Projecte d'Informàtica
The Organizer

Pol Mesalles Ripoll i Arnau Singla Manau

Gener 2015
Universitat Politècnica de Catalunya · BarcelonaTech

El programa

The Organizer és un *software* distribuït de manera lliure i gratuïta i que permet a l'usuari organitzar i consultar informació i fotografies dels viatges que ha realitzat mitjançant una *interface* via Terminal.

1 `alpha.cpp` i `beta.cpp`

En primer lloc, cal mencionar que hem dividit el projecte en dues versions del programa. La primera, `alpha.cpp`, ofereix les funcions bàsiques que demanava l'enunciat, juntament amb unes quantes variacions (ús dels fitxers, edició dels viatges, etc.), però ho fa amb una estructura basada en `arrays` i, per tant, el nombre màxim d'elements de qualsevol d'aquestes variables queda limitat per una constant global `N = 10`.

En canvi, la segona versió del programa, `beta.cpp`, es basa en una estructura de `vectors`, de manera que el nombre màxim d'elements de cada un dels «`arrays`» és ara pràcticament il·limitat. A més, aquesta versió afegeix nova funcionalitat al programa, com és la possibilitat d'indexar tots els arxius d'imatge dins d'una carpeta donada, a més de ser capaç d'obrir un navegador web i mostrar la ruta dels viatges visitats mitjançant Google Maps.

2 Estructura de dades

L'estructura de dades del nostre programa es basa principalment en `arrays`, `structs` i funcions d'aquestes variables, com es mostra en la Figura 1. Cada apartat d'informació d'un viatge `Trip` està definit amb una sèrie de variables i tipus creats per nosaltres corresponents als respectius camps d'informació que requereixen els viatges. A continuació exposarem totes les estructures de dades i la informació que conté cadascuna.

A nivell general, tenim l'estructura dels viatges de tipus `Trip`. Aquesta està formada per l'`array` (o `vector`) `countries` de tipus `Country`, una altra `taula` `albums` de tipus `Album` i tot un seguit d'`integers` que són la data, tant d'inici (`start`) com d'arribada (`end`) i, a més, en el cas de la versió `alpha.cpp`, el nombre de països (`nCountries`) i d'àlbums (`nAlbums`) que contindran les respectives `taules` i una petita valoració de tot el viatge (`score`), un `enter` entre 1 i 5.

Més concretament, quant a l'estructura de dades del tipus `Country`, que conté informació per a cada país determinat, està formada per l'`array` (o `vector`) `cities` amb les ciutats visitades del país concret, el nom del país, un comentari del país i, per últim, en cas de la versió bàsica, el nombre de ciutats de la `taula` de ciutats (`nCities`). Les tres primeres variables són de tipus `string`, mentre que l'últim és de tipus `int`.

La última estructura de dades és la d'`Album`, que conté informació sobre cada àlbum (o carpeta) de fotografies (digitals), i aquesta està formada per una `taula` de `tags` (etiquetes per a les fotografies), el `path` (ruta d'accés a la carpeta —digital— que conté les imatges), a més del nombre de tags (`nTags`) que contindrà la `taula`. Aquest nombre és de tipus `int`, i la resta són `strings`.

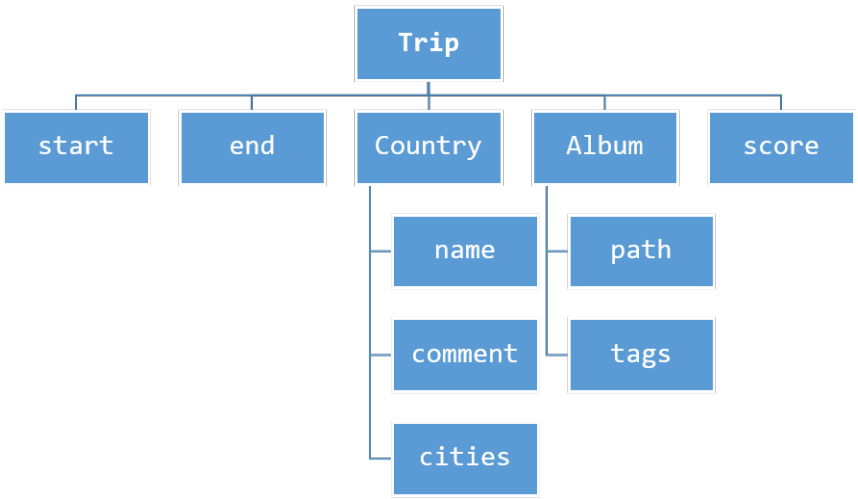

Figura 1: Esquema de l'estructura de dades del programa

Finalment, només ens queda parlar de les variables que implementem al `main()` per recollir tota aquesta informació. En primer lloc, òbviament, tenim el `vector` o `array` de viatges: `Trip trips`, que en el fons és l'estructura que recull totes les dades del programa.

A més a més, però, per millorar l'eficiència dels processos d'ordenació i visualització, fem ús del que anomenem `hash`, que consisteix en dues `taules` —`int dates, scores`— en què cada cel·la representa la posició d'un viatge concret en l'`array Trip trips`, però dins de cada `hash`, aquestes estan ordenades (d'anterior a més recent data d'inici en el cas de `dates`, i de menor a major valoració en el cas d'`scores`).

3 Estructura del programa

The Organazier ha de realitzar quatre funcions principals: afegir nous viatges, modificar viatges existents, esborrar una sèrie de viatges i, per últim, consultar informació sobre qualsevol viatge. Per complir amb aquesta funcionalitat, hem programat una sèrie de funcions per a cada una d'aquestes funcions, tot aprofitant l'estructura de dades explicada anteriorment.

Després, també és necessari dissenyar molts menús de diferent tipus, donant sempre diverses opcions a l'usuari i, per tant, també hem creat un seguit de funcions que despleguen i controlen cadascun d'aquests menús que hi ha a dins de cada una de les opcions del programa principal.

A continuació, descriurem en més detall les funcions principals del programa, agrupades segons la «categoria» a la qual pertanyen:

1. Les funcions ***OS-specific*** són aquelles que varien segons el sistema operatiu en el qual es compila el programa. Com veurem a l'últim apartat, per assegurar la màxima compatibilitat multi-plataforma, al principi del programa es defineixen unes *preprocessor directives* mitjançant l'ús dels condicionals `#ifdef`, `#elif` i `#endif` que comproven si hi ha definit algun dels macros `_WIN32`, `__linux__` o `__APPLE__`, de manera que el *software* serà prou intel·ligent per carregar les funcions corresponents a cada sistema operatiu.
   En aquest apartat trobem funcions del tipus `clearUp()`, que no fan més que una crida a `system()`, tot passant com a paràmetre la comanda corresponent a cada sistema operatiu.

2. Les funcions ***helper*** són aquelles que s'usen indistintament al llarg de tot el programa. En la seva majoria, són «accions» que permeten mostrar els diversos menús i els diferents errors a l'usuari, així la repetició innecessària de codi.
   A més, però, també hi trobem funcions com `findByDate()`, que retorna la posició d'un

viatge dins de l'array donada una data; o funcions tipus `date2int()` i `int2date()`, que transformen —de tal manera que siguin compatibles amb versions de C++ anteriors a C++ 11— un `string` de la forma `yyyy/mm/dd` (data introduïda per l'usuari) en un `int` de la forma `yyyymmdd`.

3. Les funcions ***import / export*** són les encarregades de llegir i escriure la informació en un arxiu `data.dat` que actua de «base de dades».
   Concretament, hi ha dues funcions (`readData()` i `writeData()`) que es basen en `streams` de fitxers.

4. Les funcions ***add / edit*** agrupen tot el conjunt de funcions usades a l'hora d'afegir o editar viatges. Tot i que la seva forma difereix lleugerament entre les dues versions del programa degut al funcionament de `vector.push_back()`, la idea principal és que hi ha una funció diferent per afegir dates, països, àlbums de fotografies i valoracions, de manera que podem reutilitzar part del codi d'afegir nous viatges a l'hora d'editar-los.

5. Les funcions ***remove*** inclouen tot el necessari per esborrar un viatge determinat (`erase()`) i, a continuació, actualitzar els valors del `hash` necessaris (`findHash()`, `moveHash()` i `modHash()`).

6. Les funcions ***sorting*** són les encarregades d'ordenar els `hash` de la forma més eficient possible. Per aconseguir aquest objectiu, fem ús de l'algorisme *quicksort* implementat en dues funcions per a cada tipus de dades a ordenar (`quickSortDates()`, que ordena els viatges del més antic al més recent, tot aprofitant que la data d'inici d'un viatge determinat és un enter únic[1] de la forma `yyyymmdd`; i `quickSortScores()`, que ordena els viatges segons la seva valoració en ordre creixent).
   A més, incorporem una tercera funció, `sortScores`, que s'assegura que cada grup de viatges dins d'una valoració determinada també estiguin ordenats per data.
   Cal tenir en compte, com hem mencionat anteriorment, que el nostre programa mai ordena la `taula` sencera de `Trip trips`, doncs aquest seria un procés molt llarg, sinó que ordena els `arrays` tipus `hash`. D'aquesta manera, el programa és molt més eficient i, consegüentment més ràpid, que és una qualitat molt valorada pels usuaris.

7. Les funcions ***show*** «imprimeixen» en pantalla la informació sobre un viatge concret, que agafen com a paràmetre.
   En aquesta categoria hem situat les funcions `printBasic()`, que imprimeix únicament la informació més rellevant d'un viatge, i que s'utilitza en desplegar llistes de diversos viatges, juntament amb `printAll()`, que mostra tota la informació sobre un viatge determinat introduït per l'usuari.

8. Finalment, la funció `main()`, tot un estàndard en llenguatges derivats de C, on s'implementa tota l'estructura de menús mitjançant una sèrie de bucles, a més de la funcionalitat que demana a l'usuari els camps de cerca per consultar un viatge, i tots els missatges per evitar errors i avisar a l'usuari de que no està introduint la informació de manera correcta.

4 Funcionament del *software*

Així doncs, amb totes aquestes funcions, el programa principal queda de la següent forma:

1. Primer, demana el nom a l'usuari per així fer el programa més personalitzat. En cas que no sigui el primer cop que s'obre, donarà la benvinguda a l'usuari amb el nom ja conegut.

2. Desplega el menú principal amb les quatre opcions principals i una cinquena per sortir del programa.

[1] Cal tenir en compte que el programa no pot admetre diferents viatges amb una mateixa data d'inici, doncs està adreçat a ser utilitzat per una única persona, i a més fa ús de l'`start` com a identificador únic de cada viatge a l'hora de demanar a l'usuari que esculli un viatge determinat.

3. La primera opció és la d'afegir un viatge, i si es selecciona, el programa va preguntant cada apartat d'informació que ha de contenir tot viatge. Un cop acabat, pregunta si l'usuari vol continuar afegint viatges, el guarda en la variable, i retorna al menú principal.

4. La segona opció és la d'editar un viatge ja existent. Demana que introdueixi la data de partida del viatge que vol editar, després mostra un menú de tots els apartats que té un viatge per tal que l'usuari n'escolli un i després permet que l'editi. Per acabar, pregunta si l'usuari vol continuar editant aquell viatge, el guarda en la variable, i retorna al menú principal.

5. La tercera opció és la d'esborrar un viatge. Demana que s'introdueixi la data de partida del viatge que es vol esborrar, l'esborra i pregunta a l'usuari si vol esborrar algun altre viatge. En cas negatiu, retorna al menú principal.

6. La quarta opció és la de consultar algun viatge introduït. Desplega un nou menú en el qual dóna a triar a l'usuari si vol que li ensenyi tots els viatges (ordenats de diferents maneres), els d'un país, els d'un any (o interval), o els que tenen una mateixa puntuació. El programa mostra els corresponents viatges, i pregunta a l'usuari si vol més detalls d'algun viatge. En cas de resposta afirmativa, després d'introduir la data d'inici demanada, en la versió `beta.cpp`, el programa mostrarà encara un submenú que permetrà escollir entre consultar tota la informació del viatge, veure les fotografies de cada carpeta i obrir un navegador web amb els mapes dels països visitats. Per altra banda, si la resposta és negativa, l'usuari pot tornar al menú principal.

7. La cinquena opció és la de sortir del programa. Abans de sortir el programa, pregunta si es volen guardar els canvis fets durant la sessió, i en cas afirmatiu sobreescriu les dades del fitxer `data.dat` amb els canvis realitzats.

5 Compatibilitat i problemes coneguts

Tant la versió `alpha.cpp` com la versió `beta.cpp` s'han testat —i comprovat que funcionen correctament— en els següents entorns:

1. **GNU/Linux** (concretament, Ubuntu 14.04 LTS) mitjançant el compilador per excel·lència de C/C++, el `gcc` (o `g++` com a comanda).

2. **Mac OS** (versió 10.9.5) mitjançant el mateix compilador que en Linux, el `gcc`.

3. **Microsoft Windows 8.1** (i Pro) mitjançant el compilador MinGW (que no és més que un *port* per Windows de GCC) en el *software* de desenvolupament Dev-C++ i Visual Studio 2013 Professional.

Abans d'acabar, mencionar alguns dels problemes coneguts en les versions distribuïdes actualment:

1. El `cmd` de Windows és incapaç de mostrar alguns dels caràcters *unicode* usats en el programa (com és el cas del caràcter `'\u21b3'`). Aquest problema és inexistent en el terminal `bash` inclòs en la majoria de distribucions de Linux.
   Tot i així, aquest problema es pot solucionar si s'executa el `cmd` amb l'opció `chcp 65001`.

2. El programa no comprova que la data d'acabada del viatge sigui superior (posterior) a la d'arribada actualment, tot i que això no comporta cap error de funcionament.