PROYECTO FINAL PENSAMIENTO ALGORITMICO

Por: Juanita Gómez
Ing. Mery Yolima Uribe Rios
Departamento de Ingeniería de Sistemas y Computación
Pontificia Universidad Javeriana

Documentación

ESTRUCTURAS Y ARREGLOS

1. Struct actividad: Es la estructura que contiene los datos que debe tener cada una de las actividades de la semana cultural.

Incluye el nombre, el código, la jornada, el tipo, la jornada, la duración, el día y la hora de inicio de cada actividad. El código, la duración y la hora de inicio son enteros porque contienen datos numéricos, mientras las otras son cadenas porque contienen palabras.

2. Struct actividad arregloact[30]: Para almacenar todas las actividades de la semana, se creó un arreglo de tipo Struct actividad para que guarde todos los datos de cada actividad. El tamaño es 30 porque el máximo número de actividades q puede haber en la semana es 30 porque son 5 días, 2 jornadas cada uno y 3 actividades máximo por jornada.

3. Struct usuario: Estructura que controla los datos de quien utiliza el programa, para verificar el tipo de público que puede asistir a cada actividad. Contiene información de la edad y el tipo de personal. Para ambos se manejan enteros.

4. String semana [5]: Arreglo que contiene los dias de la semana como cadenas, para imprimirlos en el horario. El tamaño es 5 porque son 5 días de la semana.

5. String horas [16]: Arreglo que contiene las horas de la semana como cadenas, para imprimirlos en el horario. El tamaño es 16 porque el horario de la semana es de 8 a 12 (4 horas en la mañana), y de 2 a 6 (4 horas en la tarde) y para mostrarlo se utilizan intervalos de media hora.

6. Int horasn[16]: Arreglo que contiene las horas de la semana como enteros (en hora militar) para poder ser comparados a la hora de ingresar las actividades, y permitir la correspondencia entre el arreglo de horas y las filas de la matriz donde se almacena el horario.

7. Struct actividad matriz[16][5]: Matriz de tipo Struct actividad que permite almacenar las actividades de la semana cultural. Tiene 5 columnas, una para cada uno de los días de la semana, y 16 filas que son los 16 intervalos de media hora en las horas designadas para las actividades durante la semana.

8. Struct jornada: Estructura que contiene un arreglo de tipo entero de tamaño 6 de tal forma que cada uno de los tipos de actividad corresponda con las posiciones del arreglo. En este caso la posición 0 corresponde a deportiva, 1 cultural, 2 artistica, 3 musical, 4 plastica y 5 de la

organización. El arreglo guarda enteros, porque estos funcionan como contadores que se explican mas adelante. Se declaro una estructura “j” de tipo Struct jornada.

9. Struct dia: Estructura que contiene un arreglo de tipo struct jornada de tamaño 2 de tal forma que cada uno de las jornadas posibles (mañana o tarde) corresponda con las posiciones del arreglo. En este caso la posición 0 corresponde a mañana y la posición 1 corresponde a tarde.

Se declaro una estructura “d” de tipo Struct dia.

10. Struct contador: Estructura que contiene un arreglo de tipo struct dia de tamaño 5 de tal forma que cada uno de los 5 días de la semana corresponda con las posiciones del arreglo. En este caso la posición 0 corresponde al lunes, 1 al martes, 2 al miércoles, 4 al jueves y 5 al miércoles.

Se declaro la estructura c1 de tipo struct contador para verificar que dos actividades del mismo tipo no queden registradas en la misma jornada del mismo día. La estructura c1 funciona de la siguiente forma.

Se le suma 1 a **c1.d[d1].j[j1].tipo[t1]** cada vez que se ingresan el día, la jornada y el tipo de cada actividad. Las variables d1, j1 y t1 determinan cual de los contadores es el que se está incrementando, dependiendo del día, la jornada y el tipo como se explico anteriormente.

FUNCIONES

1. Void llenaractividades (struct actividad arregloact[], struct actividad matriz[][5]): Función que permite registrar 5 actividades con anterioridad cargadas antes de la interacción con el sistema. En la función se llena la información de las primeras 5 actividades ocupando las primeras 5 posiciones del arreglo de actividades. Es necesario introducir los datos de cada uno de los elementos de la estructura; es decir el nombre, el código, la jornada, el tipo, la jornada, la duración, el día y la hora de inicio. Además se requieren llenar las posiciones en la matriz del horario semanal de acuerdo al día, la hora de inicio y la duración de la actividad.

La función es de tipo Void por lo que no retorna ningún valor. Como los arreglos y matrices son globales, no es necesario retornar nada a la función principal.

Los parámetros de la función, son el arreglo de actividades y la matriz de actividades, ambos de tipo “Struct actividad” y ambos necesarios para introducir los datos de las 5 actividades que se deben ingresar. Para el arreglo, no es necesario mandar el tamaño por parámetro, porque ya se sabe que el tamaño es 30, para la matriz, es necesario solo mandar el número de columnas por parámetro (5), y se conoce que el número de filas es 16.

Serie de pasos:

1. Dar valor a la variable nombre dentro del arreglo de estructuras, en la posición 0.

2. Dar valor a la variable código dentro del arreglo de estructuras, en la posición 0.
3. Dar valor a la variable tipo dentro del arreglo de estructuras, en la posición 0.
4. Dar valor a la variable jornada dentro del arreglo de estructuras, en la posición 0.
5. Dar valor a la variable duración dentro del arreglo de estructuras, en la posición 0.
6. Dar valor a la variable día dentro del arreglo de estructuras, en la posición 0.
7. Dar valor a la variable horainicio dentro del arreglo de estructuras, en la posición 0.
8. Insertar la estructura dentro de la matriz de acuerdo a la información del día, la duración y la hora de inicio. En este caso como el día es el lunes, la columna es la 0 y como la hora de inicio es 1400 y la duración son 90, se deben llenar las filas 8, 9 y 10 de la matriz.
9. Repetir el mismo procedimiento con las otras 4 actividades en las posiciones 1, 2, 3, 4 del arreglo y columnas 1, 2, 3, 4 de la matriz.

2. Void inicializarcontadores(struct contadores c1): Función que se encarga de inicializar los contadores de la estructura c1 que verifica que dos actividades del mismo tipo no queden registradas en la misma jornada del mismo día. Los contadores que incluyen a las 5 actividades que son ingresadas con anterioridad, deben ser inicializados en 1 y no en 0.

El único parámetro necesario para la función, es la estructura c1 que contiene los contadores. Dentro de esta estructura c1, se encuentra el arreglo de estructuras Struct día d[] donde a su vez esta el arreglo de estructuras Struct Jornada J[] que contiene el arreglo de los tipos de actividad .

La función es de tipo Void por lo que no retorna ningún dato. Como la estructura es global, no es necesario retornar nada a la función principal.

Serie de pasos:

Para la primera actividad ingresada, futbol, se debe tener en cuenta que el dia asignado es el lunes, por lo que la variable d1=0, la jornada de la actividad es tarde por lo que la variable j1=1 y el tipo de la actividad es deportiva así que la variable t1=0. De este modo se le da a c1.d[0].j[1].tipo[0] el valor de 1.

Debe repetirse el mismo razonamiento para determinar el valor de las variables d1, j1 y t1 de las 4 actividades restantes, e inicializar cada contador en 1.

De este modo queda:

c1.d[0].j[1].tipo[0]=1
c1.d[1].j[0].tipo[3]=1
c1.d[2].j[1].tipo[1]=1
c1.d[3].j[0].tipo[2]=1
c1.d[4].j[0].tipo[4]=1

3. Int contadormanana (struct actividad arregloact[], int manana, int i): Función que se encarga de contar el numero de actividades que hay por jornada en cada uno de los días de la semana, para poder verificar que no haya más de 3 actividades por jornada cada día.

Los parámetros para la función son el arreglo de actividades, el contador "manana" y el contador i. El primero, es de tipo Struct actividad y es el que tiene almacenada toda la información sobre la actividad, para poder verificar si su jornada es mañana o tarde. El segundo es de tipo entero, y es el parámetro que determina de cual día es que se está incrementando el contador. Al llamar esta función la función principal, se debe poner como parámetro el contador correspondiente al día en que se encuentra programada la actividad que se está verificando. Por esto, la función debe llamarse 5 veces en el main, una por cada uno de los días de la semana. Se crearon las variables mananal=0, mananama=1, mananami=0, mananaju=1, mananavi=1 que son las que se envían por parámetro en la función de acuerdo a cada uno de los días. Los contadores de martes, jueves y viernes inician en 1 porque ya existe 1 actividad en cada uno de los días por la mañana. El último parámetro es de tipo entero y es la variable que permite saber en qué posición del arreglo se debe verificar la jornada.

La función es de tipo entero por lo que retorna un entero, que es precisamente el contador que se incrementa en la función.

Serie de pasos:

1. Utilizar la función .compare para verificar si la jornada de la actividad que se está verificando es igual a "mañana".
2. Si esto se cumple aumentar el contador "manana" en 1.
3. Retornar el contador manana.

4. Int contadortarde (struct actividad arregloact[], int tarde, int i) Función que se encarga de contar el numero de actividades que hay por jornada en cada uno de los días de la semana, para poder verificar que no haya más de 3 actividades por jornada cada día.

Los parámetros para la función son el arreglo de actividades, el contador "tarde" y el contador i. El primero, es de tipo Struct actividad y es el que tiene almacenada toda la información sobre la actividad, para poder verificar si su jornada es mañana o tarde. El segundo es de tipo entero, y es el parámetro que determina de cual día es que se está incrementando el contador. Al llamar esta función la función principal, se debe poner como parámetro el contador correspondiente al día en que se encuentra programada la actividad que se está verificando. Por esto, la función debe llamarse 5 veces en el main, una por cada uno de los días de la semana. Se crearon las variables tardel=1, tardema=0, tardemi=1, tardeju=0, tardevi=0 que son las que se envían por parámetro en la función de acuerdo a cada uno de los días. Los contadores de lunes y miércoles inician en 1 porque ya existe 1 actividad en cada uno de estos días por la tarde, El último parámetro es de tipo entero y es la variable que permite saber en qué posición del arreglo se debe verificar la jornada.

La función es de tipo entero por lo que retorna un entero, que es precisamente el contador que se incrementa en la función.

Serie de pasos:

1. Utilizar la función .compare para verificar si la jornada de la actividad que se está verificando es igual a “tarde”.
2. Si esto se cumple aumentar el contador “tarde” en 1.
3. Retornar el contador tarde.

5.Int verificacionjornada(struct actividad arregloact[], int i) Funcion que verifica el tipo de jornada de la s actividades, para saber el valor de la variable j1 que permite el funcionamiento de los contadores de la estructura c1 explicada anteriormente. Si la jornada es mañana, le da el valor de 0, si es tarde le da el valor de 1.

Los parámetros para la función son el arreglo de actividades, y el contador i. El primero, es de tipo Struct actividad y es el que tiene almacenada toda la información sobre la actividad, para poder verificar si su jornada es mañana o tarde. El último parámetro es de tipo entero y es la variable que permite saber en qué posición del arreglo se debe verificar la jornada.

La variable es de tipo entero, y retorna el valor de la variable j1 dependiendo si la jornada es por la mañana o por la tarde.

Serie de pasos:

1. Declarar la variable j1 en un valor diferente de 0 y 1 (ya que son las posiciones posibles del arreglo j de tipo struct jornada.
2. Verificar si la jornada de la actividad en la posición i es mañana o tarde.
3. Si es mañana asignarle 0 a j1, si es tarde asignarle 1.

6. Void inicializarmatriz(struct actividad matriz[][5], int f): Función que inicializa la matriz de estructuras que almacena el horario de actividades para la semana cultural.

Como parámetro solo requiere la matriz de estructuras que contiene toda la información de las actividades, ubicada en el horario semanal y le variable f que representa el numero de filas de la matriz.

La función es void por lo que no devuelve ninguna variable

Serie de pasos:

1. Hacer dos ciclos for que controlen las posiciones de la matriz, variando filas y columnas y así recorrer toda la matriz

2. Inicializar con espacios la variable nombre dentro de la matriz de estructuras, en todas sus posiciones.
3. Inicializar con -1 la variable código dentro de la matriz de estructuras, en todas sus posiciones.
4. Inicializar con espacios la variable tipo dentro de la matriz de estructuras, en todas sus posiciones.
5. Inicializar con espacios la variable jornada dentro de la matriz de estructuras, en todas sus posiciones.
6. Inicializar con 0 la variable duración dentro de la matriz de estructuras, en todas sus posiciones.
7. Inicializar con espacios la variable día dentro de la matriz de estructuras, en todas sus posiciones.
8. Inicializar con 0 la variable horainicio dentro de la matriz de estructuras, en todas sus posiciones.

7. Void imprimirhorario(struct actividad matriz[][5], int f, string semana[], string horas[]):
Funcion que permite imprimir la organizacion de la de la semana cultural como un horario seminal, mostrando los dias, horas y nombre de cada actividad programada.

Los parámetros necesarios son la matriz, ya que en esta se encuentra el nombre de la actividad que se va a imprimir en cada una de las posiciones, el entero f que indica el numero de filas de la matriz, el arreglo semana que tiene guardados los días de la semana como palabras para imprimirlos en el horario semanal, y el arreglo horas de tipo cadena que tiene guardado como string las horas correspondientes a los intervalos de media hora en las horas que se pueden realizar las actividades de la semana.

La función es de tipo Void por lo que no retorna ninguna variable a la función principal.

1. Hacer in ciclo con for para imprimir cada una de las posiciones del arreglo semana en la misma fila.
2. Hacer 2 un ciclos for para imprimir cada una de las posiciones del arreglo horas y los nombres de cada una de las posiciones de la matriz como se indica:
3. Hacer un if que haga que el arreglo horas se imprima solo cuando la columna es igual a 0, y de esta forma solo queden una vez antes de la primera columna de la matriz
4. Para que todas las posiciones de la matriz queden de la misma longitud es necesario crear 2 variables que guarden el tamaño de cada posición, y el tamaño de cada nombre de la matriz.
5. Se debe hacer la resta entre estos 2 valores y hacer un ciclo for que llene con espacios hasta completar el tamaño de la posición, que fue con el cual se inicializo la matriz.
6. De este modo, se imprime la variable nombre en la matriz de estructuras en cada una de las posiciones de la matriz, seguido de los espacios necesarios hasta completar el tamaño de la celda.

8. Int leerdatos(int tamano, struct actividad arregloact[], struct usuario u1, struct actividad matriz[][5], int horasn[], struct contador c1, int i): Función que permite ingresar una actividad al usuario, o eliminarla buscándola por el código.

Los parámetros necesarios para la función son la variable entera tamaño, que determina el tamaño del arreglo de las estructuras de actividades, el arreglo de actividades de tipo struct actividad porque contiene toda la información acerca de las actividades, la estructura u1 de tipo struct usuario donde se almacena la información del usuario necesaria para validar los requisitos del tipo de público de cada tipo de actividad, la matriz de tipo struct actividad que permite almacenar las actividades como un horario semanal el arreglo de horas de tipo entero para comparar los valores de las horas de actividades, con las horas del arreglo y ubicar las actividades en la matriz, la estructura c1 de tipo struct contador, que contiene los contadores para verificar que dos actividades del mismo tipo no queden registradas en la misma jornada del mismo día y por último la variable i que funciona como contador para saber cuántas actividades han sido agregadas hasta el momento. Esta variable se manda por parámetro, y al ingresar a la función, su valor aumenta a medida que se ingresan mas actividades, por lo que debe retornarse a la función principal, para que entre por parámetro cada vez que se use la función.

Serie de pasos:

1. Realizar un ciclo para mostrar las 3 opciones que puede escoger el usuario entre ingresar una actividad, eliminar o volver al menú principal.
2. La variable opcion2 es la que permite al usuario ingresar la opción que desea escoger
3. La opción 1 permite al usuario ingresar una actividad
4. Para esto, primero se pide el nombre de la actividad en la posición i, utilizando el arreglo de actividades (arregloact[i].nombre)
5. La variable “ver” sirve para determinar el seguimiento del código.
6. El programa pide el día de la actividad correspondiente
7. Si el día ingresado por el usuario no existe, la variable ver, seguirá siendo 0 para repetir esta acción, hasta que el día sea válido.
8. Para cada uno de los días existe una condicional para determinar el valor de las variables j y d1. La primera determina la ubicación de la actividad en la matriz, utilizando la correspondencia entre la posición del día en el arreglo semana, y las columnas de la matriz. La variable d1 se utiliza aquí para determinar de qué día es el contador que debe incrementarse en la estructura c1. La variable ver cambia su valor a 2 para no repetir el ciclo de pedir el día.
9. Después de pedir el día, el programa pide la jornada de la actividad. Aquí, se le da valor a la variable j1 utilizando la función “verificacionjornada” que le da a la variable el valor 0 si es mañana o 1 si es tarde. Esta variable es utilizada para saber a qué jornada debe corresponder el contador de la estructura c1. Además se utilizan las funciones contadormanana y contadortarde con las variables mananal, mananama, mananami, mananaju, mananavi y tardel, tardema, tardemi, tardeju, tardevi como parámetros, dependiendo del día de la actividad, para verificar cuantas actividades hay por jornada cada día. Después de esto, hay un condicional que verifica que el valor de ninguno de estos contadores supere los 3. Si lo supera, entonces aparece un mensaje que indica que no puede haber más de 3 actividades por jornada por día y no permite ingresar la última

actividad reduciendo el contador i en 1 y dándole el valor 0 a la variable ver para no entrar al siguiente ciclo que pide el tipo de cada actividad.

10. Si la variable “ver” es 2, es decir no sea incumplido ninguna de las reglas de la semana cultural, el programa pasa a pedir el tipo de la actividad. Al pedir el tipo, se le da valor a la variable t1 para determinar el contador que debe aumentarse de la estructura c1. Se hace aquí también la verificación del tipo de actividad con respecto al público permitido, utilizando la variable “ver”.
11. Para las culturales y deportivas, como son aptas para todo público, la variable ver pasa automáticamente a tomar el valor 1.
12. Para las actividades artísticas, musicales y plásticas, se verifica la edad del usuario. Si es mayor a 18, la variable ver toma el valor de 1 para continuar con el siguiente ciclo, si no, toma el valor de 0 para detener el programa y no permitir el ingreso de la actividad.
13. Para las actividades de la organización, se verifica que tipo de personal es el usuario, si es autorizado la variable ver pasa a ser 0, si no pasa a ser 1.
14. Si lo ingresado por el usuario no coincide con ningún tipo, se vuelve a pedir el tipo de actividad, hasta que sea válido.
15. Después de tener el día, la jornada y tipo se utiliza la estructura de contadores c1.d[d1].j[j1].tipo[t1]++ para verificar que dos actividades del mismo tipo no queden registradas en la misma jornada. Si esto sucede, la variable “ver” toma el valor de 0 para no continuar con el siguiente ciclo.
16. El programa pasa a asignar el código de la actividad con la variable i, que aumenta después de terminar de ingresar todos los datos de cada actividad.
17. Se pide la duración de la actividad, y la hora de inicio.
18. Utilizando estos datos, se determina la fila en la que se debe ubicar la actividad, en la matriz de estructuras. Con el arreglo de horasn, se busca el valor de la hora de inicio para que en la posición correspondiente al arreglo, se ubique la actividad en la fila de la matriz. La duración, se divide en 30 utilizando una variable llamada numcasillas, y se hace un ciclo que llene el número de filas que ocupa la actividad, a partir de la primera fila que se encontró. Primero se hace un ciclo para verificar que las casillas donde quiere meterse la actividad en la matriz están vacías.
19. Cuando ya se tienen todos los datos de la actividad se utiliza la función imprimirhorario para mostrar el horario con la actividad recién ingresada.

20. Cuando el usuario escoge la opción 2, se le da la posibilidad de eliminar una actividad buscándola por el código
21. Se pide primero el código de la actividad
22. Utilizando un algoritmo de búsqueda para localizar la posición de la actividad en el arreglo de estructuras buscando por el código.
23. Cuando se encuentra la posición en el arreglo, se inicializan cada uno de los datos de la estructura en la posición hallada
24. Se repite el anterior procedimiento, pero buscando la posición de la actividad en la matriz de estructuras.
25. Cuando ya se ha eliminado la actividad, se utiliza la función imprimirhorario para mostrar el horario con la actividad eliminada

26. Si el usuario escoge la opción 3, debe volver al menu principal.

9. Void consultarportipo(struct actividad arregloact[]): Función que imprime las actividades ordenadas por tipo y en orden ascendente de acuerdo a su código. Utiliza algoritmos de búsqueda para comparar el tipo de una de las posiciones del arreglo de actividades determinada, y compararlo con el tipo que se va a imprimir.

El único parámetro necesario es el arreglo de las actividades de tipo Struct actividad ya que es el que tiene almacenada toda la información sobre la actividad, para poder verificar cual es el tipo de la actividad.

La función es de tipo Void por lo que no retorna ninguna variable a la función principal.

Serie de pasos:

1. Imprimir el titulo del primer tipo de actividad, “Deportivas”.
2. Utilizar un algoritmo de búsqueda para comparar el tipo de cada una de las posiciones del arreglo con la palabra “deportiva”, e imprimir el código y nombre de las que son de este tipo justo después de este tipo de actividad.
3. Realizar lo mismo para los otros 5 tipos de actividades.

10 .Void consultarpornombre(struct actividad arregloact[]): Funcion que permite al usuario consultar toda la información acerca de una de las actividades programadas.

El único parámetro necesario es el arreglo de las actividades de tipo Struct actividad ya que es el que tiene almacenada toda la información sobre la actividad, para poder comparar el nombre de todas las posiciones del arreglo con el nombre de la actividad que se quiere buscar.

La función es de tipo Void por lo que no retorna ninguna variable a la función principal.

Serie de pasos:

1. Pedir al usuario que ingrese el nombre de la actividad que desea buscar
2. Usar un algoritmo de búsqueda para comparar todos los nombres de las posiciones del arreglo de actividades, con el nombre ingresado por el usuario.
3. Una vez encontrada la posición del arreglo donde se encuentra el nombre buscado, imprimir todos los datos guardados en esta posición del arreglo, incluyendo código, tipo, jornada, dia, duración y hora de inicio.
4. Si no se encuentra la actividad, decir al usuario que la actividad no esta ingresada en el sistema.

11. Void horariosdisponibles(struct actividad matriz[][5], string horas[], string semana[]): Función que permite al usuario ver los horarios disponibles, mostrando los días y horas correspondientes.

Los parámetros necesarios son la matriz de estructuras, el arreglo de horas y el arreglo de los días de la semana. El primero es necesario para buscar en las posiciones de la matriz, las estructuras que aun están vacías, el segundo para conocer las horas vacias, correspondientes a las filas de la matriz y el tercero para conocer los días disponibles, correspondientes a las columnas de la matriz.

La función es de tipo Void por lo que no retorna ninguna variable a la función principal.

Serie de pasos:

1. Buscar en la matriz de estructuras, los nombres que aun se encuentren vacios, comparándolos con espacios utilizando un algoritmo de búsqueda.
2. Una vez que se encuentra una posición de la matriz donde el nombre está vacío, se toma la posición de la columna, para hallar el dia correspondiente en el arreglo semana. (Lunes=0, Martes=1, Miercoles=2, Jueves=3, Viernes=4). De la misma forma se toma la fila para hallar las horas correspondientes en el arreglo de horas.
3. Se imprimen El dia y la hora encontrados

12. Int main (int argc, char *argv[]):Funcion donde se implementan todas las funciones del sistema.

Lo primero en la función, es llenar los arreglos semana, horas y horasn con los valores correspondientes a los días de la semana, y las horas para el horario de la semana cultural.

Después se declaran las variables enteras n e i que controlan el tamaño del arreglo de estructuras de actividades y la posición de la actividad en la que se va llenando respectivamente. La variable i empieza desde 5 porque ya hay 5 actividades ingresadas anteriormente.

Posteriormente se utilizan las funciones que inicializan la matriz y el arreglo de contadores, y la que ingresa las 5 actividades que deben estar previamente establecidas.

Después de pedir los datos al usuario, se muestra el menú con las opciones del programa. Para cada opción se llama la función correspondiente.

Para el caso 1 se llama la función **imprimirhorario** que muestra la organización de actividades como horario semanal, para el caso 2 la función **leerdatos** que permite ingresar o eliminar una actividad, para el caso 3 **consultarportipo** que organiza las actividades por tipo, para el caso 4, la función **consultarpornombre** que permite buscar las actividades por nombre para mostrar la información, y para el caso 5 **horariosdisponibles** que muestra los días y horas de la semana disponibles. Si el usuario escoge la opción 6, se detiene el funcionamiento del programa.