Práctica 4

Semestre de otoño
Curso 2020/2021

Contexto de la práctica

Se desea codificar una aplicación que sirva para gestionar las reservas hechas en un hotel con varias habitaciones disponibles.

La práctica se estructura en 5 fases obligatorias incrementales que se describen detalladamente a continuación. También se propone una sexta fase que será opcional.

Material disponible en Moodle para el desarrollo de la práctica

Como complemento para la realización de la práctica, el alumno tendrá a su disposición los siguientes recursos:

- Ficheros p4f1.c, p4f2f3.c y p4f4f5.c
- Fichero ejecutable con la funcionalidad completa de la aplicación, disponible para realizar pruebas y aclarar dudas sobre el funcionamiento detallado del programa.

Enunciado detallado y proceso para realización de la práctica

PRIMERA FASE. Registro de los clientes de una habitación

En la primera fase del desarrollo de la práctica se completará el fichero **p4f1.c** que se proporciona en Moodle, siguiendo los pasos que se describen a continuación.

Paso 1: Se deben estudiar y entender los tipos de datos necesarios para gestionar el registro de clientes de una habitación, los cuales se proporcionan en el fichero p4f1.c.

La ilustración 1 muestra un diagrama de la estructura de los tipos de datos que se definen en esta fase, y los campos que contienen.

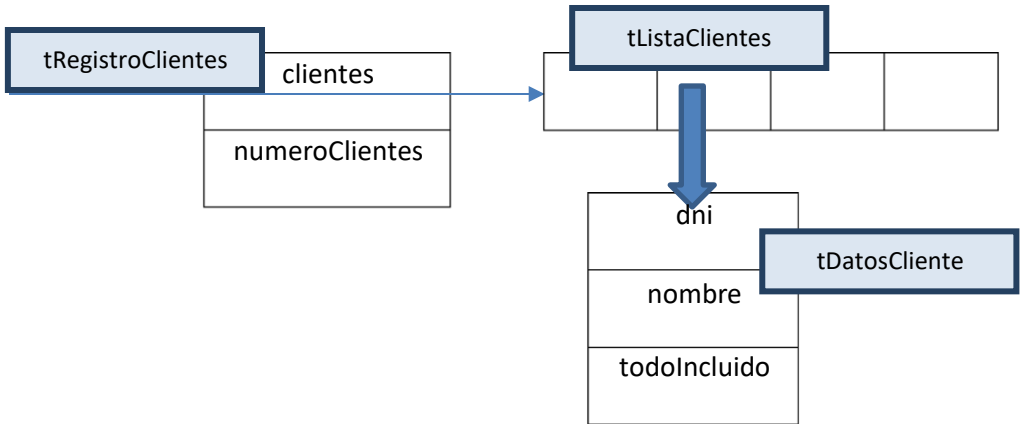


Ilustración 1. Diagrama representativo de las estructuras necesarias para el almacenamiento de datos de los clientes de una habitación

Los tipos de datos definidos se ajustan a las siguientes restricciones:

- El registro de clientes de una habitación se almacenará en una estructura de tipo **tRegistroClientes** que tiene dos campos, el **campo clientes** que es un array de tipo **tListaClientes** y el **campo numeroClientes** que es un **entero** en el que se indicará el número de clientes registrados en la habitación. Una habitación puede tener registrados un número de clientes comprendido entre 0 y 4, ambos valores incluidos.
- El array de tipo tListaClientes puede almacenar 4 elementos, cada elemento es de tipo **tDatosCliente**.
- El tipo tDatosCliente es una estructura con tres campos, el **campo dni** que es una **cadena de caracteres** en la que se pueden almacenar los **9** caracteres efectivos que conforman el número y la letra de un dni, el **campo nombre** que es una **cadena de caracteres** en la que se pueden almacenar nombres de hasta **30** caracteres efectivos, y el **campo todoIncluido** que es un valor **booleano** que contendrá el valor **true** si el cliente ha elegido la modalidad "todo incluido" y **false** si el cliente ha reservado la habitación en la modalidad "alojamiento y desayuno".

Para una definición óptima de los tipos de datos se han definido las constantes relacionadas con el tamaño del array de clientes y de las cadenas de caracteres que se utilizan.

Paso 2: Se deben **codificar** los **prototipos** de las funciones **habitacionLlena(), vaciarHabitacion(), altaCliente()** y **listarClientes()** que aparecen descritos en el fichero p4f1.c.

Paso 3: Se deben **codificar las funciones habitacionLlena(), vaciarHabitacion(), altaCliente()** y **listarClientes()** para que cumplan con la funcionalidad descrita en el correspondiente comentario en el que se describe el prototipo**.**

Nota: Los prototipos y la codificación de las funciones **existeCliente(), leerTexto ()** y **leerBooleano ()**, se proporcionan en el fichero p4f1.c.

Paso 4: Se debe **completar** el código de la función **main()**, invocando a las funciones que sea necesario en cada momento. La función main() es un programa que sirve para probar el correcto funcionamiento de las funciones codificadas en las fase 1 y de los tipos de datos definidos. El programa solicita los datos de 6 clientes.

Paso 5: Se debe **probar** el programa, introduciendo los datos de los 6 clientes que figuran en el comentario que hay dentro del main(). Si el programa se ha codificado correctamente detectaría que se intenta registrar un cliente que ya está y no lo incluiría y no dejaría registrar al último cliente por estar llena la habitación.

SEGUNDA FASE. Registro de las habitaciones disponibles de un hotel

En la segunda fase del desarrollo de la práctica se completará el fichero **p4f2f3.c** que se proporciona en Moodle, siguiendo los pasos que se describen a continuación.

Paso 1: Se deben definir los tipos de datos necesarios para gestionar la lista de habitaciones disponibles de un hotel. Se deben **mantener los nombres y la estructura de los tipos** tal y como aparecen en la figura.

La ilustración 2 muestra un diagrama orientativo de cómo deberán disponerse, y los campos que deberán contener los distintos tipos de datos.

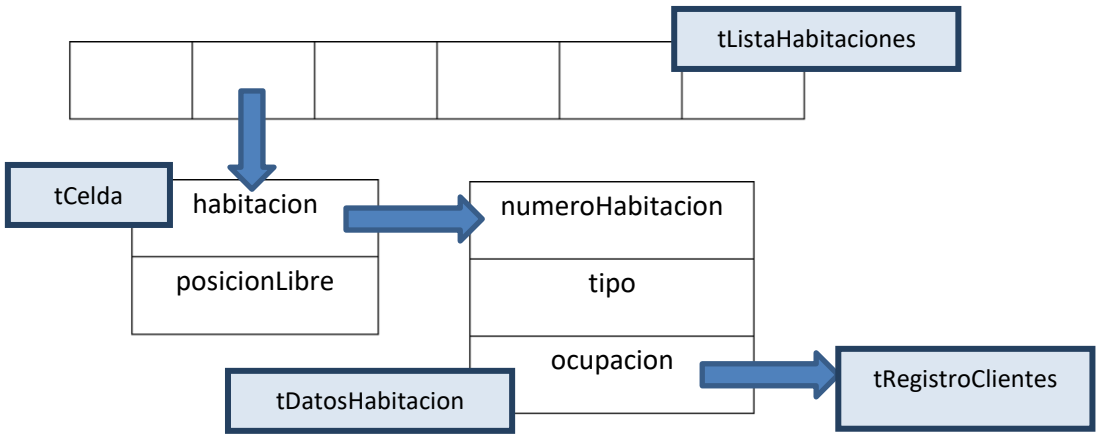


Ilustración 2. Diagrama representativo de las estructuras necesarias para el almacenamiento de datos de las habitaciones disponibles de un hotel

Se deben definir los tipos de datos acorde a las siguientes restricciones:

- Las habitaciones disponibles del hotel se almacenarán en un array de tipo **tListaHabitaciones** que para la fase de pruebas del programa tendrá un tamaño de **3 elementos**. Cada elemento del array será una estructura de tipo **tCelda**.
- El tipo tCelda es una estructura con dos campos, el **campo habitacion** que es de tipo **tDatosHabitacion** y el **campo posicionLibre** que es un valor booleano que se pondrá a **true** si el correspondiente elemento del array está vacío, y **false** si el correspondiente elemento del array tiene un valor significativo.
- El tipo tDatosHabitacion es una estructura con tres campos, el **campo numeroHabitacion** que es un **entero** en el **rango [100, 200]** en el que se almacena el número de la habitación, el **campo tipo** que es una **cadena de 10 caracteres** significativos en la que se almacena el tipo de la habitación (por ejemplo, "suite", "individual", etc) y el **campo ocupacion** de que de tipo **tRegistroClientes** (descrito y codificado en la fase 1) que contendrá los datos de los clientes registrados en la habitación.

Para una definición óptima de los tipos de datos se deben **definir las constantes** relacionadas con el tamaño del array habitaciones y de las cadenas de caracteres que se utilizan.

Paso 2: Se deben **incluir** los **prototipos** de las funciones definidos en la **fase 1** en el lugar indicado en p4f2f3.c. Se deben **codificar** los **prototipos** de las funciones **leerEnteroEnRango(), habitacionRegistrada(), hayEspacioEnHotel(), abrirHotel(), annadirHabitacion(), borrarHabitacion() y listarHabitacionesOcupadas()** que aparecen descritos en el fichero p4f2f3.c.

Paso 3: Se debe **incluir** el **código** de las funciones codificadas en la **fase 1** en el lugar indicado en p4f2f3.c. Se deben **codificar las funciones leerEnteroEnRango(), habitacionRegistrada(), hayEspacioEnHotel(), abrirHotel(), annadirHabitacion(), borrarHabitacion() y listarHabitacionesOcupadas()** para que cumplan con la funcionalidad descrita en el correspondiente comentario en el que se describe el prototipo**.**

Paso 4: Se debe **completar** el código de la función **main()**, invocando a las funciones que sea necesario en cada momento. La función main() es un programa que sirve para probar el correcto funcionamiento de las funciones codificadas en las fase 2 y de los tipos de datos definidos. El programa solicita los datos de 6 habitaciones.

Paso 5: Se debe **probar** el programa, introduciendo los datos de las 6 habitaciones que figuran en el comentario que hay dentro del main(). Si el programa se ha codificado correctamente detectaría que se intenta registrar una habitación que ya está registrada, una habitación con un número fuera de rango y no dejaría registrar la última habitación por estar lleno el array de habitaciones. Una vez registradas las habitaciones se darán de alta los clientes indicados en el comentario correspondiente en las habitaciones 100 y 200. Tras la toma de datos, se presentará un listado de ocupación por pantalla, y por último se tratará de borrar 2 habitaciones (una que existe y otra que no) y volverá a presentar por pantalla un listado de la ocupación del hotel.

TERCERA FASE. Ampliación de funcionalidades programa

En esta tercera fase se añadirán 2 nuevas funcionalidades:

- Buscar (por dni) la habitación donde se encuentra un cliente.
- Calcular número total de personas en el hotel.

Paso 1: Se deben **codificar** los **prototipos** de las funciones **buscarCliente()** y **totalClientesEnHotel()**, que aparecen descritos en el fichero p4f2f3.c.

Paso 2: Se deben **codificar** las funciones **buscarCliente()** y **totalClientesEnHotel()**, para que cumplan con la funcionalidad descrita en el correspondiente comentario en el que se describe el prototipo.

Paso 3: Se deben quitar los comentarios de la última parte del código de main() para que se pueda ejecutar este código (es el código correspondiente a la búsqueda de clientes y el cálculo del número total de clientes registrados en el hotel) y de debe **completar**, invocando a las funciones que sea necesario en cada momento.

Paso 4: Se debe **probar** el programa, introduciendo los datos que figuran en el comentario correspondiente a cada acción que se ejecuta dentro del main().

CUARTA FASE. Codificación del programa en modo menú

En esta fase se completará el fichero **p4f4f5.c** que contendrá una función main() que permita que el programa funcione a través de un menú con las siguientes opciones:

1. Registrar habitación.
2. Liberar habitación.
3. Incluir clientes en una habitación.
4. Listar ocupación completa del hotel.
5. Listar ocupación de una habitación.
6. Buscar habitación de cliente.
7. Indicar ocupación del hotel.
8. Salir de la aplicación.

Paso 1: Se debe incluir la definición de constantes, tipos y prototipos definidos en la fase 3 (incluyendo los comentarios de los prototipos de las funciones). Se debe incluir también el código de todas las funciones codificadas en las fases anteriores y que están en p4f2f3.c.

Paso 2: Se deben **codificar** los **prototipos** de las funciones **menu(), MenuRegistraHabitacion(), MenuEliminaHabitacion(), MenuRegistraClientes(), MenuListaTotal(), MenuListaHabitacion(), MenuBuscaCliente()** y **MenuCuentaClientes()**, que aparecen descritos en el fichero p4f4f5.c.

Paso 3: Se deben **codificar** las funciones **menu(), MenuRegistraHabitacion(), MenuEliminaHabitacion(), MenuRegistraClientes(), MenuListaTotal(), MenuListaHabitacion(), MenuBuscaCliente()** y **MenuCuentaClientes()**, para que cumplan con la funcionalidad descrita en el correspondiente comentario en el que se describe el prototipo.

Paso 4: Se debe **codificar** la función **main()** para que una vez abierto el hotel, presente el menú descrito en esta fase y se solicite una opción. Una vez elegida una opción se realizará la acción correspondiente que estará codificada en una de las funciones codificadas en el paso anterior. Una vez realizada la correspondiente acción se volverá a presentar el menú. Cuando se seleccione la opción 8 el programa finalizará.

Paso 5: Se debe **probar** el programa introduciendo los datos que se consideren necesarios para probar su correcto funcionamiento.

QUINTA FASE. Ficheros

En esta quinta fase se implementarán las funcionalidades relativas a la gestión de ficheros: se incluirán las funciones de *lectura de un fichero* y *escritura en un fichero*. Las funciones a codificar son **leerDatosHotel()**, **guardarDatosHotel().** Hay unas funciones análogas publicadas en las diapositivas de clase de la Unidad 6 y serán explicadas en la clase de teoría.

El nombre del fichero donde se almacenará la información de la aplicación será "datosHotel.dat".

En **main()** se incluirá lo siguiente:

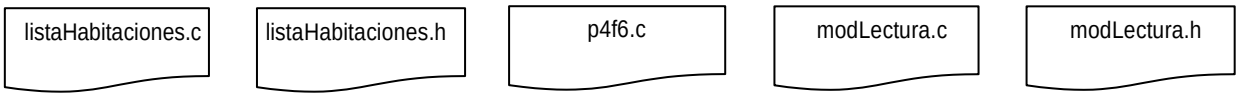
- Una vez abierto el hotel se invocará a leerDatosHotel(). Si se han podido recuperar datos del fichero se escribirá por la pantalla la frase "***Registro de clientes y habitaciones actualizado***", si no se han podido recuperar los datos se escribirá por la pantalla la frase: "***No se han encontrado datos de clientes y habitaciones. El hotel esta actualmente vacio***").
- Al salir del programa (opción 8 del menú) se invocará a guardarDatosHotel(). Se recuerda que esta función solo debe guardar en el fichero la **información** de tipo **tDatosHabitacion** de las posiciones del array de habitaciones que **no estén libres**.

SEXTA FASE. Modularidad y paso de parámetros a main(). OPCIONAL

Como fase opcional se propone reorganizar el programa en formato modular. Para ello se llevarán a un módulo todas las funciones de lectura de datos (leerTexto(), leerBooleano() y leerEnteroEnRango()), a otro módulo todas las funciones relacionadas con el tipo de datos tListaHabitaciones, de forma que en el módulo principal p4f6.c solo quede el código de las funciones main() y menu().

Al hacer los módulos se debe tener en cuenta qué funciones son externas y cuáles no.

Tras completar esta fase la estructura completa del programa quedaría de la siguiente forma:



El **nombre del fichero** en el que se guardan los datos de las habitaciones registradas en el programa al salir y del que se recuperan al arrancar, en lugar de ser la cadena de caracteres constante **"datosHotel.dat"** se introducirá como **parámetro de main()**, pudiendo poner el nombre de fichero que se desee cada vez que se ejecuta el programa.

Consideraciones generales para la realización y entrega de la práctica.

Notas sobre organización de espacios de trabajo, proyectos y archivos fuente

De acuerdo con lo especificado en las prácticas anteriores:

1. Se creará mediante el explorador de archivos la carpeta '**P4**' dentro de la carpeta dedicada a la asignatura.
2. Se deben hacer los siguientes proyectos (P4F1, P4F2F3, P4F4F5 y P4F6, este último si se hace la parte opcional).
3. La compilación, enlazado, ejecución y posible depuración de los programas se hará utilizando *Code::Blocks*.