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>
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];
int CatIndexCount = 0;

DateIndexElement DateIndexArr[MaxRec + 1];
int DateIndexCount = 0;

void ClearInput()
```

```
{
    cin.clear();
    while (cin && cin.get() != '\n') {}
    printf("Неверный ввод\n");
}

bool ReadUInt(const char *prompt, unsigned int &val)
{
    string tmp;
    while (true)
    {
        printf("%s (или # - выход): ", prompt);
        if (!(cin >> tmp))
        {
            ClearInput();
            continue;
        }
        if (tmp == "#")
        {
            printf("Отмена\n");
            return false;
        }
        try
        {
            // stoul переводит строку в число; затем приводим к unsigned int
            val = static_cast<unsigned int>(stoul(tmp));
            return true;
        }
        catch (...)
        {
            printf("Неверный ввод\n");
        }
    }
}

bool ReadDouble(const char *prompt, double &val)
{
    string tmp;
    while (true)
    {
        printf("%s (или # - выход): ", prompt);
        if (!(cin >> tmp))
        {
            ClearInput();
            continue;
        }
        if (tmp == "#")
        {
            printf("Отмена\n");
            return false;
        }
        try
        {
            val = stod(tmp);
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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