Tecnológico Nacional de México

Campus Querétaro.

**Implemente Estructura de Colas (FIFO) en la simulación de atención de procesos.**

*Alumno:*

*Arellano Ochoa Daniel Ignacio*

*Carrera:*

*Ingeniera en sistemas computacionales*

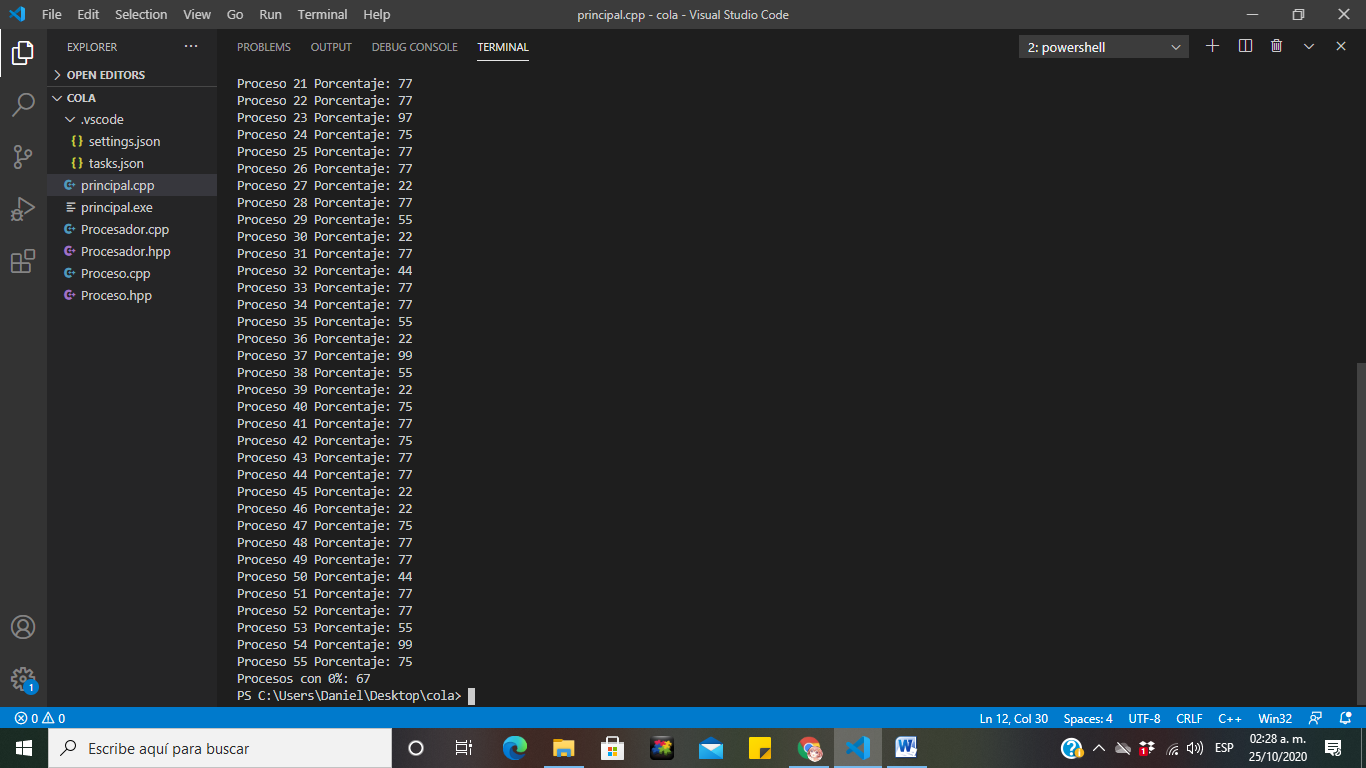
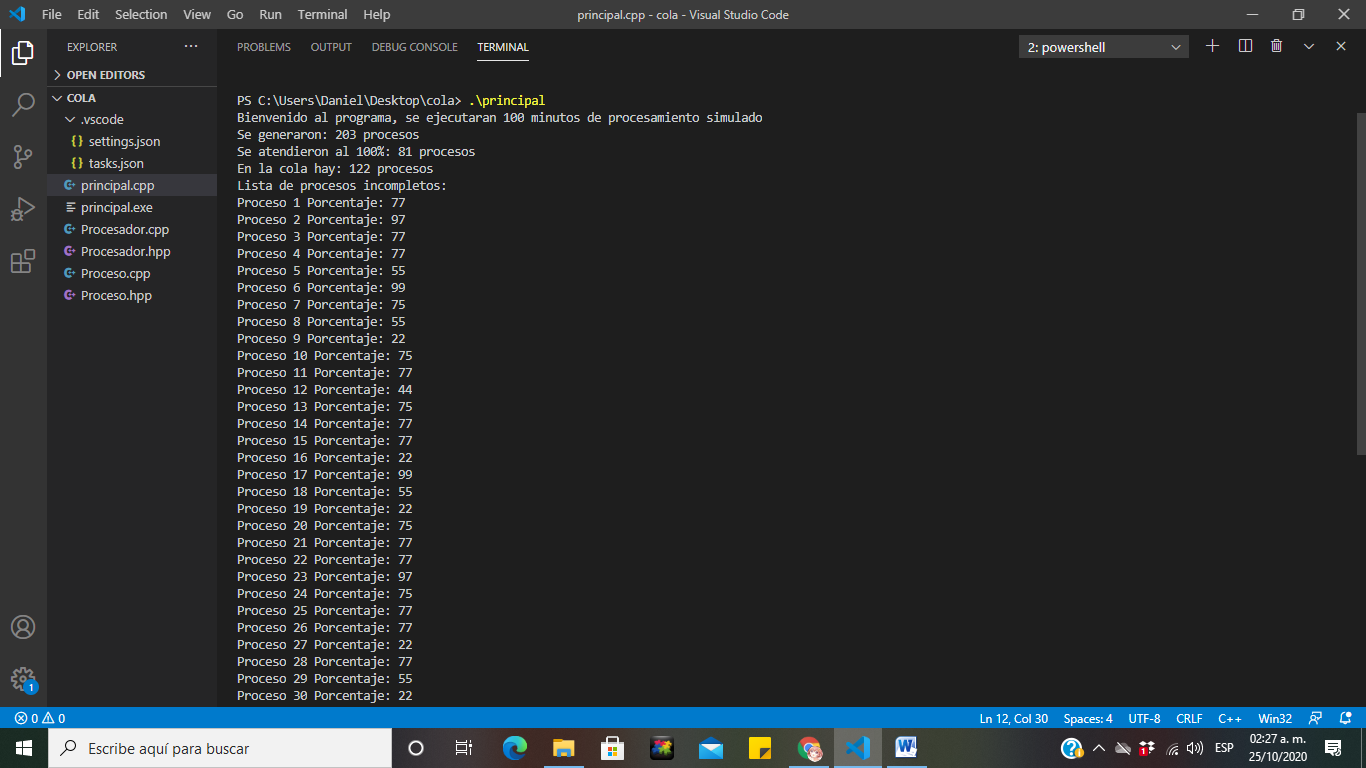
*Docente:*

*Felipe Estrada Rojas*

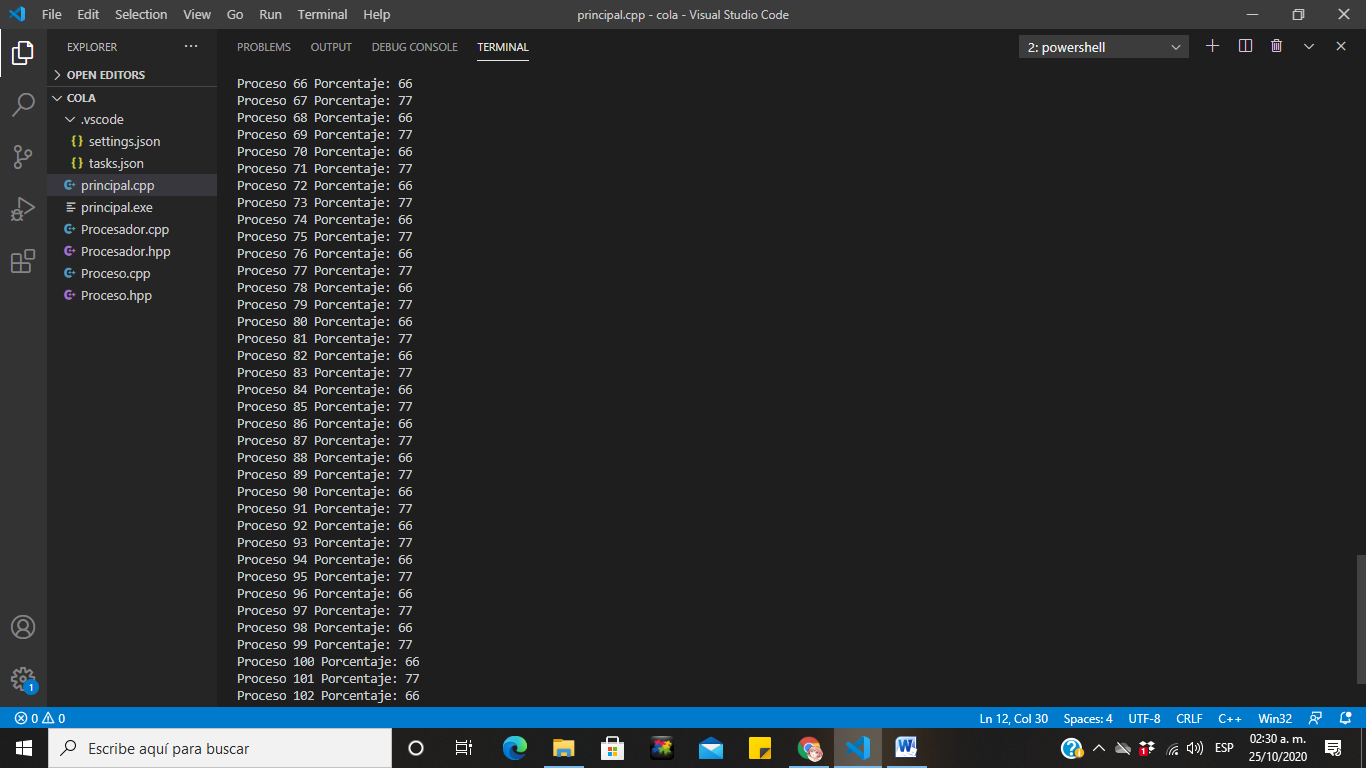
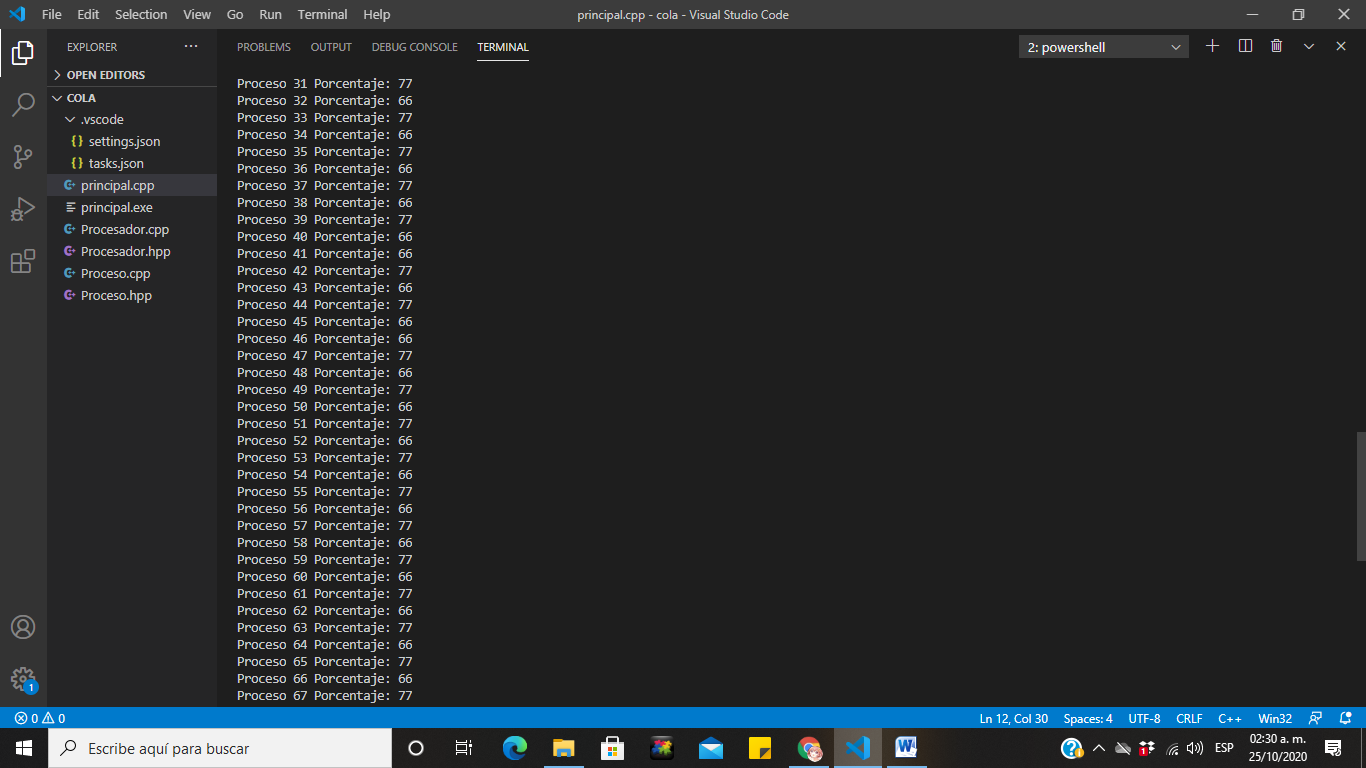
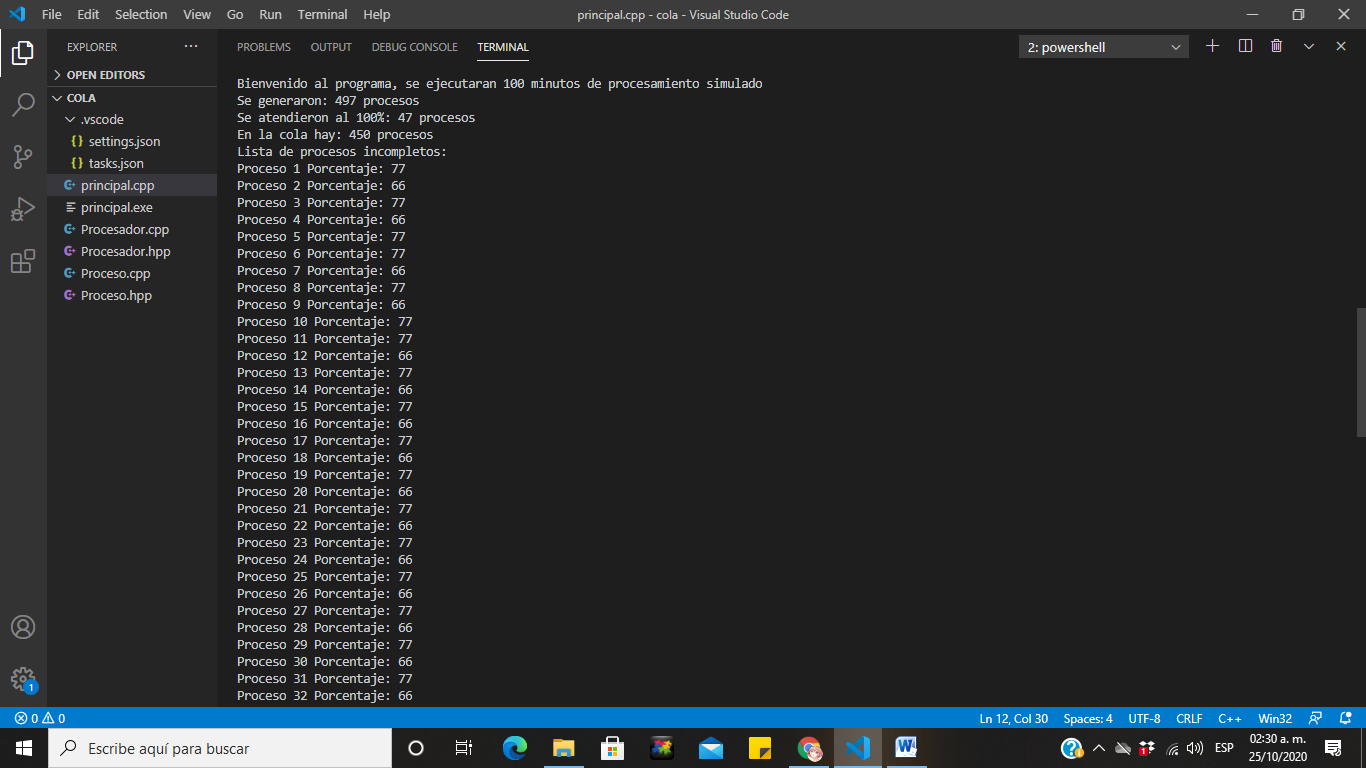
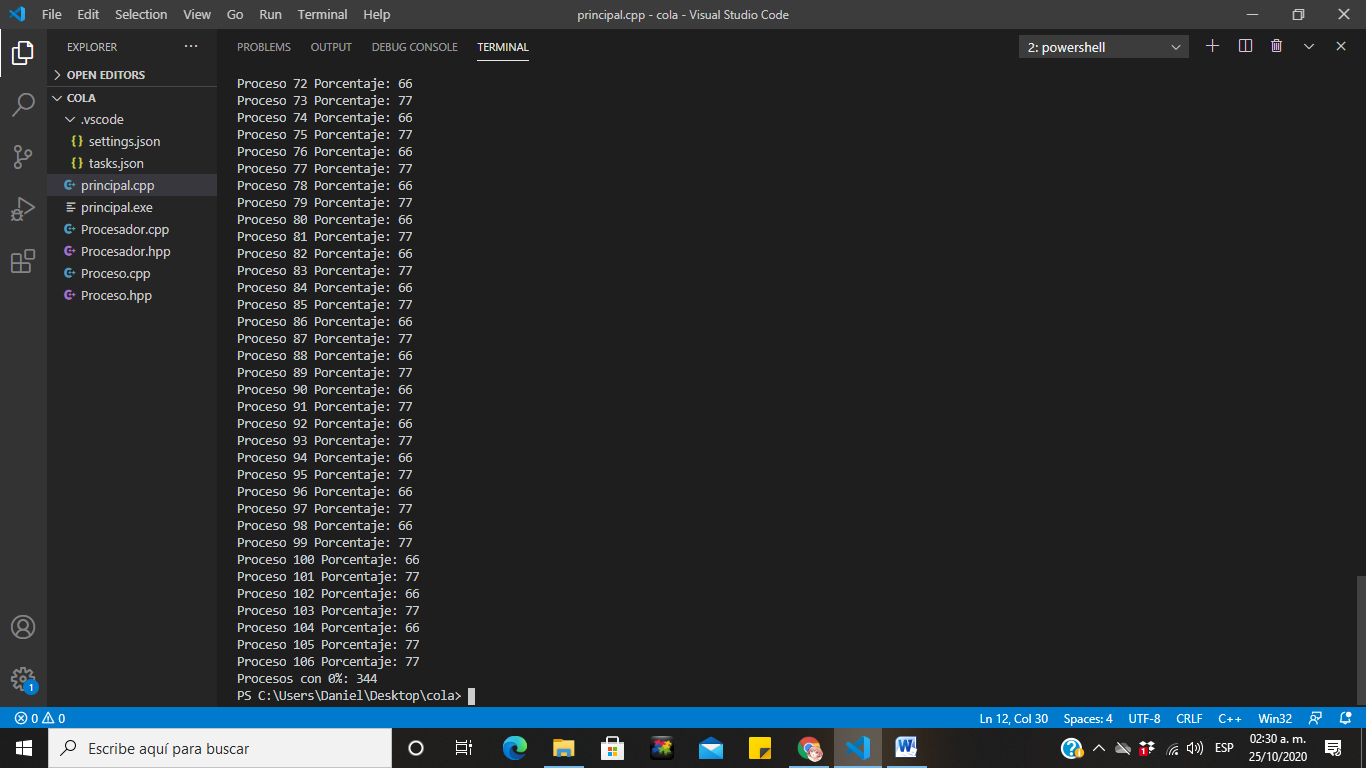
*Fecha: 24/10/2020*

**Código: El archivo cpp se anexo junto con este trabajo, de igual manera el código se encuentra hasta el final de este trabajo.**

**Ejecución:**



**En la ejecución anterior se generaron 203 procesos de los cuales 81 fueron completados al 100%, y de los 122 restantes 67 no fueron atendidos y 55 fueron atendidos por menos del 100%**

**En la ejecución anterior se generaron 497 procesos de los cuales 47 fueron completados al 100%, y de los 450 restantes, 344 no fueron atendidos y 106 fueron atendidos por menos del 100%**

**Conclusión:**

El uso de las pilas puede llegar a ser muy útil como herramienta al momento de realizar algunos programas específicos que aunque pueden ser solucionados ocupando otras herramientas como pilas o listas, es importante saber en qué momentos se ocupan las colas pues este tipo de estructuras manejan ciertos tipos de funciones que la hacen óptima para solucionar algunos problemas.

**Código:**

/\*Nombre: Arellano Ochoa Daniel Ignacio

No. de control: 19141118

Programa que simula el procesamiento durante 100 minutos en los cuales por cada minutos se genraran n procesos y se atenderan m procesos donde m y n son valores aletarios de 0 a 5

\*/

#include <string>

#include <iostream>

#include "Procesador.hpp"

#include "Proceso.hpp"

using namespace std;

//Menu de inicio del programa

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

    cout<<"Bienvenido al programa, se ejecutaran 100 minutos de procesamiento simulado"<<endl;

    Procesador\* intel=new Procesador();

    intel->procesar();

}

#ifndef Procesador\_hpp

#define Procesador\_hpp

#include "Proceso.hpp"

class Procesador

{

private:

    Proceso\* inicio;

    Proceso\* final;

    int proGenerados;

    int proAtendidos;

    int proCola;

    int proCero;

public:

    Procesador();

    void insertar(Proceso\*);

    void sacar();

    void generarProcesos(int\*);

    void atenderProcesos(int\*);

    void procesar();

    void procesosColaInc();

};

#endif /\*Procesador\_hpp\*/

#include "Procesador.hpp"

#include "Proceso.hpp"

#include <string>

#include <iostream>

#include <stdlib.h>

#include <time.h>

//clase de procesador

using namespace std;

//constructor del procesador, con datos inicializados

Procesador::Procesador(){

    this->inicio=NULL;//proceso que se encuentra al inicio de la cola

    this->final=NULL;//procesos que se encuetnra al final de la cola

    this->proGenerados=0;//procesos generados

    this->proAtendidos=0;//procesos atendido al 100%

    this->proCola=0;//procesos atendidos de 0% a 99%

    this->proCero=0;//procesos no atendido

}

//metodo para insertar un procesos dentro de la cola

void Procesador::insertar(Proceso\* p){

    if(this->proCola==0){

        this->inicio=p;

        this->final=p;

    }else{

        this->final->setAnt(p);

        this->final=p;

    }

    this->proCola++;

}

//metodo para sacar el proceso que se encuetra al inicio de la cola

void Procesador::sacar(){

    this->inicio=this->inicio->getAnt();

    this->proCola--;

}

//metodo que generar N procesos, insertandolos a la cola

void Procesador::generarProcesos(int\* N){

    Proceso\* pro;

    for (size\_t i = 0; i < \*N; i++)

    {

        pro=new Proceso();

        this->insertar(pro);

        this->proGenerados++;

    }

}

//metodo que atiende M procesos, verificando si se logro atender al 100%, si es el caso contrario se volvera a formar dentro de la cola

void Procesador::atenderProcesos(int\* M){

    srand(time(NULL));

    int pAux=0;

    Proceso\* proAux;

    for (size\_t i = 0; i < \*M; i++)

    {

        if(this->proCola>0){

            proAux=this->inicio;

            this->sacar();

            pAux=rand()%100 + 1;

            pAux= pAux + proAux->getPorcentaje();

            proAux->setPorcentaje(&pAux);

            if(proAux->getPorcentaje()<100){

                this->insertar(proAux);

            }else{

                this->proAtendidos++;

            }

        }

    }

}

//metodo que generar y ateiende N y M procesos respectivamente, durante 100 ciclos (100 minutos simulados), dando informacion al final

void Procesador::procesar(){

    srand(time(NULL));

    int M=0;//proceso a atender

    int N=0;//procesos a generar

    for (size\_t i = 0; i < 100; i++)

    {

        //cout<<"minuto"<<i<<endl;

        M= rand()%6;

        N= rand()%6;

        //cout<<"M= "<<M<<" N= "<<N<<endl;

        this->generarProcesos(&N);

        if(M>this->proCola){

            M=this->proCola;

            this->atenderProcesos(&M);

        }else{

            this->atenderProcesos(&M);

        }

    }

    cout<<"Se generaron: "<<this->proGenerados<<" procesos"<<endl;

    cout<<"Se atendieron al 100%: "<<this->proAtendidos<<" procesos"<<endl;

    cout<<"En la cola hay: "<<this->proCola<<" procesos"<<endl;

    cout<<"Lista de procesos incompletos:"<<endl;

    this->procesosColaInc();

}

//metodo que cuenta los procesos que quedaron sin atender y enumero e imprime los procesos que fueron atendidos pero no al 100%

void Procesador::procesosColaInc(){

    Proceso\* auxPro;

    int i=1;

    int aux= this->proCola;

    while (aux>0)

    {

        auxPro=this->inicio;

        this->sacar();

        if(auxPro->getPorcentaje()>0){

            cout<<"Proceso "<<i<<" Porcentaje: "<<auxPro->getPorcentaje()<<endl;

            i++;

        }else{

            this->proCero++;

        }

        this->insertar(auxPro);

        aux--;

    }

    cout<<"Procesos con 0%: "<<this->proCero<<endl;

}

#ifndef Proceso\_hpp

#define Proceso\_hpp

using namespace std;

class Proceso

{

private:

    int porcentaje;

    Proceso\* ant;

public:

    Proceso();

    int getPorcentaje();

    Proceso\* getAnt();

    void setPorcentaje(int\*);

    void setAnt(Proceso\*);

};

#endif /\*Proceso\_hpp\*/

#include "Proceso.hpp"

#include <iostream>

#include <string>

//clase de proceso

using namespace std;

//constructor con los atributos de porcentaje y ant inicializados

Proceso::Proceso(){

    this->porcentaje=0;//porcentaje atendido

    this->ant=NULL;//procesos atras de este dentro de una cola

}

//metodo que regresa el porcentaje que ha sido atendido este proceso

int Proceso::getPorcentaje(){

    return this->porcentaje;

}

//metodo que refresa el proceso que se encuentra anterior a este dentro de una cola

Proceso\* Proceso::getAnt(){

    return this->ant;

}

//metodo que permite modificar el proceso que se encuentra atras de este dentro de una cola

void Proceso::setAnt(Proceso\* p){

    this->ant=p;

}

//metodo que permite modificar el procentaje atendido de este proceso

void Proceso::setPorcentaje(int\* n){

    this->porcentaje=\*n;

}