Universidad de San Carlos de Guatemala
Facultad de Ingeniería

Curso: **Introducción a la Programación y Computación 2**

DOCUMENTACION: Segundo Proyecto

Erik Vladimir Girón Márquez
Carnet # 200313492
Sección C

Guatemala, 22 de Octubre de 2005

Introducción:

MOTHELL es un programa de administración de hotelería, , compilado para la plataforma Windows bajo MS-DOS en modo protegido 32-bit.

Este programa permite el ingreso y control de clientes, instalaciones, reservaciones, donde los comandos deberan darse desde un archivo de entrada TXT, y la salida se dará en pantalla y hacia un archivo de texto llamado SALIDA.TXT en el directorio que el usuario elija.

La interfaz del programa es muy simple, y contiene un menú principal que permite especificar lo que se desea hacer.

Este programa esta hecho en el lenguaje ANSI C/ISO C++, compilado bajo Borland Turbo C++ 3.1 activándole #define bc, y es compatible para ser compilado bajo GCC desactivándole la constante anterior.

Indice:

Documento de Análisis, Proyecto MOTHELL

Definición del Problema:

- El proyecto consiste en desarrollar un paquete de software para control de Hotelería, implementando las siguientes funciones:
    - Control de Instalaciones(Ingreso de tipos de instalación, consulta de instalaciones disponibles,
    - Control de Clientes(Ingreso de nuevos clientes, clasificación de éstos dependiendo su frecuencia de renta)
    - Control de Reservaciones(Reservar instalaciones, listar instalaciones reservadas y no reservadas)
    - Control de Consumos(Listar consumos de cada cliente dentro de las instalaciones y obtener su saldo)

Visión:

- Se espera tener un sistema de control de Hotelería completamente estable, amigable y funcional para la fecha establecida, simplificando la implementación al utilizar el paradigma de Programación Orientada a Objetos. Permitiendo automatizar los procesos anteriormente citados.

Objetivos:

- Automatizar el sistema de hospedaje de un hotel estándar, aprovechando los recursos computacionales que se ofrece en la actualidad, utilizando el lenguaje de programación C++, utilizando el paradigma de Programación orientada a objetos al 100%.
- Proporcionar una interfaz de usuario intuitiva y fácil de utilizar. Aprovechando la el muestreo de texto ASCII en pantalla (stdout).
- Generar un código estable y maduro para la fecha planificada, implantando toda la funcionalidad requerida con el menor número de errores posibles.
- Hacer que éste código generado corra en máquinas Windows, y si es posible, portarlo a GNU/Linux.
- Establecer como formato de salida un documento TXT, hacia donde se escribirá el reporte.
- Escribir un código fuente claro, con formato legible y bien documentado.

Requerimientos:

- Software: Sistema operativo basado Windows o Linux que soporte operaciones con archivos, compilador de C++ compatible con estándar.
- Hardware: Computador con microprocesador compatible con Intel i386, sistema de almacenamiento secundario con capacidad de soportar modo de escritura.
- Archivo de entrada donde se

Alcances:

- Se manejará únicamente archivos binarios dentro de las operaciones realizadas por el programa.
- La entrada de datos será únicamente por un archivo de entrada en una sintaxis específica.

Limitaciones:

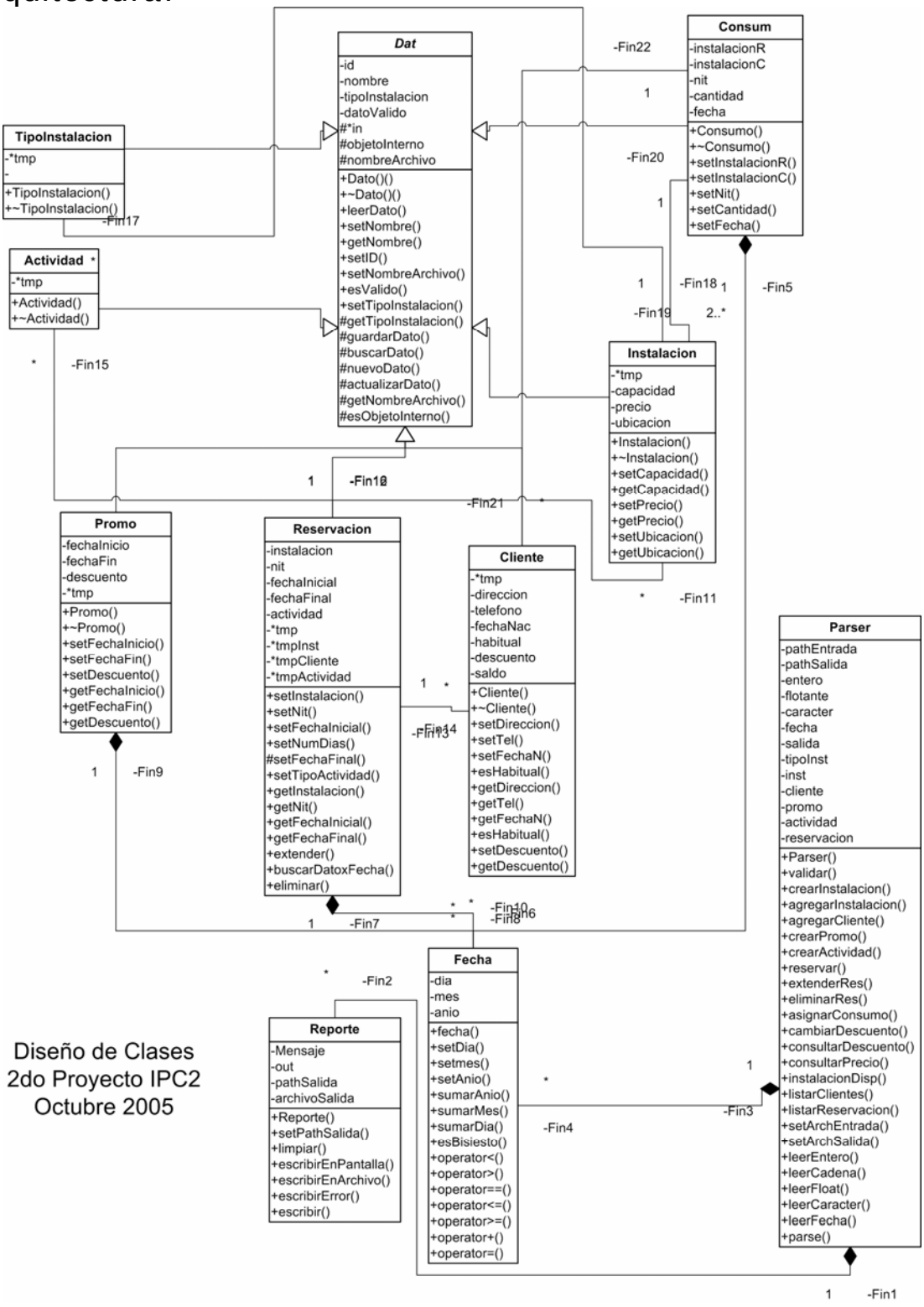
- La interfaz se usuario será únicamente en modo texto,
- El archivo de datos deberá seguir la sintaxis estricta para que se puedan hacer las operaciones correspondientes, de otro modo se ignorará la operación actual y se escribirá el error en el reporte.
- El reporte generado será únicamente en TXT, por lo que el usuario deberá verlo con su editor de texto preferido.

Tiempo Fijado:

26 – 30 de Septiembre: Análisis y Diseño
1 – 17 de Octubre: Implementación de Código, Revisiones al diseño.
18 – 19 de Octubre: Pruebas, Mejoras al Código.
20 – 21 de Octubre: Documentación y Pruebas finales.

Documento de Diseño, Proyecto MOTHELL

Arquitectura:



Definición de Clases:

A continuación se presentan la definición formal de cada clase.

```
class Menu{
 private:
  char entrada[255];
  char salida[255];
  int opcion;
public:
  Menu();
  int getOpcion();
  char * getEntrada();
  char * getSalida();
  int setPathEntrada();
  int setPathSalida();
  int mostrar();
  int ejecutar();
};

class Reporte{
 private:
  char mensaje[255];
  ofstream out;
  static char pathSalida[255];
  static char archivoSalida[24];
 public:
  Reporte();
  Reporte(char * p);
  int setPathSalida(char *);
  int limpiar();
  void escribirEnPantalla(char *);
  void escribirEnArchivo(char *);
  void escribirNumLinea();
  void escribirError(char *);
  void escribir(char*);
  void escribirReporte(char *);
};

class Fecha {
 private:
  int dia;
  int mes;
  int anio;
  char fechaStr[20];
 public:
  Fecha();
  Fecha(int,int,int);
  void setDia(int );
  void setMes(int );
  void setAnio(int );
  int getDia();
  int getMes();
  int getAnio();
  void sumarAnio();
  void sumarMes();
  void sumarDia();
  bool esBisiesto();
  bool operator>(Fecha);
  bool operator<(Fecha);
  bool operator==(Fecha);
  bool operator>=(Fecha);
  bool operator<=(Fecha);
  void operator+(int);
  void operator=(Fecha);
  char * toString();
};
```

```
class Dato{
 private:
  int id;
  char nombre[256];
  bool datoValido;
 public:
  Dato();
  Dato(bool);
  virtual ~Dato() = 0;
  int setNombre(char *);
  char * getNombre();
  int setID(int i);
  int getID();
  int setNombreArchivo(char*);
  bool esValido();
  void esValido(bool);
  void setTipoInstalacion(int);
  int getTipoInstalacion();
  char * getNombreArchivo();
  void esObjetoInterno(bool);
 protected:
  FILE * in;
  bool objetoInterno;
  fstream file;
  char nombreArchivo[17];
  virtual int guardarDato() = 0;
  virtual long buscarDato() = 0;
  int nuevoDato(void *, int);
  int actualizarDato(void*,int,long);
  int tipoInstalacionñ
};

class TipoInstalacion : public Dato{
 private:
  TipoInstalacion * tmp;
 public:
  TipoInstalacion();
  TipoInstalacion(bool);
  ~TipoInstalacion();
  int guardarDato();
  long buscarDato();
  int leerDato();
};

class Instalacion : public Dato{
 private:
  Instalacion * tmp;
  int capacidad;
  float precio;
  long ubicacion;
 public:
  Instalacion();
  Instalacion(bool);
  ~Instalacion();
  int guardarDato();
  long buscarDato();
  int leerDato(int);
  void setCapacidad(int);
  int getCapacidad();
  void setPrecio(float);
  float getPrecio();
  void setUbicacion(int);
  long getUbicacion();
  int getNivel();
  char * getStrNivel();
  int consultarPrecio();};
```

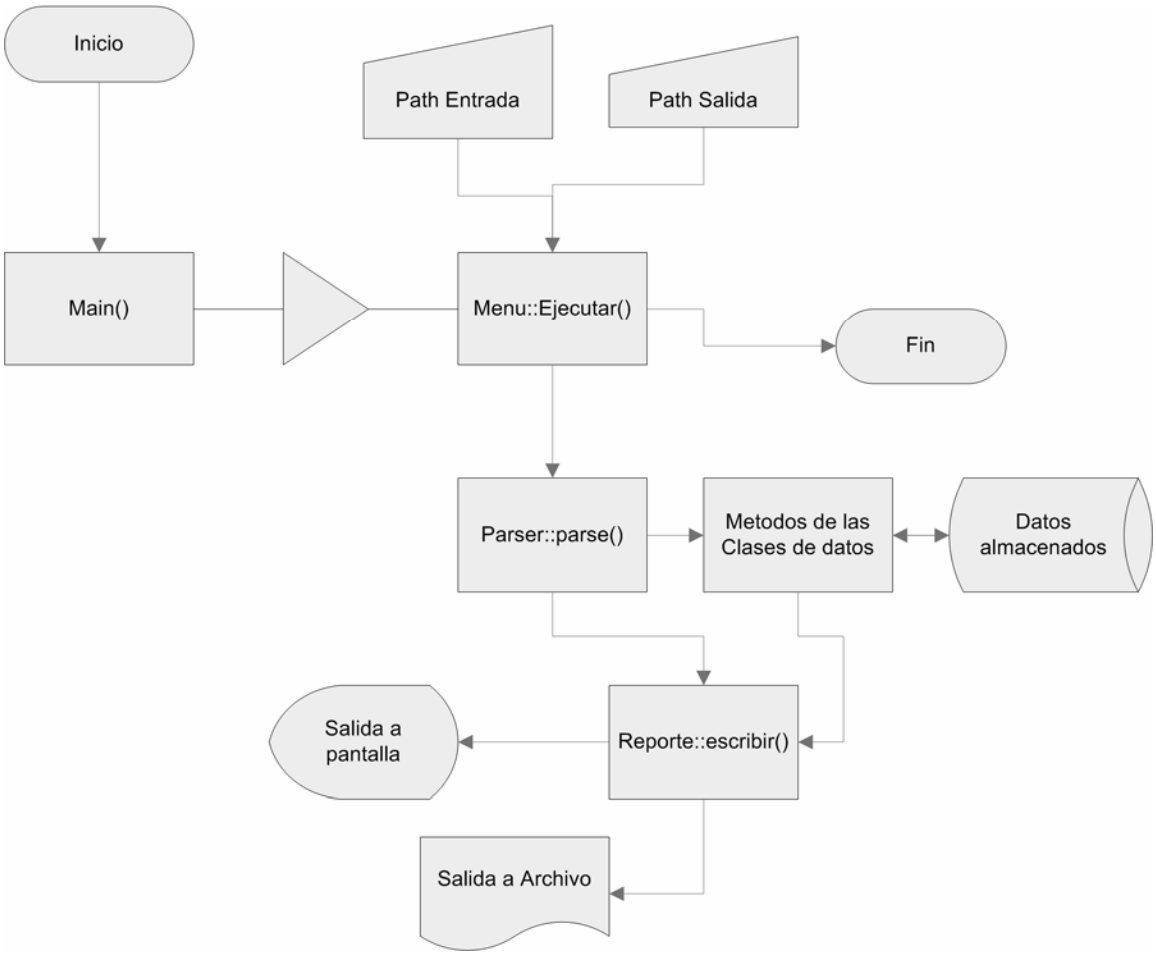
```
class Cliente : public Dato{
 private:
  char nit[16];
  Cliente * tmp;
  char direccion[255];
  char telefono[20];
  char fechaNac[255];
  bool habitual;
  static int descuento;
  float saldo;
 public:
  Cliente();
  Cliente(bool);
  ~Cliente();
  int guardarDato();
  long buscarDato();
  int leerDato(char *);
  void setNit(char *);
  char* getNit();
  void setDireccion(char *);
  void setTel(char *);
  void setFechaN(char*);
  void esHabitual(bool);
  void setHabitual(char *);
  char* getDireccion();
  char* getTel();
  char* getFechaN();
  bool esHabitual();
  int setDescuento(int);
  int getDescuento();
  int listarClientes(Reporte& );
};

class Promo : public Dato{
 private:
  Fecha fechaInicio;
  Fecha fechaFin;
  int descuento;
  Promo * tmp;
 public:
  Promo();
  Promo(bool);
  ~Promo();
  int guardarDato();
  long buscarDato();
  int leerDato();
  long buscarDatoFecha();
  void setFechaInicio(Fecha);
  void setFechaFin(Fecha);
  void setDescuento(int);
  Fecha getFechaInicio();
  Fecha getFechaFin();
  int getDescuento();
};

class Actividad : public Dato{
 private:
  Actividad * tmp;
 public:
  Actividad();
  Actividad(bool);
  ~Actividad();
  int guardarDato();
  long buscarDato();
  int leerDato();
};
```

```
class Reservacion : public Dato{
 private:
  int instalacion;
  char nit[16];
  Fecha fechaInicial;
  Fecha fechaFinal;
  int actividad;
  Reservacion * tmp;
  Instalacion * tmpInst;
  Cliente * tmpCliente;
  Actividad * tmpActividad;
  Instalacion instal;
  Cliente cliente;
  bool aplicaDescuento;
  int diasExt;
  char archTmpInst[16];
 public:
  Reservacion();
  Reservacion(bool);
  ~Reservacion();
  int guardarDato();
  long buscarDato();
  int leerDato(char *, int, Fecha);
  void setInstalacion(int);
  void setNit(char *);
  void setFechaInicial(Fecha);
  void setNumDias(int);
  void setFechaFinal(Fecha);
  void setTipoActividad(int);
  char * getNombreCliente();
  void setDiasExt(int);
  int getDiasExt();
  int getInstalacion();
  char* getNit();
  Fecha getFechaInicial();
  Fecha getFechaFinal();
  int extender();
  long buscarDatoFecha();
  int eliminar();
  void asignarDescuento();
  long buscarDatoId();
  int disponibilidad(Reporte& );
  void generarListaReservacion(Reporte& );
 private:
  int abrirArchivoInst();
  void generarReporteDisp(Reporte& );
  void borrarTmpFiles();
};
```

```
class Consumo : public Dato{
 private:
  Consumo * tmp;
  int instalacionR;
  int instalacionC;
  int nit;
  float monto;
  float precioInst;
  Cliente cliente;
  Instalacion instalC;
  Instalacion instalR;
  Reservacion reservacion;
  Fecha fecha;
 public:
  Consumo();
  Consumo(bool);
  ~Consumo();
  int guardarDato();
  long buscarDato();
  int leerDato();
  void setInstalacionR(int);
  void setInstalacionC(int);
  void setNit(char *);
  void setMonto(float);
  void setFecha(Fecha);
  int getInstalacionR();
  int getInstalacionC();
  Fecha getFecha();
  int asignarReservacion();
  char * getNombreInstalR();
  char * getNombreInstalC();
  char * getNombreCliente();
  char * getNit();
  int getIDinstalR();
  float getPrecioInstalR();
  float getMonto();
  void setDiasExt(int);
  int calcularGastoReservacion(Reporte& );
  float getDescuento(Fecha);
};
```

```
class Parser{
 private:
  char pathEntrada[255];
  char pathSalida[255];
  int entero;
  float flotante;
  char caracter;
  char cadena[255];
  Fecha fecha;
  Reporte salida;
  TipoInstalacion tipoInst;
  Instalacion inst;
  Cliente cliente;
  Promo promo;
  Actividad actividad;
  Reservacion reservacion;
  Consumo consumo;
public:
  Parser();
  int validar();
  int crearInstalacion();
  int agregarInstalacion();
  int agregarCliente();
  int crearPromo();
  int crearActividad();
  int reservar();
  int extenderReservacion();
  int eliminarReservacion();
  int asignarConsumo();
  int cambiarDescuento();
  int consultarDescuento();
  int consultarPrecio();
  int instalacionesDisp();
  int listarClientes();
  int listarReservaciones();
  int calcularGastoReservacion();
  int setArchEntrada(char *);
  int setArchSalida(char *);
  int leerEntero();
  char * leerCadena();
  float leerFloat();
  char leerCaracter();
  int leerFecha(Fecha&);
  int parse();
};
```

Lógica:
A continuación se presenta el modelo entrada y salida de datos del proyecto:



Codificación

Los archivos fuentes son:

-mothell.h:

Definicion de clases

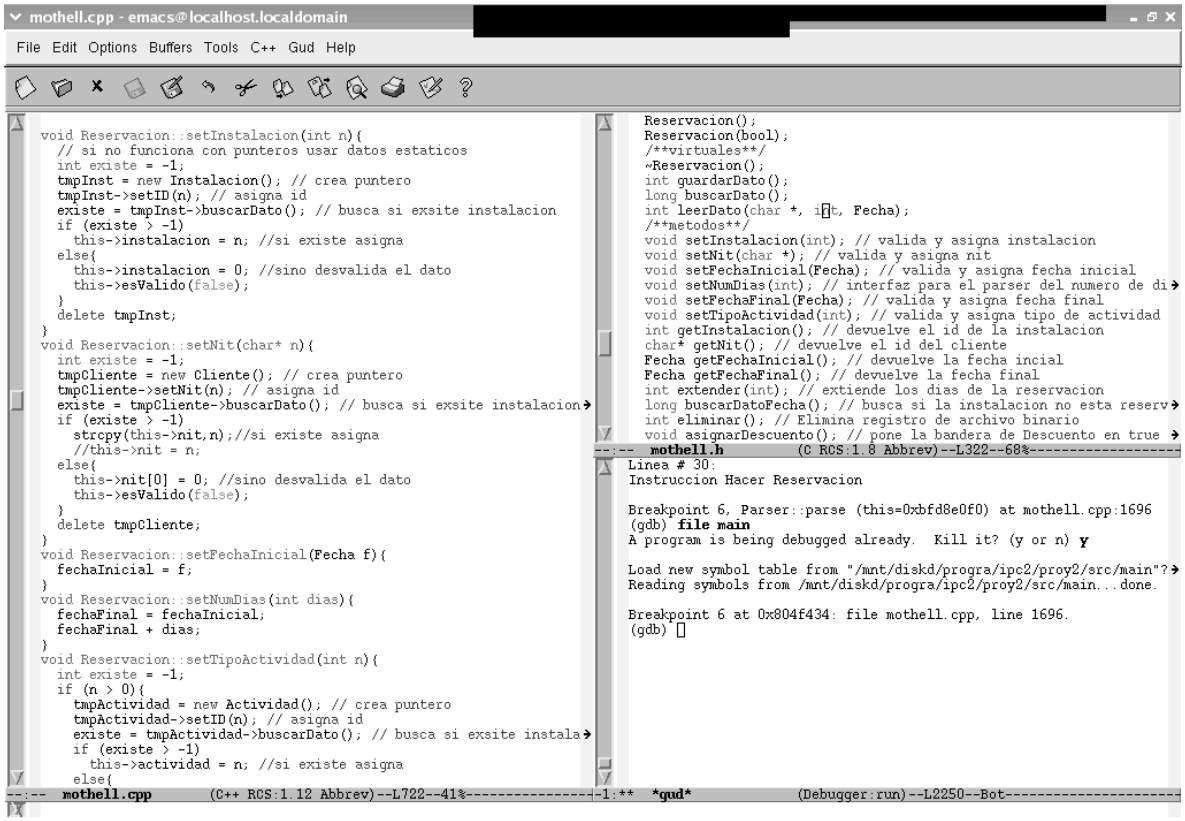
-mothell.h

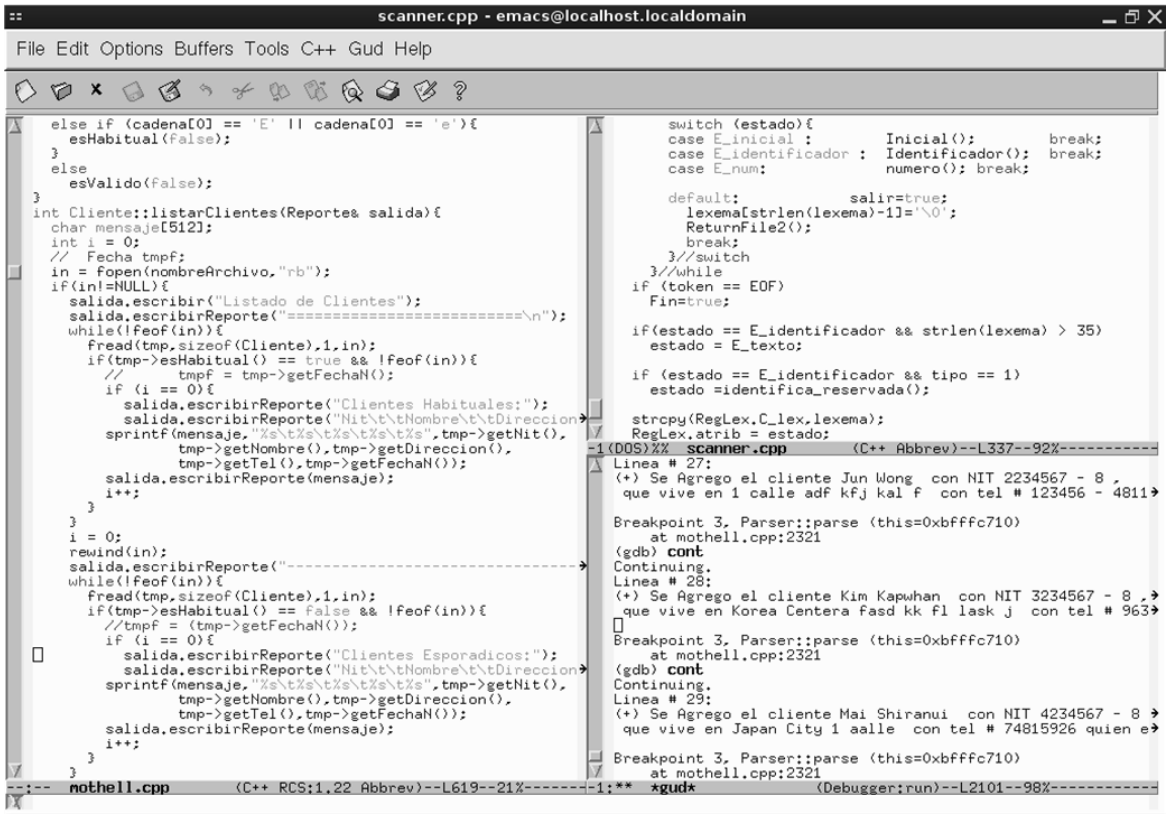
Implementacion de las clases y funcion Main

-scanner.cpp

Modulo de autómatas finitos para generar elementos lexicográficos

Pruebas:


mothell.cpp - emacs@localhost.localdomain
File Edit Options Buffers Tools C++ Gud Help
void Reservacion::setInstalacion(int n){
// si no funciona con punteros usar datos estaticos
int existe = -1;
tmpInst = new Instalacion(); // crea puntero
tmpInst->setID(n); // asigna id
existe = tmpInst->buscarDato(); // busca si exsite instalacion
if (existe > -1)
this->instalacion = n; //si existe asigna
else{
this->instalacion = 0; //sino desvalida el dato
this->esValido(false);
}
delete tmpInst;
}
void Reservacion::setNit(char* n){
int existe = -1;
tmpCliente = new Cliente(); // crea puntero
tmpCliente->setNit(n); // asigna id
existe = tmpCliente->buscarDato(); // busca si exsite instalacion
if (existe > -1)
strcpy(this->nit,n);//si existe asigna
//this->nit = n;
else{
this->nit[0] = 0; //sino desvalida el dato
this->esValido(false);
}
delete tmpCliente;
}
void Reservacion::setFechaInicial(Fecha f){
fechaInicial = f;
}
void Reservacion::setNumDias(int dias){
fechaFinal = fechaInicial;
fechaFinal + dias;
}
void Reservacion::setTipoActividad(int n){
int existe = -1;
if (n > 0){
tmpActividad = new Actividad(); // crea puntero
tmpActividad->setID(n); // asigna id
existe = tmpActividad->buscarDato(); // busca si exsite instala
if (existe > -1)
this->actividad = n; //si existe asigna
else{
--:-- mothell.cpp (C++ RCS:1.12 Abbrev)--L722--41%----------
Reservacion();
Reservacion(bool);
/**virtuales**/
~Reservacion();
int guardarDato();
long buscarDato();
int leerDato(char *, int, Fecha);
/**metodos**/
void setInstalacion(int); // valida y asigna instalacion
void setNit(char *); // valida y asigna nit
void setFechaInicial(Fecha); // valida y asigna fecha inicial
void setNumDias(int); // interfaz para el parser del numero de di
void setFechaFinal(Fecha); // valida y asigna fecha final
void setTipoActividad(int); // valida y asigna tipo de actividad
int getInstalacion(); // devuelve el id de la instalacion
char* getNit(); // devuelve el id del cliente
Fecha getFechaInicial(); // devuelve la fecha incial
Fecha getFechaFinal(); // devuelve la fecha final
int extender(int); // extiende los dias de la reservacion
long buscarDatoFecha(); // busca si la instalacion no esta reserv
int eliminar(); // Elimina registro de archivo binario
void asignarDescuento(); // pone la bandera de Descuento en true
--:-- mothell.h (C RCS:1.8 Abbrev)--L322--68%----------
Linea # 30:
Instruccion Hacer Reservacion
Breakpoint 6, Parser::parse (this=0xbfd8e0f0) at mothell.cpp:1696
(gdb) file main
A program is being debugged already. Kill it? (y or n) y
Load new symbol table from "/mnt/diskd/progra/ipc2/proy2/src/main"?
Reading symbols from /mnt/diskd/progra/ipc2/proy2/src/main...done.
Breakpoint 6 at 0x804f434: file mothell.cpp, line 1696.
(gdb)
-1:** *gud* (Debugger:run)--L2250--Bot----------


scanner.cpp - emacs@localhost.localdomain
File Edit Options Buffers Tools C++ Gud Help
else if (cadena[0] == 'E' || cadena[0] == 'e'){
esHabitual(false);
}
else
esValido(false);
}
int Cliente::listarClientes(Reporte& salida){
char mensaje[512];
int i = 0;
// Fecha tmpf;
in = fopen(nombreArchivo,"rb");
if(in!=NULL){
salida.escribir("Listado de Clientes");
salida.escribirReporte("==========\n");
while(!feof(in)){
fread(tmp,sizeof(Cliente),1,in);
if(tmp->esHabitual() == true && !feof(in)){
// tmpf = tmp->getFechaN();
if (i == 0){
salida.escribirReporte("Clientes Habituales:");
salida.escribirReporte("Nit\t\tNombre\t\tDireccion
sprintf(mensaje,"%s\t%s\t%s\t%s\t%s",tmp->getNit(),
tmp->getNombre(),tmp->getDireccion(),
tmp->getTel(),tmp->getFechaN());
salida.escribirReporte(mensaje);
i++;
}
}
i = 0;
rewind(in);
salida.escribirReporte("----------
while(!feof(in)){
fread(tmp,sizeof(Cliente),1,in);
if(tmp->esHabitual() == false && !feof(in)){
//tmpf = (tmp->getFechaN());
if (i == 0){
salida.escribirReporte("Clientes Esporadicos:");
salida.escribirReporte("Nit\t\tNombre\t\tDireccion
sprintf(mensaje,"%s\t%s\t%s\t%s\t%s",tmp->getNit(),
tmp->getNombre(),tmp->getDireccion(),
tmp->getTel(),tmp->getFechaN());
salida.escribirReporte(mensaje);
i++;
}
}
--:-- mothell.cpp (C++ RCS:1.22 Abbrev)--L619--21%----------
switch (estado){
case E_inicial : Inicial(); break;
case E_identificador : Identificador(); break;
case E_num: numero(); break;
default: salir=true;
lexema[strlen(lexema)-1]='\0';
ReturnFile2();
break;
}//switch
}//while
if (token == EOF)
Fin=true;
if(estado == E_identificador && strlen(lexema) > 35)
estado = E_texto;
if (estado == E_identificador && tipo == 1)
estado =identifica_reservada();
strcpy(RegLex.C_lex,lexema);
RegLex.atrib = estado;
-1(DOS)%% scanner.cpp (C++ Abbrev)--L337--92%----------
Linea # 27:
(+) Se Agrego el cliente Jun Wong con NIT 2234567 - 8 ,
que vive en 1 calle adf kfj kal f con tel # 123456 - 4811
Breakpoint 3, Parser::parse (this=0xbfffc710)
at mothell.cpp:2321
(gdb) cont
Continuing.
Linea # 28:
(+) Se Agrego el cliente Kim Kapwhan con NIT 3234567 - 8 ,
que vive en Korea Centera fasd kk fl lask j con tel # 963
Breakpoint 3, Parser::parse (this=0xbfffc710)
at mothell.cpp:2321
(gdb) cont
Continuing.
Linea # 29:
(+) Se Agrego el cliente Mai Shiranui con NIT 4234567 - 8
que vive en Japan City 1 aalle con tel # 74815926 quien e
Breakpoint 3, Parser::parse (this=0xbfffc710)
at mothell.cpp:2321
-1:** *gud* (Debugger:run)--L2101--98%----------