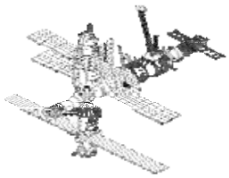

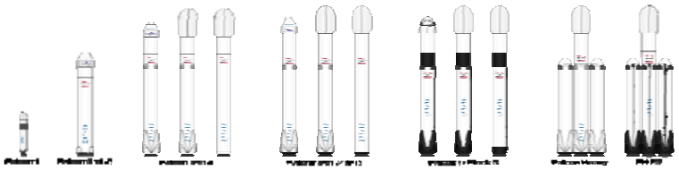

Nombre del proyecto:
ISS_SUPPLY

Autor: Diego Téllez Barrero
Profesor: D. Roger Clotet Martinez
Asignatura: Metodología de Programación
Código: 08GIIN
Grado en Ingeniería Informática

ESTRUCTURAS DE DATOS:

Para la versión 2 del programa se han convertido las variables a listas de listas, es decir, cada elemento de la lista es a su vez una lista que contiene todas las características de cada "entidad"

cohetes

Cada cohete es una lista que se compone por 2 elementos; 1º el nombre y 2º su carga útil.

peticiones

Cada petición se compone de 4 elementos en una nueva lista; 1º su id, 2º el tipo de suministro, 3º la cantidad del suministro y 4º el numero de días requerido.

lanzamientos

Cada lanzamiento compone una lista dentro de la lista general; sus elementos son 1º el nombre del cohete y 2º el número de días del lanzamiento.

cohetesEnRuta

Solo almacenaremos el nombre del cohete en este caso.

peticionAsignada

Cada elemento de la lista es otra lista que contiene la información de peticiones: 1ºIDmision, 2ºduracion, 3ºcohete y 4ºenviada (0 no 1 si)

peticionesOrdenada

Estructura utilizada para ordenar las peticiones sin alterar la lista principal. Se trabaja sobre ella modificando sus valores.

lanzamientoOrdenado

Ídem a peticionesOrdenada pero aplicado a lanzamientos.

MODULACIÓN:

El programa se ha separado en 3 ficheros, el principal y dos módulos que son:

menu.py -> se compone por una sola función:

imprimeMenu()->entero

declara y asigna los valores del menú a un diccionario local, "diccionarioMenu", posteriormente lo muestra al usuario y le pide que elija una opción que es retornada al programa principal

menuFicheros() -> entero

funciona del mismo modo que imprimeMenu pero para las opciones de ficheros.

error.py -> este módulo utiliza a su vez menu.py, además haremos uso de la librería time para controlar el tiempo de retorno al menú principal tras mostrar un error.

Tenemos en el la función errorComun(string)->int que recibe el texto __doc__ del error para mostrárselo al usuario antes de devolverle al menu principal para obtener una nueva opción. Además de mostrar el error este módulo gestiona la siguiente interacción del usuario con el menú principal.

Se ha definido una función específica para los errores en el tratamiento de ficheros. errorFichero(string)->int que funciona igual que errorComun().

FUNCIONES:

gestionaMenu(entero)

recibe la opción elegida por el usuario y con una estructura condicional anidada llama a las funciones pertinentes según la opción elegida.
En caso de elegir el usuario la opción 0 (Salir) se despide de el y cierra el programa.

gestionaFicheros(opt)

Gestiona el menú de ficheros, recibe la opción y nos dirige en consecuencia.

tipoCohete(lista)

recibe la variable global cohetes, pide al usuario los datos de un nuevo cohete (nombre y carga útil) y continua de forma recursiva pidiendo datos hasta que el usuario haya introducido todos lo que desee. Tras cada introducción de datos se muestra al usuario la información del sistema.
Se gestionan los errores para que el usuario introduzca el tipo de dato que deseemos.

peticionSuministro(lista)

Tiene un funcionamiento exactamente igual a la función anterior salvo que en este caso pide más datos específicos de cada petición (id, tipo, peso, tiempo). El resto es exactamente igual a tipoCohete. Se gestionan los errores para que el usuario introduzca el tipo de dato que deseemos.

lanzamientoDisponible(lista)

Tiene un funcionamiento exactamente igual a la función anterior salvo que en este caso pide más datos específicos de cada lanzamiento (id, tipo, peso, tiempo). El resto es exactamente igual a tipoCohete y peticionSuministo. Se gestionan los errores para que el usuario introduzca el tipo de dato que deseemos.

ordenaPeticiones(lista)

Mediante un algoritmo de ordenación de tipo burbuja y aprovechando la asignación simultanea que ofrece Python se ordenan las peticiones priorizando a las mas urgentes.

ordenaLanzamientos(lista)

Mediante un algoritmo de ordenación de tipo burbuja y aprovechando la asignación simultanea que ofrece Python se ordenan los lanzamientos priorizando la celeridad.

asignaPrioritarios(lista, lista, lista)

Esta es la función más compleja del software. Se va a encargar de recorrer todas las listas para ir asignando peticiones a lanzamientos.

simulacionDias(lista)

Esta función va a solicitar al usuario que introduzca el número de días que van a pasar. Una vez obtenido el número de días va a ir restándolos de los lanzamientos hasta que lleguen a 0 lo que significará que ha llegado a la estación y ha sido entregada. Se irá informando por consola de lo que va ocurriendo así como un resumen general del sistema.

infoSistema(lista,lista,lista)

Muestra al usuario un resumen de cohetes, peticiones y lanzamientos en ese momento.

guardaDatos(lista, lista, lista)

Esta función se encarga de escribir el fichero con el título dado por el usuario. En caso de no existir datos de cohetes, peticiones o lanzamientos nos muestra un mensaje sugiriendo la introducción de los mismos antes de guardar el fichero. Se dan más detalles a continuación en el apartado “GESTIÓN DE FICHEROS”.

cargaDatos()

Es la función encargada de recuperar los datos del fichero e incorporarlos al sistema; previamente se convierten los datos según nuestras estructuras de datos. En caso de no existir el fichero dado por el usuario mostrará un error. Se dan más detalles del funcionamiento interno en el apartado “GESTIÓN DE FICHEROS”.

threeSecDelayToMainMenu()

Es una función simple que nos da una cuenta atrás antes de redirigirnos al menú principal.

GESTIÓN DE FICHEROS

Para mantener el estado del sistema cuando el usuario cierre el programa vamos a usar ficheros de texto plano sin formato. El usuario tanto a la hora de guardar datos como al cargarlos nos dará el nombre del fichero, internamente el programa asigna en ambos casos la extensión .txt que no debe ser introducida por el usuario.

Para poder hacer un backup del sistema, el usuario ha debido introducir datos de cohetes, peticiones y lanzamientos. En caso contrario mostrará un mensaje de que faltan datos para el backup.

Usaremos un solo fichero para todo y conoceremos la estructura del mismo, es decir que sabremos en todo momento cual es la información de cada línea, para así facilitar su tratamiento a la hora de cargar. La estructura general del fichero será la siguiente:

```
COHETES:
3
['c1CK', 88.5]
['c2CK', 0.0]
['c10CK', 1000.0]

PETICIONES:
4
['p1O', ' Oxigeno', 10.0, 15, 1]
['p2O', ' Oxigeno', 200.0, 30, 1]
['p3R', ' Repuestos', 1.5, 15, 1]
['p4M', ' Modulo', 10000.0, 60, 0]

LANZAMIENTOS:
3
['c1CK', 10]
['c2CK', 20]
['c10CK', 50]

PETICIONES ASIGNADAS:
3
['p1O', 10, 'c1CK', 0]
['p3R', 10, 'c1CK', 0]
['p2O', 20, 'c2CK', 0]
```

Primero vemos el nombre del elemento guardado, después el primer dato tras el título del elemento es el número de elementos totales guardados que usaremos de forma interna para cargar la información posteriormente y tras este número vemos en cada línea cada uno de los elementos guardados.

En el caso de las “PETICIONES ASIGNADAS” solo se guardarán en el archivo si se ha hecho previamente una asignación y del mismo modo solo se cargarán posteriormente si ha sido guardado en el fichero. De este modo podemos asegurar que el usuario realice un backup del sistema después de haber hecho una asignación y sea capaz de recuperarlo para hacer una simulación posterior.

Los datos se escriben en el fichero con la estructura de una lista de Python, posteriormente haciendo uso de la función replace() desecharemos lo que no necesitamos para solo quedarnos con los datos que estarán separados por “,” y podremos separar usando la función split(), además convertiremos el tipo de dato según nos interese ya que usando este tipo de ficheros .txt el tipo de dato de la lectura es siempre string.

El usuario debe ser consciente de que la información introducida en el sistema pero no guardada será “machacada” si se carga un fichero anterior.

El usuario debe ser consciente de que la información contenida en el fichero será “machacada” si decide usar el mismo fichero para un backup nuevo con otra información.

MAIN

El bloque principal del programa da la bienvenida al usuario y llama al menú (imprimeMenu) que retorna la opción elegida por el usuario y se la envía en la llamada a gestionaMenu(entero). En principio todo el programa se desarrollará en las siguientes funciones de las que irá yendo de unas a otras sin volver a Main.