ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»**

Московский институт электроники и математики им. А.Н. Тихонова

Департамент прикладной математики

Отчёт
по лабораторной работе №8
по курсу «Алгоритмизация и программирование»
Задание № 13

ФИО студента	Номер группы	Дата
Кейер Александр Петрович	БПМ-231	3 марта 2024 г.

Москва, 2023

1 Задание (вариант № 13)

1. Данные должны храниться в бинарном файле.
2. Каждая операция с данными базы должна быть реализована как функция или набор функций.
3. Выбор и запуск требуемого режима (действия) осуществляется через меню.
4. Реализовать следующие функции обработки данных:
   - добавление записи в файл;
   - удаление заданной записи из файла по порядковому номеру записи;
   - поиск записей по заданному пользователем (любому) полю структуры;
   - редактирование (изменение) заданной записи;
   - вывод на экран содержимого файла в табличном виде.
5. Структуру (в соответствии с вариантом) определять в отдельном заголовочном файле. С помощью директив условной компиляции определить два способа ввода исходных данных в файл: пользователем с потока ввода и из заранее заполненного массива.

Данные о членах сборной команды по футболу: ФИО футболиста, название клуба, амплуа (вратарь, защитник, полузащитник, нападающий), возраст, количество проведенных за сборную матчей, количество забитых за сборную мячей.

2 Структура заголовочного файла

```
#define fieldLength 64
#define fieldSize fieldLength * sizeof(char)
#define entryLength 6

struct footballerType {
  char fullName[fieldLength];
  char clubName[fieldLength];
  char role[fieldLength];
  int age;
  int numberOfGames;
  int numberOfGoals;
};

```

Решение

```
#pragma once

#include "football.h" // Headers file.

#include <stdio.h> // Input/output library.
#include <string.h> // String library.
#include <stdlib.h> // Mmory allocation library.
#include <io.h> // File managing library.

#define FILE_NAME "footballer.bin"
#define INPUT_TYPE 'a'

// Function printing horizontal line.
void printHr(int length) {
  for (int j = 0; j < length; j++) {
    printf("- ");
  }
  printf("\n");
}

// Function printing table header.
void printTableHeader() {
  printHr(65);
  printf("%2s|%30s|%30s|%30s|%10s|%10s|%10s|\n", "ID", "
FULL NAME", "CLUB NAME", "ROLE", "AGE", "GAMES", "GOALS");
```

```
25        printHr(65);
26      }
27
28      // Function printing footballer entry.
29      void printFootballer(struct footballerType entry, int id)
        {
30        printf("%2d|%30s|%30s|%30s|%10d|%10d|%10d|\n", id,
        entry.fullName, entry.clubName, entry.role, entry.age,
        entry.numberOfGames, entry.numberOfGoals);
31        printHr(65);
32      }
33
34      // Function getting footballer id by user.
35      void getEntryIdByUser(int *entryId) {
36        printf("Enter footballer id: ");
37
38        fflush(stdin);
39        scanf("%d", entryId);
40      }
41
42      // Function getting footballer full name by user.
43      void getFootballerFullNameByUser(char fullName[
        fieldLength]) {
44        printf("Enter full name: ");
45
46        fflush(stdin);
47        fgets(fullName, fieldLength, stdin);
48        fullName[strcspn(fullName, "\n")] = 0;
49      }
50
51      // Function getting footballer club name by user.
52      void getFootballerClubNameByUser(char clubName[
        fieldLength]) {
53        printf("Enter club name: ");
54
55        fflush(stdin);
56        fgets(clubName, fieldLength, stdin);
57        clubName[strcspn(clubName, "\n")] = 0;
58      }
59
60      // Function getting footballer role by user.
61      void getFootballerRoleByUser(char role[fieldLength]) {
62        printf("Enter role: ");
63
64        fflush(stdin);
```

```
65        fgets(role, fieldLength, stdin);
66        role[strcspn(role, "\n")] = 0;
67      }
68
69      // Function getting footballer age by user.
70      void getFootballerAgeByUser(int *age) {
71        printf("Enter age: ");
72
73        fflush(stdin);
74        scanf("%d", age);
75      }
76
77      // Function getting footballer games by user.
78      void getFootballerNumberOfGamesByUser(int *numberOfGames)
        {
79        printf("Enter number of games: ");
80
81        fflush(stdin);
82        scanf("%d", numberOfGames);
83      }
84
85      // Function getting footballer goals by user.
86      void getFootballerNumberOfGoalsByUser(int *numberOfGoals)
        {
87        printf("Enter count of goals: ");
88
89        fflush(stdin);
90        scanf("%d", numberOfGoals);
91      }
92
93      // Function getting footballer by user.
94      void getFootballerByUser(struct footballerType *
      footballer) {
95        getFootballerFullNameByUser(footballer->fullName);
96        getFootballerClubNameByUser(footballer->clubName);
97        getFootballerRoleByUser(footballer->role);
98        getFootballerAgeByUser(&(footballer->age));
99        getFootballerNumberOfGamesByUser(&(footballer->
      numberOfGames));
100       getFootballerNumberOfGoalsByUser(&(footballer->
      numberOfGoals));
101     }
102
103     // Function getting file length.
104     int getFileLength(FILE *f) {
```

```
105       fseek(f, 0, SEEK_END);
106       int fileLength = ftell(f) / sizeof(struct
      footballerType);
107       fseek(f, 0, SEEK_SET);
108
109       return fileLength;
110     }
111
112     // Function creating binary file.
113     void createBinaryFile(char *fileName, struct
      footballerType entry) {
114       FILE *f = fopen(fileName, "wb");
115
116       if (!f) {
117         printf("File with name %s could not be open for write
      .\n", fileName);
118         return;
119       }
120
121       fwrite(&entry, sizeof(struct footballerType), 1, f);
122       fclose(f);
123     }
124
125     // Function appendind entry in binary file.
126     void appendEntry(char *fileName, struct footballerType
      entry) {
127       FILE *f = fopen(fileName, "ab");
128
129       if (!f) {
130         printf("File with name %s could not be open for write
      .\n", fileName);
131         return;
132       }
133
134       fwrite(&entry, sizeof(struct footballerType), 1, f);
135       fclose(f);
136     }
137
138     // Function deleting entry by id from binary file.
139     void deleteEntryById(char *fileName, int entryId) {
140       FILE *f = fopen(fileName, "r+b");
141
142       if (!f) {
143         printf("File with name %s could not be open for
      changes.\n", fileName);
```

```
144         return;
145       }
146
147       int fileLength = getFileLength(f);
148
149       struct footballerType tmpEntry;
150       struct footballerType deletedEntry;
151
152       if (entryId >= fileLength || entryId < 0) {
153         printf("Couldn't delete entry.\n");
154         return;
155       }
156
157       fseek(f, entryId * sizeof(struct footballerType),
    SEEK_SET); // Move pointer on correct position.
158       fread(&deletedEntry, sizeof(struct footballerType), 1,
    f); // Saving deleted entry.
159
160       // Making a shift.
161       for (int i = entryId + 1; i < fileLength; i++) {
162         fread(&tmpEntry, sizeof(struct footballerType), 1, f)
    ;
163         fseek(f, (i - 1) * sizeof(struct footballerType),
    SEEK_SET);
164         fwrite(&tmpEntry, sizeof(struct footballerType), 1, f
    );
165         fseek(f, (i + 1) * sizeof(struct footballerType),
    SEEK_SET);
166       }
167
168       _chsize(_fileno(f), (fileLength - 1) * sizeof(struct
    footballerType)); // Clip file.
169       fclose(f);
170       printf("Successfully deleted '%s' from binary file.\n",
     deletedEntry.fullName);
171     }
172
173     // Find all entries by full name function.
174     void findAllEntriesByFullName(char *fileName, char
    fieldValue[fieldLength]) {
175       FILE *f = fopen(fileName, "rb");
176
177       if (!f) {
178         printf("File with name %s could not be open for
    reading.\n", fileName);
```

```
179         return;
180       }
181
182       printf("\nFound all entries by full name '%s':\n",
    fieldValue);
183       printTableHeader();
184
185       struct footballerType entry;
186       int fileLength = getFileLength(f);
187       int entriesCount = 0;
188
189       for (int i = 0; i < fileLength; i++) {
190         fread(&entry, sizeof(struct footballerType), 1, f);
191
192         if (!strcmp(entry.fullName, fieldValue)) {
193           printFootballer(entry, i);
194           entriesCount++;
195         }
196       }
197
198       if (entriesCount == 0) {
199         printf("No entries were found.\n");
200       }
201
202       printf("\n");
203       fclose(f);
204     }
205
206     // Find all entries by club name function.
207     void findAllEntriesByClubName(char *fileName, char
    fieldValue[fieldLength]) {
208       FILE *f = fopen(fileName, "rb");
209
210       if (!f) {
211         printf("File with name %s could not be open for
    reading.\n", fileName);
212         return;
213       }
214
215       printf("\nFound all entries by club name '%s':\n",
    fieldValue);
216       printTableHeader();
217
218       struct footballerType entry;
219       int fileLength = getFileLength(f);
```

```
220       int entriesCount = 0;
221
222       for (int i = 0; i < fileLength; i++) {
223         fread(&entry, sizeof(struct footballerType), 1, f);
224
225         if (!strcmp(entry.clubName, fieldValue)) {
226           printFootballer(entry, i);
227           entriesCount++;
228         }
229       }
230
231       if (entriesCount == 0) {
232         printf("No entries were found.\n");
233       }
234
235       printf("\n");
236       fclose(f);
237     }
238
239     // Find all entries by role function.
240     void findAllEntriesByRole(char *fileName, char fieldValue
    [fieldLength]) {
241       FILE *f = fopen(fileName, "rb");
242
243       if (!f) {
244         printf("File with name %s could not be open for
    reading.\n", fileName);
245         return;
246       }
247
248       printf("\nFound all entries by role '%s':\n",
    fieldValue);
249       printTableHeader();
250
251       struct footballerType entry;
252       int fileLength = getFileLength(f);
253       int entriesCount = 0;
254
255       for (int i = 0; i < fileLength; i++) {
256         fread(&entry, sizeof(struct footballerType), 1, f);
257
258         if (!strcmp(entry.role, fieldValue)) {
259           printFootballer(entry, i);
260           entriesCount++;
261         }
```

```
262       }
263
264       if (entriesCount == 0) {
265         printf("No entries were found.\n");
266       }
267
268       printf("\n");
269       fclose(f);
270     }
271
272     // Find all entries by age function.
273     void findAllEntriesByAge(char *fileName, int fieldValue)
    {
274       FILE *f = fopen(fileName, "rb");
275
276       if (!f) {
277         printf("File with name %s could not be open for
    reading.\n", fileName);
278         return;
279       }
280
281       printf("\nFound all entries by age '%d':\n", fieldValue
    );
282       printTableHeader();
283
284       struct footballerType entry;
285       int fileLength = getFileLength(f);
286       int entriesCount = 0;
287
288       for (int i = 0; i < fileLength; i++) {
289         fread(&entry, sizeof(struct footballerType), 1, f);
290
291         if (entry.age == fieldValue) {
292           printFootballer(entry, i);
293           entriesCount++;
294         }
295       }
296
297       if (entriesCount == 0) {
298         printf("No entries were found.\n");
299       }
300
301       printf("\n");
302       fclose(f);
303     }
```

```
304
305     // Find all entries by games function.
306     void findAllEntriesByNumberOfGames(char *fileName, int
    fieldValue) {
307       FILE *f = fopen(fileName, "rb");
308
309       if (!f) {
310         printf("File with name %s could not be open for
    reading.\n", fileName);
311         return;
312       }
313
314       printf("\nFound all entries by number of games '%d':\n"
    , fieldValue);
315       printTableHeader();
316
317       struct footballerType entry;
318       int fileLength = getFileLength(f);
319       int entriesCount = 0;
320
321       for (int i = 0; i < fileLength; i++) {
322         fread(&entry, sizeof(struct footballerType), 1, f);
323
324         if (entry.numberOfGames == fieldValue) {
325           printFootballer(entry, i);
326           entriesCount++;
327         }
328       }
329
330       if (entriesCount == 0) {
331         printf("No entries were found.\n");
332       }
333
334       printf("\n");
335       fclose(f);
336     }
337
338     // Find all entries by goals function.
339     void findAllEntriesByNumberOfGoals(char *fileName, int
    fieldValue) {
340       FILE *f = fopen(fileName, "rb");
341
342       if (!f) {
343         printf("File with name %s could not be open for
    reading.\n", fileName);
```

```
344         return;
345       }
346
347       printf("\nFound all entries by number of goals '%d':\n"
    , fieldValue);
348       printTableHeader();
349
350       struct footballerType entry;
351       int fileLength = getFileLength(f);
352       int entriesCount = 0;
353
354       for (int i = 0; i < fileLength; i++) {
355         fread(&entry, sizeof(struct footballerType), 1, f);
356
357         if (entry.numberOfGoals == fieldValue) {
358           printFootballer(entry, i);
359           entriesCount++;
360         }
361       }
362
363       if (entriesCount == 0) {
364         printf("No entries were found.\n");
365       }
366
367       fclose(f);
368       printf("\n");
369     }
370
371     // Function updating entry by id.
372     void updateEntryById(char *fileName, int entryId, struct
    footballerType newEntry) {
373       FILE *f = fopen(fileName, "r+b");
374
375       if (!f) {
376         printf("File with name %s could not be open for
    changes.\n", fileName);
377         return;
378       }
379
380       if (entryId < 0 || entryId >= getFileLength(f)) {
381         printf("Footballer with id '%d' doesn't exist.",
    entryId);
382         return;
383       }
384
```

```
385       // Move pointer on correct position.
386       fseek(f, entryId * sizeof(struct footballerType),
    SEEK_SET);
387       fwrite(&newEntry, sizeof(struct footballerType), 1, f);
388
389       fclose(f);
390       printf("\nSuccessfully updated entry '%d'.\n", entryId)
    ;
391     }
392
393     // Function printing binary file.
394     void printBinaryFile(char *fileName) {
395       FILE *f = fopen(fileName, "rb");
396
397       if (!f) {
398         printf("File isn't valid for printing.\n");
399         return;
400       }
401
402       printTableHeader();
403
404       struct footballerType entry;
405
406       fread(&entry, sizeof(struct footballerType), 1, f);
407
408       int i = 0;
409
410       while (!feof(f)) {
411         printFootballer(entry, i++);
412
413         fread(&entry, sizeof(struct footballerType), 1, f);
414       }
415
416       fclose(f);
417     }
418
419     // Function finding very old footballer with a lot of
    goals.
420     void findVeryOldWithALotOfGoals(char *fileName) {
421       FILE *f = fopen(fileName, "rb");
422
423       if (!f) {
424         printf("File with name %s could not be open for
    reading.\n", fileName);
425         return;
```

```
426       }
427
428       printf("\nFound old guy with a lot of goals.\n");
429       printTableHeader();
430
431       struct footballerType entry;
432       struct footballerType outEntry;
433
434       int fileLength = getFileLength(f);
435       int entriesCount = 0;
436       int maximumGoals = 0;
437       int maximumAge = 0;
438
439       // Finding maximum goals number.
440       for (int i = 0; i < fileLength; i++) {
441         fread(&entry, sizeof(struct footballerType), 1, f);
442
443         if (entry.numberOfGoals > maximumGoals) {
444           maximumGoals = entry.numberOfGames;
445           entriesCount++;
446         }
447       }
448
449       fseek(f, 0, SEEK_SET);
450
451       // Finding very old people with maximum goals number.
452       for (int i = 0; i < fileLength; i++) {
453         fread(&entry, sizeof(struct footballerType), 1, f);
454
455         if (entry.numberOfGoals == maximumGoals && entry.age
    > maximumAge) {
456           maximumAge = entry.age;
457           outEntry = entry;
458           entriesCount++;
459         }
460       }
461
462       if (entriesCount == 0) {
463         printf("No entries were found.\n");
464       }
465
466       printFootballer(outEntry, 0);
467
468       fclose(f);
469       printf("\n");
```

```
470     }
471
472     // Function printing find all operations codes list.
473     void printFindAllOperationsList() {
474       printf("\n");
475       printf("%30s %3s", "full name:", "1\n");
476       printf("%30s %3s", "club name:", "2\n");
477       printf("%30s %3s", "role:", "3\n");
478       printf("%30s %3s", "age:", "4\n");
479       printf("%30s %3s", "number of goals:", "5\n");
480       printf("%30s %3s", "number of games:", "6\n");
481       printf("%30s %3s", "very old with maximum goals:", "7\n
    ");
482       printf("\n\n");
483     }
484
485     // Function starting find all entries menu.
486     void startFindAllEntriesMenu() {
487       printf("\n");
488       printFindAllOperationsList();
489       printf("Enter correct find all code: ");
490
491       int operationCode = 0;
492
493       fflush(stdin);
494       scanf("%d", &operationCode);
495
496       if (operationCode <= 0 || operationCode > 7) {
497         startFindAllEntriesMenu();
498         return;
499       }
500
501       switch (operationCode) {
502         case 1: {
503           char fullName[fieldLength];
504           getFootballerFullNameByUser(fullName);
505           findAllEntriesByFullName(FILE_NAME, fullName);
506           break;
507         }
508
509         case 2: {
510           char clubName[fieldLength];
511           getFootballerClubNameByUser(clubName);
512           findAllEntriesByClubName(FILE_NAME, clubName);
513           break;
```

```
514         }
515
516         case 3: {
517           char role[fieldLength];
518           getFootballerRoleByUser(role);
519           findAllEntriesByRole(FILE_NAME, role);
520           break;
521         }
522
523         case 4: {
524           int age;
525           getFootballerAgeByUser(&age);
526           findAllEntriesByAge(FILE_NAME, age);
527           break;
528         }
529
530         case 5: {
531           int numberOfGoals;
532           getFootballerNumberOfGoalsByUser(&numberOfGoals);
533           findAllEntriesByNumberOfGoals(FILE_NAME,
    numberOfGoals);
534           break;
535         }
536
537         case 6: {
538           int numberOfGames;
539           getFootballerNumberOfGamesByUser(&numberOfGames);
540           findAllEntriesByNumberOfGames(FILE_NAME,
    numberOfGames);
541           break;
542         }
543
544         case 7: {
545           findVeryOldWithALotOfGoals(FILE_NAME);
546           break;
547         }
548       }
549     }
550
551     // Function printing main operations codes list.
552     void printMainOperationsList() {
553       printf("\n");
554       printf("%30s %3s", "append entry:", "1\n");
555       printf("%30s %3s", "delete entry:", "2\n");
556       printf("%30s %3s", "find all entries by:", "3\n");
```

```
557       printf("%30s %3s", "update entry:", "4\n");
558       printf("%30s %3s", "print database:", "5\n");
559       printf("%30s %3s", "exit program:", "6\n");
560       printf("\n\n");
561     }
562
563     // Function starting main menu.
564     void startMenu() {
565       printf("\n");
566       printMainOperationsList();
567       printf("Enter correct operation code: ");
568
569       int operationCode = 0;
570
571       fflush(stdin);
572       scanf("%d", &operationCode);
573
574       switch (operationCode) {
575         case 1: {
576           struct footballerType footballer;
577
578           getFootballerByUser(&footballer);
579           appendEntry(FILE_NAME, footballer);
580           break;
581         }
582
583         case 2: {
584           int entryId;
585
586           getEntryIdByUser(&entryId);
587
588           deleteEntryById(FILE_NAME, entryId);
589           break;
590         }
591
592         case 3: {
593           startFindAllEntriesMenu();
594           break;
595         }
596
597         case 4: {
598           int entryId;
599           struct footballerType footballer;
600
601           getEntryIdByUser(&entryId);
```

```
602             getFootballerByUser(&footballer);
603
604             updateEntryById(FILE_NAME, entryId, footballer);
605             break;
606           }
607
608           case 5: {
609             printBinaryFile(FILE_NAME);
610             break;
611           }
612
613           case 6: {
614             return;
615           }
616         }
617
618         printf("\n");
619         startMenu();
620       }
621
622       int main() {
623         // Use user input.
624         #if INPUT_TYPE == 'u'
625         printf("User input.\n");
626
627         struct footballerType footballer;
628
629         getFootballerByUser(&footballer);
630
631         // Use array input.
632         #else
633         printf("Input from array.\n");
634
635         struct footballerType footballer = {
636           .fullName="Keyer Alexander Petrovich",
637           .clubName="BAM231",
638           .role="Goalkeeper",
639           .age=1,
640           .numberOfGames=2,
641           .numberOfGoals=3,
642         };
643         #endif
644
645         createBinaryFile(FILE_NAME, footballer);
646
```

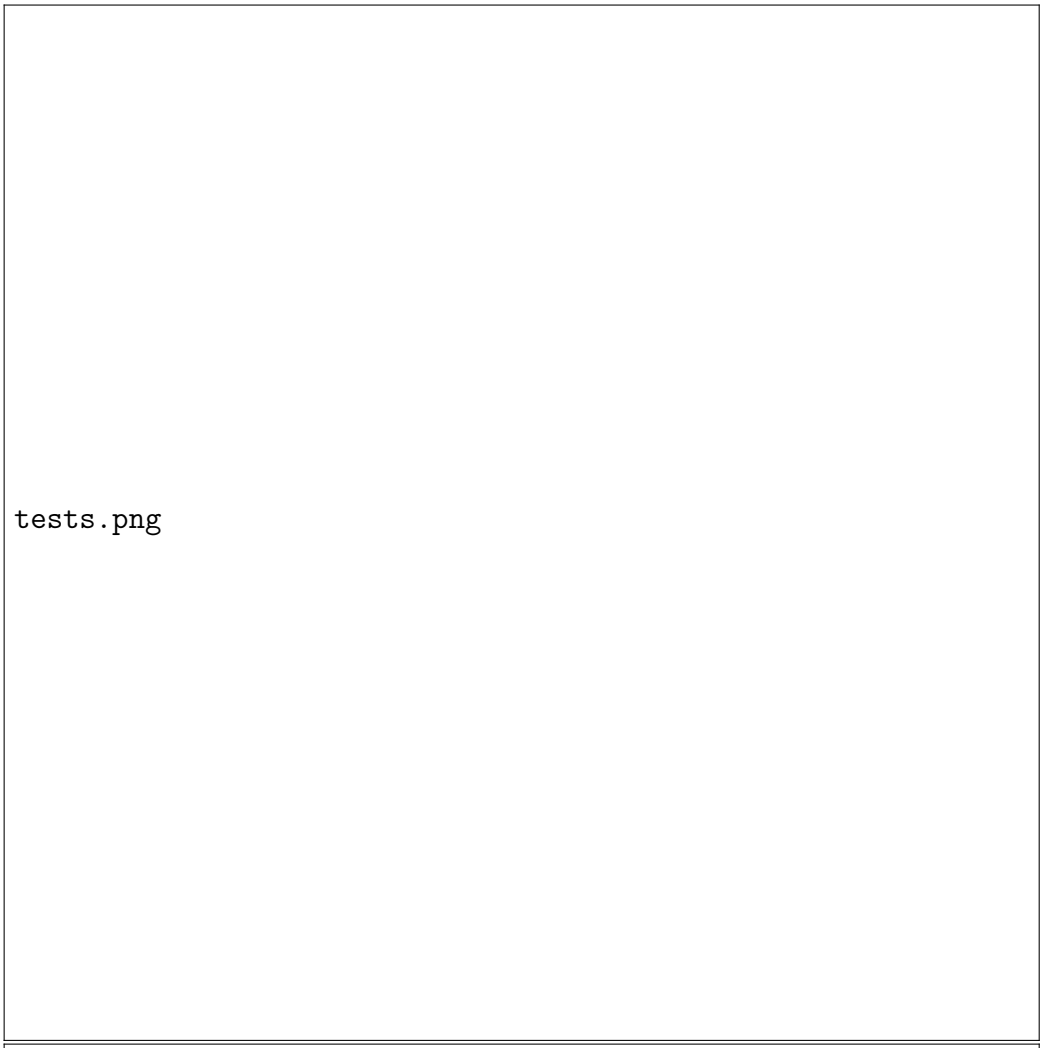
```
647         struct footballerType alex = {
648           .fullName="Alex",
649           .clubName="Innopolis",
650           .role="Guardian",
651           .age=4,
652           .numberOfGames=5,
653           .numberOfGoals=3,
654         };
655 
656         struct footballerType igor = {
657           .fullName="Igor",
658           .clubName="Innopolis",
659           .role="Goalkeeper",
660           .age=1,
661           .numberOfGames=5,
662           .numberOfGoals=3,
663         };
664 
665         struct footballerType sasha = {
666           .fullName="Sasha",
667           .clubName="ITMO",
668           .role="Guardian",
669           .age=2,
670           .numberOfGames=5,
671           .numberOfGoals=1,
672         };
673 
674         struct footballerType sasha1 = {
675           .fullName="Sasha",
676           .clubName="Innopolis",
677           .role="Forward",
678           .age=5,
679           .numberOfGames=2,
680           .numberOfGoals=3,
681         };
682 
683         struct footballerType sasha2 = {
684           .fullName="Sasha",
685           .clubName="BAM234",
686           .role="Forward",
687           .age=1,
688           .numberOfGames=3,
689           .numberOfGoals=2,
690         };
691 
```

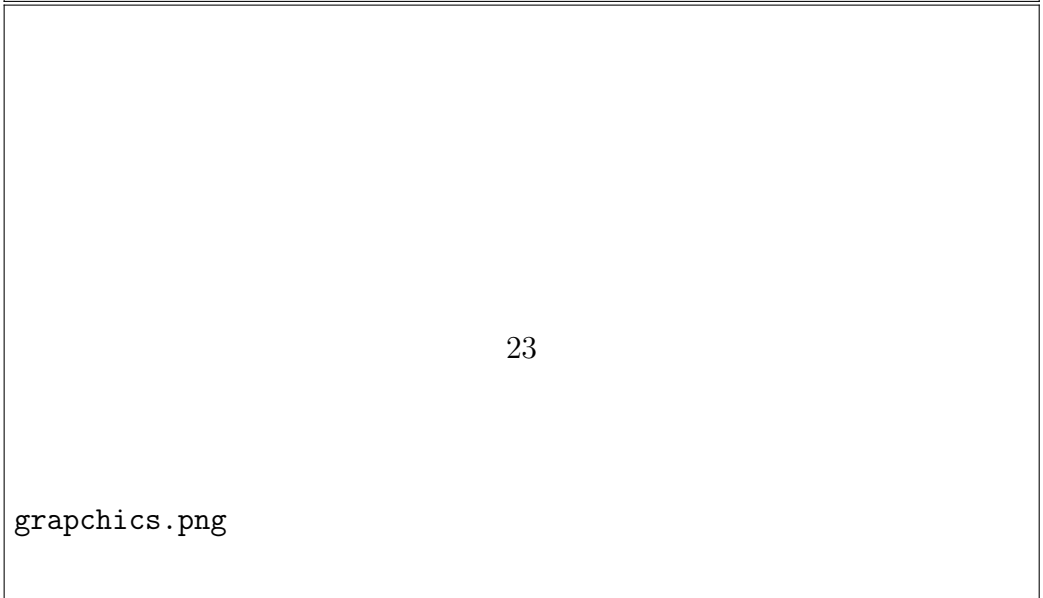
```
692         struct footballerType alex1 = {
693           .fullName="Alex",
694           .clubName="BAM233",
695           .role="Forward",
696           .age=2,
697           .numberOfGames=4,
698           .numberOfGoals=5,
699         };
700 
701         struct footballerType kostya = {
702           .fullName="Kostya",
703           .clubName="BAM234",
704           .role="Goalkeeper",
705           .age=1,
706           .numberOfGames=5,
707           .numberOfGoals=3,
708         };
709 
710         struct footballerType kostya1 = {
711           .fullName="Kostya",
712           .clubName="BAM233",
713           .role="Guardian",
714           .age=2,
715           .numberOfGames=2,
716           .numberOfGoals=2,
717         };
718 
719         struct footballerType nadya = {
720           .fullName="Nadya",
721           .clubName="BAM231",
722           .role="Guardian",
723           .age=3,
724           .numberOfGames=3,
725           .numberOfGoals=3,
726         };
727 
728         struct footballerType sonya = {
729           .fullName="Sonya",
730           .clubName="BAM232",
731           .role="Goalkeeper",
732           .age=1,
733           .numberOfGames=5,
734           .numberOfGoals=4,
735         };
736 
```

```
737         // Initializing database.
738         appendEntry(FILE_NAME, sasha2);
739         appendEntry(FILE_NAME, alex);
740         appendEntry(FILE_NAME, sasha);
741         appendEntry(FILE_NAME, kostya1);
742         appendEntry(FILE_NAME, sonya);
743         appendEntry(FILE_NAME, igor);
744         appendEntry(FILE_NAME, sasha1);
745         appendEntry(FILE_NAME, alex1);
746         appendEntry(FILE_NAME, kostya);
747         appendEntry(FILE_NAME, nadya);
748 
749         startMenu();
750 
751         return 0;
752     }
753 
```

3 Тесты

3.1 Тест 1 (графики и оценка скорости)





По времени эффективнее всех оказывается сортировка слиянием. Дальше по порядку идут: сортировка вставками, сортировка шейкером, пузырьковая сортировка. Естественно, по затратам памяти самой неэффективной оказывается сортировка слиянием, но затраты по памяти полностью компенсируются временными.

3.2 Тест 2

```
Enter correct operation code: 4
Enter footballer id: 1
Enter full name: Sanya
Enter club name: BAM234
Enter role: Forward
Enter age: 18
Enter number of games: 18
Enter count of goals: 21

Successfully updated entry '1'.
```

Успешное обновление записи по айдишнику.

3.3 Тест 3

```
                full name:  1
                club name:  2
                     role:  3
                      age:  4
            number of goals:  5
            number of games:  6
  very old with maximum goals:  7


Enter correct find all code: 5
Enter count of goals: 3

Found all entries by number of goals '3':
```

ID	FULL NAME	CLUB NAME	ROLE	AGE	GAMES	GOALS
0	Keyer Alexander Petrovich	BAM231	Goalkeeper	1	2	3
2	Alex	Innopolis	Guardian	4	5	3
6	Igor	Innopolis	Goalkeeper	1	5	3
7	Sasha	Innopolis	Forward	5	2	3
9	Kostya	BAM234	Goalkeeper	1	5	3
10	Nadya	BAM231	Guardian	3	3	3

Успешный поиск записи по какому-либо полю.

3.4 Тест 4

```
Enter correct operation code: 2
Enter footballer id: 7
Successfully deleted 'Sasha' from binary file.
```

Успешное удаление записи по айдишнику.

3.5 Тест 5

```
Enter correct operation code: 1
Enter full name: Kostya
Enter club name: BAM231
Enter role: Goalkeeper
Enter age: 19
Enter number of games: 190123
Enter count of goals: 12314



               append entry:  1
               delete entry:  2
         find all entries by:  3
               update entry:  4
             print database:  5
               exit program:  6


Enter correct operation code: 5
```

ID	FULL NAME	CLUB NAME	ROLE	AGE	GAMES	GOALS
0	Keyer Alexander Petrovich	BAM231	Goalkeeper	1	2	3
1	Sanya	BAM234	Forward	18	18	21
2	Alex	Innopolis	Guardian	4	5	3
3	Sasha	ITMO	Guardian	2	5	1
4	Kostya	BAM233	Guardian	2	2	2
5	Sonya	BAM232	Goalkeeper	1	5	4
6	Igor	Innopolis	Goalkeeper	1	5	3
7	Alex	BAM233	Forward	2	4	5
8	Kostya	BAM234	Goalkeeper	1	5	3
9	Nadya	BAM231	Guardian	3	3	3
10	Kostya	BAM231	Goalkeeper	19	190123	12314

Успешное добавление записи в конец бинарного файла.