**Sandesh Ghimire**

**CSCI 301 section 3**

**Computer Science 2**

**Project 5**

**7th October**

**Introduction**

We implemented Stack (ADT) which works on the basis of last in first out. This project contains three files, " *stack.h* “, " *stack.cpp* " and " *main.cpp* “. This program finds out the prime factor of an entered number. The input from a user is taken from the main function. After receiving the input, a function PrimeFactor is called with the input integer and value of i as a parameter. This function starts with the with a conditional loop. If the condition is satisfied the value is pushed in the stack. At last the function prints out the integers in descending order by using *pop()* function of stack.

**Data Structures:**

We used Stack as a data structure which is a Abstract data type. We declared the stack using an array. The stack has two major function i.e. *push()* and *pop().* Push function enters the integer inside the array whereas pop function takes the integer out and returns it.

**Functions:**

PrimeFactor(): It carries two parameters of integer data type (Num1 and i). The function starts with the base condition that checks if the input integer (Num1) is smaller than the square root of itself. If the base condition is satisfied it goes to 2nd condition that checks if the remainder when Num1 is divided by I is zero or not. If the second condition is satisfied it pushes the value of i and then sets the value of *Num1* to *Num1/i*. If the second condition is not satisfied, it simply increases the value of i by 1. Lastly, the postcondition pushes the number that cannot be divided by itself i.e. a prime number.

*void push(int entry):* This function takes integer as a parameter. It stores that parameter to the array and then increases the value of *used* by one.

*int pop():* This function doesn’t take any parameter. It just decreases the value of *used* by one and returns the current value inside array.

*int Size():* It returns the current used length of array.

**The Main Program:**

The main function asks for an integer as input by displaying the message. It then calls the function PrimeFactor which displays the calculated factors by manipulating the Stack data type we just created.

**Code**

**Stack.cpp**

#include <iostream>

#include "stack.h"

using namespace std;

void Stack::push(int entry){

data[used]=entry;//inserting data into the array

++used;//increasing the value of used by one

}

int Stack::pop(){

--used;//decreasing the value of used by one

return data[used];//returning the present value of array

}

**Stack.h**

#include <iostream>

#ifndef LIST\_H

#define LIST\_H

using namespace std;

class Stack{

public://public functions of the class

typedef int Item; //defining the Item using int

static const int CAPACITY=20;//setting the capacity of an array

Stack(){used=0;}; //constructor

void push(int entry);//to push the integer

int pop();//to pop out the integer

//Member functions

int Size(){return used;};//to return the size used by the array

private://private functions of class

Item data[CAPACITY];//defining array

int used;

};

#endif // list\_H

**main.cpp**

//CSCI 301

/\*

This program finds the prime factor of any numbers. The input from a user is taken from the main function.

After receiving the input a function (PrimeFactor) is called with the input integer and value of i as parameter.

This function starts with the base condition which runs the 2nd condition if satisfied. the second condition finds a

prime factor, push it to the array and change the value of Num1. With the post condition it terminates the loop . At last prints the prime factors.

\*/

#include <iostream>

//#include <cassert>

#include "stack.h"

#include "stack.cpp"

#include <cmath>

using namespace std;

void PrimeFactor(int, int);

int main()

{

Stack l;

int Num;

cout << "Enter a positive Integer from 0 to stop "; //asking an integer

cin >> Num; //Assigning the input to Num variable

int i = 2; //initializing the value of i

cout << "The prime factors of " << Num << " are: "; //Comment that displays the prime factors

PrimeFactor(Num, i); //calling the PrimeFactor function

return 0;

}

void PrimeFactor(int Num1, int i)

{

Stack l;//declearing the stack l

int value;//initializing value

bool check=false;

while (check==false){//base condition

if(i <= sqrt(Num1)) //**this is pre conditio where it checks if I is less then or equal to sqr root of Num1**

{

if (Num1 % i == 0)

{

//2nd condition where code looks for prime factor and prints it

l.push(i); //pushing the factor

Num1=Num1/i;

}

else

{

i = i + 1; //increasing the value of i

}

}

else

{

//post condition

l.push(Num1);

check=true;

}

}

while (l.Size()>0){ // to print the values

cout<<l.pop()<<" ";//pop and return the data

}

}

**User Document**

The program can be used by an integer. When the user enters an integer, the program applies the computation and prints the prime factor of entered integer in descending order. Only positive integer is accepted and the number should be within the provided range.

To compile the program simply enter:

*g++ -o main main.cpp*

To run the program, enter. */main,* then you can input an integer. After entering an integer, the program will display the prime factors of the entered number.

For example, when you run the program it asks input:

*Prompt>./main*

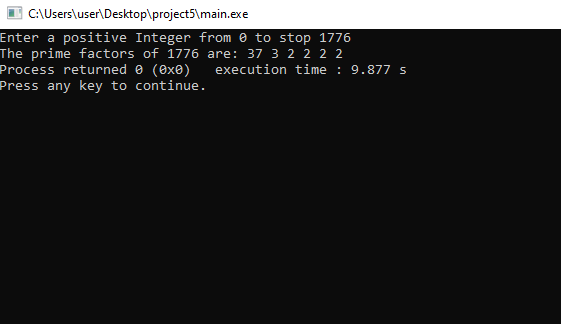
*Enter a positive Integer from 0 to stop: 123*

If you enter those two lines the program displays,

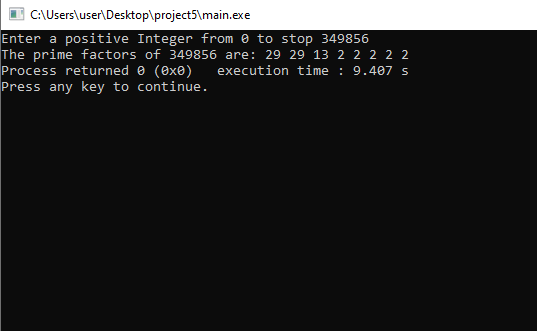
*The prime factors of 123 are: 43 \* 3*

**Testing**

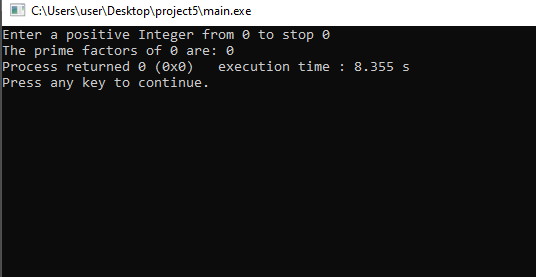
TEST1



TEST2



TEST3



**Summary**

In this project we implemented the ADT i.e. stack to find the prime factor of a positive integer. We constructed a class called stack that can push and pop the integer. They also have many different ways to do the same problem. For example: In the function *PrimeFactor* we could have set the value of *i*3 at first and then increased the value of the variable by 2. It saves a lot of time. We can do the same problem using queue ADT to print the result in increasing order. We can also use linked list to build stack instead of array. This would be more efficient because we don’t have to care about the size of the list.

 If we wanted to report each integer's prime factors in **increasing** order, would the stack be necessary or helpful? Explain.

No, because the stack uses last in first out method. we stored the value in ascending order in the stack. So, when it comes out it prints in descending order.