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

Факультет социально-экономических и компьютерных наук

Самостоятельная работа №1. Задание №3

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

Разработайте процедуры (функции), используя коды на языке C++, разработанные при выполнении предыдущих заданий:

1. Создания упорядоченного линейного списка для представления данных об объектах предметной области. Данные вводятся из файла или с клавиатуры (используйте ранее разработанные функции). Данные вставляются в порядке убывания/возрастания (неубывания/невозрастания) значений атрибутов в заданном поле структуры, представляемой сведения об объектах реального мира.
2. Ввода новых записей и включения их в линейный список с упорядочением по выбранным атрибутам по возрастанию/убыванию (невозрастанию/неубыванию) их значений.
3. Просмотр (вывод) записей из линейного списка в порядке возрастания/убывания (невозрастания/неубывания) значений ключевых атрибутов, по которым выполнена сортировка элементов списка, т. е. просмотра (вывода) данных от головы к хвосту или от хвоста к голове упорядоченного списка (разработайте итерационный и рекурсивный алгоритмы обхода списка).
4. Поиск и вывод записи/записей с заданным значением атрибута, по которому выполнена сортировка (значений всех атрибутов найденной записи, а если запись не найдена, то должно быть выведено соответствующее сообщение).
5. Удаление записи с заданным значением атрибута, по которому выполнена сортировка.

2 Введение

Реализуется продолжение программы, разработанной в задании 1. Предметная область, структура записей, формат файла и базовые операции ввода/вывода сохраняются без изменений и рассматриваются в предыдущем отчёте. Для третьего задания изменяется только часть, связанная с индексацией и поиском: вместо индексных массивов и дерева используется упорядоченный линейный список по выбранному полю (категория), список перестраивается при изменении данных.

3 Структура данных и форматы

Основной массив Expense (30 элементов) с полями Id, Category, Note, Day, Month, Year, Amount, Deleted.

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

Правила ввода: строки без символов ; и пробелов; дата проверяется по DaysInMonth (Month 1..12, Day в пределах месяца); сумма double > 0; Deleted — логическая метка (0/1).

4 Алгоритмы

Линейный список

- Структура узла: ListNode{Key, RecIndex, Next}, Key — категория, RecIndex — номер записи в массиве.
- Построение: старый список очищается, затем вставляются все не удалённые записи сортировкой вставкой по Category.

- Обходы: итеративный — от головы к хвосту (возрастание); рекурсивный — разворот через вызовы (убывание).
- Поиск: две функции — рекурсивная и итеративная; выводятся все записи с найденным ключом.

5 Операции

- Ввод/вывод/файл — как в заданиях 1–2; после ввода или чтения индекс строится отдельной командой.
- Редактирование по Id: новые поля считываются, список пересобирается.
- Удаление/восстановление: флаг Deleted ставится/снимается по Id или по категории; после этого список пересобирается.
- Физическое удаление: копируются только не удалённые записи в начало массива, RecCount уменьшается, список строится заново.

6 Проверки новых функций

Ручные тесты для списка:

1. Построение списка на sample-data.txt: обход по возрастанию/убыванию выводит все записи по сортировке категории.
2. Поиск books: найден Id = 303 в рекурсивной и итеративной версиях.
3. Дубликаты ключа: после добавления второй food обе выводятся подряд.
4. Удаление/восстановление категории transport: 302 не выводится после пометки, восстановление по Id возвращает; после физического удаления 302 пропадает окончательно.
5. Пересборка после редактирования: смена категории у записи перемещает её в новое место списка.

7 Недостатки

Сохраняются все недостатки, указанные в отчёте к заданию 1 (относительно общей структуры), а также отсутствует сортировка по двум ключам (только один упорядоченный список по категории).

8 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
```

```
14      string Note;        // короткое описание
15      unsigned int Day;
16      unsigned int Month;
17      unsigned int Year;
18      double Amount;      // сумма > 0
19      bool Deleted;       // пометка удаления
20  };
21
22  struct ListNode
23  {
24      string Key;     // ключ сортировки (категория)
25      int RecIndex;  // номер записи в массиве
26      ListNode *Next;
27  };
28
29  Expense ExpArr[MaxRec];
30  int RecCount = 0;
31
32  ListNode *Head = NULL;
33
34  void ClearInput()
35  {
36      cin.clear();
37      while (cin && cin.get() != '\n') {}
38      printf("Неверный ввод\n");
39  }
40
41  bool ReadUInt(const char *prompt, unsigned int &val)
42  {
43      string tmp;
44      while (true)
45      {
46          printf("%s (или # - выход): ", prompt);
47          if (!(cin >> tmp))
48          {
49              ClearInput();
50              continue;
51          }
52          if (tmp == "#")
53          {
54              printf("Отмена\n");
55              return false;
56          }
57          try
58          {
59              val = static_cast<unsigned int>(stoul(tmp));
60              return true;
61          }
62          catch (...)
63          {
64              printf("Неверный ввод\n");
65          }
66      }
67  }
```

```
68
69  bool ReadDouble(const char *prompt, double &val)
70  {
71      string tmp;
72      while (true)
73      {
74          printf("%s (или # - выход): ", prompt);
75          if (!(cin >> tmp))
76          {
77              ClearInput();
78              continue;
79          }
80          if (tmp == "#")
81          {
82              printf("Отмена\n");
83              return false;
84          }
85          try
86          {
87              val = stod(tmp);
88              return true;
89          }
90          catch (...)
91          {
92              printf("Неверный ввод\n");
93          }
94      }
95  }
96
97  bool ReadPosDouble(const char *prompt, double &val)
98  {
99      while (true)
100     {
101         if (!ReadDouble(prompt, val)) return false;
102         if (val > 0) return true;
103         printf("Значение должно быть > 0\n");
104     }
105 }
106
107 bool ReadString(const char *prompt, string &val)
108 {
109     while (true)
110     {
111         printf("%s (или # - выход): ", prompt);
112         getline(cin >> ws, val);
113         if (val == "#")
114         {
115             printf("Отмена\n");
116             return false;
117         }
118         if (val.find(';') != string::npos || val.find(' ') != string::npos)
119         {
120             printf("Символ ';' и пробелы запрещены\n");
121             continue;
```

```
122         }
123         return true;
124     }
125 }
126
127 bool ReadDate(unsigned int &y, unsigned int &m, unsigned int &d)
128 {
129     if (!ReadUInt("Год", y)) return false;
130     while (true)
131     {
132         if (!ReadUInt("Месяц", m)) return false;
133         if ((m >= 1) && (m <= 12)) break;
134         printf("Месяц должен быть 1..12\n");
135     }
136     while (true)
137     {
138         if (!ReadUInt("День", d)) return false;
139         int maxd = DaysInMonth[m];
140         if ((d >= 1) && (d <= static_cast<unsigned int>(maxd))) break;
141         printf("В этом месяце %d дней\n", maxd);
142     }
143     return true;
144 }
145
146 ListNode* MakeNode(string key, int recIndex)
147 {
148     ListNode *p = new ListNode;
149     p->Key = key;
150     p->RecIndex = recIndex;
151     p->Next = NULL;
152     return p;
153 }
154
155 void FreeList()
156 {
157     ListNode *p = Head;
158     while (p != NULL)
159     {
160         ListNode *tmp = p->Next;
161         delete p;
162         p = tmp;
163     }
164     Head = NULL;
165 }
166
167 void InsertSorted(string key, int recIndex)
168 {
169     ListNode *node = MakeNode(key, recIndex);
170     if (Head == NULL || key < Head->Key)
171     {
172         node->Next = Head;
173         Head = node;
174         return;
175     }
```

```
176     ListNode *cur = Head;
177     while (cur->Next != NULL && cur->Next->Key <= key)
178     {
179         cur = cur->Next;
180     }
181     node->Next = cur->Next;
182     cur->Next = node;
183 }
184 
185 void BuildList()
186 {
187     FreeList();
188     for (int i = 0; i < RecCount; i++)
189     {
190         if (!ExpArr[i].Deleted)
191         {
192             InsertSorted(ExpArr[i].Category, i);
193         }
194     }
195 }
196 
197 void PrintOne(Expense E, int Num)
198 {
199     printf("%d) Id: %u; Категория: %s; Описание: %s; Дата: %u.%u.%u; Сумма: %.2f",
200            Num, E.Id, E.Category.c_str(), E.Note.c_str(), E.Day, E.Month, E.Year, E.Amount);
201     if (E.Deleted)
202     {
203         printf(" [удалена]");
204     }
205     printf("\n");
206 }
207 
208 void PrintAll()
209 {
210     printf("Текущие записи (по порядку ввода):\n");
211     for (int i = 0; i < RecCount; i++)
212     {
213         if (!ExpArr[i].Deleted)
214         {
215             PrintOne(ExpArr[i], i + 1);
216         }
217     }
218     printf("\n");
219 }
220 
221 void PrintListAsc()
222 {
223     if (Head == NULL)
224     {
225         printf("Список пуст\n");
226         return;
227     }
228     ListNode *p = Head;
229     while (p != NULL)
```

```
230     {
231         PrintOne(ExpArr[p->RecIndex], p->RecIndex + 1);
232         p = p->Next;
233     }
234     printf("\n");
235 }
236 
237 void PrintListDescRec(ListNode *p)
238 {
239     if (p == NULL) return;
240     PrintListDescRec(p->Next);
241     PrintOne(ExpArr[p->RecIndex], p->RecIndex + 1);
242 }
243 
244 void PrintListDesc()
245 {
246     if (Head == NULL)
247     {
248         printf("Список пуст\n");
249         return;
250     }
251     PrintListDescRec(Head);
252     printf("\n");
253 }
254 
255 ListNode* SearchIter(string key)
256 {
257     ListNode *p = Head;
258     while (p != NULL)
259     {
260         if (p->Key == key)
261             return p;
262         if (p->Key > key)
263             break;
264         p = p->Next;
265     }
266     return NULL;
267 }
268 
269 ListNode* SearchRec(ListNode *p, string key)
270 {
271     if (p == NULL) return NULL;
272     if (p->Key == key) return p;
273     if (p->Key > key) return NULL;
274     return SearchRec(p->Next, key);
275 }
276 
277 void InputOne()
278 {
279     if (RecCount >= MaxRec)
280     {
281         printf("Массив заполнен\n");
282         return;
283     }
```

```
284     printf("Введите данные расхода номер %d\n", RecCount + 1);
285     printf("Id - только число, строки без пробелов, дата с учётом дней в месяце\n");
286     if (!ReadUInt("Id", ExpArr[RecCount].Id)) return;
287     if (!ReadString("Категория (строка)", ExpArr[RecCount].Category)) return;
288     if (!ReadString("Описание (строка)", ExpArr[RecCount].Note)) return;
289     if (!ReadDate(ExpArr[RecCount].Year, ExpArr[RecCount].Month, ExpArr[RecCount].Day))
        ↪   return;
290     if (!ReadPosDouble("Сумма", ExpArr[RecCount].Amount)) return;
291     ExpArr[RecCount].Deleted = false;
292     RecCount++;
293     printf("Запись добавлена\n");
294     BuildList();
295 }
296
297 void SaveToFile()
298 {
299     string FileName;
300     char Mode;
301     string modeStr;
302     printf("Введите имя файла (или # - выход): > ");
303     if (!(cin >> FileName) || FileName == "#")
304     {
305         if (!cin) ClearInput();
306         printf("Отмена\n");
307         return;
308     }
309     printf("Режим: 1 - новый файл, 2 - добавить в конец (или # - выход): > ");
310     if (!(cin >> modeStr))
311     {
312         ClearInput();
313         return;
314     }
315     if (modeStr.size() != 1)
316     {
317         printf("Неверный режим\n");
318         return;
319     }
320     Mode = modeStr[0];
321     if (Mode == '#')
322     {
323         printf("Отмена\n");
324         return;
325     }
326     if ((Mode != '1') && (Mode != '2'))
327     {
328         printf("Неверный режим\n");
329         return;
330     }
331     ofstream f;
332     if (Mode == '2')
333         f.open(FileName, ios::app);
334     else
335         f.open(FileName, ios::out);
336     if (!f.is_open())
```

```
337     {
338         printf("Файл не открыт\n");
339         return;
340     }
341     for (int i = 0; i < RecCount; i++)
342     {
343         f << ExpArr[i].Id << ";" << ExpArr[i].Category << ";" << ExpArr[i].Note << ";";
344         f << ExpArr[i].Day << ";" << ExpArr[i].Month << ";" << ExpArr[i].Year << ";";
345         f << ExpArr[i].Amount << ";" << (ExpArr[i].Deleted ? 1 : 0) << endl;
346     }
347     f.close();
348     printf("Запись в файл выполнена\n");
349 }
350 
351 string NextField(string S, size_t &pos)
352 {
353     // Формат строки: id;cat;note;day;month;year;amount;deleted
354     size_t next = S.find(';', pos);
355     if (next == string::npos)
356     {
357         string part = S.substr(pos);
358         pos = S.length();
359         return part;
360     }
361     string part = S.substr(pos, next - pos);
362     pos = next + 1;
363     return part;
364 }
365 
366 void LoadFromFile()
367 {
368     string FileName;
369     printf("Имя файла для чтения (или # - выход): > ");
370     if (!(cin >> FileName) || FileName == "#")
371     {
372         if (!cin) ClearInput();
373         printf("Отмена\n");
374         return;
375     }
376     ifstream f;
377     f.open(FileName);
378     if (!f.is_open())
379     {
380         printf("Файл не открыт\n");
381         return;
382     }
383     string line;
384     while (!f.eof())
385     {
386         getline(f, line);
387         if (line.length() == 0)
388             continue;
389         if (RecCount >= MaxRec)
390         {
```

```
391             printf("Массив заполнен, чтение остановлено\n");
392             break;
393         }
394         size_t pos = 0;
395         string fid = NextField(line, pos);
396         string fcat = NextField(line, pos);
397         string fnote = NextField(line, pos);
398         string fday = NextField(line, pos);
399         string fmonth = NextField(line, pos);
400         string fyear = NextField(line, pos);
401         string famount = NextField(line, pos);
402         string fdel = NextField(line, pos);
403         ExpArr[RecCount].Id = stoi(fid);
404         ExpArr[RecCount].Category = fcat;
405         ExpArr[RecCount].Note = fnote;
406         ExpArr[RecCount].Day = stoi(fday);
407         ExpArr[RecCount].Month = stoi(fmonth);
408         ExpArr[RecCount].Year = stoi(fyear);
409         ExpArr[RecCount].Amount = stod(famount);
410         ExpArr[RecCount].Deleted = (fdel == "1");
411         RecCount++;
412     }
413     f.close();
414     BuildList();
415     printf("Чтение из файла завершено\n");
416 }
417 
418 void ShowListAsc()
419 {
420     PrintListAsc();
421 }
422 
423 void ShowListDesc()
424 {
425     PrintListDesc();
426 }
427 
428 void SearchByCategoryRec()
429 {
430     if (Head == NULL)
431     {
432         printf("Список пуст\n");
433         return;
434     }
435     string key;
436     if (!ReadString("Введите категорию для поиска (рекурсивно)", key)) return;
437     ListNode *p = SearchRec(Head, key);
438     if (p == NULL)
439     {
440         printf("Не найдено\n\n");
441         return;
442     }
443     while (p != NULL && p->Key == key)
444     {
```

```
445         PrintOne(ExpArr[p->RecIndex], p->RecIndex + 1);
446         p = p->Next;
447     }
448     printf("\n");
449 }
450 
451 void SearchByCategoryIter()
452 {
453     if (Head == NULL)
454     {
455         printf("Список пуст\n");
456         return;
457     }
458     string key;
459     if (!ReadString("Введите категорию для поиска (итеративно)", key)) return;
460     ListNode *p = SearchIter(key);
461     if (p == NULL)
462     {
463         printf("Не найдено\n\n");
464         return;
465     }
466     while (p != NULL && p->Key == key)
467     {
468         PrintOne(ExpArr[p->RecIndex], p->RecIndex + 1);
469         p = p->Next;
470     }
471     printf("\n");
472 }
473 
474 void EditById()
475 {
476     unsigned int id;
477     printf("Id - только число\n");
478     if (!ReadUInt("Введите Id записи для редактирования", id)) return;
479     for (int i = 0; i < RecCount; i++)
480     {
481         if ((!ExpArr[i].Deleted) && (ExpArr[i].Id == id))
482         {
483             if (!ReadString("Новая категория (строка)", ExpArr[i].Category)) return;
484             if (!ReadString("Новое описание (строка)", ExpArr[i].Note)) return;
485             if (!ReadDate(ExpArr[i].Year, ExpArr[i].Month, ExpArr[i].Day)) return;
486             if (!ReadPosDouble("Новая сумма", ExpArr[i].Amount)) return;
487             printf("Запись изменена\n");
488             BuildList();
489             return;
490         }
491     }
492     printf("Запись не найдена\n");
493 }
494 
495 void DeleteByCategory()
496 {
497     if (Head == NULL)
498     {
```

```
499         printf("Список пуст\n");
500         return;
501     }
502     string key;
503     if (!ReadString("Категория для удаления", key)) return;
504     bool found = false;
505     ListNode *p = Head;
506     while (p != NULL && p->Key <= key)
507     {
508         if (!ExpArr[p->RecIndex].Deleted && p->Key == key)
509         {
510             ExpArr[p->RecIndex].Deleted = true;
511             found = true;
512         }
513         p = p->Next;
514     }
515     if (found)
516         printf("Все записи категории %s помечены как удалённые\n", key.c_str());
517     else
518         printf("Не найдено\n");
519     BuildList();
520 }
521
522 void DeleteById()
523 {
524     unsigned int id;
525     printf("Id - только число\n");
526     if (!ReadUInt("Id для удаления", id)) return;
527     bool found = false;
528     for (int i = 0; i < RecCount; i++)
529     {
530         if (!ExpArr[i].Deleted && ExpArr[i].Id == id)
531         {
532             ExpArr[i].Deleted = true;
533             found = true;
534             printf("Запись с Id %u помечена как удалённая\n", id);
535             break;
536         }
537     }
538     if (!found) printf("Запись с таким Id не найдена\n");
539     BuildList();
540 }
541
542 void RestoreById()
543 {
544     unsigned int id;
545     printf("Id - только число\n");
546     if (!ReadUInt("Id записи для восстановления", id)) return;
547     for (int i = 0; i < RecCount; i++)
548     {
549         if (ExpArr[i].Id == id && ExpArr[i].Deleted)
550         {
551             ExpArr[i].Deleted = false;
552             printf("Запись восстановлена\n");
```

```
553             BuildList();
554             return;
555         }
556     }
557     printf("Удалённая запись не найдена\n");
558 }
559 
560 void PackDeleted()
561 {
562     int j = 0;
563     for (int i = 0; i < RecCount; i++)
564     {
565         if (!ExpArr[i].Deleted)
566         {
567             ExpArr[j] = ExpArr[i];
568             j++;
569         }
570     }
571     RecCount = j;
572     BuildList();
573     printf("Удалённые записи вычищены\n");
574 }
575 
576 void BuildListMenu()
577 {
578     BuildList();
579     if (Head == NULL)
580         printf("Список пуст\n");
581     else
582         printf("Список построен\n");
583 }
584 
585 int main()
586 {
587     char ch = '0';
588     string cmd;
589 
590     do
591     {
592         printf("Меню:\n");
593         printf("1 - ввод новой записи\n");
594         printf("2 - вывод записей по вводу\n");
595         printf("3 - запись в файл\n");
596         printf("4 - чтение из файла\n");
597         printf("5 - построить список по категории\n");
598         printf("6 - вывод списка (возрастание)\n");
599         printf("7 - вывод списка (убывание)\n");
600         printf("8 - поиск по категории (рекурсивно)\n");
601         printf("9 - поиск по категории (итеративно)\n");
602         printf("c - редактировать по Id\n");
603         printf("d - удалить по категории\n");
604         printf("h - удалить по Id\n");
605         printf("e - восстановить по Id\n");
606         printf("f - физическое удаление помеченных\n");
```

```
607         printf("0 - выход\n");
608         if (!(cin >> cmd))
609         {
610             ClearInput();
611             continue;
612         }
613         if (cmd.size() != 1)
614         {
615             printf("Команда не распознана\n\n");
616             continue;
617         }
618         ch = cmd[0];
619         printf("\n");
620 
621         switch (ch)
622         {
623         case '1':
624             InputOne();
625             break;
626         case '2':
627             PrintAll();
628             break;
629         case '3':
630             SaveToFile();
631             break;
632         case '4':
633             LoadFromFile();
634             break;
635         case '5':
636             BuildListMenu();
637             break;
638         case '6':
639             ShowListAsc();
640             break;
641         case '7':
642             ShowListDesc();
643             break;
644         case '8':
645             SearchByCategoryRec();
646             break;
647         case '9':
648             SearchByCategoryIter();
649             break;
650         case 'c':
651             EditById();
652             break;
653         case 'd':
654             DeleteByCategory();
655             break;
656         case 'e':
657             RestoreById();
658             break;
659         case 'f':
660             PackDeleted();
```

```
661             break;
662         case 'h':
663             DeleteById();
664             break;
665         case '0':
666             break;
667         default:
668             printf("Нет такой команды\n");
669             break;
670         }
671         printf("\n");
672     } while (ch != '0');
673 
674     FreeList();
675     return 0;
676 }
```