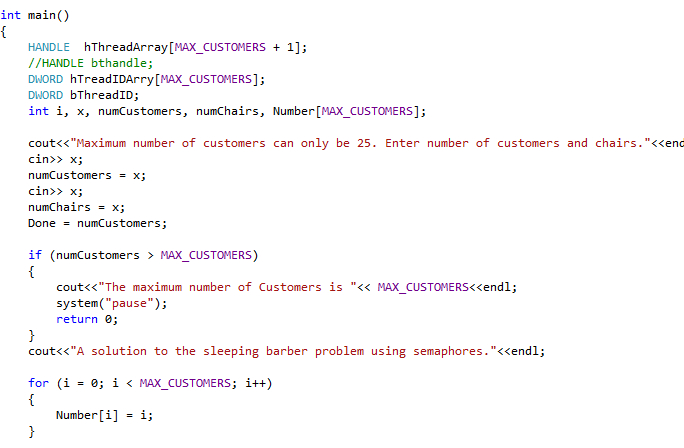
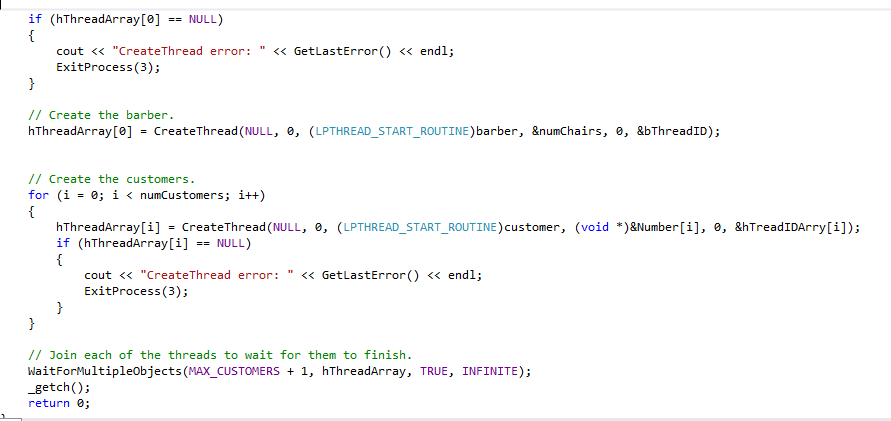
**The Sleeping-Barber Problem**

**In this problem can only handle 25 customers as max, customers sometime enter the shop while Barber is sleeping. It is the customer desire if he or she wants to wait or leave. That why, customers choose to enter waiting room is random. This program will max out number of customers waiting on the number of seats available. First customer enter the waiting room will wake up the barber and the other people in the waiting room will wait. Customers whom do not want to wait will leave the shop.**

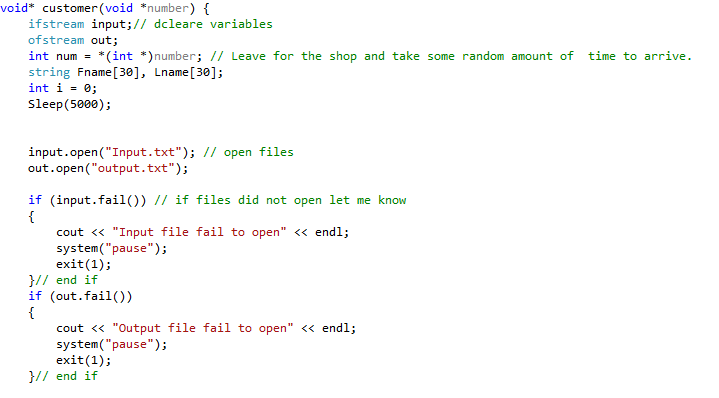
**Flow diagram:**

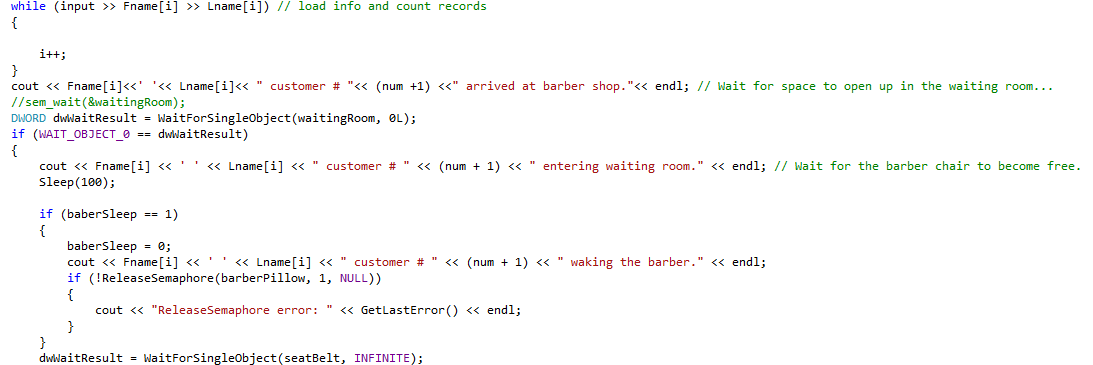
**Functions description:**

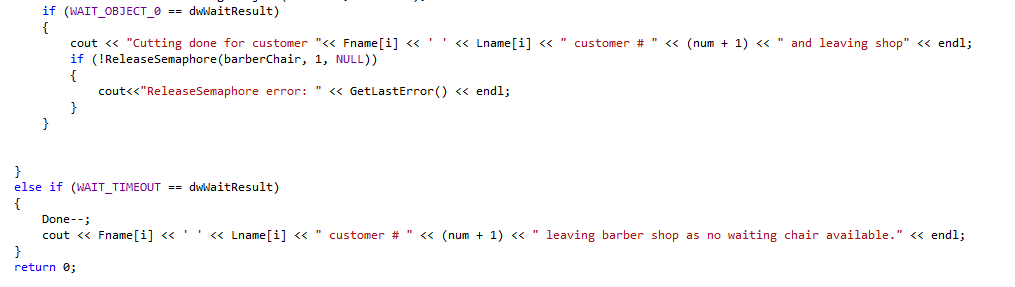
****

****

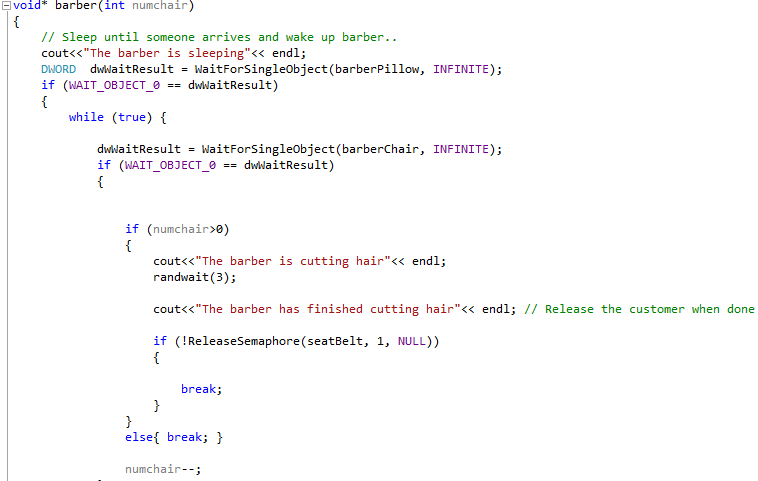
**Main function will allow user to enter number of customers and chairs to determent how many customers can wait otherwise they will leave the shop. Also, will create all threads and call for other functions. Finally, it will say if barber sleeping or not.**

****

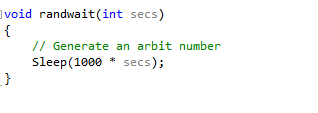
****

****

**Customers function will load customer’s names. Show arrival to the shop and determent if a customer wants to stay or leave. First waiting customer will wake up the barber if he/she is sleeping. When barber awake he/she will start cutting hair and inform user when customer done.**

****

**Barber function will shoe process of barber during cutting hair.**

****

**Randwait function is to determent the amount of time the barber sleeping and how much time a customer willing to wait.**

**Code:**

/\*

Abdullah Alajaj

PP2

3/19/2017

\*/

#include <fstream>

#include <stdio.h>

#include<conio.h>

#include<windows.h>

#include <iostream>

#include<string>

// The maximum number of customers threads.

#define MAX\_CUSTOMERS 25 // Functions prototypes

void \*barber(int);

void \*customer(void \*num);

void randwait(int secs);

using namespace std;

//Define the semaphores.

// waiting Room Limits the number of customers allowed to enter the waiting room at one time.

HANDLE waitingRoom;

// barberChair ensures mutually exclusive access to the barber chair.

HANDLE barberChair;

// barberPillow is used to allow the barber to sleep until a customer arrives.

HANDLE barberPillow;

// seatBelt is used to make the customer to wait until the barber is done cutting his/her hair.

HANDLE seatBelt;

// Flag to stop the barber thread when all customers have been serviced.

static int Done = 0;

int baberSleep = 1;

int main()

{

HANDLE hThreadArray[MAX\_CUSTOMERS + 1];

//HANDLE bthandle;

DWORD hTreadIDArry[MAX\_CUSTOMERS];

DWORD bThreadID;

int i, x, numCustomers, numChairs, Number[MAX\_CUSTOMERS];

cout<<"Maximum number of customers can only be 25. Enter number of customers and chairs."<<endl;

cin>> x;

numCustomers = x;

cin>> x;

numChairs = x;

Done = numCustomers;

if (numCustomers > MAX\_CUSTOMERS)

{

cout<<"The maximum number of Customers is "<< MAX\_CUSTOMERS<<endl;

system("pause");

return 0;

}

cout<<"A solution to the sleeping barber problem using semaphores."<<endl;

for (i = 0; i < MAX\_CUSTOMERS; i++)

{

Number[i] = i;

}

waitingRoom = CreateSemaphore(NULL, numChairs, numChairs, L"waitingRoom"); // Initialize the semaphores with initial values

barberChair = CreateSemaphore(NULL, 0, 1, L"barberChair");

barberPillow = CreateSemaphore(NULL, 0, 1, L"barberPillow");

seatBelt = CreateSemaphore(NULL, 1, 1, L"seatBelt");

if (hThreadArray[0] == NULL)

{

cout << "CreateThread error: " << GetLastError() << endl;

ExitProcess(3);

}

// Create the barber.

hThreadArray[0] = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)barber, &numChairs, 0, &bThreadID);

// Create the customers.

for (i = 0; i < numCustomers; i++)

{

hThreadArray[i] = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)customer, (void \*)&Number[i], 0, &hTreadIDArry[i]);

if (hThreadArray[i] == NULL)

{

cout << "CreateThread error: " << GetLastError() << endl;

ExitProcess(3);

}

}

// Join each of the threads to wait for them to finish.

WaitForMultipleObjects(MAX\_CUSTOMERS + 1, hThreadArray, TRUE, INFINITE);

\_getch();

return 0;

}

void\* customer(void \*number) {

ifstream input;// dcleare variables

ofstream out;

int num = \*(int \*)number; // Leave for the shop and take some random amount of time to arrive.

string Fname[30], Lname[30];

int i = 0;

Sleep(5000);

input.open("Input.txt"); // open files

out.open("output.txt");

if (input.fail()) // if files did not open let me know

{

cout << "Input file fail to open" << endl;

system("pause");

exit(1);

}// end if

if (out.fail())

{

cout << "Output file fail to open" << endl;

system("pause");

exit(1);

}// end if

while (input >> Fname[i] >> Lname[i]) // load info and count records

{

i++;

}

cout << Fname[i]<<' '<< Lname[i]<< " customer # "<< (num +1) <<" arrived at barber shop."<< endl; // Wait for space to open up in the waiting room...

//sem\_wait(&waitingRoom);

DWORD dwWaitResult = WaitForSingleObject(waitingRoom, 0L);

if (WAIT\_OBJECT\_0 == dwWaitResult)

{

cout << Fname[i] << ' ' << Lname[i] << " customer # " << (num + 1) << " entering waiting room." << endl; // Wait for the barber chair to become free.

Sleep(100);

if (baberSleep == 1)

{

baberSleep = 0;

cout << Fname[i] << ' ' << Lname[i] << " customer # " << (num + 1) << " waking the barber." << endl;

if (!ReleaseSemaphore(barberPillow, 1, NULL))

{

cout << "ReleaseSemaphore error: " << GetLastError() << endl;

}

}

dwWaitResult = WaitForSingleObject(seatBelt, INFINITE);

if (WAIT\_OBJECT\_0 == dwWaitResult)

{

cout << "Cutting done for customer "<< Fname[i] << ' ' << Lname[i] << " customer # " << (num + 1) << " and leaving shop" << endl;

if (!ReleaseSemaphore(barberChair, 1, NULL))

{

cout<<"ReleaseSemaphore error: " << GetLastError() << endl;

}

}

}

else if (WAIT\_TIMEOUT == dwWaitResult)

{

Done--;

cout << Fname[i] << ' ' << Lname[i] << " customer # " << (num + 1) << " leaving barber shop as no waiting chair available." << endl;

}

return 0;

}

void\* barber(int numchair)

{

// Sleep until someone arrives and wake up barber..

cout<<"The barber is sleeping"<< endl;

DWORD dwWaitResult = WaitForSingleObject(barberPillow, INFINITE);

if (WAIT\_OBJECT\_0 == dwWaitResult)

{

while (true) {

dwWaitResult = WaitForSingleObject(barberChair, INFINITE);

if (WAIT\_OBJECT\_0 == dwWaitResult)

{

if (numchair>0)

{

cout<<"The barber is cutting hair"<< endl;

randwait(3);

cout<<"The barber has finished cutting hair"<< endl; // Release the customer when done

if (!ReleaseSemaphore(seatBelt, 1, NULL))

{

break;

}

}

else{ break; }

numchair--;

}

}

}

return 0;

}

void randwait(int secs)

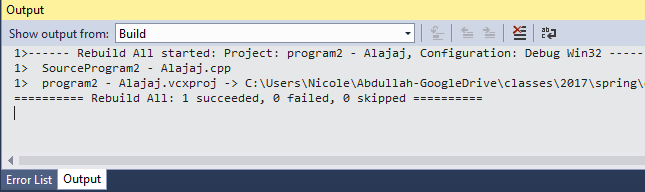
{

// Generate an arbit number

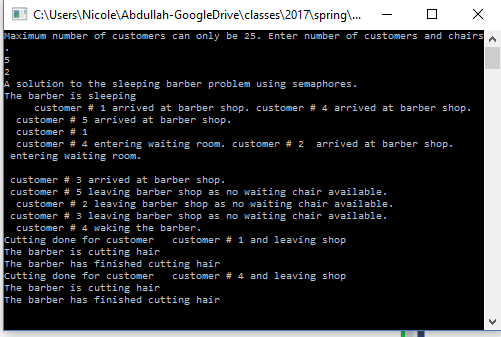
Sleep(1000 \* secs);

}

**Compiled and executed:**

* **Open code file (program2.sln) from program2 – Alajaj file. “this will open Visual studio 2013”**
* **To compile the code :**
  + **From top Manu bar : BUILD >> rebuild solution**
* **Now you will see this which mean to compiled correctly**
* ****
* **To execute the code : DEBUG >> start debugging or press F5**

**Result:**

****