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

Факультет социально-экономических и компьютерных наук

Самостоятельная работа №1. Задание №1

по дисциплине «Теоретические основы информатики»

Раздел 4 - Сортировка и поиск

Выполнил: студент Яцишин Л.С. (индивидуальная работа), уч. группа ПСАПР-25-2

Пермь, 2025

Содержание

1 Задание

Опишите массив записей с разработанной структурой.
Разработайте процедуры (функции):

1. Ввода данных с клавиатуры для описанного массива записей (без сортировки).
2. Вывода данных (записей из массива) на экран в порядке их ввода в массив. Формат вывода данных построчно (каждая запись в отдельной строке) на экран определите самостоятельно.
3. Вывода данных (записей из массива) в текстовый файл в порядке их ввода в массив. Формат вывода данных построчно в файл (каждая запись - в отдельной строке) определите самостоятельно - выберите разделители между полями (атрибутами) записей. При выводе записей в файл используйте два режима (по выбору пользователя):
   a) создание нового файла;
   b) дополнение новыми записями существующего файла (записи добавляются в конец файла).
4. Ввода данных в массив (записей) из текстового файла выбранного формата.
5. Построения индекса массива по значениям заданного атрибута записей в массиве.
6. Построения индекса массива по значениям вычисленного поля на основе атрибутов записей в массиве.
7. Вывода элементов массива (записей) на экран в порядке сортировки в индексах (отдельная функция по каждому индексу):
   a) по возрастанию значений ключевых полей;
   b) по убыванию значений ключевых полей.
8. Поиска элементов по значениям ключевых атрибутов с использованием бинарного поиска (поиска делением пополам) в построенных индексах. В случае, если элемент не найден, должно быть выведено соответствующее сообщение. Если элемент с заданным для поиска значением ключа найден в индексе, должны быть выведены значения всех атрибутов из соответствующей записи в массиве введённых данных. Разработайте два варианта процедур поиска по индексам: по одному атрибуту - рекурсивную функцию поиска, по другому - итерационную.
9. Редактирования записей в массиве с возможностью изменения значений атрибутов, по которым выполнена индексация записей массива. При этом соответствующие изменения вносятся и в индексы, т.е. в индексе может измениться порядок записей, чтобы сохранить порядок сортировки при изменении ключевых значений.
10. Удаления записи с заданным значением ключевого атрибута из массива с соответствующим изменением индекса. Удаление может быть выполнено путём внесения пометки об удалении записи в специальное поле (помеченные записи могут быть восстановлены специальной функцией), при этом физическое удаление помеченных для удаления записей выполняется отдельной функцией.

Отчёт о выполнении задания 1 самостоятельной работы - файл, содержащий описания решений (описание структуры данных (записей в массиве) с комментариями по выбору типов для каждого атрибута; структуры индексов с описанием полей; формата записей в текстовом файле; алгоритмов для всех функций в виде блок-схем; тестов для всех функций; кода разработанной программы на C++),

2 Предметная область

Я выбрал учёт личных расходов. Каждая запись - одна трата: числовой Id, категория, короткое описание, дата покупки и сумма. Также хранится пометка удаления, чтобы можно

было восстановить запись или полностью удалить ее позже.

3 Структура данных и форматы

Основной массив Expense (30 элементов):
- Id (unsigned int) - числовой ключ
- Category, Note (string) - строки (без ; и пробелов)
- Day, Month, Year (unsigned int) - дата
- Amount (double) - сумма
- Deleted (bool) - флаг удаления

Индексы:
- CatIndexElement{CatKey, RecN} - категория, сортировка вставками с барьером
- DateIndexElement{DateKey, RecN} - дата, ключ вычисляется как $YYYY \times 10000 + MM \times 100 + DD$, сортировка выбором

Строятся только по не удалённым.

Формат файла: id;category;note;day;month;year;amount;deleted

Ограничения и проверки

- Id: целое > 0, читается как unsigned int
- Category: без символов ; и пробелов.
- Note: без символов ; и пробелов
- Дата: Month 1..12, Day в пределах DaysInMonth[Month], Year - без ограничений
- Amount: double > 0
- Deleted: логическая метка, сохраняется в файле как 0/1

4 Алгоритмы

Ввод/вывод

Реализованр: ввод одной записи с проверками, вывод массива в порядке ввода, запись в файл (новый/append), чтение из файла до переполнения массива.

В любой момент при отправлении символа "#" просиходит выход в основное меню.

Сортировка индексов

- Вставками с барьером (категория): сложность $O(n^2)$
- Выбором (дата): сложность $O(n^2)$.

Поиск

- По категории: рекурсивный бинарный поиск в отсортированном CatIndex.
- По дате: итеративный бинарный поиск в DateIndex.
- Вывод по индексам: возрастающий/убывающий проход по индексному массиву.
- Вывод списка категорий: линейный проход по индексному массиву.

Редактирование и удаление

Редактирование: запись ищется по Id, считываются новые поля (с проверками) и индексы пересобираются, для обновления порядка ключей.

Удаление/восстановление: ставится или снимается флаг Deleted по Id либо по категории (все вхождения), затем индексы пересобираются.

Физическое удаление: записи без Deleted копируются в начало массива, счётчик уменьшается (помеченные уходят за пределы RecCount и могут быть перезаписаны), индексы строятся заново.

5 Тестирование

Ручные проверки:

1. Загрузка sample-data.txt: вывод по вводу - 7 записей.
2. Построение индексов: вывод по категориям в порядке возрастания и убывания и вывод по датам в порядке возрастания и убывания.
3. Поиск: категория books (нашло запись 303), дата 13.05.2025 (нашло 304).
4. Удаление по категории transport: запись 302 помечена, в индексе не выводится; восстановление по категории возвращает.
5. Физическое удаление после нескольких логических удалений: размер массива уменьшается, индексы перестроены.
6. Проверка дубликатов Id: добавлен Id 303 повторно, поиск по Id выводит первую найденную запись, защиты нет.
7. Проверка повторной записи файла: чтение sample-data.txt, затем сохранение в тот же файл - строки продублировались (ожидаемо, защиты нет).
8. Проверка строк: ввод строки с пробелом или ';' отклоняется, корректный ввод проходит.
9. Проверка месяца(февраль): ввод даты 29.02.2025 отклонён, 28.02.2025 принят.
10. Проверка пересборки индексов: после редактирования без пункта «построить индексы» поиск по индексу даёт старые данные; после пересборки - новые.
11. Неверный ввод чисел/дат: при вводе текста вместо числа выводится ошибка, ввод повторяется до корректного значения.

Пример файла данных (sample-data.txt)

```
301;food;lunch;10;5;2025;820;0
302;transport;subway;11;5;2025;140;0
303;books;study;12;5;2025;1900;0
304;sport;gym;13;5;2025;3100;0
305;utilities;electricity;15;5;2025;4200;0
306;health;dentist;18;5;2025;5600;0
307;gifts;birthday;20;5;2025;2300;0
```

6 Недостатки

1. Нет високосных лет: в феврале всегда 28 дней, поиск по датам в високосный год будет неточным.
2. Нет проверки уникальности Id: можно добавить дубликаты.
3. Если читать из файла, а потом сразу добавлять в него, записи сохранятся второй раз (нет защиты от дубликатов при экспорте).

4. Поиск по категории и по дате выдает только первое совпадение в индексе, остальные записи с тем же ключом не выводятся.
5. Строки Category/Note вводятся без пробелов и без символа ;.
6. Индексы нужно пересобирать через меню после ввода/чтения, иначе поиск/вывод по индексам работают на старых данных. (оставленно специально)

7 Код программы

```
#include <iostream>
#include <string>
#include <fstream>
#include <stdio.h>
#include <limits>
#include <cstdlib>

using namespace std;

const int MaxRec = 30; // Количество записей в массиве
const int DaysInMonth[13] = { 0,31,28,31,30,31,30,31,31,30,31,30,31 }; // Календарь для проверки дат

struct Expense
{
    unsigned int Id;   // числовой ключ
    string Category;   // краткая категория
    string Note;       // короткое описание
    unsigned int Day;
    unsigned int Month;
    unsigned int Year;
    double Amount;     // сумма > 0
    bool Deleted;      // пометка удаления
};

struct CatIndexElement
{
    string CatKey;
    int RecN;
};

struct DateIndexElement
{
    int DateKey;
    int RecN;
};

Expense ExpArr[MaxRec];
int RecCount = 0;

CatIndexElement CatIndex[MaxRec + 1]; // с барьером
int CatIndexCount = 0;

DateIndexElement DateIndexArr[MaxRec + 1]; // с барьером
```

```
44  int DateIndexCount = 0;
45
46  void ClearInput()
47  {
48      cin.clear();
49      cin.ignore(numeric_limits<streamsize>::max(), '\n');
50      printf("Неверный ввод\n");
51  }
52
53  bool ReadUInt(const char *prompt, unsigned int &val)
54  {
55      string tmp;
56      while (true)
57      {
58          printf("%s (или # - выход): ", prompt);
59          if (!(cin >> tmp))
60          {
61              ClearInput();
62              continue;
63          }
64          if (tmp == "#")
65          {
66              printf("Отмена\n");
67              return false;
68          }
69          try
70          {
71              // stoul переводит строку в число; затем приводим к unsigned int
72              val = static_cast<unsigned int>(stoul(tmp));
73              return true;
74          }
75          catch (...)
76          {
77              printf("Неверный ввод\n");
78          }
79      }
80  }
81
82  bool ReadDouble(const char *prompt, double &val)
83  {
84      string tmp;
85      while (true)
86      {
87          printf("%s (или # - выход): ", prompt);
88          if (!(cin >> tmp))
89          {
90              ClearInput();
91              continue;
92          }
93          if (tmp == "#")
94          {
95              printf("Отмена\n");
96              return false;
97          }
```

```
98          try
99          {
100             val = stod(tmp);
101             return true;
102         }
103         catch (...)
104         {
105             printf("Неверный ввод\n");
106         }
107     }
108 }
109
110 bool ReadPosDouble(const char *prompt, double &val)
111 {
112     // Только положительное значение
113     while (true)
114     {
115         if (!ReadDouble(prompt, val)) return false;
116         if (val > 0) return true;
117         printf("Значение должно быть > 0\n");
118     }
119 }
120
121 bool ReadString(const char *prompt, string &val)
122 {
123     while (true)
124     {
125         printf("%s (или # - выход): ", prompt);
126         getline(cin >> ws, val);
127         if (val == "#")
128         {
129             printf("Отмена\n");
130             return false;
131         }
132         if (val.find(';') != string::npos || val.find(' ') != string::npos)
133         {
134             printf("Символ ';' и пробелы запрещены\n");
135             continue;
136         }
137         return true;
138     }
139 }
140
141 bool ReadDate(unsigned int &y, unsigned int &m, unsigned int &d)
142 {
143     if (!ReadUInt("Год", y)) return false; // сначала год
144     while (true)
145     {
146         if (!ReadUInt("Месяц", m)) return false; // проверяем 1..12
147         if ((m >= 1) && (m <= 12)) break;
148         printf("Месяц должен быть 1..12\n");
149     }
150     while (true)
151     {
```

```
152         if (!ReadUInt("День", d)) return false; // проверяем дни в месяце
153         int maxd = DaysInMonth[m];
154         if ((d >= 1) && (d <= static_cast<unsigned int>(maxd))) break;
155         printf("В этом месяце %d дней\n", maxd);
156     }
157     return true;
158 }
159 
160 int CalcDateKey(Expense E)
161 {
162     // Вычисляем ключ даты вида YYYYMMDD
163     return (E.Year * 10000 + E.Month * 100 + E.Day);
164 }
165 
166 string Key(CatIndexElement IE)
167 {
168     return (IE.CatKey);
169 }
170 
171 int Key(DateIndexElement IE)
172 {
173     return (IE.DateKey);
174 }
175 
176 int OrderCatByInsert(CatIndexElement *A, int AN)
177 {
178     int C = 0; // пересылки
179     CatIndexElement x;
180     int i, j;
181 
182     for (i = 2; i < AN; i++)
183     {
184         x = A[i];
185         A[0] = x;
186         j = i - 1;
187         while (Key(x) < Key(A[j]))
188         {
189             C++;
190             A[j + 1] = A[j];
191             j = j - 1;
192         }
193         A[j + 1] = x;
194     }
195     return (C);
196 }
197 
198 int OrderDateBySelection(DateIndexElement A[], int AN)
199 {
200     int C = 0; // пересылки
201     DateIndexElement x;
202     int i, j;
203     int MinEl;
204     int MinIndex;
205 
```

```
206     for (i = 1; i < AN - 1; i++)
207     {
208         MinEl = Key(A[i]);
209         MinIndex = i;
210         for (j = i + 1; j <= AN - 1; j++)
211         {
212             if (Key(A[j]) < MinEl)
213             {
214                 MinEl = Key(A[j]);
215                 MinIndex = j;
216                 C++;
217             }
218         }
219         x = A[MinIndex];
220         A[MinIndex] = A[i];
221         A[i] = x;
222         C++;
223     }
224 
225     return (C);
226 }
227 
228 int RecBinarySearchCat(CatIndexElement Arr[], int L, int R, string K)
229 {
230     int M;
231 
232     if (L <= R)
233     {
234         M = ((R - L) / 2) + L;
235         if (Key(Arr[M]) == K)
236         {
237             return M;
238         }
239         else
240         {
241             if (Key(Arr[M]) > K)
242                 R = M - 1;
243             else
244                 L = M + 1;
245             return RecBinarySearchCat(Arr, L, R, K);
246         }
247     }
248     return -1;
249 }
250 
251 int IterBinarySearchDate(DateIndexElement Arr[], int AN, int K)
252 {
253     int L = 1;
254     int R = AN;
255     int M;
256 
257     if ((AN > 0) && (K >= Key(Arr[L])) && (K <= Key(Arr[R])))
258     {
259         while (L <= R)
```

```
260         {
261             M = ((R - L) / 2) + L;
262             if (Key(Arr[M]) == K)
263             {
264                 return M;
265             }
266             else
267             {
268                 if (Key(Arr[M]) > K)
269                     R = M - 1;
270                 else
271                     L = M + 1;
272             }
273         }
274     }
275     return -1;
276 }
277 
278 void PrintOne(Expense E, int Num)
279 {
280     printf("%d) Id: %u; Категория: %s; Описание: %s; Дата: %u.%u.%u; Сумма: %.2f",
281            Num, E.Id, E.Category.c_str(), E.Note.c_str(), E.Day, E.Month, E.Year, E.Amount);
282     if (E.Deleted)
283     {
284         printf(" [удалена]");
285     }
286     printf("\n");
287 }
288 
289 void PrintAll()
290 {
291     printf("Текущие записи (по порядку ввода):\n");
292     for (int i = 0; i < RecCount; i++)
293     {
294         if (!ExpArr[i].Deleted)
295         {
296             PrintOne(ExpArr[i], i + 1);
297         }
298     }
299     printf("\n");
300 }
301 
302 void InputOne()
303 {
304     if (RecCount >= MaxRec)
305     {
306         printf("Массив заполнен\n");
307         return;
308     }
309     printf("Введите данные расхода номер %d\n", RecCount + 1);
310     printf("Id - только число, строки без пробелов, дата с учётом дней в месяце\n");
311     if (!ReadUInt("Id", ExpArr[RecCount].Id)) return;
312     if (!ReadString("Категория (строка)", ExpArr[RecCount].Category)) return;
313     if (!ReadString("Описание (строка)", ExpArr[RecCount].Note)) return;
```

```
314     if (!ReadDate(ExpArr[RecCount].Year, ExpArr[RecCount].Month, ExpArr[RecCount].Day))
        ↪   return;
315     if (!ReadPosDouble("Сумма", ExpArr[RecCount].Amount)) return;
316     ExpArr[RecCount].Deleted = false;
317     RecCount++;
318     printf("Запись добавлена\n");
319 }
320
321 void SaveToFile()
322 {
323     string FileName;
324     char Mode;
325     string modeStr;
326     printf("Введите имя файла (или # - выход): > ");
327     if (!(cin >> FileName) || FileName == "#")
328     {
329         if (!cin) ClearInput();
330         printf("Отмена\n");
331         return;
332     }
333     printf("Режим: 1 - новый файл, 2 - добавить в конец (или # - выход): > ");
334     if (!(cin >> modeStr))
335     {
336         ClearInput();
337         return;
338     }
339     if (modeStr.size() != 1)
340     {
341         printf("Неверный режим\n");
342         return;
343     }
344     Mode = modeStr[0];
345     if (Mode == '#')
346     {
347         printf("Отмена\n");
348         return;
349     }
350     if ((Mode != '1') && (Mode != '2'))
351     {
352         printf("Неверный режим\n");
353         return;
354     }
355     ofstream f;
356     if (Mode == '2')
357         f.open(FileName, ios::app);
358     else
359         f.open(FileName, ios::out);
360     if (!f.is_open())
361     {
362         printf("Файл не открыт\n");
363         return;
364     }
365     for (int i = 0; i < RecCount; i++)
366     {
```

```
367         // Сохраняются все записи, включая помеченные deleted
368         f << ExpArr[i].Id << ";" << ExpArr[i].Category << ";" << ExpArr[i].Note << ";";
369         f << ExpArr[i].Day << ";" << ExpArr[i].Month << ";" << ExpArr[i].Year << ";";
370         f << ExpArr[i].Amount << ";" << (ExpArr[i].Deleted ? 1 : 0) << endl;
371     }
372     f.close();
373     printf("Запись в файл выполнена\n");
374 }
375 
376 string NextField(string S, size_t &pos)
377 {
378     // Формат строки: id;cat;note;day;month;year;amount;deleted
379     size_t next = S.find(';', pos);
380     if (next == string::npos)
381     {
382         string part = S.substr(pos);
383         pos = S.length();
384         return part;
385     }
386     string part = S.substr(pos, next - pos);
387     pos = next + 1;
388     return part;
389 }
390 
391 void LoadFromFile()
392 {
393     string FileName;
394     printf("Имя файла для чтения (или # - выход): > ");
395     if (!(cin >> FileName) || FileName == "#")
396     {
397         if (!cin) ClearInput();
398         printf("Отмена\n");
399         return;
400     }
401     ifstream f;
402     f.open(FileName);
403     if (!f.is_open())
404     {
405         printf("Файл не открыт\n");
406         return;
407     }
408     string line;
409     while (!f.eof())
410     {
411         getline(f, line);
412         if (line.length() == 0)
413             continue;
414         if (RecCount >= MaxRec)
415         {
416             printf("Массив заполнен, чтение остановлено\n");
417             break;
418         }
419         size_t pos = 0;
420         string fid = NextField(line, pos);
```

```
421         string fcat = NextField(line, pos);
422         string fnote = NextField(line, pos);
423         string fday = NextField(line, pos);
424         string fmonth = NextField(line, pos);
425         string fyear = NextField(line, pos);
426         string famount = NextField(line, pos);
427         string fdel = NextField(line, pos);
428         ExpArr[RecCount].Id = stoi(fid);
429         ExpArr[RecCount].Category = fcat;
430         ExpArr[RecCount].Note = fnote;
431         ExpArr[RecCount].Day = stoi(fday);
432         ExpArr[RecCount].Month = stoi(fmonth);
433         ExpArr[RecCount].Year = stoi(fyear);
434         ExpArr[RecCount].Amount = stod(famount);
435         ExpArr[RecCount].Deleted = (fdel == "1");
436         RecCount++;
437     }
438     f.close();
439     printf("Чтение из файла завершено\n");
440 }
441 
442 void BuildCatIndex()
443 {
444     CatIndexCount = 0; // пересоздаём индекс
445     for (int i = 0; i < RecCount; i++)
446     {
447         if (!ExpArr[i].Deleted)
448         {
449             CatIndexCount++;
450             CatIndex[CatIndexCount].CatKey = ExpArr[i].Category;
451             CatIndex[CatIndexCount].RecN = i;
452         }
453     }
454     if (CatIndexCount > 1)
455     {
456         OrderCatByInsert(CatIndex, CatIndexCount + 1);
457     }
458 }
459 
460 void BuildDateIndex()
461 {
462     DateIndexCount = 0; // пересоздается индекс
463     for (int i = 0; i < RecCount; i++)
464     {
465         if (!ExpArr[i].Deleted)
466         {
467             DateIndexCount++;
468             DateIndexArr[DateIndexCount].DateKey = CalcDateKey(ExpArr[i]);
469             DateIndexArr[DateIndexCount].RecN = i;
470         }
471     }
472     if (DateIndexCount > 1)
473     {
474         OrderDateBySelection(DateIndexArr, DateIndexCount + 1);
```

```
475     }
476 }
477
478 void ShowByCatIndex(bool Asc)
479 {
480     if (CatIndexCount == 0)
481     {
482         printf("Индекс пуст\n");
483         return;
484     }
485     if (Asc)
486     {
487         for (int i = 1; i <= CatIndexCount; i++)
488         {
489             PrintOne(ExpArr[CatIndex[i].RecN], CatIndex[i].RecN + 1);
490         }
491     }
492     else
493     {
494         for (int i = CatIndexCount; i >= 1; i--)
495         {
496             PrintOne(ExpArr[CatIndex[i].RecN], CatIndex[i].RecN + 1);
497         }
498     }
499     printf("\n");
500 }
501
502 void ShowByDateIndex(bool Asc)
503 {
504     if (DateIndexCount == 0)
505     {
506         printf("Индекс пуст\n");
507         return;
508     }
509     if (Asc)
510     {
511         for (int i = 1; i <= DateIndexCount; i++)
512         {
513             PrintOne(ExpArr[DateIndexArr[i].RecN], DateIndexArr[i].RecN + 1);
514         }
515     }
516     else
517     {
518         for (int i = DateIndexCount; i >= 1; i--)
519         {
520             PrintOne(ExpArr[DateIndexArr[i].RecN], DateIndexArr[i].RecN + 1);
521         }
522     }
523     printf("\n");
524 }
525
526 void ListCategories()
527 {
528     if (CatIndexCount == 0)
```

```
529     {
530         printf("Категории отсутствуют, постройте индекс по категории\n");
531         return;
532     }
533     printf("Список категорий (по индексу):\n");
534     for (int i = 1; i <= CatIndexCount; i++)
535     {
536         printf("%s\n", CatIndex[i].CatKey.c_str());
537     }
538     printf("\n");
539 }
540 
541 void SearchByCategory()
542 {
543     if (CatIndexCount == 0)
544     {
545         printf("Сначала постройте индекс по категории\n");
546         return;
547     }
548     string key;
549     if (!ReadString("Введите категорию для поиска", key)) return;
550     int pos = RecBinarySearchCat(CatIndex, 1, CatIndexCount, key);
551     if (pos == -1)
552     {
553         printf("Не найдено\n");
554     }
555     else
556     {
557         PrintOne(ExpArr[CatIndex[pos].RecN], CatIndex[pos].RecN + 1);
558     }
559     printf("\n");
560 }
561 
562 void SearchByDate()
563 {
564     if (DateIndexCount == 0)
565     {
566         printf("Сначала постройте индекс по дате\n");
567         return;
568     }
569     unsigned int d, m, y;
570     if (!ReadDate(y, m, d)) return;
571     int key = y * 10000 + m * 100 + d;
572     int pos = IterBinarySearchDate(DateIndexArr, DateIndexCount, key);
573     if (pos == -1)
574     {
575         printf("Не найдено\n");
576     }
577     else
578     {
579         PrintOne(ExpArr[DateIndexArr[pos].RecN], DateIndexArr[pos].RecN + 1);
580     }
581     printf("\n");
582 }
```

```
583
584 void EditById()
585 {
586     unsigned int id;
587     printf("Id - только число\n");
588     if (!ReadUInt("Введите Id записи для редактирования", id)) return;
589     for (int i = 0; i < RecCount; i++)
590     {
591         if ((!ExpArr[i].Deleted) && (ExpArr[i].Id == id))
592         {
593             if (!ReadString("Новая категория (строка)", ExpArr[i].Category)) return;
594             if (!ReadString("Новое описание (строка)", ExpArr[i].Note)) return;
595             if (!ReadDate(ExpArr[i].Year, ExpArr[i].Month, ExpArr[i].Day)) return;
596             if (!ReadPosDouble("Новая сумма", ExpArr[i].Amount)) return;
597             printf("Запись изменена\n");
598             BuildCatIndex();
599             BuildDateIndex();
600             return;
601         }
602     }
603     printf("Запись не найдена\n");
604 }
605
606 void DeleteByCategory()
607 {
608     if (CatIndexCount == 0)
609     {
610         printf("Сначала постройте индекс по категории\n");
611         return;
612     }
613     string key;
614     if (!ReadString("Категория для удаления", key)) return;
615     bool found = false;
616     for (int i = 0; i < RecCount; i++)
617     {
618         if (!ExpArr[i].Deleted && ExpArr[i].Category == key)
619         {
620             ExpArr[i].Deleted = true;
621             found = true;
622         }
623     }
624     if (found)
625         printf("Все записи категории %s помечены как удалённые\n", key.c_str());
626     else
627         printf("Не найдено\n");
628     BuildCatIndex();
629     BuildDateIndex();
630 }
631
632 void DeleteById()
633 {
634     unsigned int id;
635     printf("Id - только число\n");
636     if (!ReadUInt("Id для удаления", id)) return;
```

```
637     bool found = false;
638     for (int i = 0; i < RecCount; i++)
639     {
640         if (!ExpArr[i].Deleted && ExpArr[i].Id == id)
641         {
642             ExpArr[i].Deleted = true;
643             found = true;
644             printf("Запись с Id %u помечена как удалённая\n", id);
645             break;
646         }
647     }
648     if (!found) printf("Запись с таким Id не найдена\n");
649     BuildCatIndex();
650     BuildDateIndex();
651 }
652 
653 void RestoreById()
654 {
655     unsigned int id;
656     printf("Id - только число\n");
657     if (!ReadUInt("Id записи для восстановления", id)) return;
658     for (int i = 0; i < RecCount; i++)
659     {
660         if (ExpArr[i].Id == id && ExpArr[i].Deleted)
661         {
662             ExpArr[i].Deleted = false;
663             printf("Запись восстановлена\n");
664             BuildCatIndex();
665             BuildDateIndex();
666             return;
667         }
668     }
669     printf("Удалённая запись не найдена\n");
670 }
671 
672 void RestoreByCategory()
673 {
674     if (CatIndexCount == 0)
675     {
676         printf("Сначала постройте индекс по категории\n");
677         return;
678     }
679     string key;
680     if (!ReadString("Категория для восстановления", key)) return;
681     bool found = false;
682     for (int i = 0; i < RecCount; i++)
683     {
684         if (ExpArr[i].Deleted && ExpArr[i].Category == key)
685         {
686             ExpArr[i].Deleted = false;
687             found = true;
688         }
689     }
690     if (found)
```

```
691     {
692         printf("Записи категории %s восстановлены\n", key.c_str());
693         BuildCatIndex();
694         BuildDateIndex();
695     }
696     else
697     {
698         printf("Удалённых записей с такой категорией нет\n");
699     }
700 }
701 
702 void PackDeleted()
703 {
704     int j = 0; // новая позиция не удаленных записей
705     for (int i = 0; i < RecCount; i++)
706     {
707         if (!ExpArr[i].Deleted)
708         {
709             ExpArr[j] = ExpArr[i];
710             j++;
711         }
712     }
713     RecCount = j;
714     BuildCatIndex();
715     BuildDateIndex();
716     printf("Удалённые записи вычищены\n");
717 }
718 
719 void BuildAllIndexes()
720 {
721     // строятся оба индекса
722     BuildCatIndex();
723     BuildDateIndex();
724 }
725 
726 int main()
727 {
728     char ch = '0';
729     string cmd;
730 
731     do
732     {
733         printf("Меню:\n");
734         printf("1 - ввод новой записи\n");
735         printf("2 - вывод записей по вводу\n");
736         printf("3 - запись в файл\n");
737         printf("4 - чтение из файла\n");
738         printf("5 - построить индексы\n");
739         printf("6 - вывод по индексу категории (возрастание)\n");
740         printf("7 - вывод по индексу категории (убывание)\n");
741         printf("8 - вывод по индексу даты (возрастание)\n");
742         printf("9 - вывод по индексу даты (убывание)\n");
743         printf("a - поиск по категории (рекурсивно)\n");
744         printf("b - поиск по дате (итеративно)\n");
```

```
745         printf("g - показать все категории\n");
746         printf("c - редактировать по Id\n");
747         printf("d - удалить по категории\n");
748         printf("h - удалить по Id\n");
749         printf("e - восстановить по Id\n");
750         printf("i - восстановить по категории\n");
751         printf("f - физическое удаление помеченных\n");
752         printf("0 - выход\n");
753         if (!(cin >> cmd))
754         {
755             ClearInput();
756             continue;
757         }
758         if (cmd.size() != 1)
759         {
760             printf("Команда не распознана\n\n");
761             continue;
762         }
763         ch = cmd[0];
764         printf("\n");
765 
766         switch (ch)
767         {
768         case '1':
769             InputOne();
770             break;
771         case '2':
772             PrintAll();
773             break;
774         case '3':
775             SaveToFile();
776             break;
777         case '4':
778             LoadFromFile();
779             break;
780         case '5':
781             BuildAllIndexes();
782             printf("Индексы построены\n");
783             break;
784         case '6':
785             ShowByCatIndex(true);
786             break;
787         case '7':
788             ShowByCatIndex(false);
789             break;
790         case '8':
791             ShowByDateIndex(true);
792             break;
793         case '9':
794             ShowByDateIndex(false);
795             break;
796         case 'a':
797             SearchByCategory();
798             break;
```

```
799         case 'b':
800             SearchByDate();
801             break;
802         case 'c':
803             EditById();
804             break;
805         case 'd':
806             DeleteByCategory();
807             break;
808         case 'e':
809             RestoreById();
810             break;
811         case 'f':
812             PackDeleted();
813             break;
814         case 'g':
815             ListCategories();
816             break;
817         case 'h':
818             DeleteById();
819             break;
820         case 'i':
821             RestoreByCategory();
822             break;
823         case '0':
824             break;
825         default:
826             printf("Нет такой команды\n");
827             break;
828         }
829         printf("\n");
830     } while (ch != '0');
831 
832     return 0;
833 }
```