|  |  |
| --- | --- |
| **Author Identification Block** | |
| **Author:** | Chris Graff |
| **Student ID:** | \*20274911 |
| **E-Mail:** | [cgraff@uco.edu](mailto:cgraff@uco.edu) |
| **Course:** | CMSC 2613 – Programming 2 |
| **CRN:** | 21641, Spring 2012 |
| **Project:** | p04 |
| **Due:** | February 10, 2012 |
| **Account:** | tt025 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Scoring Block** | | | |
| **Component** | **Available** | **Earned** | **Explanation** |
| Compilation |  |  |  |
| Submission Instructions | 2 |  |  |
| Author Identification | 1 |  |  |
| Modularity | 3 |  |  |
| Command Line | 3 |  |  |
| Input file | 3 |  |  |
| Output file | 3 |  |  |
| Execution | 10 |  |  |
| **Total** | **25** |  |  |

#Author: Chris Graff

#StudentID#: \*20274911

#Email: cgraff@uco.edu

#Course: CMSC2613 Programming II

#CRN: 21641, Spring 2012

#Project: p04

#Due: February 10th, 2012

#Account: tt025

#--------------------------------------------------------------------------

#File p02make contains instructions for the UNIX utility make. Instructions

#in file p02make direct the creation of p01.

#--------------------------------------------------------------------------

#--------------------------------------------------------------------------

#-------------------------------------------------------------------

#object files

#-------------------------------------------------------------------

obj = p04.o Sim04.o Queue04.o

#-------------------------------------------------------------------

# Bind object files and create the executable p04

#-------------------------------------------------------------------

p04: ${obj}

g++ -o p04 ${obj} -lm

#-------------------------------------------------------------------

# Compile file p04.cpp that processes the command line and exercises

# class Sim & Queue.

#-------------------------------------------------------------------

p04.o: p04.cpp Sim04.h Queue04.h

g++ -c -g p04.cpp

#-------------------------------------------------------------------

# Compile file Queue04.cpp that contains member functions of class

# Queue

#-------------------------------------------------------------------

Queue04.o: Queue04.cpp Queue04.h

g++ -c -g Queue04.cpp

#-------------------------------------------------------------------

# Compile file Sim04.cpp that contains member functions of class

# Sim

#-------------------------------------------------------------------

Sim04.o: Sim04.cpp Sim04.h Queue04.h

g++ -c -g Sim04.cpp

#-------------------------------------------------------------------

tt025@cs:~$ cat p04.cpp

#include <cstdlib>

#include <iostream>

#include <math.h>

#include <string.h>

#include <iosfwd>

#include <iomanip>

#include <fstream>

#include <string>

#include "Queue04.h"

#include "Sim04.h"

#include <ios>

using namespace std;

//----------------------------------------------------------------

//Author: Chris Graff

//StudentID#: \*20274911

//Email: cgraff@uco.edu

//Course: CMSC2613 Programming II

//CRN: 21641, Spring 2012

//Project: p04

//Due: February 10th, 2012

//Account: tt025

//----------------------------------------------------------------

struct CommandLineException

{

CommandLineException (int max, int actual)

{

cout <<endl <<"Too many command line arguements." <<endl;

cout <<"A maximum of " <<max <<" arguements are permitted." <<endl;

cout <<actual <<" arguements were entered." <<endl;

}

};

struct FileException

{

FileException (char\* filename)

{

cout <<endl <<"File " <<filename <<" could not be opened or doesn't exist" <<endl;

}

};

int main (int argc, char\* argv[])

{

try

{

char iFileName[255], oFileName[255];

switch (argc)

{

case 1:

cout <<"Enter the input file name:";

cin >> iFileName;

cout <<"Enter the output file name:";

cin >> oFileName;

break;

case 2:

strcpy(iFileName, argv[1]);

cout <<"Enter the output file name:";

cin >> oFileName;

break;

case 3:

strcpy(iFileName, argv[1]);

strcpy(oFileName, argv[2]);

break;

default:

throw CommandLineException (2, argc-1);

break;

}

ifstream i(iFileName);

if (!i)

throw FileException(iFileName);

ofstream o(oFileName);

if (!o)

throw FileException(oFileName);

Sim S;

S.Run(i, o);

i.close();

o.close();

}

catch (...)

{

cout <<"Program terminated." <<endl;

exit(EXIT\_FAILURE);

}

}

#ifndef Sim04\_h

#define Sim04\_h

#include "Queue04.h"

#include <fstream>

#include <cstdlib>

//----------------------------------------------------------------

//Author: Chris Graff

//StudentID#: \*20274911

//Email: cgraff@uco.edu

//Course: CMSC2613 Programming II

//CRN: 21641, Spring 2012

//Project: p04

//Due: February 10th, 2012

//Account: tt025

//----------------------------------------------------------------

using namespace std;

class Sim:public Queue

{

int max;

int served;

int total;

double average;

void arrival(int t);

void depart(int t);

void printitem(ostream& o, string a, double v, string b);

void printitem(ostream& o, string a, int v, string b);

public:

Sim();

void Run(istream& i, ostream& o);

};

#endif

#include <cstdlib>

#include <iostream>

#include <iomanip>

#include <fstream>

#include "Sim04.h"

#include "Queue04.h"

//----------------------------------------------------------------

//Author: Chris Graff

//StudentID#: \*20274911

//Email: cgraff@uco.edu

//Course: CMSC2613 Programming II

//CRN: 21641, Spring 2012

//Project: p04

//Due: February 10th, 2012

//Account: tt025

//----------------------------------------------------------------

using namespace std;

Sim::Sim()

{

max = served = total = 0;

average = 0.0;

}

struct invalidcode

{

invalidcode (int bad)

{

cout << endl << "The code ";

cout << bad << " is bad." << endl;

}

};

void Sim::arrival(int t)

{

if (isfull()) return;

enq(t);

}

void Sim::depart(int t)

{

if(isempty()) return;

int d = t - deq();

if (d > max) max = d;

total += d;

++served;

}

void Sim::printitem(ostream& o, string a, double v, string b)

{

o << setw(30) << left << a;

o << fixed;

o << setw(10) << setprecision(0) << right << v;

o << setw(20) << right << b;

o << endl;

}

void Sim::printitem(ostream& o, string a, int v, string b)

{

o << setw(30) << left << a;

o << setw(10) << right << v;

o << setw(20) << right << b;

o << endl;

}

void Sim::Run(istream& i, ostream& o)

{

int time = 0;

for(;;)

{

int code;

i >> code;

if(i.eof())

break;

switch (code)

{

case 1:

break;

case 2:

arrival(time);

break;

case 3:

depart(time);

break;

case 4:

arrival(time);

depart(time);

break;

default:

throw invalidcode(code);

}

time += 15;

}

average = total/served;

printitem(o, "Customers served", served, "customers");

printitem(o, "Maximum response", max, "seconds");

printitem(o, "Simulated mean response", average, "seconds");

}

#ifndef Queue04\_h

#define Queue04\_h

#include <cstdlib>

//----------------------------------------------------------------

//Author: Chris Graff

//StudentID#: \*20274911

//Email: cgraff@uco.edu

//Course: CMSC2613 Programming II

//CRN: 21641, Spring 2012

//Project: p04

//Due: February 10th, 2012

//Account: tt025

//----------------------------------------------------------------

using namespace std;

class Queue

{

int newest;

int oldest;

int\* Q;

int size;

int count;

public:

Queue(int sz=100);

~Queue();

bool isfull();

bool isempty();

void enq(int v);

int deq(void);

int length(void);

};

#endif

#include <cstdlib>

#include <iostream>

#include <string>

#include "Queue04.h"

//----------------------------------------------------------------

//Author: Chris Graff

//StudentID#: \*20274911

//Email: cgraff@uco.edu

//Course: CMSC2613 Programming II

//CRN: 21641, Spring 2012

//Project: p04

//Due: February 10th, 2012

//Account: tt025

//----------------------------------------------------------------

using namespace std;

struct qexcept

{

qexcept(string m)

{

cout << endl << "The Queue is " << m << "."<< endl;

}

};

Queue::Queue(int sz):size(sz),count(0),oldest(0),newest(-1)

{

Q = new int[size];

}

Queue::~Queue(){if(Q) delete[] Q;}

bool Queue::isfull()

{

return(count >= (size-1));

}

bool Queue::isempty()

{

return(count <= 0);

}

void Queue::enq(int v)

{

if(isfull()) throw qexcept("full");

newest = (newest++) % size;

Q[newest] = v;

++count;

}

int Queue::deq(void)

{

if(isempty()) throw qexcept("empty");

int v = Q[oldest];

oldest = (oldest ++) % size;

--count;

return v;

}

int Queue::length(void)

{

return count;

}