# Assignment_16-08-2024

Author: ThanhTH10
Date: 19/08/2024

---

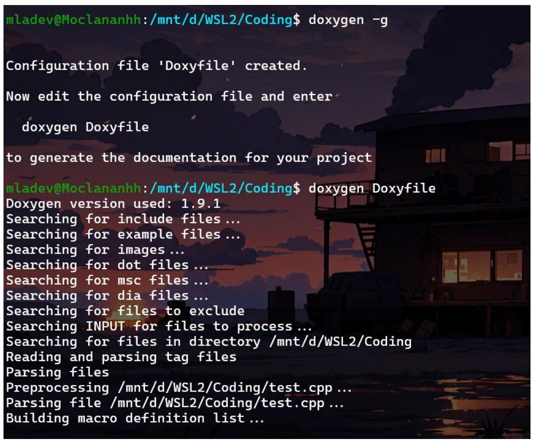
## August-16:

1. How to generate documentation with doxygen tool for c++ programs.

### Step 1: Comment in cpp file

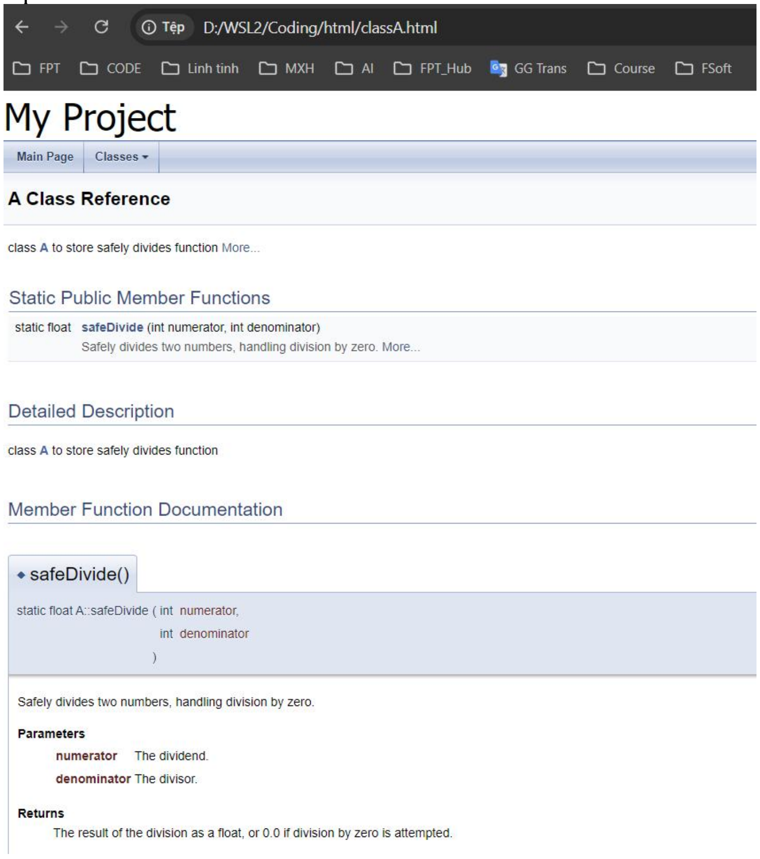
```cpp
#include <stdio.h>
/**
 * @brief class A to store safely divides function
 *
 */
class A
{
public:
    /**
     * @brief Safely divides two numbers, handling division by zero.
     *
     * @param numerator The dividend.
     * @param denominator The divisor.
     * @return The result of the division as a float, or 0.0 if division by zero is attempted.
     */
    static float safeDivide(int numerator, int denominator)
    {
        if (denominator == 0)
        {
            printf("Error: Division by zero is not allowed.\n");
            return 0.0;
        }

        return (float)numerator / denominator;
    }
};
/**
 * @brief Main function to test safe division
 *
 * @return int
 */
int main()
{
    int num1, num2;
    float result;
    printf("Enter numerator: ");
    scanf("%d", &num1);
    printf("Enter denominator: ");
    scanf("%d", &num2);
    result = A::safeDivide(num1, num2);
    if (num2 != 0)
    {
        printf("Result: %.2f\n", result);
    }
    return 0;
}
```

### Step 2: Config doxygen file and run command



```
mladev@Moclananhh:/mnt/d/WSL2/Coding$ doxygen -g

Configuration file 'Doxyfile' created.

Now edit the configuration file and enter

  doxygen Doxyfile

to generate the documentation for your project

mladev@Moclananhh:/mnt/d/WSL2/Coding$ doxygen Doxyfile
Doxygen version used: 1.9.1
Searching for include files...
Searching for example files...
Searching for images...
Searching for dot files...
Searching for msc files...
Searching for dia files...
Searching for files to exclude
Searching INPUT for files to process...
Searching for files in directory /mnt/d/WSL2/Coding
Reading and parsing tag files
Parsing files
Preprocessing /mnt/d/WSL2/Coding/test.cpp...
Parsing file /mnt/d/WSL2/Coding/test.cpp...
Building macro definition list...
```

Step 3:
Open index.html



My Project

A Class Reference

class A to store safely divides function More...

Static Public Member Functions

static float safeDivide (int numerator, int denominator)
Safely divides two numbers, handling division by zero. More...

Detailed Description

class A to store safely divides function

Member Function Documentation

◆ safeDivide()

```
static float A::safeDivide ( int  numerator,
                             int  denominator
                           )
```

Safely divides two numbers, handling division by zero.

Parameters
numerator The dividend.
denominator The divisor.

Returns
The result of the division as a float, or 0.0 if division by zero is attempted.

2.
int main()
{
char *s=”rama”; -> Why it is not part of stack/heap?
printf(“%s”,s);
}

The string literal "rama" is not allocated on the stack or heap. Instead, it is stored in a special section of the memory called the **read-only data segment** (sometimes called the **text segment** or **constant data segment**).

**- String Literals are Immutable**: String literals in C are stored in a read-only section of the memory. This is because string literals are typically immutable (cannot be modified). Thus, "rama" is placed in this read-only data segment by the compiler, which persists for the entire lifetime of the program.

- **Pointer** s **on Stack**: The pointer s itself is stored on the stack because it is a local variable of the main function. However, the string "rama" that s points to is in the read-only data segment.

- **Not on Heap**: The string literal "rama" is not stored on the heap because the heap is used for dynamic memory allocation

**Memory Layout:**

- **Stack**: The pointer s is stored on the stack.
- **Read-only Data Segment**: The string literal "rama" is stored in a read-only memory segment.
- **Heap**: Not used in this case because no dynamic memory allocation has been performed.

## 3.Create Distance class with feet and inches as data members. Perform add, subtract with + and – operator overloading

```
#include <iostream>
using namespace std;

class Distance
{
    float feet;
    float inches;
public:
    Distance(float f = 0.0, float i = 0.0) : feet(f), inches(i) {}
    void display()
    {
        cout << feet << " feet, " << inches << " inches" << endl;
    }
    Distance operator+(const Distance &d);
    Distance operator-(const Distance &d);
};
Distance Distance::operator+(const Distance &d)
{
    float totalFeet = feet + d.feet;
    float totalInches = inches + d.inches;
    if (totalInches >= 12.0)
    {
        int extraFeet = totalInches / 12.0;
        totalFeet += extraFeet;
        totalInches -= extraFeet * 12.0;
    }
    return Distance(totalFeet, totalInches);
}
Distance Distance::operator-(const Distance &d)
{
    float totalFeet = feet - d.feet;
    float totalInches = inches - d.inches;
    if (totalInches < 0)
    {
        totalFeet -= 1.0;
        totalInches += 12.0;
    }
    return Distance(totalFeet, totalInches);
}
int main()
{
    Distance d1(5, 6.5);
    Distance d2(3, 8.2);
    Distance d3 = d1 + d2;
    d3.display();
    Distance d4 = d1 - d2;
    d4.display();
    return 0;
}
```

Output:

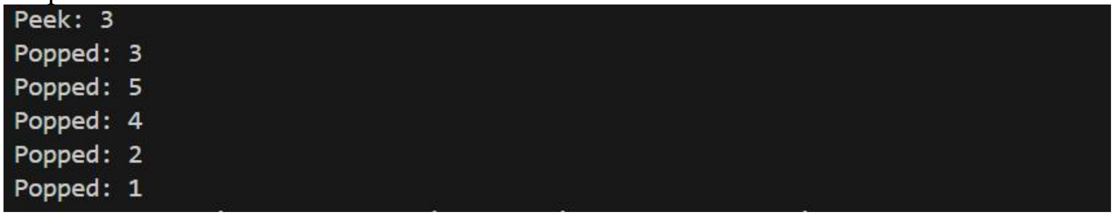
```
9 feet, 2.7 inches
1 feet, 10.3 inches
```

4. Create Stack with dynmaic memory allocation new, delete and perform push and pop operations

```
#include <iostream>
using namespace std;

class Stack
{
    int *arr;
    int top;
    int capacity;
public:
    Stack(int initCapcity = 10) : top(-1), capacity(initCapcity)
    {
        arr = new int[capacity];
    }
    ~Stack()
    {
        delete[] arr;
    }
    void push(int element);
    int pop();
    int peek();
    bool isEmpty();
    bool isFull();
};
bool Stack::isFull()
{
    return top == capacity - 1;
}
bool Stack::isEmpty()
{
    return top == -1;
}
void Stack::push(int element)
{
    if (isFull())
    {
        // if the stack is full, reallocate with double the capacity
        int *newArr = new int[capacity * 2];
        for (int i = 0; i < capacity; i++)
        {
            newArr[i] = arr[i];
        }
        delete[] arr;
        arr = newArr;
        capacity *= 2;
    }
```

```cpp
    arr[++top] = element;
}
int Stack::pop()
{
    if (isEmpty())
    {
        cerr << "Stack empty" << endl;
        return -1;
    }
    return arr[top--];
}
int Stack::peek()
{
    if (isEmpty())
    {
        cerr << "Stack empty" << endl;
        return -1;
    }
    return arr[top];
}
int main(int argc, char const *argv[])
{
    Stack s;
    s.push(1);
    s.push(2);
    s.push(3);
    cout << "Peek: " << s.peek() << endl; // prints 3
    int popped = s.pop();
    cout << "Popped: " << popped << endl; // prints 3
    s.push(4);
    s.push(5);
    while (!s.isEmpty())
    {
        cout << "Popped: " << s.pop() << endl;
    }
    return 0;
}
```

Output:



```
Peek: 3
Popped: 3
Popped: 5
Popped: 4
Popped: 2
Popped: 1
```