Programming Project

Terry Marris January 2010

7 Results

The results are derived from the runner's times. First, we deal with just one result. Then we deal with the file of results.

7.1 result.h

A result has a runner number, a name, a race time and a category. A category is either half marathon or full marathon.

```
/* result.h: interface */
/* ver 1.0 20Jan2011 */

#ifndef RESULT_H
#define RESULT_H

#include "entry.h"

typedef struct RESULT {
  char number[NUMBERSIZE];
  char name[NAMESIZE];
  char time[10];
  Category category;
} RESULT, *Result;

/* resultError: reports fatal errors, halts program */
void resultError(const char *report);

/* resultWarning: reports warnings */
int resultWarning(const char *report);

/* newResult: returns a new result */
Result newResult();

/* copyResult: returns a duplicate of the given result */
Result copyResult(const Result result);

/* setResultNumber: amends result number */
Result setResultNumber(const char *number, const Result result);

/* setResultName: amends result name */
Result setResultName(const char *name, const Result result);

/* setResultTime: amends result race time */
Result setResultTime(const char *time, const Result result);

/* setResultCategory: amends result category */
Result setResultCategory(Category category, const Result result);
```

```
/* setResult: from the given number, name, time and category */
Result setResult(const char *number, const char *name,
                 const char *time, Category category);

/* resultNumber: returns result number */
char *resultNumber(const Result result);

/* resultName: returns result name */
char *resultName(const Result result);

/* resultTime: returns result time */
char * resultTime(const Result result);

/* resultCategory: returns result category */
Category resultCategory(const Result result);

/* printResultTitle: displays results title */
int printResultsTitle(const char *title);

/* printResultHeadings: displays headings */
int printResultHeadings();

/* printResult: displays result */
int printResult(const Result result);

#endif
```

7.2 result.c

```
/* result.c: a result has a number, a category, a name and a time */
/* ver 1.0 20Jan2011 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "utility.h"
#include "entry.h"
#include "result.h"

/* resultError: reports fatal errors, halts program */
void resultError(const char *report)
{
  printf("\n%s\n", report);
  exit(EXIT_FAILURE);
}

/* resultWarning: reports warnings */
int resultWarning(const char *report)
{
  printf("%s\n", report);
  getStr("Press return to continue ...", 5);
  return 0;
}
```

```
/* newResult: returns a new blank result */
Result newResult()
{
  Result result;

  result = malloc(sizeof(struct RESULT));
  if (result == NULL)
    resultError("newResult: out of memory");
  strcpy(result->number, "0");
  strcpy(result->name, "");
  strcpy(result->time, "00:00:00");
  result->category = unknownCategory;
  return result;
}

/* copyResult: returns a duplicate of the given result */
Result copyResult(const Result result)
{
  Result r;

  if (result == NULL) {
    resultWarning("copyResult: cannot copy a null result");
    return (Result)NULL;
  }
  r = newResult();
  strcpy(r->number, result->number);
  strcpy(r->name, result->name);
  strcpy(r->time, result->time);
  r->category = result->category;
  return r;
}

/* setResultNumber: amends result number */
Result setResultNumber(const char *number, const Result result)
{
  Result r;

  if (result == NULL) {
    resultWarning("setResultNumber: result is NULL");
    return (Result)NULL;
  }
  r = copyResult(result);
  strcpy(r->number, number);
  return r;
}

/* setResultName: amends result name */
Result setResultName(const char *name, const Result result)
{
  Result r;

  if (result == NULL) {
    resultWarning("setResultName: result is NULL");
    return (Result)NULL;
  }
  r = copyResult(result);
  strcpy(r->name, name);
  return r;
}
```

```
/* setResultTime: amends result race time */
Result setResultTime(const char *time, const Result result)
{
  Result r;

  if (result == NULL) {
    resultWarning("setResultName: result is NULL");
    return (Result)NULL;
  }
  r = copyResult(result);
  strcpy(r->time, time);
  return r;
}

/* setResultCategory: amends result category */
Result setResultCategory(Category category, const Result result)
{
  Result r;

  if (result == NULL) {
    resultWarning("setResultName: result is NULL");
    return (Result)NULL;
  }
  r = copyResult(result);
  r->category = result->category;
  return r;
}

/* setResult: from the given number, name, time and category */
Result setResult(const char *number, const char *name,
                 const char *time, Category category)
{
  Result result = newResult();

  result = setResultNumber(number, result);
  result = setResultName(name, result);
  result = setResultTime(time, result);
  result = setResultCategory(category, result);
  return result;
}

/* resultNumber: returns result number */
char *resultNumber(const Result result)
{
  return result->number;
}

/* resultName: returns result name */
char *resultName(const Result result)
{
  return result->name;
}

/* resultTime: returns result time */
char *resultTime(const Result result)
{
  return result->time;
}
```

```
/* resultCategory: returns result category */
Category resultCategory(const Result result)
{
  return result->category;
}

/* printResultTitle: displays results title */
int printResultsTitle(const char *title)
{
  printf("%s\n", title);
  return 0;
}

/* printResultHeadings: displays headings */
int printResultHeadings(const char *title)
{
  printf("%s\n", title);
  printf("%-6s %-15s %s\n", "Number", "Name", "Time");
  return 0;
}

/* printResult: displays result */
int printResult(const Result result)
{
  printf("%6s %-15s %s\n", resultNumber(result),
          resultName(result), resultTime(result));
  return 0;
}
```

result.c is tested in *testresults.c*.

7.3 results.h

results.h is refreshingly small.

```
/* results.h: interface for results.c */
/* ver 1.0 21Jan2011 */

#ifndef RESULTS_H
#define RESULTS_H

/* resultsFileError: reports fatal errors, halts program */
void resultsFileError(const char *report);

/* resultsFileWarning: displays warning */
int resultsFileWarning(const char *report);

/* makeResultsfiles: constructs results file from
   runner and entry files */
int makeResultsFiles(const char *entryFileName,
                     const char *runnerFileName,
                     const char *halfMFileName,
                     const char *fullMFileName);

/* printResultsFile: displays results for either half or
   full marathon */
int printResultsFile(const char *fileName, const char *title);

#endif
```

7.4 results.c

```
/* results.c: creates results file for both
   half and full marathon */
/* ver 1.0 21Jan2011 */

#include <stdio.h>
#include <stdlib.h>
#include "utility.h"
#include "entrybase.h"
#include "runner.h"
#include "result.h"

/* resultsFileError: reports fatal errors, halts program */
void resultsFileError(const char *report)
{
  printf("\n%s\n", report);
  exit(EXIT_FAILURE);
}

/* resultsFileWarning: displays warning */
int resultsFileWarning(const char *report)
{
  printf("\n%s\n", report);
  getStr("Press return to continue ... ", 5);
  return 0;
}

/* NOTE: essentially the same error handling functions have been
   written several times now.  Perhaps we need modules for
   error handling, file handling, time handling, and application
   for a better modular structure?
*/

/* makeResultsfiles: constructs results file from
   runner and entry files */
int makeResultsFiles(const char *entryFileName,
                     const char *runnerFileName,
                     const char *halfMFileName,
                     const char *fullMFileName)
/* basically, we retrieve records from the runners file and
separate the contents into two files. We need the name from the
entries file! */
{
  FILE *runnerFile;
  FILE *halfMFile;
  FILE *fullMFile;
  RUNNER runner;
  RESULT result;
  char *number = malloc(NUMBERSIZE);
  char *name = malloc(NAMESIZE);
  char *time = malloc(10);
  Category category;

  if (number == NULL || name == NULL)
    resultsFileError("makeResultsFile: out of memory");
  runnerFile = fopen(runnerFileName, "rb");
  halfMFile = fopen(halfMFileName, "wb");
  fullMFile = fopen(fullMFileName, "wb");
```

```
  fread(&runner, sizeof(RUNNER), 1, runnerFile);
  while (!feof(runnerFile)) {
    number = runnerNumber(&runner);
    name = getNameFromFile(number, entryFileName);
    time = raceTimeToString(runnerRaceTime(&runner));
    category = categoryFromFile(number, entryFileName);
    result = *setResult(number, name, time, category);
    if (category == half)
      fwrite(&result, sizeof(RESULT), 1, halfMFile);
    else if (category == full)
      fwrite(&result, sizeof(RESULT), 1, fullMFile);
    fread(&runner, sizeof(RUNNER), 1, runnerFile);
  }
  fclose(runnerFile);
  fclose(halfMFile);
  fclose(fullMFile);
  return 0;
}

/* printResultsFile: displays results for either half or
   full marathon */
int printResultsFile(const char *fileName, const char *title)
{
  FILE *file;
  RESULT result;

  file = fopen(fileName, "rb");
  if (file == NULL)
    resultsFileError("printResultsFile: cannot open file");
  printResultHeadings(title);
  fread(&result, sizeof(RESULT), 1, file);
  while (!feof(file)) {
    printResult(&result);
    fread(&result, sizeof(RESULT), 1, file);
  }
  fclose(file);
  return 0;
}
```

7.5 testresults.c

The filenames are again hardwired into the program.

```
/* testresults.c: tests results file for both
   half and full marathon */
/* ver 1.0 21Jan2011 */

#include <stdio.h>
#include "results.h"

int main()
{
  char *entryFileName = "entries2010.dat";
  char *runnerFileName = "run2010.dat";
  char *halfMFileName = "hmresults2010.dat";
  char *fullMFileName = "fmresults2010.dat";
```

```
  makeResultsFiles(entryFileName, runnerFileName,
                   halfMFileName, fullMFileName);
  printResultsFile(halfMFileName, "Half Marathon Results 2010");
  printf("\n");
  printResultsFile(fullMFileName, "Full Marathon Results 2010");
  printf("\n");

  return 0;
}
```