assignment_1/18.Makefile_CPP$ ./main
Add: 15
Subtract: 5
Multiply: 50
Divide: 2
mladev@Moclananhh:/mnt/d/WSL2/Coding/2.CPP/2.Coding/1.AssignmentCode/1.Assignment_1/18.Makefile_CPP$
```

Create test coverage

Step 1: Create test_main.cpp

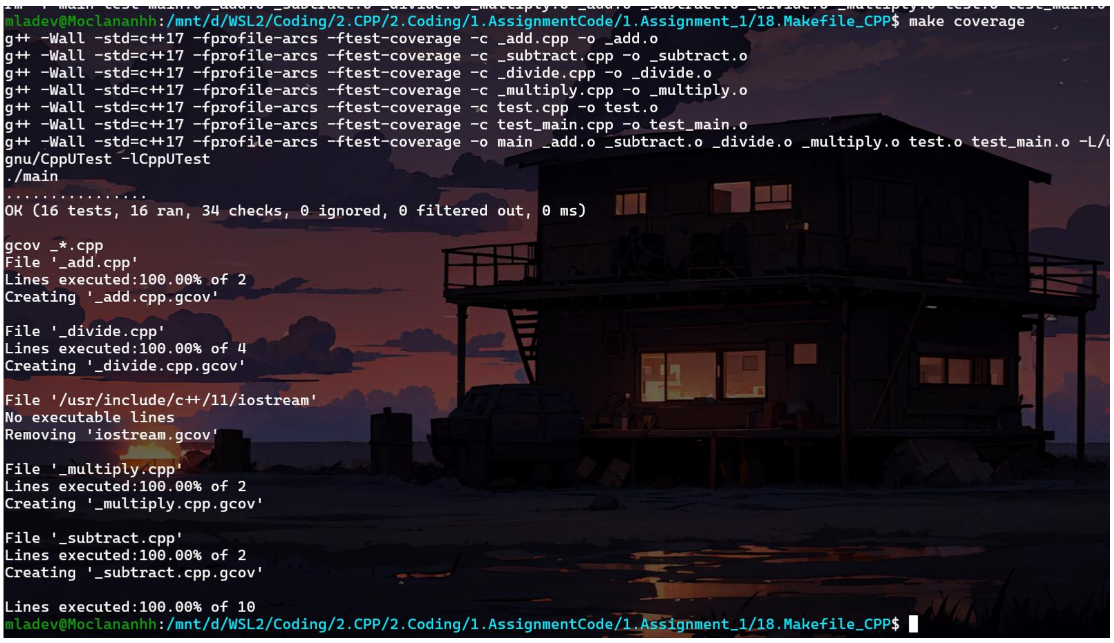
```
#include <CppUTest/CommandLineTestRunner.h>
int main(int ac, char **av)
{
    return CommandLineTestRunner::RunAllTests(ac, av);
}
```

Step 2: Create test.cpp

```
#include <CppUTest/Utest.h>
#include <CppUTest/UtestMacros.h>
#include "CppUTest/TestHarness.h"
#include "mylib.h"
#include <stdexcept>

// Test Group
TEST_GROUP(DivideGroup){};
// Test case for normal division
TEST(DivideGroup, HandlesPositiveNumbers)
{
    DOUBLES_EQUAL(5.0, divide(10.0, 2.0), 0.0001);
    DOUBLES_EQUAL(3.0, divide(9.0, 3.0), 0.0001);
}
// Test case for division by zero
TEST(DivideGroup, HandlesDivisionByZero)
{
    CHECK_THROWS(std::invalid_argument, divide(10.0, 0.0));
}
// Test case for negative numbers
TEST(DivideGroup, HandlesNegativeNumbers)
{
    DOUBLES_EQUAL(-5.0, divide(-10.0, 2.0), 0.0001);
    DOUBLES_EQUAL(-5.0, divide(10.0, -2.0), 0.0001);
    DOUBLES_EQUAL(5.0, divide(-10.0, -2.0), 0.0001);
}
// Test case for division resulting in a fraction
TEST(DivideGroup, HandlesFractionResult)
{
    DOUBLES_EQUAL(2.5, divide(10.0, 4.0), 0.0001);
}





// Test Group
TEST_GROUP(AddGroup){};
// Test case for adding two positive numbers
TEST(AddGroup, HandlesPositiveNumbers)
{
    DOUBLES_EQUAL(10.0, add(3.0, 7.0), 0.0001);
    DOUBLES_EQUAL(5.0, add(0.0, 5.0), 0.0001);
}
// Test case for adding two negative numbers
TEST(AddGroup, HandlesNegativeNumbers)
{
    DOUBLES_EQUAL(-10.0, add(-3.0, -7.0), 0.0001);
    DOUBLES_EQUAL(-10.0, add(-5.0, -5.0), 0.0001);
}
// Test case for adding a positive and a negative number
TEST(AddGroup, HandlesMixedSignNumbers)
{
    DOUBLES_EQUAL(4.0, add(-3.0, 7.0), 0.0001);
    DOUBLES_EQUAL(-4.0, add(3.0, -7.0), 0.0001);
}
// Test case for adding zero
TEST(AddGroup, HandlesZero)
{
    DOUBLES_EQUAL(0.0, add(0.0, 0.0), 0.0001);
    DOUBLES_EQUAL(10.0, add(10.0, 0.0), 0.0001);
    DOUBLES_EQUAL(-10.0, add(0.0, -10.0), 0.0001);
}





// Test Group
TEST_GROUP(SubtractGroup){};
// Test case for subtracting two positive numbers
TEST(SubtractGroup, HandlesPositiveNumbers)
{
    DOUBLES_EQUAL(3.0, subtract(10.0, 7.0), 0.0001);
    DOUBLES_EQUAL(0.0, subtract(5.0, 5.0), 0.0001);
}
// Test case for subtracting two negative numbers
TEST(SubtractGroup, HandlesNegativeNumbers)
{
    DOUBLES_EQUAL(-2.0, subtract(-7.0, -5.0), 0.0001);
    DOUBLES_EQUAL(-5.0, subtract(-10.0, -5.0), 0.0001);
}
// Test case for subtracting a positive number from a negative number
```

```
TEST(SubtractGroup, HandlesMixedSignNumbers)
{
    DOUBLES_EQUAL(-10.0, subtract(-3.0, 7.0), 0.0001);
    DOUBLES_EQUAL(10.0, subtract(3.0, -7.0), 0.0001);
}
// Test case for subtracting zero
TEST(SubtractGroup, HandlesZero)
{
    DOUBLES_EQUAL(10.0, subtract(10.0, 0.0), 0.0001);
    DOUBLES_EQUAL(-10.0, subtract(-10.0, 0.0), 0.0001);
    DOUBLES_EQUAL(0.0, subtract(0.0, 0.0), 0.0001);
}



// Test Group
TEST_GROUP(MultiplyGroup){};
// Test case for multiplying two positive numbers
TEST(MultiplyGroup, HandlesPositiveNumbers)
{
    DOUBLES_EQUAL(20.0, multiply(4.0, 5.0), 0.0001);
    DOUBLES_EQUAL(0.0, multiply(0.0, 5.0), 0.0001);
}
// Test case for multiplying two negative numbers
TEST(MultiplyGroup, HandlesNegativeNumbers)
{
    DOUBLES_EQUAL(20.0, multiply(-4.0, -5.0), 0.0001);
    DOUBLES_EQUAL(25.0, multiply(-5.0, -5.0), 0.0001);
}
// Test case for multiplying a positive number by a negative number
TEST(MultiplyGroup, HandlesMixedSignNumbers)
{
    DOUBLES_EQUAL(-20.0, multiply(-4.0, 5.0), 0.0001);
    DOUBLES_EQUAL(-15.0, multiply(3.0, -5.0), 0.0001);
}
// Test case for multiplying by zero
TEST(MultiplyGroup, HandlesZero)
{
    DOUBLES_EQUAL(0.0, multiply(10.0, 0.0), 0.0001);
    DOUBLES_EQUAL(0.0, multiply(0.0, 10.0), 0.0001);
    DOUBLES_EQUAL(0.0, multiply(0.0, 0.0), 0.0001);
}
```

Step 3: Run test



```
rm -f main test main.o _add.o _subtract.o _divide.o _multiply.o _add.o _subtract.o _divide.o _multiply.o test.o test_main.o
mladev@Moclananhh:/mnt/d/WSL2/Coding/2.CPP/2.Coding/1.AssignmentCode/1.Assignment_1/18.Makefile_CPP$ make coverage
g++ -Wall -std=c++17 -fprofile-arcs -ftest-coverage -c _add.cpp -o _add.o
g++ -Wall -std=c++17 -fprofile-arcs -ftest-coverage -c _subtract.cpp -o _subtract.o
g++ -Wall -std=c++17 -fprofile-arcs -ftest-coverage -c _divide.cpp -o _divide.o
g++ -Wall -std=c++17 -fprofile-arcs -ftest-coverage -c _multiply.cpp -o _multiply.o
g++ -Wall -std=c++17 -fprofile-arcs -ftest-coverage -c test.cpp -o test.o
g++ -Wall -std=c++17 -fprofile-arcs -ftest-coverage -c test_main.cpp -o test_main.o
g++ -Wall -std=c++17 -fprofile-arcs -ftest-coverage -o main _add.o _subtract.o _divide.o _multiply.o test.o test_main.o -L/u
gnu/CppUTest -lCppUTest
./main
................
OK (16 tests, 16 ran, 34 checks, 0 ignored, 0 filtered out, 0 ms)

gcov _*.cpp
File '_add.cpp'
Lines executed:100.00% of 2
Creating '_add.cpp.gcov'

File '_divide.cpp'
Lines executed:100.00% of 4
Creating '_divide.cpp.gcov'

File '/usr/include/c++/11/iostream'
No executable lines
Removing 'iostream.gcov'

File '_multiply.cpp'
Lines executed:100.00% of 2
Creating '_multiply.cpp.gcov'

File '_subtract.cpp'
Lines executed:100.00% of 2
Creating '_subtract.cpp.gcov'

Lines executed:100.00% of 10
mladev@Moclananhh:/mnt/d/WSL2/Coding/2.CPP/2.Coding/1.AssignmentCode/1.Assignment_1/18.Makefile_CPP$
```