UNIVERSIDAD NACIONAL DE MISIONES

FACULTAD DE INGENIERÍA

Carrera:

INGENIERÍA EN COMPUTACIÓN

Asignatura:

FUNDAMENTOS DE COMPUTACIÓN

(Dpto. Electrónica)

Tema:

PROYECTO FINAL.

Autores:

BORGES NOGEIRA, Martin Elías.

KRZYZANOWSKI CLARK, Alejandro.

NEUMANN, Miguel Ángel.

Docentes responsables:

RENDÓN, Alicia Beatriz.

GROOSS, Juan Pablo.

Fecha de realización: 20/05/20

Fecha de presentación: --/05/20

Fecha de aprobación: /05/20

Oberá-Misiones

2020

Índice:

.

INTRODUCCIÓN

El presente informe plasma en papel el proceso de desarrollo del programa presentado en el marco del Proyecto Final de cursado de la materia Fundamentos de Computación I correspondiente al segundo año de la carrera Ingeniería en Computación, sus objetivos y alcances como así también un pequeño instructivo de uso para el mismo que fue llevado a cabo por el conjunto de alumnos Borges Nogueira, Martin Elías; Krzyzanowski-Clark, Alejandro y Neumann Miguel Ángel.

DEFINICIÓN DEL PROBLEMA

El problema que se planteó solucionar, en base a la consigna propuesta por la catedra de dicha materia, fue el de poder recolectar, almacenar y manejar los datos asociados a un negocio abocado al estilismo específicamente en las áreas de maquillaje, peinado y venta de productos relacionados con el cuidado personal (como ser cremas, champús, etc.), para facilitar su administración y proporcionar un panorama claro de su situación como así también poder presentar un respaldo numérico para futuras decisiones empresariales como ser la compra de cierta maquinaria o destinar recursos a cierta área específica del negocio.
Para esto, se tomará como punto de partida la posibilidad de aplicar estructuras de datos.

En un análisis inicial, el grupo separo los datos provenientes del negocio en tres estructuras básicas: los datos de los clientes, los de los productos que vende y utiliza el local y la información de los turnos reservados por los clientes

OBJETIVOS DEL PROYECTO DE SOFTWARE

En base al panorama planteado, el conjunto de alumnos ideo un plan para utilizar estructuras de datos que puedan, mediante un programa, recolectar y organizar la información correspondiente a los turnos de cada cliente, el inventario del negocio, las cuentas de los clientes, las preferencias de los mismos y cierta parte de las finanzas que corresponden al negocio mencionado. De esta manera, ponemos en práctica los conocimientos adquiridos durante el cursado de la materia en el presente año, buscando alcanzar los siguientes objetivos:

- Proporcionar una estructura confiable que permita armar un cronograma de trabajo (calendario) con los turnos agendados y días libres.

- Tener un registro completo de los clientes e inventario.

- Establecer un escenario de consulta para los dueños y encargados acerca de la situación actual del negocio brindando información sobre los rubros más solicitados, los productos más vendidos, los trabajos más populares, etc.

- Llevar una trazabilidad numérica clara del negocio para decisiones futuras a largo y mediano plazo.

- Brindar los datos necesarios para realizar un diagnóstico del emprendimiento y sus partes.

CARACTERIZACIÓN DEL CONTEXTO DEL PROBLEMA

Es necesario desde un principio definir claramente los horizontes o alcances del proyecto ya que debido al tiempo corto de desarrollo con el que se cuenta no es posible realizar un proyecto muy extenso y por ello debido a la naturaleza del mismo, se pensó en un primer limite, que plantea abarcar completamente las funciones correspondientes al manejo básico de las estructuras que contendrán la información de los turnos, el inventario y los clientes como ser crear y eliminar elementos, mostrar el arreglo y guardar los datos.
Luego, si el tiempo disponible lo permitiera, desarrollar funciones que relacionen los datos agrupados en dichas estructuras para obtener las funciones mencionadas anteriormente, como por ejemplo, saber cuántos los trabajos agendados puedo realizar con el stock disponible actualmente.

Habiendo establecido unos límites provisorios, que quedarán perfectamente definidos conforme se desarrolle el proyecto, se decidió trabajar con una parte representativa del problema para no tener que plantear todos los posibles casos y no extender tanto el desarrollo.

Fueron parte del planteo inicial, pero debido a cuestiones de tiempo no se incluyeron en el presente desarrollo las funciones correspondientes a las relaciones más profundas y extensas de los datos entre las distintas estructuras; por ejemplo ciertas funciones que sirven como complemento, como ser alarmas de compra de inventario para cierto producto o la creación de ofertas para mayor movilidad del inventario y la productividad y funciones de manejo estadístico del negocio como los rubros de mayor demanda o rentabilidad.

DEFINICIÓN DEL ALCANCE DEL SISTEMA A DESARROLLAR

Una vez delimitado el marco del problema, se intentará de manera provisoria definir el alcance del código a escribir, las funciones que tendrá, los procesos que desarrollará y especialmente los datos que este solicitará.

Para hacerlo, se determinó que la mejor manera de proceder sería dividir la información conformando un registro de clientes, un calendario de turnos y una lista de productos de inventario. Para la lista de clientes, se debe contar con una estructura que almacene la información particular de cada cliente como ser Nombre, Sexo, Edad, Número de contacto del Cliente entre otros, y con dicha información trabajar cuestiones relacionadas a los turnos de cada uno, su saldo para con el negocio, sus pedidos y demás.
Para esto, el programa deberá permitir el registro de un cliente y su información. Además, para el manejo de los turnos el programa deberá contar con alguna estructura que tome los datos concernientes a cada cliente y realice una cruza de datos para conocer si por ejemplo un cliente presenta cierta deuda o saldo a favor al momento de solicitar un turno.

Por otro lado, el programa deberá contemplar las funciones de asignación de un turno a cierto día del año y poseer cierto límite para dicha asignación (en base a un valor límite de trabajo preestablecido por el gerente del local), debiendo contar además con campos particulares para el manejo del inventario en cada sesión de trabajo y contar con una sección que tenga en cuenta el inventario en sí; todo lo anterior sujeto a la condiciones actuales de tiempo disponible para su desarrollo.

CRONOGRAMA TENTATIVO DE TRABAJO

Los tiempos a continuación presentados fueron planteados en una etapa inicial del desarrollo teniendo en mente el límite provisorio, ya que en base al poco tiempo disponible se pensó terminar la etapa inicial del proyecto y luego de haber terminado dicha etapa profundizar más en los desarrollos del programa.

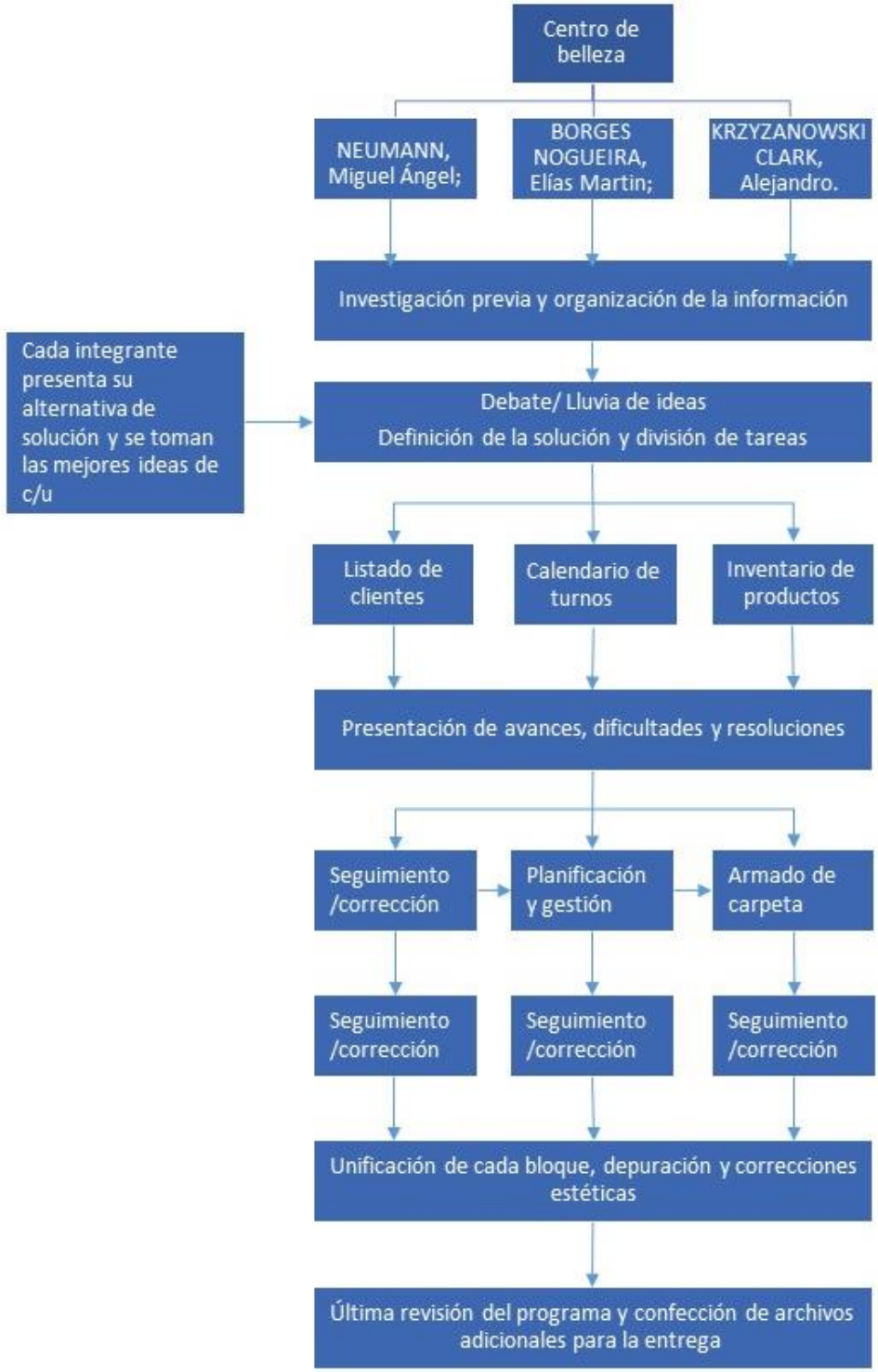
Facultad de Ingeniería OBERA

Universidad Nacional de Misiones

N°	ACTIVIDAD	TIEMPO (semanas)				
		1	2	3	4	5
1	Plantaear algún problema/ algoritmo					
2	Reunión Creativa (lluvia de ideas)					
3	Division de tareas y consenso de directivas generales					
4	Planificar el tipo de estructura de datos y codificar					
5	Informe de avaces y puesta en comun de novedades (problemas, decisiones, etc)					
6	Compilación, ejecución, depuración					
7	Unificacion de codigos, depuración final y arreglos finales					
8	Revisión general de: carpeta y del programa					

Sem./Act. N°	Descripción	Tiempo(horas)				
		1	2	3	4	5 ó +
Sem 1/Act 1	Busqueda de un problea a solucionar					
Sem 2/Act 2	Hacer una reunión virtual para exponer ideas					
Sem 3/Act 3	Analisis del problema. Datos de entrada y salida					
Sem 3/Act 4	Video conferencia para ver estado de avance y realizar modificaiones					
Sem 3/Act 5	Planificar el tipo de estructura de datos					
Sem 3/Act 6	Video conferencia para ver estado de avance y realizar modificaiones					
Sem 4/Act 7	(+) Codificación, compilación, ejecución y depuración					
Sem 4/Act 8	(+) Codificación, compilación, ejecución y depuración					
Sem 4/Act 9	Video conferencia para ver estado de avance					
Sem 4/Act 10	(+) Codificación, compilación, ejecución y depuración					
Sem 4/Act 11	(+) Codificación, compilación, ejecución y depuración					
Sem 5/Act 12	(+) Revisión general de: carpeta y del programa					
Sem 5/Act 13	Video conferencia para hablar sobre modificacaciones y del estado de la carpeta					
Sem 5/Act 14	(+) Revisión general de: carpeta y del programa					

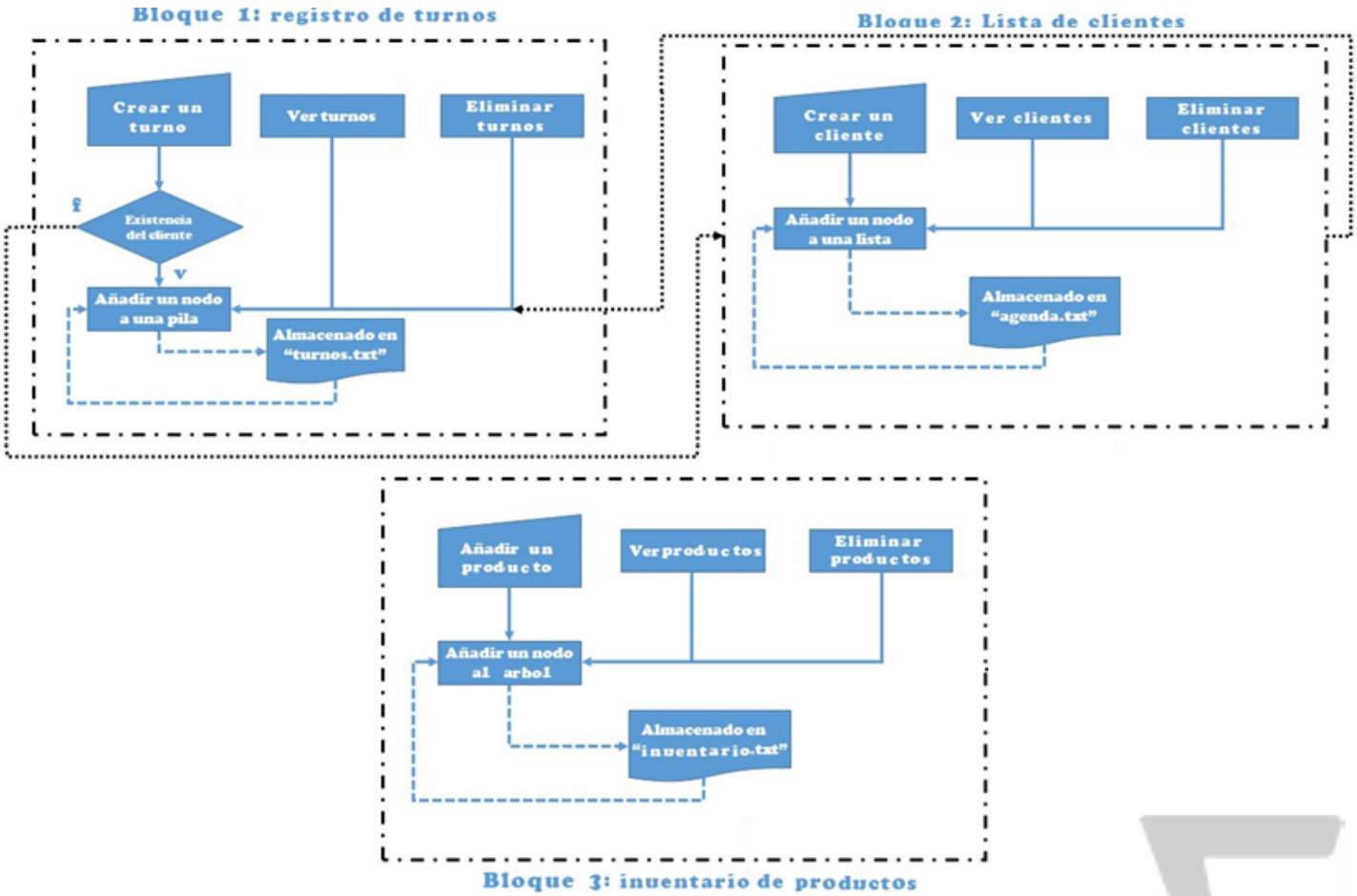
Para ayudar a visualizar la dinámica de trabajo adoptada por el grupo, se incluyó un organigrama para brindar un apoyo gráfico a la división de áreas llevada a cabo.



ALTERNATIVAS DE SOLUCIÓN CONSIDERADAS

Para comenzar la resolución del problema, el grupo determinó que la mejor manera de proceder seria recabar información y luego, en base a ésta, cada integrante plantearía una solución por si solo para que estas soluciones sean expuestas a los restantes miembros del grupo fomentando así, una variedad de alternativas, para finalmente "juntar" lo mejor de cada una y armar el planteo de la solución definitiva

Diagrama de relación entre los bloques de código:



Como se mencionó anteriormente, el problema presenta tres grandes fuentes de información (clientes, turnos e inventario) y desde un primer momento se contó con una idea bastante clara del objetivo que se buscaba en cuanto a las funciones básicas del programa. Luego de una lluvia de ideas que definió la estructura general, se decidió dividir el trabajo en las estructuras correspondientes para cada fuente de información tomando como guía los campos que se detallan a continuación:

Diseño de Datos, Diseño Arquitectónico y Diseño de Interfaces.

En cuanto al diseño de interfaces se buscó casi desde un comienzo, mantener una estética simple de entender y regular para todas las partes del programa, por lo que el ítem será tratado al final al ser común a las tres estructuras; en cuanto a los dos restantes se desarrollarán separadamente.

Calendario de Turnos:

El diseño de ésta se basa en poder almacenar turnos de clientes en un calendario. Fue pensada para constituirse en dos partes:

- struct turno, en ella se registran los campos n_turno, cod_trabajo, nombres.
- struct dia, aquí hay un campo de tipo turno el cual almacena turnos para la mañana y tarde, este se relaciona con la estructura anterior.
- Por último, se tiene un vector cuyo identificador es "c_mes[31]" en el cual se pretende cargar turnos para 31 días del mes.

```
struct turno{
    int n_turno;
    char cod_trabajo[30];
    char nombre[60];
};

struct dia{
    struct turno t_MoT[2][9];
}c_mes[31];
```

c_mes[0]

i	TURNO					
	MAÑANA			TARDE		
	t_MoT[0][i]			t_MoT[1][i]		
	n_turno	cod_trabajo[30]	nombre[60]	n_turno	cod_trabajo[30]	nombre[60]
0	-1	-	-	0	-	-
1	-1	-	-	0	-	-
2	**1**	**A**	**Juana Perez**	0	-	-
3	0	-	-	0	-	-
4	0	-	-	0	-	-
5	0	-	-	0	-	-
6	0	-	-	0	-	-
7	0	-	-	0	-	-
8	0	-	-	0	-	-
9	0	-	-	0	-	-

Por cuestiones de tiempo y comodidad se decidió usar un vector de tipo estructura, en donde se almacenarían todos los campos para poder archivar un turno y así disponer de todos los turnos de un mes.

En un primer momento, se pensó establecer una estructura que contara con 12 elementos como los mostrados anteriormente, es decir, con un elemento struct día para cada mes y dentro de estos 31 elementos struct turno para disponer de todos los turnos de un año, pero la idea fue descartada rápidamente debido a que se requeriría de mucha memoria para ser llevada a cabo.

En su ligar se pensó utilizar una misma estructura para todos los meses e ir guardado la información de cada mes en un archivo asociado. A continuación, se describe cada campo que compone a la estructura:

Campo	Tipo de dato:	Descripción:
n_turno	int	Cantidad de turnos, sirve para controlar los turnos que existen en un día.
cod_trabajo[30]	array char	Código interno de la peluquería para identificar qué tipo de trabajo se realza (siendo '**A**' maquillaje por ej.)
nombre[60]	array char	Cadena de caracteres para almacenar el nombre del cliente.
t_MoT[2][9]	matriz struct	En esta matriz se almacenan los campos anteriores.
c_mes[31]	vector struct	Este vector representa los días del mes y en él se incluyen todos los campos anteriores.

En el campo n_turno, se tiene pensado almacenar valores como:

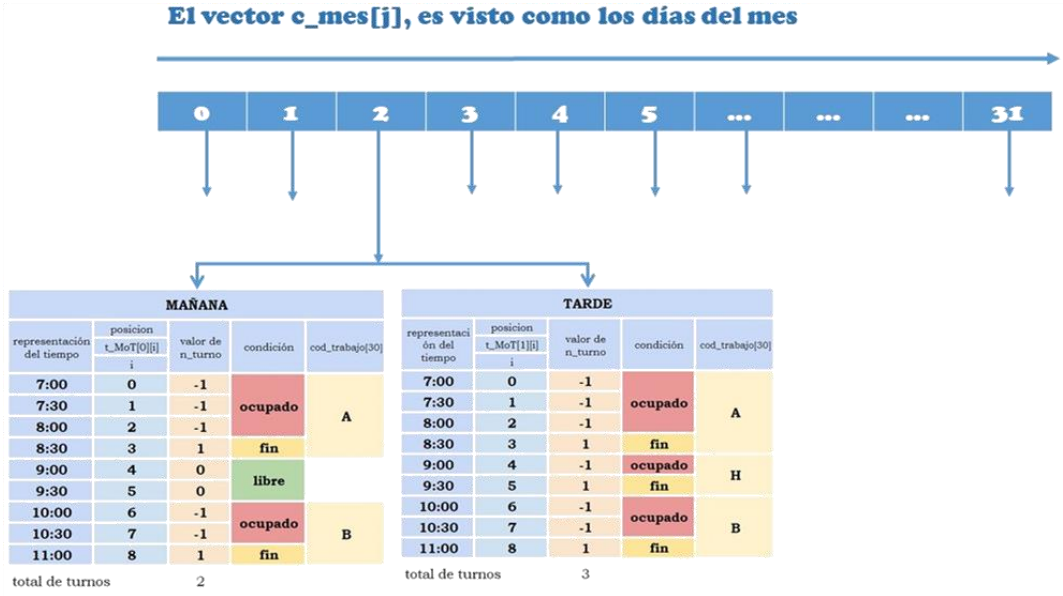
- -1: Representa que el turno se encuentra ocupado.
- 0: Representa que in turno libre.
- 1: Representa el final de un turno, además es usado para contar la cantidad de turnos en un día.

Para el caso de t_MoT[2][9], matriz de tipo struct turno la cual significa "turno mañana o turno tarde", la fila [0] representa los turnos de la mañana correspondiente a un día particular y la fila [1] representa los turnos de la tarde, las nueve filas son usadas para representar intervalos de tiempo de 30 minutos. El campo "n_turno" y esta matriz se relacionan de la siguiente manera:

Como se aprecia en la imagen de la derecha, para cada posición i de la matriz se corresponde un valor numérico de los antes mencionados.

La imagen muestra dos turnos agendados de 7 a 8:30 horas y de 10:00 a 11:00 horas.

MAÑANA				
representación del tiempo	posicion t_MoT[0][i] i	valor de n_turno	condición	cod_trabajo[30]
7:00	0	-1	ocupado	A
7:30	1	-1		
8:00	2	-1		
8:30	3	1	fin	
9:00	4	0	libre	
9:30	5	0		
10:00	6	-1	ocupado	B
10:30	7	-1		
11:00	8	1	fin	
total de turnos		2		



MAÑANA

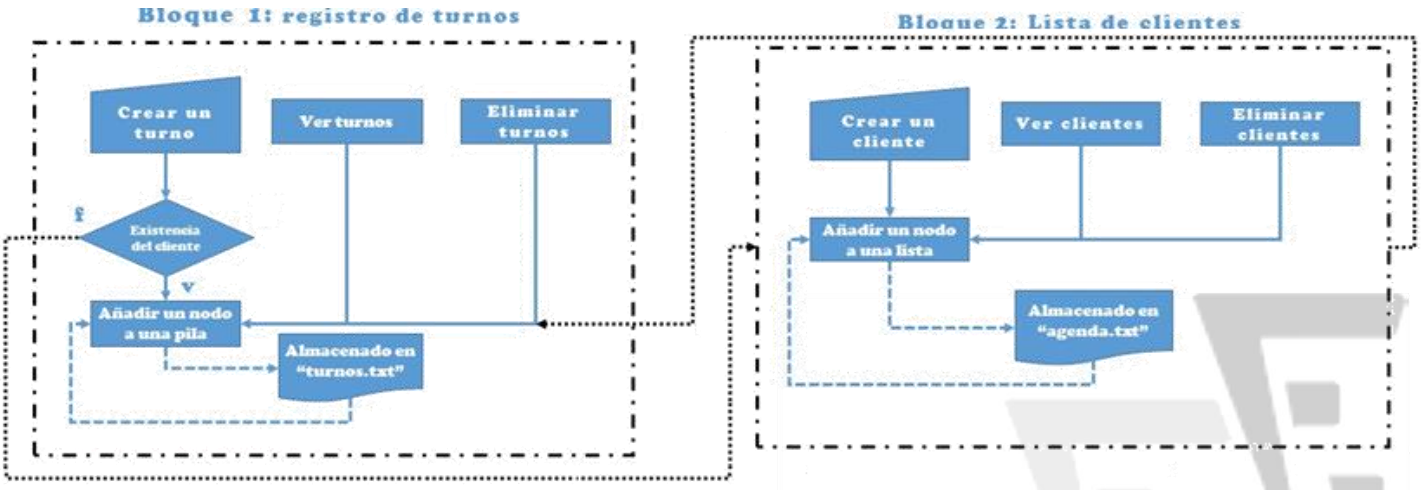
representación del tiempo	posicion t_MoT[0][i] i	valor de n_turno	condición	cod_trabajo[30]
7:00	0	-1	ocupado	A
7:30	1	-1		
8:00	2	-1		
8:30	3	1	fin	
9:00	4	0	libre	
9:30	5	0		
10:00	6	-1	ocupado	B
10:30	7	-1		
11:00	8	1	fin	
total de turnos		2		

TARDE

representación del tiempo	posicion t_MoT[1][i] i	valor de n_turno	condición	cod_trabajo[30]
7:00	0	-1	ocupado	A
7:30	1	-1		
8:00	2	-1		
8:30	3	1	fin	
9:00	4	-1	ocupado	H
9:30	5	1	fin	
10:00	6	-1	ocupado	B
10:30	7	-1		
11:00	8	1	fin	
total de turnos		3		

En la imagen anterior, se puede apreciar una visión más general de cómo se ubicaría cada dato dentro del vector c_mes y la interacción de cada parte. Para cada día del mes, existen dos subestructuras (turno mañana y turno tarde), en ellas se almacenarán los turnos utilizando las marcas {-1,0,1} según corresponda.

A continuación, se muestra de forma general el cómo trabajan e interactúan los módulos del código para generar turnos.



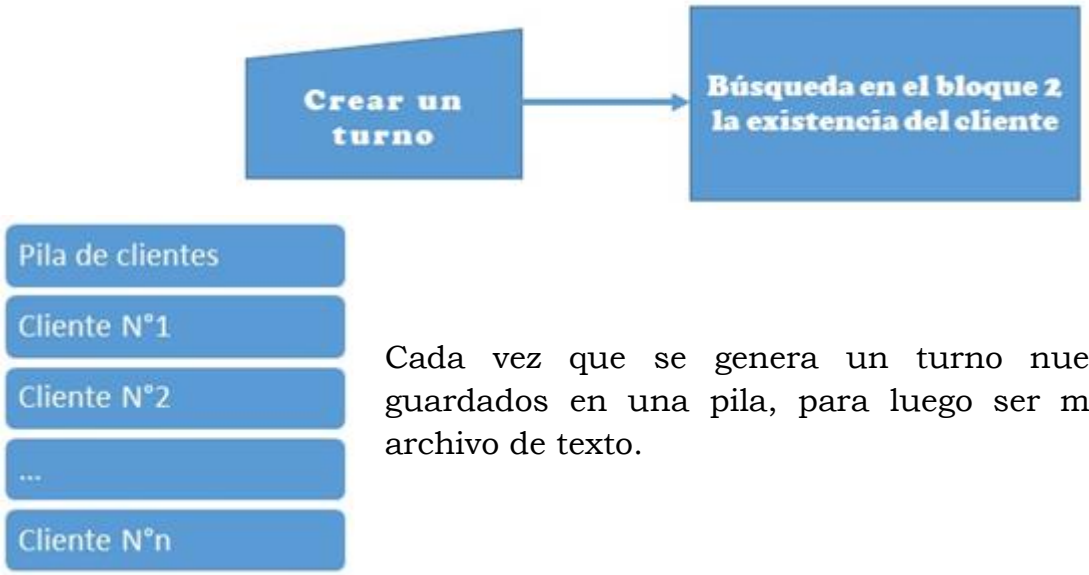
Como se puede observar en el diagrama anterior el bloque 1 es el que se encarga de la gestión de turnos. Los pasos que sigue son:

1. Ingreso fecha, turno, horario y datos de cliente,
2. Búsqueda del cliente.
3. Almacenamiento del cliente.

Luego, se necesita cargar el día, el turno (sea mañana o tarde), y por último el horario inicial y final.

Para la carga del día se valida el engarzo para verificar una fecha posible, ya que el vector "c_dia[31]" contiene 31 elementos. Por último, se solicita el ingreso de los horarios inicial y final, donde su validación admite los valore enteros del intervalo [0,8], ya que la matriz que lo contiene posee la siguiente forma MoT[2][9].

Antes de crear el turno, se verifica que en el bloque 2 exista dicho cliente, en caso contrario será creado y luego es añadido a la pila; si el cliente ya se encuentra registrado, nada más es añadido a la pila.



Cada vez que se genera un turno nuevo, estos son guardados en una pila, para luego ser mandados a un archivo de texto.

El almacenamiento de toda la información de los clientes, tunos y productos ocurre al cerrar el programa, en ese instante se llama a una función que trabaja con archivos de texto y guarda dicha información. Fueron seleccionados los archivos de texto sobre los binarios por dos motivos particulares: la posibilidad de acceder a la información cuando sea necesario sin tener que ejecutar el programa y la capacidad de estos de ser modificados sin comprometer futuras ejecuciones del mismo.

Además, cada vez que se inicia el programa, se lee ese archivo asociado a cada estructura y de esa manera se mantiene el registro de los datos entre distintas ejecuciones.

En consecuencia, con lo ya dicho conjuntamente a cada estructura se disponen de las funciones básicas de carga/eliminación, modificación y muestra de datos.

Pero también hubo otras consideraciones:

En la etapa inicial, se consideró guardar los turnos en una matriz donde las filas representaban meses y las columnas días; en cada casillero de esta matriz habría una lista de clientes, y por cada cliente se dispondría de una lista secundaria para cargar una "x" cantidad de turnos tanto a la mañana como a la tarde, como se muestra en las siguientes imágenes:

```
struct servicio{

    char nombre_servicio[60];
    float costo;

};
struct cliente{

    char nombre_apellido[60];
    char sexo;
    int edad;
    struct servicio servicios;

} matriz_turnos[12][31];
```

La matriz que se ve presenta en la imagen anterior, se compondría de ceros cuando haya turnos disponibles, y con cualquier otro número mayor a cero si hubiera más de un turno. Para cada día de un mes especifico, se puede obtener una lista de turnos con "x" cantidad de clientes.

Registro de Clientes:

Para este bloque del programa, se pensó rápidamente en utilizar las estructuras denominadas listas, debido a que no se conoce el número de clientes y este además no es fijo. Al momento de elegir el tipo de estructura a utilizar se plantearon distintas alternativas para la misma, pero al ver los beneficios que ésta aportaba al sistema, optamos por ella considerando:

La posibilidad de insertar/eliminar elementos.
La asignación de memoria dinámica durante la ejecución del programa.

Habiendo determinado que tipo de estructura seria la utilizada, restaba saber el tipo de lista a utilizar, ya que se dispone de:

- Listas simplemente enlazadas.
- Listas doblemente enlazadas.
- Listas circulares simplemente enlazadas.
- Listas circulares doblemente enlazadas.

Cabe aclarar que cualquiera de los cuatro tipos mencionados hace referencia a una lista enlazada, pero el grupo consideró que en base a las ideas que se tenían en mente desarrollar lo más conveniente sería utilizar la lista doblemente enlazada, ya que posee características favorables para que el programa tenga un mejor rendimiento.

Si bien las listas doblemente enlazadas requieren mayor cantidad de memoria por nodo y sus operaciones básicas resultan más "costosas" en términos informáticos, ofrecen una mayor facilidad para manipulación ya que permiten el acceso secuencial a lista en ambas direcciones facilitando tanto el recorrido como la eliminación e inserción de nuevos elementos.

La lista doblemente enlazada utiliza dos apuntadores globales, los cuales contienen las direcciones de memoria de los elementos en los extremos del arreglo.

Cada nodo debe tener dos apuntadores, uno que indique al elemento siguiente dentro de la lista y otro que indique el anterior (los mismos están graficados con flechas en el diagrama anterior), donde cada elemento representará un cliente; dicho elemento estará compuesto por un tipo dato de tipo struct ya que permite almacenar distintos tipos a consideración del programador de datos en una única variable.

El nombre de la estructura creada para los elementos de la lista fue struct cliente.
Esta estructura surge de la necesidad de almacenar la información de aquellos clientes que utilizan alguno de los servicios disponibles en el comercio, para tener un registro de los mismo, calcular información adicional o enviar recordatorios de turnos o deudas partiendo de los siguientes campos:

```
struct cliente
{
    char c_nombre[60];
    char c_sexo;
    int c_edad;
    char c_num_tel[15];
    struct cliente *anterior;
    struct cliente *siguiente;

};
```

Campo	Tipo de dato	Descripción
c_nombre[30]	array (char)	Nombre del cliente.
c_sexo	char	Tipo de sexo (Masculino o Femenino).
c_edad	int	Edad del cliente.
c_num_tel	array(char)	Número de teléfono del cliente.
anterior	Puntero	Puntero que almacenará la dirección de memoria del cliente anterior a él, hacia la izquierda de su posición.
siguiente	Puntero	Puntero que almacenará la dirección de memoria del cliente posterior a él, hacia la derecha de su posición.

Descripción detallada:

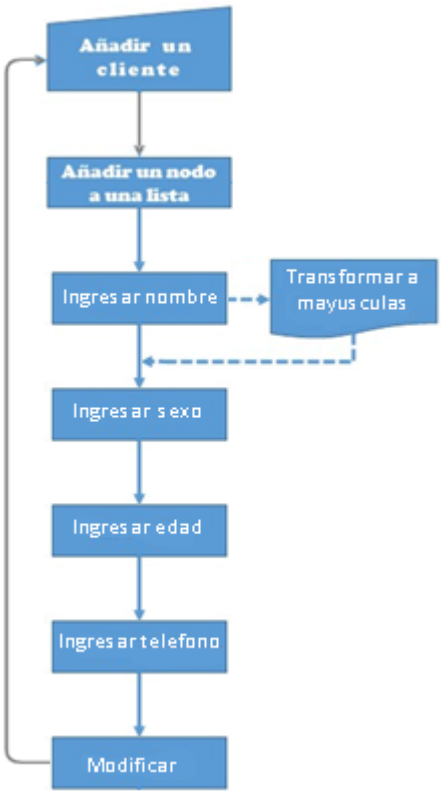
c_nombre: arreglo de caracteres en el cual el usuario deberá ingresar el nombre del cliente sin importar el tipo (mayúscula o minúscula), dado que se utilizarán las funciones del lenguaje incluidas en las distintitas librerías para poder cambiar todas los caracteres que componen el string en mayúsculas para en un futuro no tener ningún tipo de inconveniente al tratar de encontrar dicho cliente comparando los strings.

c_sexo: Consta de un solo carácter en el cual el usuario al registrar 'F' o 'M' si dicho cliente es de sexo femenino o masculino respectivamente.

c_num_tel: cadena de caracteres que contendrá el número de teléfono del cliente, como la cantidad de dígitos de un número difiere con otro de distinta característica o de un teléfono fijo y no se trabajará con la información del número como tal, se optó por almacenar esta información con este tipo de dato.

En este punto del desarrollo se determinó “liberar” la función principal del programa (main) para las llamadas a los menúes particulares de cada estructura (que contendrán las opciones de manejo básico) y los menúes de las funciones que relacionen los datos de las tres estructuras obteniendo información implícita partiendo de dichos datos.

Considerando lo anterior, el menú particular de esta estructura dispondría de al menos 6 opciones las cuales trabajarían con los punteros globales “primero” y “ultimo” para poder así realizar todas las operaciones necesarias en el menú en cuestión.



Una vez cargados, se muestran todos los campos y luego el programa ofrece la posibilidad de poder cambiar o modificar algún campo que se haya podido cargar erróneamente.

Dicho formato de impresión de datos del cliente se lo repite en todo el programa, para poder así tener un formato uniforme en todo el menú con el objetivo de una facilidad de comprensión y evitar todo tipo de confusión.

Todas las funciones ubican un elemento mediante el nombre que se ingresó utilizando al mismo como elemento identificador, pero se podrían agregar para otras aplicaciones funciones que realicen búsquedas por otros campos como ser sexo o deudas pendientes para con el negocio. Además, todas las funciones validan que exista una lista ante de ejecutar las operaciones correspondientes; se destaca que debido a la decisión de guardar la información en un archivo de texto, se consideró el caso de querer eliminar tanto un solo elemento como la de eliminar todos los elementos, y debido a las consecuencias de su accionar; imposibilidad de volver atrás, se realiza una validación anterior si es que el usuario realmente quiso acceder a dicha función.

Llegado el final de la ejecución de cualquier función, se estableció (siguiendo la línea de mantener los estándares estéticos en todo el programa) que la única manera de regresar a un menú o fase anterior fuera utilizando la letra ‘S’ para luego poder utilizar otras funciones.

Facultad de Ingeniería OBERA

Universidad Nacional de Misiones

Inventario de Productos:

La siguiente estructura fue planteada para almacenar la información pertinente al inventario de productos que se comercializan o son utilizados como material en los procesos o trabajos que se realizan el negocio sobre el cual se desarrolla el presente proyecto (séanse procesos relacionados con el peinado o corte de cabello, el maquillaje, la manicura o la venta de dichos productos).

El primer planteo de la estructura que almacenaría los datos del inventario arrojo como resultado de dicho planteo la utilización de una lista, pero se decidió finalmente utilizar un árbol binario para realizar dicha tarea ya que la catedra exigía en el desarrollo del programa al menos una estructura no lineal y el grupo consideró que el inventario supondría la manera más integra de hacerlo debido a la eficiencia de dicha estructura a la hora de localizar un elemento.
En un principio la estructura que conformaría cada elemento del árbol en cuestión tendrá los siguientes datos:

```
struct inventario {
        char nombre_prod [50];
        int cantidad;
        int cod_producto;
        struct inventario *padre;
        struct inventario *h_izq;
        struct inventario *h_der;
};
```

Campo:	Tipo de dato:	Descripción:
nombre_prod	array char	Nombre del producto (crema para peinar, jabón, etc.).
cantidad	int	Cantidad disponible.
cod_producto	Int	Código interno para dicho producto.
padre	puntero	Puntero que almacenará la dirección de memoria del elemento del nivel anterior a él.
h_izq	puntero	Puntero que almacenará la dirección de memoria del elemento del nivel posterior a él, hacia la izquierda de su posición.
h_der	puntero	Puntero que almacenará la dirección de memoria del elemento del nivel posterior a él, hacia la derecha de su posición.

El campo “cantidad” deberá en su ingreso ser validado como un número positivo o cero; además, otro aspecto a considerar para las funciones que se tienen planteadas es que el tamaño de las unidades de cada elemento varia:

Por ej.: un champú posee embaces de 1 L y una crema promedio de 500 ml, por lo que se propuso realizar un análisis de los productos y sus presentaciones para conocer el panorama al que se debe adaptar el programa resultando en el siguiente listado de productos, que contiene algunos de los productos más utilizados en el local, siendo una porción representativa del inventario del comercio:

Nombre del Producto:	Código:	Tamaño del envase:	Cantidad usual en stock:
Tintura Moreno	cp_001	60 g	15 u
Tintura Castaño Claro	cp_002	60 g	15 u
Tintura Castaño Oscuro	cp_003	60 g	15 u
Tintura Rubia Claro	cp_004	60 g	15 u
Tintura Rubia Oscuro	cp_005	60 g	15 u
Tintura Pelirroja	cp_006	60 g	15 u
Esmalte de Uñas Transparente	cp_007	60 g	10 u
Esmalte de Uñas Rojo	cp_008	60 g	5 u
Esmalte de Uñas Negro	cp_009	60 g	5 u
Esmalte de Uñas Blanco	cp_010	60 g	5 u
Agua Oxigenada	cp_011	1L	15 u
Polvo Decolorante	cp_012	360 g	30 u
Champú	cp_013	5 L≡ 5000 ml	2 u
Acondicionador	cp_014	5 L≡ 5000 ml	2 u
Cera	cp_015	30 g	15 u
Fijador	cp_016	180 ml/120 g	5 u
Uñas Postizas	cp_017	1 u	5 u
Crema para Peinar	cp_018	30 g	10 u
Rubor	cp_019	10 g	5 u
Lápiz Labial Rojo	cp_020	1 u	5 u
Lápiz Labial Negro	cp_021	1 u	5 u
Lápiz Labial Marrón	cp_022	1 u	5 u
Desmaquillante Facial	cp_023	120 ml	5 u
Crema para Manos/Piel	cp_024	30 g	5 u
Toallas de Papel/Algodón	cp_025	10 u	10 u
Alcohol en Gel	cp_026	60 g	10 u
Quitaesmalte	cp_027	300 ml	5 u

Entiéndase: u ≡ unidad/es, g ≡ gramos, ml ≡ mililitros y L ≡ litros.

El grupo consideró que de funcionar el programa para una porción representativa de los datos del problema debería hacerlo con el conjunto completo de datos. Luego se procede a determinar los trabajos que realiza el local e introducirlos al programa:

Nombre del Trabajo:	Código:	Productos Utilizados en el Proceso:
Peinados	A	013, 014, 016, 018, 026.
Teñido (cambio de color desde las raíces)	B	001-006, 013, 018, 026.
Mechas (cambio de color en las puntas)	C	001-006, 011-013, 018, 026.
Decoloración	D	011-013, 018, 026.
Maquillaje	E	019-026.
Manicura	F	007-011, 017, 024-027.
Depilaciones	G	015, 023-026.
Cortes de Cabello	H	016, 026.

Nótese que deben diferenciarse las tinturas a realizar utilizando el color negro (moreno), de las que requieran el tono pelirrojo, por lo que en cada categoría en la que existan diferentes tipos de productos a utilizar se crearan subcategorías (B1: tintura morena, B7: tintura pelirroja); además se considerará que todos los trabajos realizados siguen un único procedimiento, es decir, que el algoritmo no toma en cuenta si un cliente acude al local habiendo lavado su cabello o desmaquillando su rostro y se computaran de igual manera los insumos correspondientes.

Esta situación podría salvarse estableciendo un conjunto de salvedades o subcategorías de cada subcategoría como podría ser B7.1, pero dado que el tiempo requerido para establecer con eficiencia cada caso o situación es mayor al disponible, se simplificara la resolución estableciendo la siguiente codificación:

Trabajo:	Código:	Trabajo:	Código:
Peinados	A	Decoloración Rubio Oscuro	D5
Tintura Moreno	B1	Decoloración Pelirroja	D6
Tintura Castaño Claro	B2	Maquillaje Labial Rojo	E1
Tintura Castaño Oscuro	B3	Maquillaje Labial Negro	E2
Tintura Rubia Claro	B4	Maquillaje Labial Marrón	E3
Tintura Rubia Oscuro	B5	Manicura Transparente	F1
Tintura Pelirroja	B6	Manicura Roja	F2
Mechas	C	Manicura Negra	F3
Decoloración Morena	D1	Manicura Blanca	F4
Decoloración Castaño Claro	D2	Depilaciones	G
Decoloración Castaño Oscuro	D3	Cortes de Cabello	H
Decoloración Rubio Claro	D4		

Una vez obtenidos los datos representativos del inventario y habiendo establecido los trabajos que realiza el local (una vez más tomando una parte representativa del conjunto de trabajos realizados en la vida real), se procedió a determinar la cantidad promedio de cada producto a ser utilizado en cada proceso:

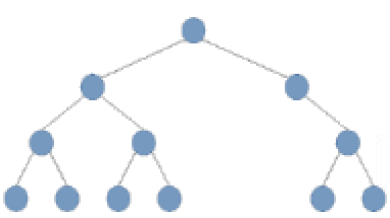
Nombre del Producto:	Tamaño del envase:	Cantidad utilizada por trabajo:
Tintura Moreno	60 g	1 u
Tintura Castaño Claro	60 g	1 u
Tintura Castaño Oscuro	60 g	1 u
Tintura Rubia Claro	60 g	1 u
Tintura Rubia Oscuro	60 g	1 u
Tintura Pelirroja	60 g	1 u
Esmalte de Uñas Transparente	15 ml	10 % ≡ 1,5 g
Esmalte de Uñas Rojo	15 ml	10 % ≡ 1,5 g
Esmalte de Uñas Negro	15 ml	10 % ≡ 1,5 g
Esmalte de Uñas Blanco	15 ml	10 % ≡ 1,5 g
Agua Oxigenada	1L ≡ 1000 ml	20 % ≡ 20 ml
Polvo Decolorante	360 g	20 % ≡ 72 g
Champú	5 L ≡ 5000 ml	2 % ≡ 100 ml
Acondicionador	5 L ≡ 5000 ml	2 % ≡ 100 ml
Cera	30 g	1 u
Fijador	180 ml/120 g	10 % ≡ 18 ml
Uñas Postizas	1 u	1 u
Crema para Peinar	30 g	33,3 % ≡ 10 g
Rubor	10 g	0,1 % ≡ 0,01 g
Lápiz Labial Rojo	4 g	0,5 % ≡ 0,02 g
Lápiz Labial Negro	4 g	0,5 % ≡ 0,02 g
Lápiz Labial Marrón	4 g	0,5 % ≡ 0,02 g
Desmaquillante Facial	120 ml	20 % ≡ 24 ml
Crema para Manos/Piel	30 g	33,3 % ≡ 10 g
Toallas de Papel/Algodón	10 u	5 u
Alcohol en Gel	60 g	½ u
Quitaesmalte	300 ml	25 % ≡ 75 ml

Todo esto es realizado para así poder crear una función que determine la cantidad de trabajos de cierto tipo a poder realizar en base a la cantidad de turnos agendados y las cantidades de los productos requeridos, por ejemplo para x cantidad de tinturas programadas el stock es suficiente, justo o faltante.

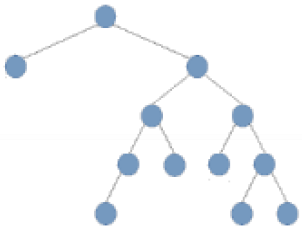
Por la naturaleza de los arboles binarios se necesita una forma de organizar el árbol evaluando cierto campo de la estructura que componen a sus elementos y por ello en base a lo anteriormente expuesto, se consideró que el campo código de producto (cod_producto) realizara dicha función presentándose algunos inconvenientes ya que para sacar provecho de la estructura elegida se debe contar con un árbol más o menos equilibrado, ya que un árbol degenerado se comporta como un lista ocupando los parámetros de un árbol.

Como solución se determinó establecer una secuencia de asignación de códigos de producto independiente al orden de carga, es decir, que los códigos serán asignados en función de la cantidad de productos que el árbol posea cargados al momento de la inserción sin depender de otro factor lo que resulta bastante práctico considerando que la carga puede realizarse en un orden variado.

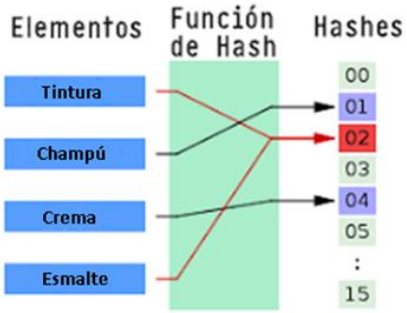
Dicho esto, al trabajar con una cantidad acotada y representativa de productos, el primer código asignado será el de la mitad de la cantidad total de productos (considerando un margen proporcional para agregar productos extra) en el caso del problema será para una cantidad analizada de 27 unidades (más 3 adicionales) el código 15 (30/2) y así sucesivamente procediendo a los resultados enteros de las fracciones cuartas, octavas y dieciseisavas del total para llegar a un árbol lo más equilibrado posible.



Árbol equilibrado



Árbol desequilibrado



La alternativa considerada y solución pensada para el caso de trabajar con el conjunto completo de productos del inventario fue utilizar una función hash que devuelva un código ante el ingreso del nombre de un producto, pero fue descartada debido al tamaño de la solución en comparación al problema.

Como complemento, se ideo cortar el código del programa para presentar a modo de extensión un ejecutable que permitiera únicamente crear o modificar los archivos de cada estructura para facilitar su modificación completa o reestructuración, pero fue descartada debido a que posteriormente se estableció utilizar archivos de texto para almacenar la información.

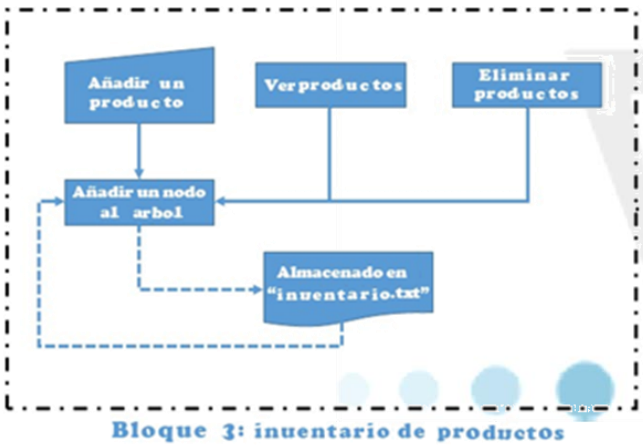
Continuando con el diseño de las funciones de manejo básico de la estructura, el grupo se encontró con el siguiente problema: al ser el campo código de trabajo una referencia interna del programa, para la búsqueda de un elemento el usuario deberá conocer los códigos correspondientes a cada producto. Para solucionar el tema y permitir la búsqueda de los elementos por nombre se decidió utilizar un arreglo lineal, denominado "vector guía", que contenga los nombres de los productos cargados y sus respectivos códigos asignados debido a la dificultad de recorrer el árbol evaluando el campo nombre.

```
struct guia {
        int cod_producto;          ← Fila de Códigos
        char nombre_prod [50];     ← Fila de Nombres
};
```

01	02	03	04	...	...	30
Champú	Tintura	Esmalte	Crema	'\0'	'\0'	'\0'

arreglo de tipo struct guía

Como se puede ver en la ilustración anterior, los espacios libres del vector guía se visualizan con un carácter nulo de código ASSCI (inicialización de la fila de nombres) para que al cargar un elemento se ocupe la menor posición disponible solucionando el problema de la asignación de códigos preestablecidos y el problema de la búsqueda de un elemento con el mismo array, ya que al encontrar un nombre en el vector se accede a la posición de su código correspondiente.



Bloque 3: inuentario de productos

Como se puede observar en el diagrama anterior, debido a la complejidad de los vínculos pensados para relacionar los datos del inventario de productos y las estructuras antes mencionadas, no se han podido concretar las funciones que lo hacen; mas allá de esto, el bloque de código permite cargar un producto al árbol binario llamando a una función que primeramente recorre el vector guía hasta hallar un cupo libre (nombre con el carácter nulo de ASSCI) y obteniendo el código asignado de la posición libre encontrada antes.

Luego de que la función ubique la posición correspondiente del nuevo elemento dentro del árbol binario mediante recursividad, se procede a reservar la memoria requería e ingresar los campos que componen a la estructura inventario de haber podido reservarse correctamente la memoria para preguntar luego si se desea modificar alguno de ellos.

Para el proceso de eliminación, se pide ingresar el nombre del producto que se busca eliminar y con dicho dato recorrer el vector guía para encontrar su posición y su correspondiente código; posteriormente se muestran los datos de dicho producto y se confirma la decisión. Además se ideo otra función que elimina todos los productos cargados repitiendo el proceso hasta que el nodo maestro (que contiene la dirección de memoria del elemento raíz) sea igual a NULL.

Para la modificación de algún campo de cierto producto, se procede de manera similar para hallar al elemento y luego de enseñar sus datos sobrescribir los anteriores en caso de validar la decisión. Cabe destacar que al ingresar un nombre, se verifica que no se encuentre cargado anteriormente invalidando la existencia de dos productos con un mismo nombre y para asegurar comparaciones exitosas se minimizan todos los caracteres que componen al nombre ingreso mediante una función.

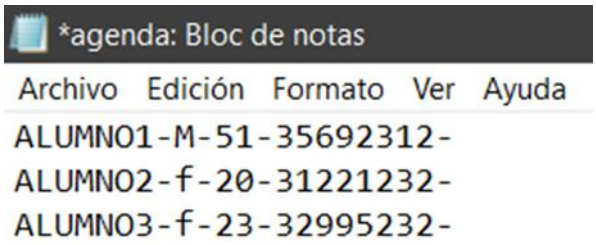
Además, se seleccionó el recorrido in-orden para mostrar el árbol binario debido a que recorre el árbol en orden ascendente y al utilizar un formato tipo tabla para enseñar los datos (considerada una forma natural para observar un listado de productos por el grupo), los códigos de producto quedan ordenados de menor a mayor; en conjunto a la función de mostrar todos los productos cargados, se agregaron una función que enseña todos los productos agotados y otra que muestra en pantalla los datos de un único producto.

Para finalizar, al cerrar el programa se almacena la información en un archivo de texto (al igual que las otras estructuras) utilizando como carácter separados al guion medio “ - ” y separando los datos de cada producto por renglones.

Luego de todo lo anterior, como último paso se libera la memoria reservada para todas las estructuras mediante la función correspondiente que llama a las funciones que eliminan todos los elementos de cada estructura.

Guardado de datos:

Al finalizar el programa, la información debe ser guardada y para ello se implementó la utilización de archivos de texto como se mencionó anteriormente, creando o sobrescribiendo un archivo .txt asociado a cada estructura al salir del programa.
Cada dato del cliente esta separa por un "-" para su correcta lectura posterior.



Al utilizar archivos de texto, y poseer conocimiento de la funcionalidad del mismo, se puede ver o editar el contenido del mismo, es decir, los datos de cada cliente sin la necesidad de abrir el programa.

Esta opción nos pareció muy interesante al plantearnos la interrogativa de qué tipo de archivo manejar, para posteriormente, ser capaces de colocar en el manual de usuario las aclaraciones e informaciones correspondientes para que el usuario disponga de todos estos recursos.

Lectura de datos:

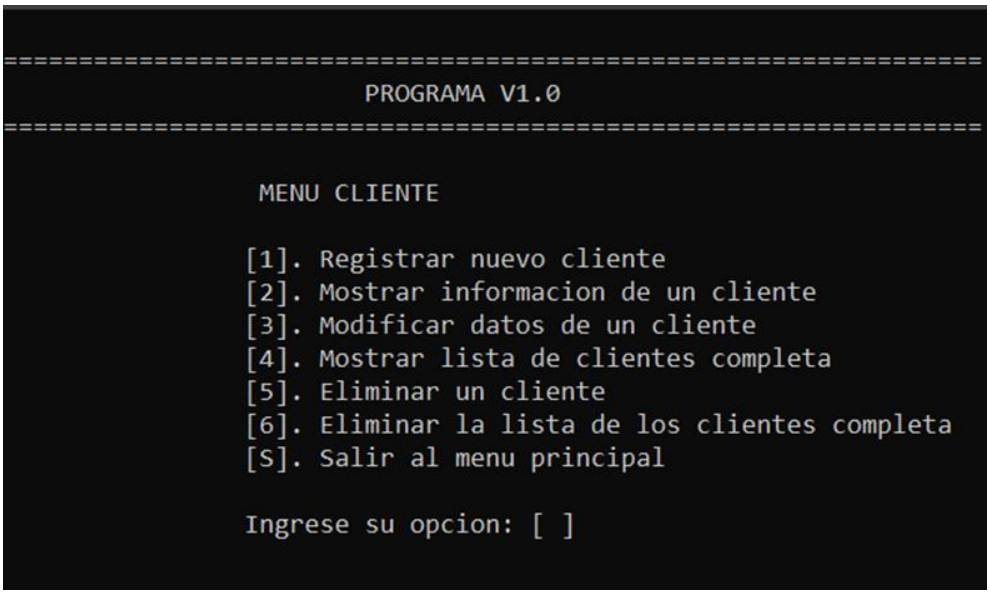
La lectura de datos, se realiza una vez abierto el programa; los datos son leídos carácter por carácter hasta llegar a la marca distintiva "-" y luego ser asignados a las variables correspondientes (un nombre a un string y un número a un entero).

Al cargar los datos se imprime en pantalla si la carga de los mismos, y la correspondiente lectura del archivo, fue exitosa o si hubo algún error durante el proceso.

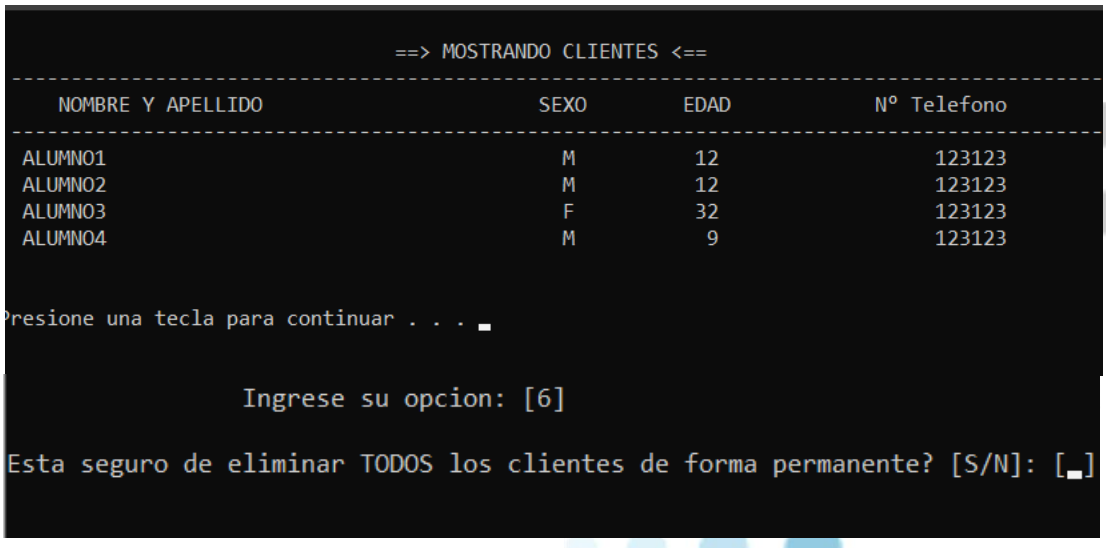
Para el caso de esta estructura no se consideraron alternativas ya que se tenía muy en claro que una lista serviría adecuadamente para el problema.

Diseño de Interfaces:

La primer decisión tomada para el diseño de interfaces con el usuario fue mantener un estilo visual en todas las pantallas del programa y utilizar la misma validación de si o no (s/n) cada vez que deba hacerse. Luego se determinó que se debían resaltar los “títulos” de cada menú para facilitar la orientación de usuario en el programa.



En la parte superior se puede observar el resultado final del diseño de los menúes (en la imagen el menú de la estructura cliente) y las decisiones que desembocaron en el fueron sangrar o identar todo el texto, centrarlo a la pantalla y agruparlo de manera tal de forma que se identifiquen claramente un título, una serie de opciones y una oración de requerimiento.



FUNCIONES Y PROCESOS QUE REALIZA EL SISTEMA

El programa, hasta el momento de la entrega, se encuentra en el límite provisorio de alcance establecido anteriormente debido básicamente al tiempo que se tuvo disponible para el desarrollo, pero se estima que en un plazo igual o menor de tiempo adicional se pueden concluir las operaciones hasta completar el desarrollo con las funciones más complejas.

El sistema permite manipular los elementos de tres estructuras de datos independientes una lista doblemente enlazada, una pila y un árbol binario que contienen información proveniente del negoción de estética sobre el cual se basa el mismo, dicha información presenta tres fuentes claras: el registro de clientes, el calendario de turnos y el inventario de productos. Específicamente el código brinda las opciones de manejo básico para cada una de las estructuras que utiliza el programa para manejar la información: agregar, eliminar o modificar un elemento y mostrar todo el arreglo.

Por último, la información de cada estructura es almacenada en un archivo de texto independiente para cada una de ellas y se libera la memoria asignada dinámicamente por el programa.

ENTRADAS DE DATOS DEL SISTEMA

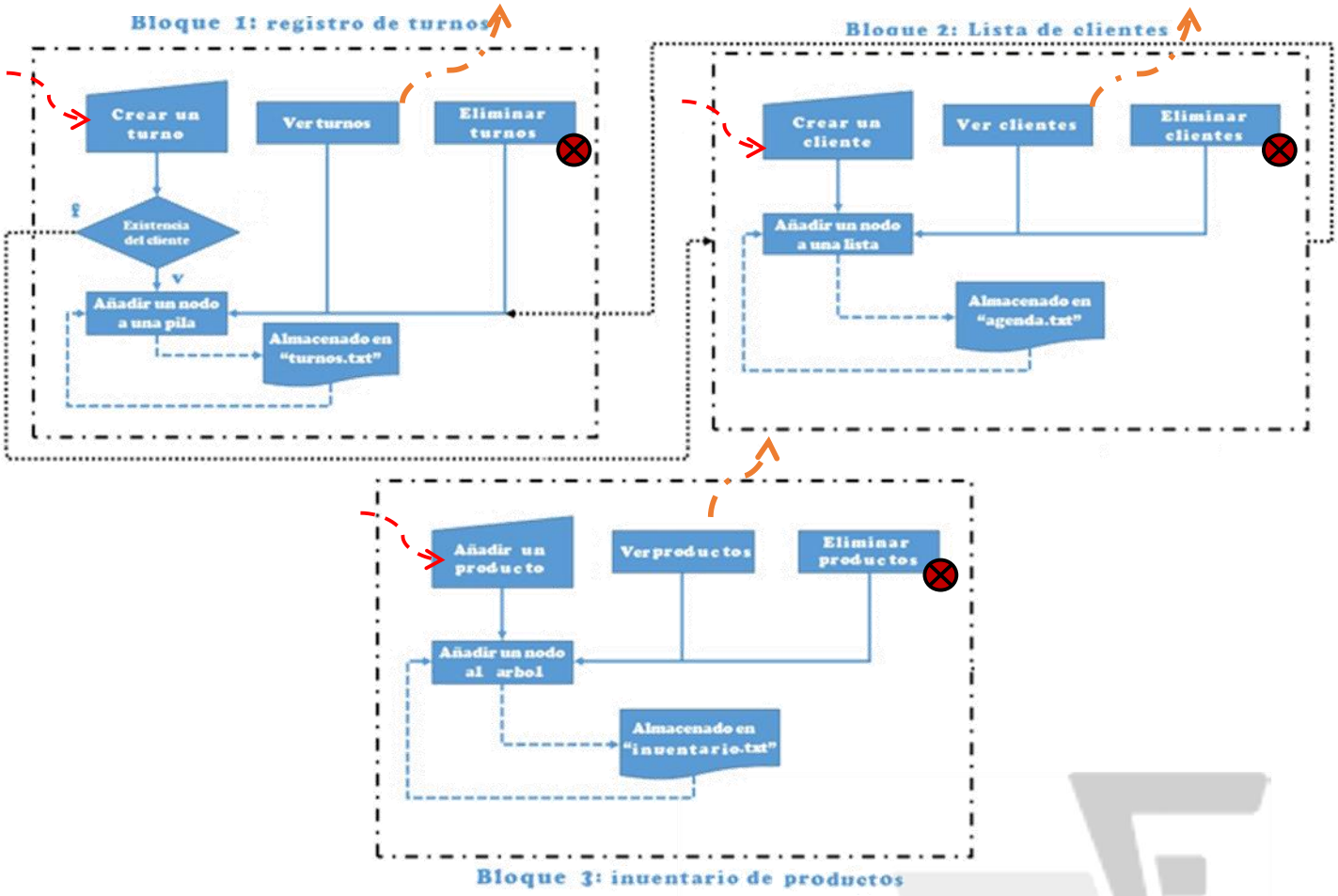
Dependiendo de la estructura sobre la cual se trabaje, los datos requeridos por el sistema serán distintos, como se listan a continuación exceptuando la validación que será general para todo el sistema:

Registro de clientes:	Calendario de Turnos:	Inventario de Productos:
_ Nombre del Cliente. _ Sexo. _ Edad. _ Número Telefónico.	_ Nombre del Cliente. _ Tipo de trabajo. _ Día de turno. _ Hora de inicio y final.	_ Nombre del Producto. _ Cantidad Disponible.

SALIDAS DE INFORMACIÓN DEL SISTEMA

Dado que no se pudo completar el desarrollo de las funciones más complejas del programa, que relacionan de manera profunda la información ingresada, la información enseñada por pantalla será la misma pero organizada.

ESQUEMAS DE LAS INTERFACES USUARIO-SISTEMA



Reutilizando el diagrama mostrado anteriormente, las flechas azules representan procesos internos a cada bloque de código y las flechas negras procesos entre bloques de código; las flechas rojas representan pantallas de ingreso de información (que varía según la estructura sobre la cual se trabaje como se explicó anteriormente), las flechas naranjas representan las pantallas donde se muestra dicha información y las marcas rojas con cruces negras hacen referencia a las validaciones requeridas por el sistema.

CONCLUSIONES

Como conclusión general, el grupo destaca que fue sumamente provechosa la experiencia de desarrollo de un proyecto desde cero, para cumplir ciertos requisitos establecidos por la cátedra en una situación semejante a las que seguramente se presentarán durante el desarrollo de nuestra actividad profesional, ya que tuvimos un paneo general de diversos temas que conforman a un profesional como ser:

_ Conocimiento general de las etapas de desarrollo de un proyecto informático.
_ Desarrollo de las capacidades de un profesional para el trabajo en equipo.
_ Exploración de la capacidades (virtudes y carencias) propias.
_ Compresión de los costos de tiempo que requiere el desarrollo de un programa.
_ Valoración de distintas alternativas y puntos de vista para tomar decisiones.

Por otro lado, el grupo resalta que la puesta en práctica de los conocimientos adquiridos desde el comienzo del cursado de la carrera hasta el momento fue un ejercicio evaluativo excelente ya que requirió de la utilización de todos los conceptos vistos desde los más básicos hasta los más complejos exigiendo en muchos casos el replanteamiento de alguno de ellos para corregirse o reformarse, logrando mejorar la formación profesional de cada integrante.

BIBLIOGRAFÍA

El grupo se valió principalmente del material brindado por la cátedra durante el transcurso del cursado (aunque también se visitó en repetidas ocasiones el material brindado el año anterior), pero por un tema de desconocimiento sobre cómo abordar un proyecto como el aquí desarrollado se recurrió a bibliografía que habla de los pasos a seguir para desarrollar un sistema que fueron orientativos utilizando la plataforma eLibro.net:

_ JOYANES AGUILAR, Luis. Programación en C: Metodología, Algoritmos y Estructuras de Datos (2da. Ed.). Editorial Mc-Graw Hill. 2005.

WEBGRAFÍA

Por otro lado, en ocasiones particulares se consultaron páginas relacionadas al contenido de la materia para evacuar duda con respecto a funciones del lenguaje:

_ Validar sólo ingreso de números en C. (2016, 31 octubre). Stack Overflow en español. https://es.stackoverflow.com/questions/30801/validar-s%C3%B3lo-ingreso-de-n%C3%BAmeros-en-c

_ C library function - atoi() - Tutorialspoint. (s. f.). Tutorials Poin. Recuperado 17 de junio de 2020, de https://www.tutorialspoint.com/c_standard_library/c_function_atoi.htm