

Міністерство освіти і науки України
Національний університет «Львівська політехніка»
кафедра інформаційних систем та мереж

Григорович Віктор

Алгоритмізація та програмування

Файли

Навчальний посібник

2020

Григорович Віктор Геннадійович

Алгоритмізація та програмування. Файли. Навчальний посібник.

Дисципліна «Алгоритмізація та програмування» вивчається на першому курсі, – з цієї дисципліни розпочинається цикл предметів, що стосуються програмування та розробки програмного забезпечення.

У посібнику містяться теоретичні відомості, приклади, методичні вказівки з їх розв'язування, варіанти лабораторних завдань та питання і завдання з контролю знань з теми «Файли».

Розглядаються наступні роботи лабораторного практикуму:

Лабораторна робота № 10.1. Пошук символів у текстовому файлі

Лабораторна робота № 10.2. Пошук слів у текстовому файлі

Лабораторна робота № 10.3. Опрацювання текстового файлу

Лабораторна робота № 10.4. Опрацювання рядків текстового файлу

Лабораторна робота № 11.1. Бінарні файли

Лабораторна робота № 11.2. Послідовний пошук у бінарному файлі

Лабораторна робота № 11.3. Впорядкування та бінарний пошук у бінарному файлі

Лабораторна робота № 11.4. Опрацювання бінарного файлу

Лабораторна робота № 11.5. Структури, об'єднання та бінарні файли

В посібнику наведено рекомендований набір лабораторних робіт та приклади їх виконання.

Відповідальний за випуск – Григорович В.Г.

Стислий зміст

Вступ.....	20
Теми 10, 11. Файли.....	22
Стисло та головне про файли.....	22
Опрацювання файлів засобами мови С.....	23
Файлові потоки: опрацювання файлів в стилі С++.....	29
Теоретичні відомості.....	38
Файлове введення / виведення в С та С++.....	38
Загальні поняття.....	38
Опрацювання файлів засобами мови С.....	42
Файлові потоки: опрацювання файлів в стилі С++.....	64
Операції з фізичними файлами.....	99
Приклади.....	101
Опрацювання текстового файлу.....	101
Опрацювання бінарного файлу.....	101
Впорядкування бінарних файлів (рекомендації).....	103
Вилучення компонент з файлів (рекомендації).....	104
Проблеми при повторному скануванні файлу.....	107
Лабораторний практикум.....	114
Оформлення звітів про виконання лабораторних робіт.....	114
10. Текстові файли.....	116
11. Бінарні файли.....	158
Питання та завдання для контролю знань.....	220
Текстові файли.....	220
Бінарні файли.....	221
Файли.....	223
Файлові потоки.....	227
Стан потоку.....	228
Буфер потоку.....	228
Вказівник файлового потоку (файловий вказівник).....	229
Опрацювання двійкових файлів за допомогою потоків.....	229
Предметний покажчик.....	231
Література.....	233

Зміст

Вступ.....	20
Теми 10, 11. Файли.....	22
Стисло та головне про файли.....	22
Опрацювання файлів засобами мови C.....	23
Відкриття файлу.....	23
Закриття файлу.....	25
Опрацювання текстових файлів.....	25
Запис та зчитування символу.....	25
Запис символу – функції <code>putc()</code> та <code>fputc()</code>	25
Зчитування символу – функції <code>getc()</code> та <code>fgetc()</code>	26
Перевірка кінця файлу – функція <code>feof()</code>	26
Введення / виведення рядків: функції <code>fputs()</code> та <code>fgets()</code>	27
Опрацювання бінарних файлів.....	27
Функції <code>fread()</code> та <code>fwrite()</code>	27
Прямий доступ: функція <code>fseek()</code>	28
Файлові потоки: опрацювання файлів в стилі C++.....	29
Файлові потоки та організація введення / виведення.....	30
Оголошення файлової змінної.....	30
Відкриття файлу.....	30
Закриття файлу.....	31
Режими відкриття файлу.....	31
Визначення кінця файлу.....	33
Опрацювання текстових файлових потоків.....	33
Відкриття файлу.....	33
Послідовний перегляд та зчитування текстового файлу.....	34
Запис у текстовий файл.....	34
Опрацювання бінарних файлових потоків.....	34
Відкриття бінарного файлу.....	34
Зчитування з бінарного файлу.....	34
Запис у бінарний файл.....	35
Прямий доступ: навігація по бінарному файлу.....	36
Теоретичні відомості.....	38
Файлове введення / виведення в C та C++.....	38
Загальні поняття.....	38

Опрацювання файлів засобами мови C	42
Потоки та файли	42
Потоки	42
Текстові потоки	42
Двійкові (бінарні) потоки	43
Файли.....	43
Основи файлової системи.....	44
Файлова змінна – вказівник на структуру FILE	45
Відкриття файлу	45
Закриття файлу	47
Опрацювання текстових файлів.....	48
Запис та зчитування символу	48
Запис символу – функції <code>putc()</code> та <code>fputc()</code>	48
Зчитування символу – функції <code>getc()</code> та <code>fgetc()</code>	48
Приклад використання функцій <code>fopen()</code> , <code>getc()</code> , <code>putc()</code> та <code>fclose()</code>	49
Перевірка кінця файлу – функція <code>feof()</code>	50
Введення / виведення рядків: функції <code>fputs()</code> та <code>fgets()</code>	51
Функція <code>rewind()</code>	52
Опрацювання бінарних файлів	53
Функції <code>fread()</code> та <code>fwrite()</code>	53
Прямий доступ: функції <code>fseek()</code> та <code>ftell()</code>	57
Додаткові можливості опрацювання файлів	60
Функція <code>ferror()</code>	60
Примусовий запис потоку – функція <code>fflush()</code>	60
Функції <code>fprintf()</code> та <code>fscanf()</code>	61
Стандартні потоки.....	62
Пере-направлення потоків: функція <code>freopen()</code>	62
Файлові потоки: опрацювання файлів в стилі C++.....	64
Файлові потоки та організація введення / виведення	65
Оголошення файлової змінної	65
Відкриття файлу	66
Закриття файлу	66
Режими відкриття файлу	67
Визначення кінця файлу	68

Передавання файлів у функції	71
Опрацювання текстових файлових потоків	72
Відкриття файлу	72
Послідовний перегляд та зчитування текстового файлу	73
Запис у текстовий файл	76
Опрацювання бінарних файлових потоків	76
Відкриття бінарного файлу	76
Зчитування з бінарного файлу	76
Запис у бінарний файл	77
Приклад опрацювання бінарного файлу	78
Прямий доступ: навігація по бінарному файлу	80
Додаткові можливості опрацювання файлових потоків	85
Функції управління введенням-виведенням	85
get ()	85
put ()	87
getline ()	87
gcount ()	87
ignore ()	88
peek ()	88
putback ()	88
unget ()	89
flush ()	89
Перевірка помилок при виконанні файлових операцій	90
Визначення та встановлення стану потоку	91
Зчитування / запис літерних рядків <code>string</code> у бінарний файл	94
Спосіб 1. Використання рядків в стилі мови C	95
Спосіб 2. Запис та зчитування довжини рядка <code>string</code> та його вмісту	97
Операції з фізичними файлами	99
remove () – вилучення файлу	99
rename () – перейменування файлу	99
Приклади	101
Опрацювання текстового файлу	101
Опрацювання бінарного файлу	101
Впорядкування бінарних файлів (рекомендації)	103

Вилучення компонент з файлів (рекомендації).....	104
Вилучення компонент з текстового файлу	105
Вилучення компонент з бінарного файлу.....	106
Проблеми при повторному скануванні файлу.....	107
Сканування текстового файлу.....	107
Сканування бінарного файлу	110
Лабораторний практикум	114
Оформлення звітів про виконання лабораторних робіт	114
Вимоги до оформлення звіту про виконання лабораторних робіт №№ 10.1–10.4, 11.1–11.5.....	114
Зразок оформлення звіту про виконання лабораторних робіт №№ 10.1–10.4, 11.1–11.5	114
10. Текстові файли.....	116
Приклад виконання лабораторних завдань	116
Лабораторна робота № 10.1. Пошук символів у текстовому файлі	119
Питання, які необхідно вивчити та пояснити на захисті.....	119
Оформлення звіту.....	119
Варіанти лабораторних завдань	119
Варіант 1.....	119
Варіант 2.....	119
Варіант 3.....	120
Варіант 4.....	120
Варіант 5.....	120
Варіант 6.....	120
Варіант 7.....	120
Варіант 8.....	120
Варіант 9.....	120
Варіант 10.....	121
Варіант 11.....	121
Варіант 12.....	121
Варіант 13.....	121
Варіант 14.....	121
Варіант 15.....	121
Варіант 16.....	121
Варіант 17.....	122

Варіант 18.....	122
Варіант 19.....	122
Варіант 20.....	122
Варіант 21.....	122
Варіант 22.....	122
Варіант 23.....	122
Варіант 24.....	123
Варіант 25.....	123
Варіант 26.....	123
Варіант 27.....	123
Варіант 28.....	123
Варіант 29.....	123
Варіант 30.....	123
Варіант 31.....	124
Варіант 32.....	124
Варіант 33.....	124
Варіант 34.....	124
Варіант 35.....	124
Варіант 36.....	124
Варіант 37.....	124
Варіант 38.....	125
Варіант 39.....	125
Варіант 40.....	125
Лабораторна робота № 10.2. Пошук слів у текстовому файлі.....	126
Питання, які необхідно вивчити та пояснити на захисті.....	126
Оформлення звіту.....	126
Варіанти лабораторних завдань	126
Варіант 1.....	126
Варіант 2.....	126
Варіант 3.....	127
Варіант 4.....	127
Варіант 5.....	127
Варіант 6.....	127
Варіант 7.....	127
Варіант 8.....	127

Варіант 9.....	128
Варіант 10.....	128
Варіант 11.....	128
Варіант 12.....	128
Варіант 13.....	128
Варіант 14.....	128
Варіант 15.....	129
Варіант 16.....	129
Варіант 17.....	129
Варіант 18.....	129
Варіант 19.....	130
Варіант 20.....	130
Варіант 21.....	130
Варіант 22.....	130
Варіант 23.....	130
Варіант 24.....	130
Варіант 25.....	130
Варіант 26.....	131
Варіант 27.....	131
Варіант 28.....	131
Варіант 29.....	131
Варіант 30.....	131
Варіант 31.....	131
Варіант 32.....	132
Варіант 33.....	132
Варіант 34.....	132
Варіант 35.....	132
Варіант 36.....	132
Варіант 37.....	133
Варіант 38.....	133
Варіант 39.....	133
Варіант 40.....	133
Лабораторна робота № 10.3. Опрацювання текстового файлу.....	134
Питання, які необхідно вивчити та пояснити на захисті.....	134
Оформлення звіту.....	134

Варіанти лабораторних завдань	134
Варіант 1.....	134
Варіант 2.....	135
Варіант 3.....	135
Варіант 4.....	135
Варіант 5.....	136
Варіант 6.....	136
Варіант 7.....	136
Варіант 8.....	137
Варіант 9.....	137
Варіант 10.....	137
Варіант 11.....	137
Варіант 12.....	138
Варіант 13.....	138
Варіант 14.....	138
Варіант 15.....	139
Варіант 16.....	139
Варіант 17.....	139
Варіант 18.....	140
Варіант 19.....	140
Варіант 20.....	140
Варіант 21.....	141
Варіант 22.....	141
Варіант 23.....	141
Варіант 24.....	142
Варіант 25.....	142
Варіант 26.....	142
Варіант 27.....	142
Варіант 28.....	143
Варіант 29.....	143
Варіант 30.....	143
Варіант 31.....	144
Варіант 32.....	144
Варіант 33.....	144
Варіант 34.....	145

Варіант 35.....	145
Варіант 36.....	145
Варіант 37.....	146
Варіант 38.....	146
Варіант 39.....	146
Варіант 40.....	147
Лабораторна робота № 10.4. Опрацювання рядків текстового файлу	149
Питання, які необхідно вивчити та пояснити на захисті.....	149
Оформлення звіту.....	149
Варіанти лабораторних завдань	149
Варіант 1.....	149
Варіант 2.....	150
Варіант 3.....	150
Варіант 4.....	150
Варіант 5.....	150
Варіант 6.....	151
Варіант 7.....	151
Варіант 8.....	151
Варіант 9.....	151
Варіант 10.....	151
Варіант 11.....	152
Варіант 12.....	152
Варіант 13.....	152
Варіант 14.....	152
Варіант 15.....	152
Варіант 16.....	153
Варіант 17.....	153
Варіант 18.....	153
Варіант 19.....	153
Варіант 20.....	153
Варіант 21.....	153
Варіант 22.....	153
Варіант 23.....	154
Варіант 24.....	154
Варіант 25.....	154

Варіант 26.....	154
Варіант 27.....	154
Варіант 28.....	154
Варіант 29.....	155
Варіант 30.....	155
Варіант 31.....	155
Варіант 32.....	155
Варіант 33.....	156
Варіант 34.....	156
Варіант 35.....	156
Варіант 36.....	156
Варіант 37.....	156
Варіант 38.....	156
Варіант 39.....	157
Варіант 40.....	157
11. Бінарні файли.....	158
Приклад виконання лабораторних завдань	158
Лабораторна робота № 11.1. Бінарні файли	161
Мета роботи	161
Питання, які необхідно вивчити та пояснити на захисті.....	161
Оформлення звіту.....	161
Варіанти лабораторних завдань	161
Варіант 1.....	161
Варіант 2.....	162
Варіант 3.....	162
Варіант 4.....	162
Варіант 5.....	162
Варіант 6.....	162
Варіант 7.....	162
Варіант 8.....	162
Варіант 9.....	163
Варіант 10.....	163
Варіант 11.....	163
Варіант 12.....	163
Варіант 13.....	163

Варіант 14.....	163
Варіант 15.....	164
Варіант 16.....	164
Варіант 17.....	164
Варіант 18.....	164
Варіант 19.....	164
Варіант 20.....	164
Варіант 21.....	165
Варіант 22.....	165
Варіант 23.....	165
Варіант 24.....	165
Варіант 25.....	165
Варіант 26.....	165
Варіант 27.....	166
Варіант 28.....	166
Варіант 29.....	166
Варіант 30.....	166
Варіант 31.....	166
Варіант 32.....	166
Варіант 33.....	167
Варіант 34.....	167
Варіант 35.....	167
Варіант 36.....	167
Варіант 37.....	167
Варіант 38.....	167
Варіант 39.....	168
Варіант 40.....	168
Лабораторна робота № 11.2. Послідовний пошук у бінарному файлі	169
Мета роботи	169
Питання, які необхідно вивчити та пояснити на захисті.....	169
Оформлення звіту.....	169
Варіанти лабораторних завдань	169
Варіант 1.....	170
Варіант 2.....	170
Варіант 3.....	170

Варіант 4.....	170
Варіант 5.....	170
Варіант 6.....	171
Варіант 7.....	171
Варіант 8.....	171
Варіант 9.....	171
Варіант 10.....	171
Варіант 11.....	171
Варіант 12.....	171
Варіант 13.....	171
Варіант 14.....	171
Варіант 15.....	172
Варіант 16.....	172
Варіант 17.....	172
Варіант 18.....	172
Варіант 19.....	172
Варіант 20.....	172
Варіант 21.....	172
Варіант 22.....	173
Варіант 23.....	173
Варіант 24.....	173
Варіант 25.....	173
Варіант 26.....	173
Варіант 27.....	173
Варіант 28.....	173
Варіант 29.....	173
Варіант 30.....	173
Варіант 31.....	174
Варіант 32.....	174
Варіант 33.....	174
Варіант 34.....	174
Варіант 35.....	174
Варіант 36.....	174
Варіант 37.....	174
Варіант 38.....	175

Варіант 39.....	175
Варіант 40.....	175
Лабораторна робота № 11.3. Впорядкування та бінарний пошук у бінарному файлі.....	176
Мета роботи	176
Питання, які необхідно вивчити та пояснити на захисті.....	176
Оформлення звіту.....	176
Варіанти лабораторних завдань	176
Варіант 1.....	177
Варіант 2.....	178
Варіант 3.....	178
Варіант 4.....	178
Варіант 5.....	179
Варіант 6.....	179
Варіант 7.....	179
Варіант 8.....	180
Варіант 9.....	180
Варіант 10.....	180
Варіант 11.....	181
Варіант 12.....	181
Варіант 13.....	182
Варіант 14.....	182
Варіант 15.....	182
Варіант 16.....	183
Варіант 17.....	183
Варіант 18.....	184
Варіант 19.....	184
Варіант 20.....	184
Варіант 21.....	185
Варіант 22.....	185
Варіант 23.....	185
Варіант 24.....	186
Варіант 25.....	186
Варіант 26.....	186
Варіант 27.....	187
Варіант 28.....	187

Варіант 29.....	188
Варіант 30.....	188
Варіант 31.....	188
Варіант 32.....	189
Варіант 33.....	189
Варіант 34.....	189
Варіант 35.....	190
Варіант 36.....	190
Варіант 37.....	191
Варіант 38.....	191
Варіант 39.....	191
Варіант 40.....	192
Лабораторна робота № 11.4. Опрацювання бінарного файлу	193
Мета роботи	193
Питання, які необхідно вивчити та пояснити на захисті.....	193
Оформлення звіту.....	193
Варіанти лабораторних завдань	193
Варіант 1.....	194
Варіант 2.....	194
Варіант 3.....	194
Варіант 4.....	195
Варіант 5.....	195
Варіант 6.....	195
Варіант 7.....	196
Варіант 8.....	196
Варіант 9.....	196
Варіант 10.....	196
Варіант 11.....	197
Варіант 12.....	197
Варіант 13.....	197
Варіант 14.....	198
Варіант 15.....	198
Варіант 16.....	198
Варіант 17.....	199
Варіант 18.....	199

Варіант 19.....	199
Варіант 20.....	200
Варіант 21.....	200
Варіант 22.....	201
Варіант 23.....	201
Варіант 24.....	201
Варіант 25.....	202
Варіант 26.....	202
Варіант 27.....	202
Варіант 28.....	203
Варіант 29.....	203
Варіант 30.....	203
Варіант 31.....	203
Варіант 32.....	204
Варіант 33.....	204
Варіант 34.....	204
Варіант 35.....	205
Варіант 36.....	205
Варіант 37.....	205
Варіант 38.....	206
Варіант 39.....	206
Варіант 40.....	206
Лабораторна робота № 11.5. Структури, об'єднання та бінарні файли.....	208
Мета роботи	208
Питання, які необхідно вивчити та пояснити на захисті.....	208
Оформлення звіту.....	208
Варіанти лабораторних завдань	208
Варіант 1.....	209
Варіант 2.....	209
Варіант 3.....	209
Варіант 4.....	210
Варіант 5.....	210
Варіант 6.....	210
Варіант 7.....	210
Варіант 8.....	211

Варіант 9.....	211
Варіант 10.....	211
Варіант 11.....	211
Варіант 12.....	212
Варіант 13.....	212
Варіант 14.....	212
Варіант 15.....	212
Варіант 16.....	213
Варіант 17.....	213
Варіант 18.....	213
Варіант 19.....	214
Варіант 20.....	214
Варіант 21.....	214
Варіант 22.....	214
Варіант 23.....	215
Варіант 24.....	215
Варіант 25.....	215
Варіант 26.....	215
Варіант 27.....	216
Варіант 28.....	216
Варіант 29.....	216
Варіант 30.....	217
Варіант 31.....	217
Варіант 32.....	217
Варіант 33.....	217
Варіант 34.....	218
Варіант 35.....	218
Варіант 36.....	218
Варіант 37.....	218
Варіант 38.....	219
Варіант 39.....	219
Варіант 40.....	219
Питання та завдання для контролю знань	220
Текстові файли.....	220
Бінарні файли.....	221

Файли.....	223
Файлові потоки.....	227
Стан потоку.....	228
Буфер потоку	228
Вказівник файлового потоку (файловий вказівник).....	229
Опрацювання двійкових файлів за допомогою потоків	229
Предметний покажчик	231
Література	233

Вступ

Дисципліна «Алгоритмізація та програмування» вивчається на першому курсі, – з цієї дисципліни розпочинається цикл предметів, що стосуються програмування та розробки програмного забезпечення.

Тема «Файли» продовжує розгляд питань, що стосуються організації введення / виведення, розпочатий при вивченні теми «Лінійні програми» в розділах «Організація введення / виведення засобами мови С» та «Організація введення / виведення засобами мови С++». Тема «Файли» охоплює наступні розділи:

В розділах «Файлове введення / виведення в С та С++» та «Загальні поняття» коротко характеризуються системи введення / виведення в мовах С та С++; описуються поняття файлу, фізичного файлу, файлу операційної системи, логічного файлу, файлової змінної, програмного файлу; файлового введення та виведення; відкриття та закриття файлу; буферу обміну та каналу зв'язку; описується загальна схема опрацювання файлу; наводиться поняття файлового вказівника (вказівника поточної позиції), вказівника введення (вказівника зчитування) та вказівника виведення (вказівника запису) та поняття текстових і бінарних файлів.

В розділі «Опрацювання файлів засобами мови С» розглядаються питання зв'язку потоків та файлів; описуються основи файлової системи: поняття файлової змінної та робота з нею, процедури відкриття та закриття файлів; наводиться опис опрацювання текстових файлів: запис та зчитування символу, перевірка кінця файлу, введення / виведення рядків та «перемотка» файлу; описано порядок опрацювання бінарних файлів: робота з функціями `fread()` та `fwrite()` та організацію прямого доступу до файлу; наводиться опис додаткових можливостей опрацювання файлів: визначення, чи трапилася помилка при виконанні файлової операції; примусового запису потоку та функцій `fprintf()` та `fscanf()`.

Розділ «Файлове введення / виведення в С та С++» містить наступні параграфи: файлові потоки та організація введення / виведення (описано оголошення файлової змінної, відкриття та закриття файлу, режими відкриття файлу, визначення кінця файлу та передавання файлів у функції); опрацювання текстових файлових потоків (відкриття файлу, послідовний перегляд та зчитування текстових файлів, запис у текстовий файл); опрацювання бінарних файлових потоків (відкриття бінарного файлу, зчитування та запис у бінарний файл, навігація по бінарному файлу); додаткові можливості опрацювання файлових потоків (функції управління введенням-виведенням, перевірка помилок при виконанні файлових операцій, визначення та встановлення стану потоку).

Розділ «Операції з фізичними файлами» присвячено операціям вилучення та перейменування файлів.

В розділі «Приклади» розв’язано задачі опрацювання текстового файлу (зчитування файлу, що містить цілі числа, та запис додатних чисел в один файл, від’ємних – в інший); опрацювання бінарного файлу (програмне формування бінарного файлу, що містить цілі числа, за допомогою генератора випадкових чисел та запис додатних чисел в один файл, від’ємних – в інший); наведено рекомендації стосовно впорядкування бінарних файлів та вилучення компонент із текстового та бінарного файлів; описано проблеми, що виникають при скануванні файлів та наведено рекомендації щодо їх подолання.

В посібнику містяться теоретичні відомості, приклади, методичні вказівки з їх розв’язування, питання і завдання з контролю знань; наведено рекомендований набір лабораторних робіт з теми «Файли», приклади їх виконання та прийнятий стиль оформлення C/C++ програм.

Теми 10, 11. Файли

Стисло та головне про файли

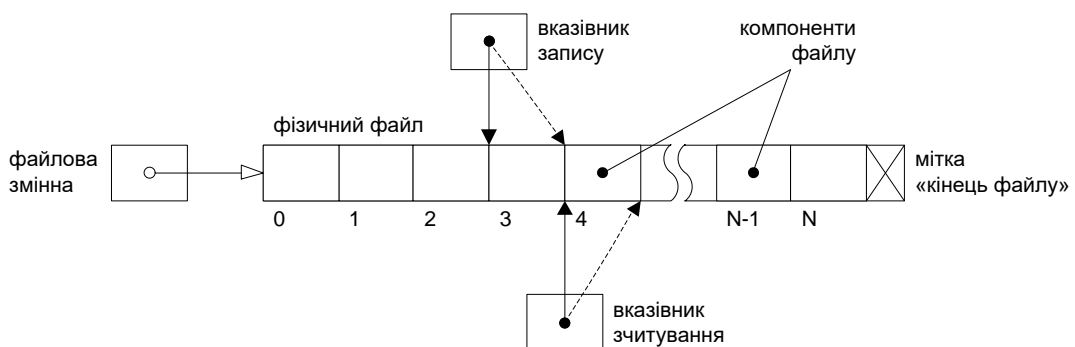
Загальна схема роботи з файлом:

- 1) Оголосити файлову змінну;
- 2) Відкрити файл;
- 3) Виконати необхідні дії з файлом (опрацювання файлу);
- 4) Закрити файл.

Опрацювання файлу

Всі дії з файлом виконуються по-компонентно. Це означає, що в кожний момент часу програма опрацьовує лише одну поточну компоненту файлу. Такий підхід дозволяє економити пам'ять: оскільки внутрішня (оперативна) пам'ять набагато дорожча зовнішньої (дискової), то набагато дешевше в оперативній пам'яті тримати лише ту частину інформації, яку в цей момент опрацьовує програма, а всі решта частин – зберігати на диску.

Для позначення поточної компоненти файлу використовується **файловий вказівник** (або **вказівник поточної позиції**) – службова змінна, яка створюється автоматично при створенні файлової змінної. В деяких системах програмування (Pascal, Delphi, C) файловий вказівник один і той самий для операції запису та для операції зчитування, в інших системах (C++) – є окремі файлові вказівники запису і зчитування. **Файловий вказівник запису (вказівник виведення)** позначає ту компоненту, яка буде поточною при виконанні операції запису, – тобто, дані із програми будуть записуватися саме у цю компоненту. **Файловий вказівник зчитування (вказівник введення)** – позначає компоненту, яка буде поточною при виконанні зчитування, – тобто, дані будуть зчитуватися саме із цієї компоненти. Після виконання операцій зчитування / запису відповідний файловий вказівник автоматично переходить до наступної компоненти файлу.



Текстові та бінарні файли

Всі файли за способом трактування їх вмісту можна поділити на дві категорії: текстові та бінарні.

В **текстових файлах** інформація автоматично конвертується (як і при консольному введенні / виведенні): при виведенні – із внутрішнього представлення до текстового вигляду, при введенні – із текстового подання до внутрішнього представлення. При цьому, текстове (символьне) представлення інформації означає, що всі символи у текстовому файлі поділяються на два види: звичайні символи, які відображаються текстовими редакторами, та форматні символи, які текстовими редакторами не відображаються, а керують способом відображення звичайних символів. Такими форматними символами є мітка «кінець рядка» – пара символів з кодами 13 – «переведення рядка» (перехід до нового рядка) і 10 – «повернення каретки» (перехід на початок рядка): `'\n'` та `'\r'` – результат натискання клавіші **Enter**, та мітка «кінець файлу» – символ з кодом 26 – результат натискання комбінації клавіш **Ctrl+Z**.

Таким чином, вміст текстового файлу – це символи, організовані у логічні рядки.

В **бінарних файлах** інформація подається у внутрішньому представленні – в двійкових кодах, які використовуються комп'ютером. Мітка «кінець файлу» для бінарних файлів – поняття умовне, ніяких спеціальних символів для неї не передбачено, вона реалізується записом у відповідному каталозі реальної довжини файлу.

Опрацювання файлів засобами мови C

Відкриття файлу

Функція `fopen()` відкриває потік, пов'язує з ним певний файл та повертає вказівник на блок управління цим файлом. Далі в цьому розділі файлом будемо називати дисковий файл.

Прототип функції:

```
FILE *fopen(const char *ім'я-файлу, const char *режим);
```

де

ім'я-файлу – вказівник на нуль-термінальний літерний рядок, який описує допустиме в операційній системі ім'я файлу (і може містити шлях до файлу);

режим – вказівник на нуль-термінальний літерний рядок, що описує спосіб, яким буде відкрито цей файл.

Наступна таблиця описує допустимі режими відкриття файлів.

Допустимі значення режимів відкриття файлу	
Режим	Призначення
r	Відкрити текстовий файл для зчитування
w	Створити текстовий файл для запису
a	Відкрити текстовий файл для додавання даних в його кінець
rb	Відкрити двійковий файл для зчитування
wb	Створити двійковий файл для запису
ab	Відкрити двійковий файл для додавання даних в його кінець
r+	Відкрити текстовий файл для зчитування / запису
w+	Створити текстовий файл для зчитування / запису
a+	Відкрити текстовий файл для додавання даних в його кінець або створити текстовий файл для зчитування / запису
r+b	Відкрити двійковий файл для зчитування / запису
w+b	Створити двійковий файл для зчитування / запису
a+b	Відкрити двійковий файл для додавання даних в його кінець або створити двійковий файл для зчитування / запису

Рядки, подібні до **"r+b"**, можна записувати у вигляді **"rb+"**.

Функція `fopen()` повертає файлову змінну – вказівник на блок управління файлом. Ніколи в програмі не слід змінювати його значення! Якщо при відкритті файлу відбувається помилка, то функція `fopen()` поверне порожній вказівник (**NULL**).

Наступний фрагмент демонструє використання функції для відкриття файлу **"1.txt"** для запису:

```
FILE *f;
f = fopen("1.txt", "w");
```

Хоча такий код технічно вірний, проте стилістично краще буде переписати його у вигляді:

```
FILE *f;

if ((f = fopen("1.txt", "w")) == NULL)
{
    cerr << "Помилка при відкритті файлу.\n";
    exit(1);
}
```

Такий спосіб дозволяє виявити помилки, які трапилися при відкритті файлу, призначеного для запису (наприклад, диск захищений від запису або відсутність вільного місця на диску). Завжди потрібно спочатку отримати підтвердження, що файл успішно відкритий, а лиш тоді виконувати з файлом інші операції.

Якщо спробувати відкрити файл лише для зчитування (режим **"r"**), а він не існує, то функція `fopen()` поверне помилку (значення **NULL**).

Якщо спробувати відкрити файл в режимі додавання даних у його кінець (режим **"a"**), а сам цей файл не існує, то він буде створений. Якщо ж такий файл вже існував, то його попередній вміст не буде змінений: нові дані будуть добавлятися у кінець файлу, після тих, які вже існували у файлі.

Якщо файл відкривається для запису (режим **"w"**), і виявляється, що файл не існує, то він буде створений. Якщо такий файл вже існує, то його попередній вміст буде втрачено – файл створюється заново.

Різниця між режимами **"r+"** та **"w+"** полягає в тому, що у випадку, якщо файл не існує, то при відкритті в режимі **"r+"** він не буде створений, а в режимі **"w+"** – буде. Якщо файл вже існував, то відкриття в режимі **"w+"** приведе до втрати його вмісту, а відкриття в режимі **"r+"** залишає вміст файлу неушкодженим.

Закриття файлу

Функція `fclose()` закриває потік, відкритий за допомогою `fopen()`. При цьому: у файл записуються всі дані, які ще залишалися в дисковому буфері; звільняється файлова змінна – блок управління файлом, пов'язаний з цим потоком; файл закривається на рівні операційної системи. Якщо не закрити файл, то це приведе до втрати даних, пошкодження файлів на рівні операційної системи та можливих помилок при виконанні програми.

Прототип функції:

```
int fclose(FILE *файлова-змінна);
```

де

файлова-змінна – вказівник на блок управління файлом, отриманий в результаті виклику `fopen()`.

При успішному виконанні функція повертає 0. У випадку помилки повертається **EOF** (тобто, -1) Точну інформацію про причину помилки можна отримати за допомогою функції `ferror()`, яка буде описана далі. Зазвичай помилки при закритті файлу відбуваються лише тоді, коли диск був видалений із системи, або коли на диску не залишилося вільного місця.

Опрацювання текстових файлів

Запис та зчитування символу

Запис символу – функції `putc()` та `fputc()`

Для запису символу є дві повністю еквівалентні функції: `putc()` та `fputc()`. Їх є дві лише для збереження сумісності із попередніми версіями C.

Функція `putc()` (`fputc()`) записує символи у файл, вже відкритий функцією `fopen()` в режимі запису.

Прототип функції:

```
int putc(int ch, FILE *файлова-змінна);  
int fputc(int ch, FILE *файлова-змінна);
```

де

файлова-змінна — вказівник на блок управління файлу, отриманий в результаті виклику функції `fopen()`;

ch — символ, що записується у файл. Хоча *ch* визначено як ціле число типу `int`, записується лише молодший байт.

Якщо функція `putc()` (`fputc()`) виконана успішно, вона повертає в якості результату записаний символ, в іншому випадку — повертає значення `EOF`. Значення `EOF` визначене як `-1`.

Зчитування символу – функції `getc()` та `fgetc()`

Для зчитування символів є також дві (для збереження сумісності із попередніми версіями C) повністю еквівалентних функції: `getc()` та `fgetc()`.

Функція `getc()` (`fgetc()`) зчитує символи із файлу, вже відкритого функцією `fopen()` в режимі зчитування.

Прототип функції:

```
int getc(FILE *файлова-змінна);  
int fgetc(FILE *файлова-змінна);
```

де

файлова-змінна — вказівник на блок управління файлу, отриманий в результаті виклику функції `fopen()`;

Якщо функція `getc()` (`fgetc()`) виконана успішно, вона повертає значення типу `int`, в молодшому байті якого буде прочитаний символ, а старші байти будуть заповнені нулями.

Якщо досягнуто кінець файлу, то функція `getc()` (`fgetc()`) повертає значення `EOF`, тому введення символів можна організувати так:

```
do {  
    ch = getc(f);  
} while ( ch != EOF );
```

Якщо ж при зчитуванні з файлу трапилася інша помилка, то функція `getc()` (`fgetc()`) теж поверне значення `EOF`. Для отримання інформації про причину помилки, слід використати функцію `ferror()`.

Перевірка кінця файлу – функція `feof()`

Для перевірки, чи досягнуто кінець файлу, використовувати функцію `getc()` (`fgetc()`) — не правильно, бо ця функція повертає `EOF` і у випадку досягнення кінця файлу, і у випадку іншої помилки введення. Крім того, при опрацюванні двійкових файлів може бути прочитане

значення, яке збігається з **EOF**. Тоді програма повідомить, що досягнуто кінець файлу, а насправді цього і не відбулося.

Тому для перевірки досягнення кінця файлу слід використовувати функцію `feof()`. Прототип цієї функції:

```
int feof(FILE *файлова-змінна);
```

Функція повертає ненульове значення **true** (істина) лише при спробі виконати зчитування після того, як буде досягнуто кінець файлу, та нульове значення **false** (хибність) в іншому випадку. Наприклад, якщо файл містить 10 байтів, і ми прочитали 10 байтів із файлу, то функція `feof()` поверне значення 0 (**false**), – оскільки, хоча вказівник поточної позиції знаходиться в кінці файлу, ми ще не зробили спробу зчитування після кінця файлу. Лише при спробі зчитати 11-ий байт функція `feof()` поверне ненульове значення (**true**).

Введення / виведення рядків: функції `fputs()` та `fgets()`

Функції `fgets()` та `fputs()` подібні до функцій `getc()`, `fgetc()` та `putc()`, `fputc()`, лише вони зчитують / записують не символи, а літерні рядки.

Їх прототипи:

```
char *fgets(char *рядок, int довжина, FILE *файлова-змінна);  
int fputs(const char *рядок, FILE *файлова-змінна);
```

Функція `fputs()` записує у файл, з яким пов'язана *файлова-змінна*, літерний рядок *рядок*. У випадку помилки повертає значення **EOF**.

Функція `fgets()` зчитує із потоку, пов'язаного із *файловою-змінною*, символи та записує їх у літерний *рядок*, і робить це до тих пір, поки не буде прочитаний символ нового рядка, або кількість прочитаних символів не стане дорівнювати *довжина* - 1. Якщо прочитано символ нового рядка, то він записується у *рядок*, цим функція `fgets()` відрізняється від `gets()` (введення рядків з клавіатури). Отриманий рядок буде завершуватися нуль-символом **'\0'**. При успішному завершенні функція повертає *рядок*, у випадку помилки – порожній вказівник **NULL**.

Опрацювання бінарних файлів

Функції `fread()` та `fwrite()`

Ці функції дозволяють зчитувати та записувати блоки даних будь-якого типу. Їх прототипи:

```
size_t fread(void *буфер, size_t розмір, size_t кількість, FILE *фз);  
size_t fwrite(const void *буфер, size_t розмір, size_t кількість, FILE *фз);
```

де

- буфер* – вказівник на *елемент даних*:
 для функції `fread()` – область пам'яті, в яку будуть записані дані, прочитані з файлу;
 для функції `fwrite()` – вказівник на область пам'яті, яка містить дані, що будуть записані у файл;
- розмір* – довжина одного елементу даних в байтах (розмір області пам'яті, на яку налаштований вказівник *буфер*);
- кількість* – визначає, скільки елементів даних буде прочитано (`fread`) чи записано (`fwrite`);
- фз* – *файлова змінна* – вказівник на блок управління файлом, пов'язаний із вже відкритим потоком.

Функція `fread()` повертає кількість прочитаних елементів. Якщо досягнуто кінця файлу або трапилася помилка при зчитуванні, то результат функції `fread()` може бути меншим значення *кількість*.

Функція `fwrite()` повертає кількість записаних елементів. Якщо трапилася помилка при записуванні, то результат функції `fwrite()` може бути меншим значення *кількість*.

Прямий доступ: функція `fseek()`

Функція `fseek()` встановлює вказівник поточної позиції. Її прототип:

```
int fseek(FILE *файлова-змінна, long кількість-байт, int початок-відліку);
```

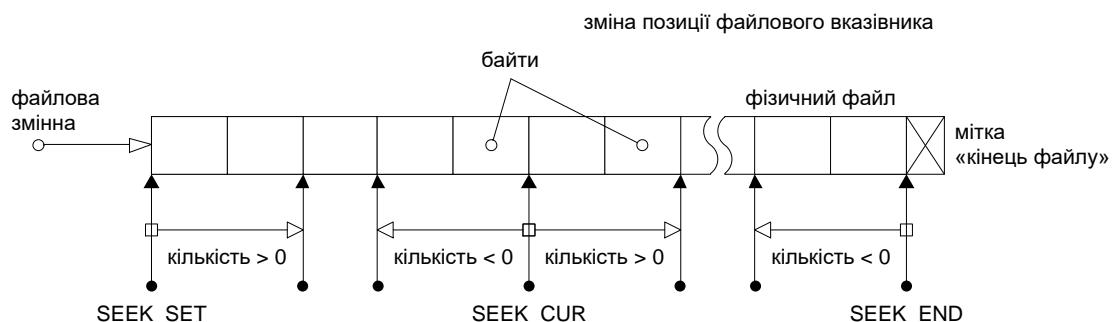
де

- файлова-змінна* – вказівник на блок управління файлом, результат виклику функції `fopen()`;
- кількість-байт* – кількість байт, відносно *початку-відліку*, визначає переміщення вказівника поточної позиції у нове положення;
- початок-відліку* – одне із наступних трьох значень:

Початок відліку	Значення
Початок файлу	<code>SEEK_SET</code>
Поточна позиція	<code>SEEK_CUR</code>
Кінець файлу	<code>SEEK_END</code>

При успішному завершенні функція `fseek()` повертає значення 0.

Наприклад, щоб встановити вказівник поточної позиції на вказану *кількість-байт* від початку файлу, *початок-відліку* має дорівнювати `SEEK_SET`, від поточної позиції – `SEEK_CUR`, від кінця файлу – `SEEK_END`:



Файлові потоки: опрацювання файлів в стилі C++

Крім засобів опрацювання файлів в стилі мови C, мова C++ дозволяє використовувати файлові потоки, реалізовані класами `ifstream`, `ofstream` та `fstream`. Ці класи визначені в заголовному файлі `fstream`, тому для його використання необхідно вказати директиву підключення:

```
#include <fstream>
```

Функції опрацювання файлових потоків, що належать потоковим типам `ifstream`, `ofstream` чи `fstream`, зазвичай викликаються наступним чином:

Для змінної файлового потоку, оголошеної так:

```
ifstream f;
```

або так:

```
ofstream f;
```

або так:

```
fstream f;
```

функції опрацювання потоку викликаються як окрема команда:

```
f.<функція>(<аргументи>);
```

або – як елемент виразу у складі команди:

```
<змінна> = f.<функція>(<аргументи>);
```

або – як елемент виразу у складі умови:

```
if ( f.<функція>(<аргументи>) == <значення> )...
```

Тобто, для виклику функції, яка має опрацювати файловий потік, зазвичай потрібно вказати: ім'я змінної, пов'язаної із файловим потоком (файлової змінної); крапку; ім'я відповідної функції та її аргументи.

Потік – це набір символів, які передаються від одного (довільного) пристрою до іншого (теж довільного) пристрою.

Файлові потоки та організація введення / виведення

Оголошення файлової змінної

Термін *файлова змінна* у цьому розділі означає *змінну файлового потоку*, або скорочено *файловий потік*.

В загальному випадку оголошення файлової змінної виглядає наступним чином:

```
файловий-потіковий-тип файлова-змінна;
```

Файлова змінна, що позначатиме файл, який можна і писати і читати, оголошується, наприклад, так:

```
fstream f;
```

Файлова змінна, що позначатиме файл, який можна лише читати, оголошується, наприклад, так:

```
ifstream fin;
```

Файлова змінна, що позначатиме файл, який можна лише писати, оголошується, наприклад, так:

```
ofstream fout;
```

Відкриття файлу

Після оголошення файлової змінної її необхідно пов'язати з конкретним фізичним файлом, тобто – **відкрити файл**, це виконується за допомогою функції `open()` (в квадратних дужках записано необов'язкові аргументи функції):

```
файлова-змінна.open("ім'я-файлу"[, режим-відкриття-файлу]);
```

де *"ім'я-файлу"* – нуль-термінальний літерний рядок, що містить ім'я файлу, записане згідно правил операційної системи.

Наприклад, для вище оголошених файлових змінних, команди відкриття відповідних файлів можуть бути такими:

```
f.open("1.txt");  
fin.open("2.txt");  
fout.open("3.txt");
```

Команди оголошення файлової змінної та відкриття файлу можна поєднати (в квадратних дужках записано необов'язкові аргументи):

```
файловий-потіковий-тип файлова-змінна("ім'я-файлу"[, режим-відкриття-файлу]);
```

Для вищенаведених прикладів альтернативою буде:

```
fstream f("1.txt");  
ifstream fin("2.txt");  
ofstream fout("3.txt");
```

Якщо файл відкрити не вдалося, то файлова змінна буде повертати значення 0:

```

if ( ! f )
{
    cout << "Файл не відкрито!" << endl;
    return;
}

```

Перевірити успішність відкриття файлу можна також за допомогою функції `is_open()`, яка повертає 1, якщо файлову змінну вдалося пов'язати з відкритим файлом:

```

if ( ! f.is_open() )
{
    cout << "Файл не відкрито!" << endl;
    return;
}

```

Закриття файлу

Після завершення дій з файлом його необхідно **закрити**, це виконує функція `close()`:

```

файлова-змінна.close();

```

Для файлових змінних, описаних у попередніх прикладах:

```

f.close();
fin.close();
fout.close();

```

Коли ми закриваємо файл, то всі дані, які програма записувала у цей файл, скидаються на диск, і операційна система оновлює запис в каталозі для цього файлу.

Якщо файлова змінна – локальна, тобто оголошена у деякому блоці, то при виході з блоку така змінна буде знищена (як і кожна локальна змінна). При цьому автоматично закривається файл, пов'язаний із цією змінною.

Таким чином, замість фрагменту коду

```

{
    fstream f;
    f.open("1.txt");
    ...                // опрацювання файлу
    ...                // опрацювання файлу
    f.close();
}

```

можна використовувати наступний фрагмент:

```

{
    fstream f("1.txt");
    ...                // опрацювання файлу
    ...                // опрацювання файлу
}

```

Режими відкриття файлу

При відкритті файлу можна вказати другий аргумент – режим відкриття.

Режим відкриття файлу визначається константою або комбінацією констант.

Константи режимів наведені в наступній таблиці:

Режими відкриття та їх призначення

Режим відкриття	Призначення
<code>ios::app</code>	Відкриває файл в режимі додавання, файловий вказівник розміщується в кінці файлу.
<code>ios::ate</code>	Розміщує файловий вказівник в кінці файлу.
<code>ios::binary</code>	Відкрити файл як бінарний.
<code>ios::in</code>	Відкриває файл для введення (зчитування).
<code>ios::nocreate</code>	Якщо вказаний файл не існує – то не створювати нового файлу і повернути помилку.
<code>ios::noreplace</code>	Якщо вказаний файл існує – то не замінювати його, операція відкриття файлу має бути перервана і має повернути помилку.
<code>ios::out</code>	Відкрити файл для виведення (запису).
<code>ios::trunc</code>	Перезаписати вміст існуючого файлу.

Якщо необхідно поєднати кілька констант режимів відкриття файлу, то це слід робити за допомогою бітової (розрядної) операції | «або».

За умовчанням, з файловими потоковими типами пов'язані певні режими:

```
fstream f(const char *name, ios::openmode mode = ios::in | ios::out);
```

```
ifstream f(const char *name, ios::openmode mode = ios::in);
```

```
ofstream f(const char *name, ios::openmode mode = ios::out | ios::trunc);
```

– тобто, поточковий тип `fstream` дозволяє одночасно читати і писати файл (режим `ios::in | ios::out`); поточковий тип `ifstream` дозволяє лише читати файл (режим `ios::in`), при цьому файл має існувати; а поточковий тип `ofstream` дозволяє лише писати файл (режим `ios::out | ios::trunc`), при цьому: якщо файл не існував, то він буде створений, а якщо існував – то його вміст буде очищено (`ios::trunc`).

Якщо файл потрібно відкрити в режимі до-запису у кінець файлу, то це можна робити так:

```
fstream f("FName.txt", ios::out | ios::app);
```

або так:

```
ofstream f("FName.txt", ios::app);
```

Є різниця між режимами відкриття `ios::ate` та `ios::app`.

Якщо файл відкривається в режимі додавання (`ios::app`), то все виведення у файл буде здійснюватися в позицію, що починається з поточного положення кінця файлу, не залежно від операцій позиціонування файлових вказівників. В режимі `ios::ate` (з англ. «at end» – з кінця) можна буде змінити позицію виведення у файл та здійснювати запис даних, починаючи з цієї позиції.

Для потоків виведення (`ofstream`) режим відкриття за умовчанням визначений як `ios::out` | `ios::trunc`, тобто можна не вказувати режим відсікання (`ios::trunc`) файлу. Проте для потоків введення-виведення (`fstream`) цей режим необхідно вказувати явно.

Файли, які відкриваються для виведення (`ios::out`), створюються, якщо вони ще не існують.

Визначення кінця файлу

Типовою файловою операцією є зчитування вмісту файлу, поки не буде досягнуто кінець файлу. Для визначення кінця файлу можна використовувати функцію `eof()` файлового потоку. Ця функція повертає значення 0, якщо ще не досягнуто кінця файлу, та 1, якщо досягнуто кінець файлу. Використовуючи цикл `while`, програма може зчитувати вміст файлу, поки не буде досягнуто кінець файлу:

```
while ( ! f.eof() )    // поки не кінець файлу
{
    ...                // опрацювання файлу
}
```

Цикл виконується, поки функція `eof()` повертає значення 0 (хибність).

Опрацювання текстових файлових потоків

Відкриття файлу

Відкриття файлового потоку введення (тобто, відкриття файлу для зчитування):

```
ifstream fin("1.txt");
```

Відкриття файлового потоку виведення (тобто, відкриття файлу для запису):

```
ofstream fout("1.txt");
```

Відкриття файлового потоку виведення в режимі до-запису у кінець файлу (тобто, відкриття файлу для до запису у кінець):

```
ofstream f("1.txt", ios::app);
```

Відкриття файлового потоку введення-виведення (тобто, відкриття файлу для зчитування і запису):

```
fstream fin("1.txt");
```

Коли файл відкривається в текстовому режимі, відбувається наступне:

- при введенні (зчитуванні) даних з файлу кожна пара символів `'\r' + '\n'` (повернення каретки + переведення рядка) перетворюється у символ переведення рядка (`'\n'`);

- при виведенні (запису) даних у файл кожний символ *переведення рядка* (`'\n'`) перетворюється у пару символів `'\r' + '\n'` (*повернення каретки + переведення рядка*).

Послідовний перегляд та зчитування текстового файлу

Всі команди потокового введення з клавіатури можна застосовувати і для потокового зчитування з файлу. При відкриванні файлу вказівник зчитування встановлюється на початок файлу. При зчитуванні даних з файлу важливою є перевірка досягнення мітки «кінець файлу» – операція зчитування виконується до тих пір, поки не буде досягнуто кінець файлу.

Запис у текстовий файл

Подібно до того, як всі команди потокового введення з клавіатури виконуються і для потокового введення (зчитування) з файлу, так і всі команди потокового виведення на екран виконуються і для потокового виведення (запису) у файл. Для управління форматом виведення слід підключити файл заголовків маніпуляторів потоку:

```
#include <iomanip>
```

Опрацювання бінарних файлових потоків

Відкриття бінарного файлу

Для відкриття бінарного файлового потоку слід вказати режим `ios::binary`, наприклад:

```
fstream f("1.dat", ios::binary);
```

Оскільки зазвичай з одним і тим самим бінарним файлом виконують операції і зчитування і запису в одному сеансі роботи, то бінарні файли, як правило, відкривають за допомогою потоку введення-виведення `fstream`.

Зчитування з бінарного файлу

Загальний вигляд команди зчитування даних з бінарного файлу:

```
файлова-змінна.read(адреса-області-пам'яті, sizeof(область-пам'яті));
```

де:

файлова-змінна – файлова змінна, пов'язана із відкритим бінарним файлом;

адреса-області-пам'яті – вказівник на змінну (адреса змінної), значення якої буде зчитане з файлу;

`sizeof(область-пам'яті)` – розмір змінної – тої області пам'яті, яка буде заповнена зчитаними з файлу даними; це ж – і розмір зчитаних з файлу даних.

Дані зчитуються з тої компоненти файлу, яка позначена позицією поточного значення файлового вказівника зчитування. При відкриванні файлу вказівник зчитування встановлюється на початок файлу. При зчитуванні даних з файлу важливою є перевірка досягнення мітки «кінець файлу».

Оскільки потік визначається як послідовність символів, що передаються від одного довільного пристрою до іншого, то для зчитування даних із бінарного файлу необхідно привести адресу змінної (якщо це – не символьна змінна) до типу «вказівник на символ», наприклад:

```
int x;  
f.read((char*) &x, sizeof(x));
```

Запис у бінарний файл

Загальний вигляд команди запису даних у бінарний файл:

```
файлова-змінна.write(адреса-області-пам'яті, sizeof(область-пам'яті));
```

де:

файлова-змінна – файлова змінна, пов'язана із відкритим бінарним файлом;

адреса-області-пам'яті – вказівник на змінну (адреса змінної), значення якої буде записане у файл;

sizeof(область-пам'яті) – розмір змінної – тої області пам'яті, вміст якої буде записаний у файл.

Дані записуються у ту компоненту файлу, яка позначена позицією поточного значення файлового вказівника запису. Якщо в процесі запису нові дані перекрили мітку «кінець файлу», то розмір файлу автоматично збільшується – при закриванні файлу мітка «кінець файлу» буде автоматично записана після всіх даних. При відкриванні файлу вказівник запису встановлюється на початок файлу. Якщо бінарний файл був відкритий в режимі «відсікання» попереднього вмісту (*ios::trunc*) – то після завершення файлових операцій – при закриванні файлу – в поточну позицію вказівника запису записується мітка «кінець файлу», – це приводить до втрати попереднього вмісту файлу.

Оскільки потік визначається як послідовність символів, що передаються від одного довільного пристрою до іншого, то для запису даних у бінарний файл необхідно привести адресу змінної (якщо це – не символьна змінна) до типу «вказівник на символ», наприклад:

```
int x;  
f.write((char*) &x, sizeof(x));
```

Прямий доступ: навігація по бінарному файлу

Навігація по бінарному файлу здійснюється за допомогою функцій позиціонування (встановлюють позицію у потоці) та функцій отримання позиції (повертають значення поточної позиції).

Функції `seekg()` та `seekp()` використовуються для позиціонування вхідного та вихідного файлового потоку відповідно (тобто, для встановлення позиції файлових вказівників зчитування та запису). Позиція файлового вказівника вказується в байтах.

Прототипи функцій позиціонування:

```
istream& seekg(long pos);  
istream& seekg(long offset, seek_dir dir);  
ostream& seekp(long pos);  
ostream& seekp(long offset, seek_dir dir);
```

де:

`pos` – абсолютна позиція у файлі відносно початку файлу – вказується в байтах;

`offset` – зміщення в байтах у файлі відносно позиції, яка визначається параметром `dir`; якщо `offset > 0`, то зміщення відбувається вперед, в напрямку до кінця файлу; якщо `offset < 0`, то зміщення відбувається назад, в напрямку до початку файлу.

`dir` – визначає позицію, відносно якої виконується зміщення.

Параметр `dir` може набувати значень відповідно до визначення перелічуваного типу `seek_dir` в файлі заголовків `ios`:

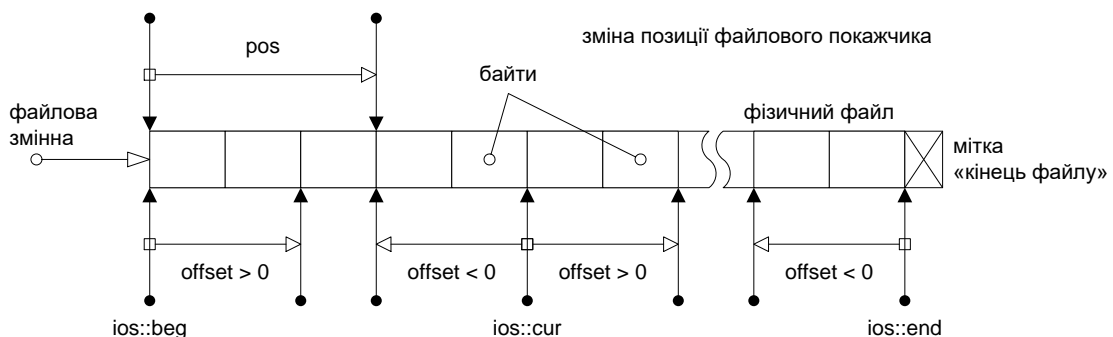
```
enum seek_dir { beg, cur, end };
```

де константи переліку визначають:

`ios::beg` – зміщення виконується від початку файлу;

`ios::cur` – зміщення виконується відносно поточної позиції файлового вказівника;

`ios::end` – зміщення виконується від кінця файлу.



Функції `tellg()` та `tellp()` дозволяють отримати значення поточної позиції в байтах у файловому потоці введення чи виведення (тобто, значення номеру позиції файлового

вказівника зчитування та запису, відповідно). Результат цих функцій – кількість байт від початку файлу до поточної позиції файлового вказівника.

Прототипи цих функцій:

```
long tellg();  
long tellp();
```

Теоретичні відомості

Файлове введення / виведення в С та С++

Мова С – це фундамент С++, тому С++ підтримує всю файлову систему С. Тобто, при перенесенні старого коду із С на С++ не потрібно змінювати процедури введення-виведення. В С++ є своя власна система введення-виведення, яка повністю підтримує всі можливості аналогічної системи С. При написанні програм на мові С++ зазвичай більш зручно використовувати саме її систему введення-виведення, проте якщо необхідно, то можна використати систему мови С.

Загальні поняття

Часто буває потрібно зберігати дані між сеансами роботи програми. В таких випадках використовуються **файли**.

Насамперед розглянемо терміни: файл, файловий буфер, канал зв'язку із файлом, текстовий файл, бінарний файл, фізичний файл, файл операційної системи, логічний файл, програмний файл, файлова змінна, файловий вказівник зчитування, файловий вказівник запису.

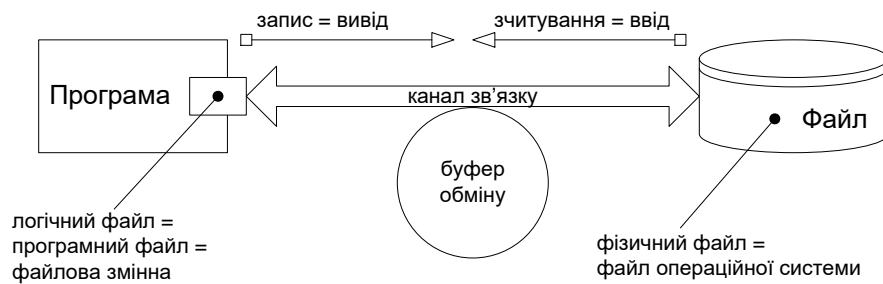
Перше значення терміну **файл – фізичний файл**: або **файл операційної системи**. Це – сукупність даних, яка зберігається на зовнішньому носії та має своє ім'я (записане згідно правил операційної системи).

Друге значення терміну **файл – файлова змінна, логічний файл** або **програмний файл**. Це – представник фізичного файлу у програмі, що містить службову інформацію, необхідну для опрацювання файлу.

Аналогічно до консольного введення / виведення, термін **виведення** означає запис у файл, а термін **введення** – зчитування з файлу.

Якщо програма має зчитувати дані з файлу чи записувати дані у файл, то такий файл необхідно **відкрити**. При цьому виділяється **канал зв'язку** із файлом та утворюються системні **буфери** для обміну інформацією із файлом.

Буфер обміну (або **файловий буфер**) – це область пам'яті, яка виділяється операційною системою і призначена для накопичення інформації, що має бути записана у файл, чи інформації, зчитаної з файлу. Потреба в буфері обміну зумовлена необхідністю підвищення швидкодії файлових операцій зчитування / запису: взаємодія із зовнішніми носіями відбувається значно повільніше, ніж дії з пам'яттю.



Кожна команда запису даних приводить до того, що дані записуються у буфер, а не безпосередньо у файл. Попереднє накопичення даних в буфері і запис у файл даних за допомогою однієї операції виконується швидше, ніж багатократні операції запису кожної порції інформації безпосередньо у файл. Справа в тому, що багатократні операції запису безпосередньо у файл вимагають виконання підготовчих та завершальних дій обслуговування дискової системи для кожної операції, тоді як при буферизованому виведенні ці дії виконуються лише один раз, коли при заповненні буферу дані із нього автоматично переносяться у файл. Такими підготовчими діями є: ввімкнення двигуна, розкручування пакету дисків до утворення повітряної подушки та позиціонування пакету магнітних головок. Відповідні завершальні дії – це: переведення пакету магнітних головок у безпечне положення та вимкнення двигуна.

Аналогічно при виконанні операції зчитування: для прискорення виконання цієї операції із файлу зчитується великий блок і заноситься у буфер. Коли програмі необхідно прочитати дані, то ці дані необхідними для програми невеликими порціями зчитуються з буферу, а не безпосередньо з файлу.

Після завершення опрацювання файлу його необхідно **закрити**. При цьому:

- 1) вміст буферу обміну примусово переноситься на зовнішній носій, тобто – у файл;
- 2) знищується буфер – звільняється область пам'яті, що була виділена для системного буферу обміну;
- 3) звільняється канал зв'язку – розривається зв'язок між файловою змінною та фізичним файлом;
- 4) після останньої компоненти файлу записується мітка «кінець файлу».

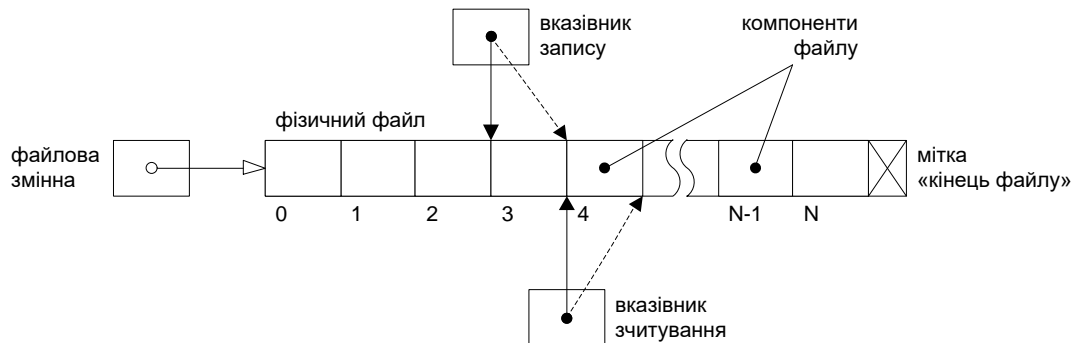
Загальна схема роботи з файлом::

- 1) Оголосити файлову змінну;
- 2) Відкрити файл;
- 3) Виконати необхідні дії з файлом (опрацювання файлу);
- 4) Закрити файл.

Опрацювання файлу

Всі дії з файлом виконуються по-компонентно. Це означає, що в кожний момент часу програма опрацьовує лише одну поточну компоненту файлу. Такий підхід дозволяє економити пам'ять: оскільки внутрішня (оперативна) пам'ять набагато дорожча зовнішньої (дискової), то набагато дешевше в оперативній пам'яті тримати лише ту частину інформації, яку в цей момент опрацьовує програма, а всі решта частин – зберігати на диску.

Для позначення поточної компоненти файлу використовується **файловий вказівник** (або **вказівник поточної позиції**) – службова змінна, яка створюється автоматично при створенні файлової змінної. В деяких системах програмування (Pascal, Delphi, C) файловий вказівник один і той самий для операції запису та для операції зчитування, в інших системах (C++) – є окремі файлові вказівники запису і зчитування. **Файловий вказівник запису (вказівник виведення)** позначає ту компоненту, яка буде поточною при виконанні операції запису, – тобто, дані із програми будуть записуватися саме у цю компоненту. **Файловий вказівник зчитування (вказівник введення)** – позначає компоненту, яка буде поточною при виконанні зчитування, – тобто, дані будуть зчитуватися саме із цієї компоненти. Після виконання операцій зчитування/запису відповідний файловий вказівник автоматично переходить до наступної компоненти файлу.



Текстові та бінарні файли

Всі файли за способом трактування їх вмісту можна поділити на дві категорії: текстові та бінарні.

В **текстових файлах** інформація автоматично конвертується (як і при консольному введенні / виведенні): при виведенні – із внутрішнього представлення до текстового вигляду, при введенні – із текстового подання до внутрішнього представлення. При цьому, текстове (символьне) представлення інформації означає, що всі символи у текстовому файлі поділяються на два види: звичайні символи, які відображаються текстовими редакторами, та форматні символи, які текстовими редакторами не відображаються, а керують способом відображення звичайних символів. Такими форматними символами є мітка «кінець рядка» –

пара символів з кодами 13 – «переведення рядка» (перехід до нового рядка) і 10 – «повернення каретки» (перехід на початок рядка): '\n' та '\r' – результат натискання клавіші **Enter**, та мітка «кінець файлу» – символ з кодом 26 – результат натискання комбінації клавіш **Ctrl+Z**.

Таким чином, вміст текстового файлу – це символи, організовані у логічні рядки.

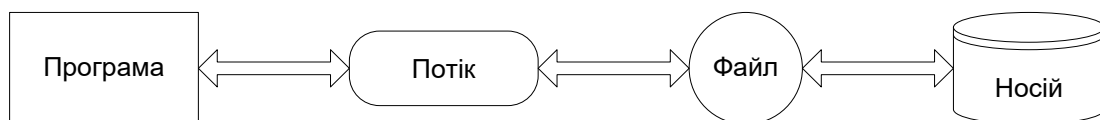
В **бінарних файлах** інформація подається у внутрішньому представленні – в двійкових кодах, які використовуються комп'ютером. Мітка «кінець файлу» для бінарних файлів – поняття умовне, ніяких спеціальних символів для неї не передбачено, вона реалізується записом у відповідному каталозі реальної довжини файлу.

Опрацювання файлів засобами мови C

В цьому розділі описуються засоби файлового введення-виведення, які описані в стандарті мови C.

Потоки та файли

Перед початком вивчення файлової системи мови C слід зрозуміти, в чому полягає різниця між *потоками* та *файлами*. В системі введення-виведення C для програм підтримується єдиний інтерфейс, який не залежить від того, до якого конкретного пристрою здійснюється доступ. Тобто, в цій системі між програмою та пристроєм розміщується щось більш загальне, ніж сам пристрій. Такий узагальнений пристрій введення або виведення (пристрій більш високого рівня абстракції) називається *поток*ом, а сам цей конкретний пристрій – називається *файлом* (хоча файл – теж абстрактне поняття). Важливо розуміти, яким чином відбувається взаємодія потоків та файлів.



Потоки

Файлова система мови C призначена для роботи із різноманітними пристроями (зокрема, терміналами, дисковими тощо). Ці фізичні пристрої сильно відрізняються один від іншого, проте буферизована файлова система представляє їх у вигляді логічного пристрою, який називається потоком. Всі потоки – однотипні, оскільки вони не залежать від фізичних пристроїв, тому та сама функція, яка виконує виведення на консоль, може виконувати запис до дискового файлу. Потоки бувають двох видів: текстові та бінарні (двійкові).

Текстові потоки

Текстовий потік – це послідовність символів. Згідно стандарту C, текстовий потік організований у вигляді рядків, кожний з яких закінчується символом нового рядка; в кінці останнього рядка цей символ – не обов'язковий. В текстовому потоці можуть виконуватися певні перетворення символів, – наприклад, символ нового рядка може замінюватися парою символів – повернення каретки (поняття ще з епохи друкарських машинок, означає перехід на початок рядка) та переведення рядка (це означає – перехід до нового рядка). Тому може і не бути однозначної відповідності між символами, які записуються (зчитуються), і тими, які зберігаються на зовнішньому носії. Окрім цього, кількість тих символів, які записуються

(зчитуються), і тих, які зберігаються на зовнішньому носії, можуть не збігатися із-за можливих перетворень (таких, як автоматичне перетворення числових даних із двійкового внутрішнього представлення до відповідного текстового зображення).

Двійкові (бінарні) потоки

Двійковий потік – це послідовність байтів, яка взаємно однозначно відповідає байтам на зовнішньому носії, причому ніяких перетворень символів не відбувається. Кількість байтів, які записуються (зчитуються), і тих, які зберігаються на зовнішньому носії, теж однакова. Проте в кінці двійкового потоку може додаватися певна кількість нульових байтів, які використовуються для заповнення вільного місця в блоці пам'яті, щоб точно заповнити сектор диску.

Файли

В мові C файлом може бути все що завгодно, починаючи з дискового файлу і закінчуючи терміналом чи принтером. Потік пов'язують із певним файлом, виконуючи операцію *відкриття*. Як тільки файл відкрито, можна виконувати обмін інформацією між ним та програмою.

Проте не у всіх файлів однакові можливості. Наприклад, до дискового файлу можливий прямий доступ, а до багатьох видів принтерів – ні. Таким чином, можна сформулювати важливий принцип стосовно системи введення-виведення мови C: всі потоки однакові, а файли – ні.

Якщо файл підтримує *запити на місцеположення (вказівник поточної позиції)*, то при відкритті такого файлу *вказівник поточної позиції* встановлюється на початок файлу. При зчитуванні з файлу (або запису до файлу) кожного символу *вказівник поточної позиції* збільшується, забезпечуючи тим самим просування по файлу.

Файл від'єднується від потоку (тобто, розривається зв'язок між файлом та потоком) за допомогою операції *закриття*. При закриванні файлу, відкритого для виведення (з метою запису), вміст пов'язаного з ним потоку записується на зовнішній пристрій. Цей процес, який називають *примусовим звільненням вмісту буферу* або *дописуванням потоку*, гарантує, що жодні дані не залишаться випадково в буфері диску. Якщо програма завершується нормально (тобто, коли функція `main()` повертає управління операційній системі, або коли викликається функція `exit()`), то всі файли закриваються автоматично. У випадку аварійного завершення програми (у випадку неполадки або коли викликається функція `abort()`), файли не закриваються.

Кожний потік, пов'язаний з файлом, має певну структуру – *блок управління файлом*, що містить інформацію про файл; тип цієї структури – **FILE**. Блок управління файлом – невелика область пам'яті, яка виділяється операційною системою для зберігання інформації про відкритий файл, і зазвичай містить інформацію про ідентифікатор файлу, його місце розміщення на диску та вказівник поточної позиції у файлі.

Основна мета розмежування файлів та потоків – забезпечити єдиний інтерфейс. Для виконання всіх операцій введення-виведення використовуються потоки, які не залежать від реалізації файлів. Використання потоків означає на логічному рівні застосування лише одної файлової системи: введення-виведення від кожного пристрою автоматично перетворюється системою введення-виведення у потік, яким програмі дуже легко керувати.

Основи файлової системи

Файлова система мови C складається із набору пов'язаних між собою функцій. Найбільш поширені з них подані в наступній таблиці. Для роботи з ними потрібно підключити файл заголовку `<stdio.h>`.

Деякі функції файлової системи C	
Ім'я	Призначення
<code>fopen()</code>	Відкриває файл
<code>fclose()</code>	Закриває файл
<code>putc()</code>	Записує символ у файл
<code>fputc()</code>	Те ж саме, що і <code>putc()</code>
<code>getc()</code>	Зчитує символ з файлу
<code>fgetc()</code>	Те ж саме, що і <code>getc()</code>
<code>fgets()</code>	Зчитує рядок з файлу
<code>fputs()</code>	Записує рядок у файл
<code>fseek()</code>	Встановлює вказівник поточної позиції на певний байт файлу
<code>ftell()</code>	Повертає значення вказівника поточної позиції у файлі
<code>fprintf()</code>	Для файлу те саме, що <code>printf()</code> для консолі
<code>fscanf()</code>	Для файлу те саме, що <code>scanf()</code> для консолі
<code>feof()</code>	Повертає значення true (істина), якщо досягнуто кінець файлу
<code>ferror()</code>	Повертає значення true , якщо трапилася помилка
<code>rewind()</code>	Встановлює вказівник поточної позиції на початок файлу
<code>rename()</code>	Перейменовує фізичний файл
<code>remove()</code>	Знищує фізичний файл
<code>fflush()</code>	Допишує потік у файл (примусово звільняє буфер)

У файлі заголовку `<stdio.h>` описано прототипи функцій введення-виведення та визначено три типи: `size_t`, `fpos_t` та `FILE`. `size_t` та `fpos_t` – різновиди цілого без-знакового типу, `FILE` описано в наступному параграфі. Крім того, в `<stdio.h>` описані значення `NULL`, `EOF`, `FOPEN_MAX`, `SEEK_SET`, `SEEK_CUR` та `SEEK_END`. `NULL` – порожній (`null`) вказівник. `EOF`, часто визначається як `-1`, – це значення, яке повертається тоді, коли робиться спроба виконати зчитування після кінці файлу. `FOPEN_MAX` визначає максимальну кількість одночасно відкритих файлів. Інші значення використовуються разом з функцією `fseek()`, яка здійснює операції прямого доступу до файлу.

Файлова змінна – вказівник на структуру `FILE`

Файлова змінна – це те, що поєднує в єдине ціле усю систему введення-виведення мови C. В цьому розділі терміни *файлова змінна*, *вказівник на структуру `FILE`* та *вказівник на блок управління файлом* – означають те саме. Цей вказівник налаштований на структуру, яка містить різноманітну інформацію про файл, зокрема, його ім'я, статус та вказівник поточної позиції у файлі. Терміни *файловий вказівник* та *вказівник поточної позиції* – будемо в цьому розділі вважати еквівалентними.

Файлова змінна визначає конкретний файл і використовується відповідним потоком при виконанні операцій введення-виведення: для виконання над файлами операцій зчитування і запису програми мають використовувати відповідні файлові змінні, пов'язані з цими файлами.

Оголошення файлової змінної має такий загальний вигляд:

```
FILE* файлова-змінна;
```

Наприклад:

```
FILE* f;
```

Відкриття файлу

Функція `fopen()` відкриває потік, пов'язує з ним певний файл та повертає вказівник на блок управління цим файлом. Далі в цьому розділі файлом будемо називати дисковий файл.

Прототип функції:

```
FILE *fopen(const char *ім'я-файлу, const char *режим);
```

де

ім'я-файлу – вказівник на нуль-термінальний літерний рядок, який описує допустиме в операційній системі ім'я файлу (і може містити шлях до файлу);

режим – вказівник на нуль-термінальний літерний рядок, що описує спосіб, яким буде відкрито цей файл.

Наступна таблиця описує допустимі режими відкриття файлів.

Допустимі значення режимів відкриття файлу	
Режим	Призначення
r	Відкрити текстовий файл для зчитування
w	Створити текстовий файл для запису
a	Відкрити текстовий файл для додавання даних в його кінець
rb	Відкрити двійковий файл для зчитування
wb	Створити двійковий файл для запису
ab	Відкрити двійковий файл для додавання даних в його кінець
r+	Відкрити текстовий файл для зчитування / запису
w+	Створити текстовий файл для зчитування / запису
a+	Відкрити текстовий файл для додавання даних в його кінець або створити текстовий файл для зчитування / запису
r+b	Відкрити двійковий файл для зчитування / запису
w+b	Створити двійковий файл для зчитування / запису
a+b	Відкрити двійковий файл для додавання даних в його кінець або створити двійковий файл для зчитування / запису

Рядки, подібні до **"r+b"**, можна записувати у вигляді **"rb+"**.

Функція `fopen()` повертає файлову змінну – вказівник на блок управління файлом. Ніколи в програмі не слід змінювати його значення! Якщо при відкритті файлу відбувається помилка, то функція `fopen()` поверне порожній вказівник (**NULL**).

Наступний фрагмент демонструє використання функції для відкриття файлу **"1.txt"** для запису:

```
FILE *f;
f = fopen("1.txt", "w");
```

Хоча такий код технічно вірний, проте стилістично краще буде переписати його у вигляді:

```
FILE *f;

if ((f = fopen("1.txt", "w")) == NULL)
{
    cerr << "Помилка при відкритті файлу.\n";
    exit(1);
}
```

Такий спосіб дозволяє виявити помилки, які трапилися при відкритті файлу, призначеного для запису (наприклад, диск захищений від запису або відсутність вільного місця на диску). Завжди потрібно спочатку отримати підтвердження, що файл успішно відкритий, а лиш тоді виконувати з файлом інші операції.

Якщо спробувати відкрити файл лише для зчитування (режим **"r"**), а він не існує, то функція `fopen()` поверне помилку (значення **NULL**).

Якщо спробувати відкрити файл в режимі додавання даних у його кінець (режим "a"), а сам цей файл не існує, то він буде створений. Якщо ж такий файл вже існував, то його попередній вміст не буде змінений: нові дані будуть добавлятися у кінець файлу, після тих, які вже існували у файлі.

Якщо файл відкривається для запису (режим "w"), і виявляється, що файл не існує, то він буде створений. Якщо такий файл вже існує, то його попередній вміст буде втрачено – файл створюється заново.

Різниця між режимами "r+" та "w+" полягає в тому, що у випадку, якщо файл не існує, то при відкритті в режимі "r+" він не буде створений, а в режимі "w+" – буде. Якщо файл вже існував, то відкриття в режимі "w+" приведе до втрати його вмісту, а відкриття в режимі "r+" залишає вміст файлу неушкодженим.

Файл можна відкрити або в одному із текстових, або в одному із двійкових режимів. В більшості реалізацій в текстових режимах кожна комбінація кодів повернення каретки (ASCII 13) та кінця рядка (ASCII 10) перетворюється при введенні (зчитуванні з файлу) у символ нового рядка ('\n'). При виведенні (запису у файл) відбувається зворотне перетворення: символи нового рядка ('\n') перетворюються в комбінацію кодів повернення каретки (ASCII 13) та кінця рядка (ASCII 10). В двійкових режимах такі перетворення не виконуються.

Максимальна кількість одночасно відкритих файлів визначається значенням `FOPEN_MAX`. Зазвичай це значення не менше 8, точно можна довідатися в налаштуваннях операційної системи (параметр `FILES`) та в документації системи програмування.

Закриття файлу

Функція `fclose()` закриває потік, відкритий за допомогою `fopen()`. При цьому: у файл записуються всі дані, які ще залишалися в дисковому буфері; звільняється файлова змінна – блок управління файлом, пов'язаний з цим потоком; файл закривається на рівні операційної системи. Якщо не закрити файл, то це приведе до втрати даних, пошкодження файлів на рівні операційної системи та можливих помилок при виконанні програми.

Прототип функції:

```
int fclose(FILE *файлова-змінна);
```

де

файлова-змінна – вказівник на блок управління файлом, отриманий в результаті виклику `fopen()`.

При успішному виконанні функція повертає 0. У випадку помилки повертається `EOF` (тобто, -1). Точну інформацію про причину помилки можна отримати за допомогою функції

`ferror()`, яка буде описана далі. Зазвичай помилки при закритті файлу відбуваються лише тоді, коли диск був видалений із системи, або коли на диску не залишилося вільного місця.

Опрацювання текстових файлів

Запис та зчитування символу

Запис символу – функції `putc()` та `fputc()`

Для запису символу є дві повністю еквівалентні функції: `putc()` та `fputc()`. Їх є дві лише для збереження сумісності із попередніми версіями C.

Функція `putc()` (чи `fputc()`) записує символи у файл, вже відкритий функцією `fopen()` в режимі запису.

Прототипи функцій:

```
int putc(int ch, FILE *файлова-змінна);  
int fputc(int ch, FILE *файлова-змінна);
```

де

файлова-змінна – вказівник на блок управління файлу, отриманий в результаті виклику функції `fopen()`;

ch – символ, що записується у файл. Хоча *ch* визначено як ціле число типу `int`, записується лише молодший байт.

Якщо функція `putc()` (`fputc()`) виконана успішно, вона повертає в якості результату записаний символ, в іншому випадку – повертає значення `EOF`. Значення `EOF` визначене як `-1`.

Зчитування символу – функції `getc()` та `fgetc()`

Для зчитування символів є також дві (для збереження сумісності із попередніми версіями C) повністю еквівалентних функції: `getc()` та `fgetc()`.

Функція `getc()` (`fgetc()`) зчитує символи із файлу, вже відкритого функцією `fopen()` в режимі зчитування.

Прототип функції:

```
int getc(FILE *файлова-змінна);  
int fgetc(FILE *файлова-змінна);
```

де

файлова-змінна – вказівник на блок управління файлу, отриманий в результаті виклику функції `fopen()`;

Якщо функція `getc()` (`fgetc()`) виконана успішно, вона повертає значення типу `int`, в молодшому байті якого буде прочитаний символ, а старші байти будуть заповнені нулями.

Якщо досягнуто кінець файлу, то функція `getc()` (`fgetc()`) повертає значення `EOF`, тому введення символів можна організувати так:


```
do {
    ch = getc(f);
} while ( ch != EOF );
```

Якщо ж при зчитуванні з файлу трапилася інша помилка, то функція `getc()` (`fgetc()`) теж поверне значення `EOF`. Для отримання інформації про причину помилки, слід використати функцію `ferror()`.

Приклад використання функцій `fopen()`, `getc()`, `putc()` та `fclose()`

В наступній програмі символи зчитуються з клавіатури та записуються у дисковий файл до тих пір, поки користувач не введе знак проценту. Ім'я файлу вводить користувач.

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <stdio.h>

using namespace std;

int main()
{
    FILE *f;
    char ch;

    char fname[61];
    cout << "Enter file name: ";
    cin.getline(fname, sizeof(fname));

    if ((f = fopen(fname, "w")) == NULL)
    {
        cerr << "Error opening file '" << fname << "'" << endl;
        exit(1);
    }

    cout << "Enter your text (enter '%' to cancel): " << endl;
    do {
        ch = getchar();
        putc(ch, f);
    } while (ch != '%');

    fclose(f);

    return 0;
}
```

Наступна програма зчитує та виводить на екран по одному символу вміст текстового файлу, ім'я якого вводить користувач.

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <stdio.h>

using namespace std;

int main()
{
    FILE* f;
    char ch;
```

```

const int LEN = 61;
char fname[LEN];
cout << "Enter file name: ";
cin.getline(fname, LEN);

if ((f = fopen(fname, "r")) == NULL)
{
    cerr << "Error opening file '" << fname << "'" << endl;
    exit(1);
}

ch = getc(f);

while (ch != EOF)
{
    cout << ch;
    ch = getc(f);
}

fclose(f);

return 0;
}

```

Перевірка кінця файлу – функція `feof()`

Для перевірки, чи досягнуто кінець файлу, використовувати функцію `getc()` (`fgetc()`) – не правильно, бо ця функція повертає `EOF` і у випадку досягнення кінця файлу, і у випадку іншої помилки введення. Крім того, при опрацюванні двійкових файлів може бути прочитане значення, яке збігається з `EOF`. Тоді програма повідомить, що досягнуто кінець файлу, а насправді цього і не відбулося.

Тому для перевірки досягнення кінця файлу слід використовувати функцію `feof()`. Прототип цієї функції:

```
int feof(FILE *файлова-змінна);
```

Функція повертає ненульове значення `true` (істина) лише при спробі виконати зчитування після того, як буде досягнуто кінець файлу, та нульове значення `false` (хибність) в іншому випадку.

Наприклад, якщо файл містить 10 байтів, і ми прочитали 10 байтів із файлу, то функція `feof()` поверне значення `0` (`false`), – оскільки, хоча вказівник поточної позиції знаходиться в кінці файлу, ми ще не зробили спробу зчитування після кінця файлу. Лише при спробі зчитати 11-ий байт функція `feof()` поверне ненульове значення (`true`).

Наступний фрагмент зчитує символи з файлу до тих пір, поки не буде досягнуто кінець файлу:

```

while ( !feof(f) )
    ch = getc(f);

```

Цей спосіб можна застосовувати і до текстових, і до двійкових файлів.

Наступна програма копіює файли. Файли відкриваються в двійковому режимі, функція `feof()` перевіряє чи не досягнуто кінець файлу:

```
// Копіювання файлу
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <stdio.h>

using namespace std;

int main()
{
    FILE *in, *out;
    char inName[61], outName[61];

    cout << "Enter input file name: ";
    cin.getline(inName, sizeof(inName));

    if ((in = fopen(inName, "rb")) == NULL)
    {
        cerr << "Error opening input file" << endl;
        exit(1);
    }

    cout << "Enter output file name: ";
    cin.getline(outName, sizeof(outName));

    if ((out = fopen(outName, "wb")) == NULL)
    {
        cerr << "Error opening output file" << endl;
        exit(1);
    }

    char ch;

    // Цей код копіює файл
    while ( !feof(in) )
    {
        ch = getc(in);
        if ( !feof(in) )
            putc(ch, out);
    }

    fclose(in);
    fclose(out);

    return 0;
}
```

Введення / виведення рядків: функції `fputs()` та `fgets()`

Функції `fgets()` та `fputs()` подібні до функцій `getc()` (`fgetc()`) та `putc()` (`fputc()`), лише вони зчитують / записують не символи, а літерні рядки.

Їх прототипи:

```
char *fgets(char *рядок, int довжина, FILE *файлова-змінна);
int fputs(const char *рядок, FILE *файлова-змінна);
```

Функція `fputs()` записує у файл, з яким пов'язана *файлова-змінна*, літерний *рядок*. У випадку помилки повертає значення `EOF`.

Функція `fgets()` зчитує із потоку, пов'язаного із *файловою-змінною*, символи та записує їх у літерний *рядок*, і робить це до тих пір, поки не буде прочитаний символ нового рядка, або кількість прочитаних символів не стане дорівнювати *довжина* - 1. Якщо прочитано символ нового рядка, то він записується у *рядок*, цим функція `fgets()` відрізняється від `gets()` (введення рядків з клавіатури). Отриманий рядок буде завершуватися нуль-символом `'\0'`. При успішному завершенні функція повертає *рядок*, у випадку помилки – порожній вказівник `NULL`.

Наступна програма зчитує рядки з клавіатури і записує їх у файл. Для завершення роботи слід ввести порожній рядок. Оскільки функція `getline()` не записує символу відокремлення рядків, його приходится вставляти перед кожним рядком, що записується у файл:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <iostream>

using namespace std;

int main()
{
    const int LEN = 80;
    char s[LEN];
    FILE* f;

    char fname[61];
    cout << "Enter file name: "; cin.getline(fname, sizeof(fname));

    if ((f = fopen(fname, "w")) == NULL)
    {
        cerr << "Error opening file '" << fname << "'" << endl;
        exit(1);
    }

    do {
        cout << "Enter string (empty - to exit): ";
        cin.getline(s, LEN - 1); // на 1 менше, щоб було місце для '\n'
        strcat(s, "\n");        // додавання символу нового рядка
        fputs(s, f);
    } while (*s != '\n');

    return 0;
}
```

Функція `rewind()`

Ця функція встановлює вказівник поточної позиції на початок файлу (виконує «перемотування» файлу). Її прототип:

```
int rewind(FILE *файлова-змінна);
```

Модифікуємо попередню програму, щоб вона одразу після створення виводила вміст файлу, для цього після завершення введення програма «перемотує» файл до початку та зчитує його вміст. Тепер файл потрібно відкрити в режимі "w+" – зчитування-запису:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <iostream>

using namespace std;

int main()
{
    const int LEN = 80;
    char s[LEN];
    FILE* f;

    char fname[61];
    cout << "Enter file name: "; cin.getline(fname, sizeof(fname));

    if ((f = fopen(fname, "w+")) == NULL)
    {
        cerr << "Error opening file '" << fname << "'" << endl;
        exit(1);
    }

    do {
        cout << "Enter string (empty - to exit): ";
        cin.getline(s, LEN - 1); // на 1 менше, щоб було місце для '\n'
        strcat(s, "\n");        // додавання символу нового рядка
        fputs(s, f);
    } while (*s != '\n');

    rewind(f); // "перемотали" файл до початку

    while (!feof(f)) // поки не кінець файлу
    {
        fgets(s, LEN - 1, f); // зчитуємо файл і
        cout << s;           // виводимо його вміст на екран
    }

    return 0;
}
```

Опрацювання бінарних файлів

Функції fread() та fwrite()

Ці функції дозволяють зчитувати та записувати блоки даних будь-якого типу. Їх прототипи:

```
size_t fread(void *буфер, size_t розмір, size_t кількість, FILE *фз);
size_t fwrite(const void *буфер, size_t розмір, size_t кількість, FILE *фз);
```

де

буфер – вказівник на елемент даних:

для функції `fread()` – область пам'яті, в яку будуть записані дані, прочитані з файлу;

для функції `fwrite()` – вказівник на область пам'яті, яка містить дані, що будуть записані у файл;

розмір – довжина одного елементу даних в байтах (розмір області пам'яті, на яку налаштований вказівник *буфер*);

кількість – визначає, скільки елементів даних буде прочитано (`fread`) чи записано (`fwrite`), і це – не кількість дублікатів даних! – див. Приклад 1;

фз – *файлова змінна* – вказівник на блок управління файлом, пов'язаний із вже відкритим потоком.

Функція `fread()` повертає кількість прочитаних елементів. Якщо досягнуто кінця файлу або трапилася помилка при зчитуванні, то результат функції `fread()` може бути меншим значення *кількість*.

Функція `fwrite()` повертає кількість записаних елементів. Якщо трапилася помилка при записуванні, то результат функції `fwrite()` може бути меншим значення *кількість*.

Приклад 1

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <stdio.h>

using namespace std;

int main()
{
    char fname[61];
    cout << "Enter file name: "; cin >> fname;

    FILE* f;
    f = fopen(fname, "wb");

    int x = 5;
    fwrite(&x, sizeof(x), 3, f); // 3 - це не кількість екземплярів змінної x:
                                // починаючи з адреси &x оперативної пам'яті
                                // буде зчитано три значення типу int та
                                // записано у файл

    fclose(f);

    f = fopen(fname, "rb");

    while (!feof(f))
    {
        if (fread(&x, sizeof(x), 1, f) != 1)
            if (feof(f))
            {
                cerr << "Error reading file." << endl;
                break;
            }
    }
```

```

        cout << "x = " << x << endl;
    }
    fclose(f);

    return 0;
}

```

```

Microsoft Visual Studio Debug Console
Enter file name: 1.dat
x = 5
x = -858993460
x = -858993460
Error reading file.

C:\Users\Віктор\source\repos\Project5\Debug\Project5.exe (process 9320) exited with code 0.
Press any key to close this window . . .

```

– записано значення змінної `x`, вказано кількість = 3.

Але в оперативній пам'яті починаючи з адреси `&x` розміщено лише одну просту змінну `x`, там немає даних для ще двох значень типу `int`. Тому у файл потрапили «невстановлені» значення, що показує наступний цикл зчитування з файлу.

Приклад 2

```

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <stdio.h>
using namespace std;

struct Employee
{
    char name[64];
    int age;
    float salary;
};

void enter_save(char* fname);
void load_print(char* fname);

int main()
{
    char fname[61];
    char ch;
    do
    {
        cout << endl;
        cout << "Select:" << endl;
        cout << "[1] - enter & save data;" << endl;
        cout << "[2] - load & print data;" << endl << endl;
        cout << "[0] - exit." << endl << endl;
        cout << "Your choise: "; cin >> ch;

        switch (ch)
        {
            case '0':
                break;
            case '1':
                cin.get(); // очищуємо буфер клавіатури – щоб не було символу
                cin.sync(); // "кінець рядка", який залишився після вводу числа

```

```

        cout << "Enter file name: "; cin.getline(fname, sizeof(fname));
        enter_save(fname);
        break;
    case '2':
        cin.get(); // очищуємо буфер клавіатури – щоб не було символу
        cin.sync(); // "кінець рядка", який залишився після вводу числа
        cout << "Enter file name: "; cin.getline(fname, sizeof(fname));
        load_print(fname);
        break;
    default:
        cout << "Error input! ";
    }
} while (ch != '0');

return 0;
}

void enter_save(char* fname)
{
    FILE* f;
    if ((f = fopen(fname, "wb")) == NULL)
    {
        cerr << "Error opening file '" << fname << "'" << endl;
        return;
    }

    Employee worker;

    char ch;
    do
    {
        cout << endl;
        cout << "name:   "; cin.sync(); cin >> worker.name;
        cout << "age:    "; cin >> worker.age;
        cout << "salary: "; cin >> worker.salary;

        if (fwrite(&worker, sizeof(Employee), 1, f) != 1)
        {
            cerr << "Error writing file." << endl;
            return;
        }

        cout << "Continue? (Y/N) "; cin >> ch;
    } while (ch == 'Y' || ch == 'y');

    fclose(f);
}

void load_print(char* fname)
{
    FILE* f;
    if ((f = fopen(fname, "rb")) == NULL)
    {
        cerr << "Error opening file '" << fname << "'" << endl;
        return;
    }

    Employee worker;

    while (!feof(f))
    {
        if (fread(&worker, sizeof(Employee), 1, f) != 1)

```



```

        if (feof(f))
        {
            cerr << "Error reading file." << endl;
            return;
        }

        cout << endl;
        cout << "name:  " << worker.name << endl;
        cout << "age:   " << worker.age << endl;
        cout << "salary: " << worker.salary << endl;
    }
    fclose(f);
}

```

Прямий доступ: функції `fseek()` та `ftell()`

Функція `fseek()` встановлює вказівник поточної позиції. Її прототип:

```
int fseek(FILE *файлова-змінна, long кількість-байт, int початок-відліку);
```

де

файлова-змінна – вказівник на блок управління файлом, результат виклику функції

`fopen()`;

кількість-байт – кількість байт, відносно *початку-відліку*, визначає переміщення

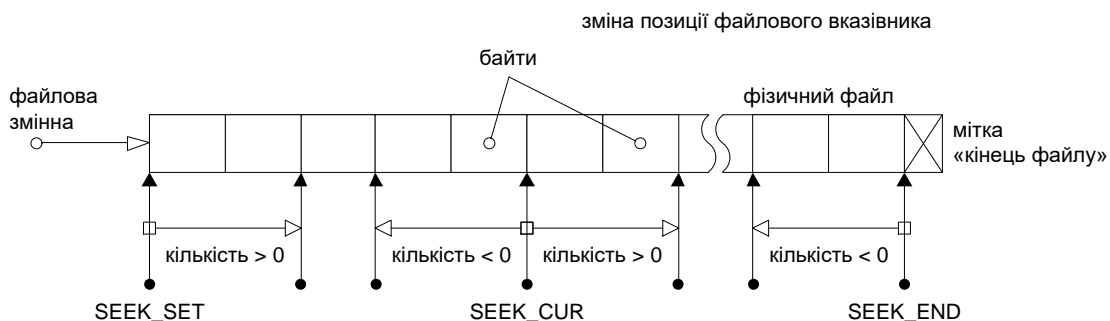
вказівника поточної позиції у нове положення;

початок-відліку – одне із наступних трьох значень:

Початок відліку	Значення
Початок файлу	<code>SEEK_SET</code>
Поточна позиція	<code>SEEK_CUR</code>
Кінець файлу	<code>SEEK_END</code>

При успішному завершенні функція `fseek()` повертає значення 0.

Наприклад, щоб встановити вказівник поточної позиції на вказану *кількість-байт* від початку файлу, *початок-відліку* має дорівнювати `SEEK_SET`, від поточної позиції – `SEEK_CUR`, від кінця файлу – `SEEK_END`:



Приклад:

```
// Навігація по файлу
```

```

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <stdio.h>

using namespace std;

int main()
{
    char FileName[61];

    cout << "Enter file name: ";
    cin.getline(FileName, sizeof(FileName));

    FILE* f1;
    if ((f1 = fopen(FileName, "wb")) == NULL)
    {
        cerr << "Error opening file with write mode." << endl;
        return 1;
    }

    for (int i = 1; i <= 100; i++)
        fwrite(&i, sizeof(int), 1, f1);

    fclose(f1);

    FILE* f2;
    if ((f2 = fopen(FileName, "rb")) == NULL)
    {
        cerr << "Error opening file with read mode." << endl;
        return 1;
    }

    int i;

    fseek(f2, 0 * (long)sizeof(int), SEEK_SET); // 1
    fread(&i, sizeof(int), 1, f2);             // 2
    cout << i << endl;

    fseek(f2, 4 * (long)sizeof(int), SEEK_SET); // 3
    fread(&i, sizeof(int), 1, f2);             // 4
    cout << i << endl;

    fseek(f2, 1 * (long)sizeof(int), SEEK_CUR); // 5
    fread(&i, sizeof(int), 1, f2);             // 6
    cout << i << endl;

    fseek(f2, -5 * (long)sizeof(int), SEEK_CUR); // 7
    fread(&i, sizeof(int), 1, f2);             // 8
    cout << i << endl;

    fseek(f2, -1 * (long)sizeof(int), SEEK_END); // 9
    fread(&i, sizeof(int), 1, f2);             // 10
    cout << i << endl;

    fclose(f2);

    return 0;
}

```

Виведення:

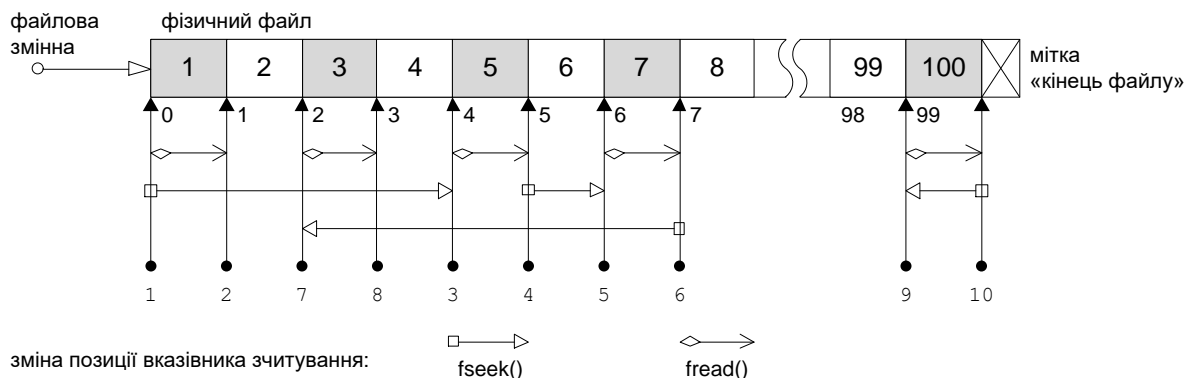
Enter file name: 1.dat

1
5
7
3
100

Програма демонструє навігацію по файлу за допомогою функції встановлення позиції файлового вказівника зчитування `fseek()` та автоматичне переміщення файлового вказівника після виконання операції введення `fread()`:

Спочатку (після введення користувачем імені файлу) у вказаний файл записуються 100 цілих чисел, потім цей файл закривається (що забезпечує перенесення даних з буферу у фізичний файл) і повторно відкривається для зчитування.

Перша команда позиціонування (`// 1`) встановлює файловий вказівник зчитування на найпершу компоненту файлу – компоненту з позицією 0 від початку файлу (оскільки після відкриття файлу вказівник встановлюється на початок файлу, то цю команду можна не вказувати – результат буде той самий). Наступна команда (`// 2`) зчитує значення першої компоненти файлу (1), при цьому автоматично переміщується вказівник зчитування – на компоненту з позицією 1.



Команда `// 3` встановлює вказівник зчитування на компоненту з позицією 4 від початку файлу, а наступна команда `// 4` – зчитує значення цієї компоненти (5), при цьому автоматично переміщується вказівник зчитування – на компоненту з позицією 5.

Команда `// 5` переводить вказівник зчитування від поточної позиції на 1 позицію в напрямку «вперед» – до кінця файлу, тобто – на компоненту з позицією 6 ($=5+1$). Потім команда `// 6` зчитує значення цієї компоненти (7), при цьому автоматично переміщується вказівник зчитування – на компоненту з позицією 7.

Команда `// 7` переводить вказівник зчитування від поточної позиції на 5 позицій в напрямку «назад» – до початку файлу, тобто – на компоненту з позицією 2 ($=7-5$). Потім команда `// 8` зчитує значення цієї компоненти (3), при цьому автоматично переміщується вказівник зчитування – на компоненту з позицією 3.

Команда `// 9` встановлює вказівник зчитування на 1 позицію в напрямку «назад» – від кінця до початку файлу, тобто – на компоненту з позицією 99 ($=100-1$). Потім команда `// 10` зчитує значення цієї компоненти (100), при цьому автоматично переміщується вказівник зчитування – в кінець файлу, тобто в позицію з номером 100.

При навігації по файлу слід номер позиції множити на довжину компоненти – бо функції навігації вказують зміщення в байтах, а не в кількостях компонент.

У цьому прикладі номер позиції множиться на `sizeof(int)` – оскільки розглядається файл цілих чисел. Отримане значення слід привести до типу `long`, за це відповідає конструкція `(long)sizeof(int)`, – бо зміщення вказується в байтах відносно позиції, яка визначається другим аргументом функції `fseek()`.

Поточне значення вказівника поточної позиції у файлі можна визначити за допомогою функції `ftell()`. Її прототип:

```
long ftell(FILE *файлова-змінна);
```

Ця функція повертає значення вказівника поточної позиції у файлі, пов'язаному з *файловою-змінною*. При невдалому виклику вона повертає -1.

Зазвичай прямий доступ використовується лише для двійкових файлів. Проте, якщо файл містить лише текст, то все рівно за необхідності цей файл можна відкрити і в двійковому режимі. Ніякі обмеження, пов'язані із вмістом файлу, операцій прямого доступу не стосуються. Ці обмеження стосуються лише файлів, відкритих в *текстовому режимі*.

Додаткові можливості опрацювання файлів

Функція `ferror()`

Ця функція визначає, чи трапилася помилка при виконанні файлової операції. Її прототип:

```
int ferror(FILE *файлова-змінна);
```

Вона повертає ненульове значення `true` (істина), якщо при останній операції з файлом трапилася помилка, та 0 – значення `false` (хибність) в іншому випадку. Після кожної файлової операції стан помилки оновлюється, тому функцію `ferror()` слід викликати одразу після підозрілої операції, інакше дані про помилку будуть втрачені.

Примусовий запис потоку – функція `fflush()`

Ця функція використовується для примусового запису вихідного потоку у файл. Її прототип:

```
int fflush(FILE *файлова-змінна);
```

Вона записує всі дані, що знаходяться у буфері до файлу, пов'язаного із *файловою-змінною*. Якщо викликати функцію `fflush()` з аргументом `null` (порожній вказівник замість *файлової-змінної*), то будуть примусово записані дані з буферів у всі відкриті для запису файли. Після успішного виконання функція `fflush()` повертає 0, в іншому випадку – значення `EOF`.

Функції `fprintf()` та `fscanf()`

Крім основних функцій файлового введення-виведення, які розглядалися в попередніх параграфах, мова C має також функції `fprintf()` та `fscanf()`, які призначені для форматного файлового запису та зчитування даних різних типів. Ці функції аналогічні функціям `printf()` та `scanf()`, призначеним для форматного консольного виведення та введення відповідно.

Прототипи функцій `fprintf()` та `fscanf()` наступні:

```
int fprintf(FILE *файлова-змінна, const char *керівний-рядок, ...);
int fscanf(FILE *файлова-змінна, const char *керівний-рядок, ...);
```

де

файлова-змінна – вказівник на блок управління файлом, отриманий в результаті виклику функції `fopen()`.

Приклад: програма, яка зчитує з клавіатури рядок та ціле значення і записує їх у файл на диску. Потім програма зчитує дані з файлу та виводить їх на екран, причому вміст файлу можна переглядати текстовим редактором.

```
// використання fscanf() та fprintf()
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <io.h>
#include <stdlib.h>

int main()
{
    FILE* f;
    char s[80];
    int t;

    if ((f = fopen("1.txt", "w")) == NULL)
    {
        printf("Error opening file.\n");
        exit(1);
    }

    printf("Enter string and integer value: ");
    fscanf(stdin, "%s%d", s, &t);    // введення з клавіатури

    fprintf(f, "%s %d", s, t);      // запис у файл
    fclose(f);

    if ((f = fopen("1.txt", "r")) == NULL)
    {
        printf("Error opening file\n");
    }
}
```

```

        exit(1);
    }

    fscanf(f, "%s%d", s, &t);        // зчитування з файлу
    fprintf(stdout, "%s %d", s, t); // виведення на екран

    return 0;
}

```

Попередження: може здатися, що записувати дані різних типів у файли та зчитувати їх з файлів найлегше за допомогою функцій `fprintf()` та `fscanf()`, проте це – не ефективно, бо в текстовому вигляді дані записуються із перетвореннями, тому ці функції потребують додаткових ресурсів та більше часу на виконання. Якщо слід враховувати розмір файлу та час виконання програми, то більш ефективним буде використовувати функції `fwrite()` та `fread()`.

Стандартні потоки

Для програми на мові C, на початку її виконання автоматично відкриваються три потоки: `stdin` – стандартний потік введення, `stdout` – стандартний потік виведення та `stderr` – стандартний потік помилок. Зазвичай ці потоки пов'язані з консоллю, проте їх можна пере-направити на інший пристрій. `stdin` використовується для введення з консолі, а `stdout` та `stderr` – для виведення на консоль.

Оскільки стандартні потоки – це вказівники на блоки управління відповідними файлами (файлові змінні), то вони можуть використовуватися системою файлового введення-виведення і для введення-виведення на консоль (як в попередній програмі).

В якості файлових змінних потоки `stdin`, `stdout` та `stderr` можна використовувати в будь-якій програмі, де використовується змінна типу `FILE*`.

Не слід забувати, що `stdin`, `stdout` та `stderr` – це не файлові змінні в звичайному значенні цього слова (це – вже визначені та відкриті потоки), тому їх не можна відкривати функцією `fopen()` та закривати функцією `fclose()`.

Пере-направлення потоків: функція `freopen()`

Для пере-направлення потоків можна використати функцію `freopen()`. Вона пов'язує потік з новим файлом. Її прототип:

```
FILE *freopen(const char *ім'я-файлу, const char *режим, FILE *потік);
```

де

ім'я-файлу – вказівник на рядок, що містить ім'я файлу, який потрібно пов'язати з потоком, на який вказує вказівник *потік*.

Файл відкривається в режимі *режим*, цей параметр може набувати всіх тих значень, що і відповідний параметр функції `foren()`. Як і функція `foren()`, якщо функція `freopen()` виконалася успішно, то вона повертає *потік*, якщо ж були помилки, – то значення `NULL`.

Файлові потоки: опрацювання файлів в стилі C++

Крім засобів опрацювання файлів в стилі мови C, мова C++ дозволяє використовувати файлові потоки, реалізовані класами `ifstream`, `ofstream` та `fstream`. Ці класи визначені в заголовному файлі `fstream`, тому для його використання необхідно вказати директиву підключення:

```
#include <fstream>
```

Тема «Класи» не вивчається в рамках дисципліни «Алгоритмізація та програмування», тому файлові потокові типи `ifstream`, `ofstream` та `fstream` у цьому розділі описані дещо спрощено – так, щоб ними можна було користуватися, не маючи знань з об'єктно-орієнтованого програмування.

Функції опрацювання файлових потоків, що належать потоковим типам `ifstream`, `ofstream` чи `fstream` зазвичай викликаються наступним чином:

Для змінної файлового потоку, оголошеної так:

```
ifstream f;
```

або так:

```
ofstream f;
```

або так:

```
fstream f;
```

функції опрацювання потоку викликаються як окрема команда:

```
f.<функція>(<аргументи>);
```

або – як елемент виразу у складі команди:

```
<змінна> = f.<функція>(<аргументи>);
```

або – як елемент виразу у складі умови:

```
if ( f.<функція>(<аргументи>) == <значення> )...
```

Тобто, для виклику функції, яка має опрацювати файловий потік, зазвичай потрібно вказати: ім'я змінної, пов'язаної із файловим потоком (файлової змінної); крапку; ім'я відповідної функції та її аргументи.

Потік – це набір символів, які передаються від одного (довільного) пристрою до іншого (теж довільного) пристрою.

У цьому розділі терміни *файловий потоковий тип*, *потіковий тип*, *тип файлового потоку*, *тип потоку*, *файловий потоковий клас*, *клас файлового потоку*, *потіковий клас*,

клас потоку – еквівалентні. Так само в цьому розділі еквівалентними є терміни *файлова змінна*, *файловий потік*, *змінна файлового потоку*. Для того, щоб розрізнити файлові потоки, які дозволяють виконувати операції зчитування, запису, зчитування і запису, використовують терміни *файлові потоки введення*, *файлові потоки виведення*, *файлові потоки введення-виведення* відповідно.

Файлові потоки та організація введення / виведення

В цьому розділі будемо розглядати лише засоби файлового потокового введення / виведення (тобто, засоби мови C++). Засоби, які надає мова C стосовно опрацювання файлів – в цьому розділі розглядатися не будуть (як і в попередніх темах, коли розглядалися лише засоби мови C++ організації консольного введення / виведення).

Для доступу до необхідних ресурсів потрібно підключити файл заголовку `<fstream>`, який описує засоби файлового потокового введення / виведення. В цьому файлі, зокрема, описано файлові потокові типи:

- `fstream` (file stream – англ.: файловий потік) – дозволяє як зчитувати, так і записувати дані (файловий потік введення-виведення).
- `ifstream` (input file stream – англ.: потік файлового введення) – дозволяє лише зчитувати дані з файлу.
- `ofstream` (output file stream – англ.: потік файлового виведення) – дозволяє лише записувати дані у файл.

Оголошення файлової змінної

Термін *файлова змінна* в цьому розділі означає змінну *файлового потоку*, або скорочено *файловий потік*.

В загальному випадку оголошення файлової змінної виглядає наступним чином:

файловий-потіковий-тип файлова-змінна;

Файлова змінна, що позначатиме файл, який можна і писати і читати, оголошується, наприклад, так:

```
fstream f;
```

Файлова змінна, що позначатиме файл, який можна лише читати, оголошується, наприклад, так:

```
ifstream fin;
```

Файлова змінна, що позначатиме файл, який можна лише писати, оголошується, наприклад, так:

```
ofstream fout;
```

Відкриття файлу

Після оголошення файлової змінної її необхідно пов'язати з конкретним фізичним файлом, тобто – **відкрити файл**, це виконується за допомогою функції `open()` (в квадратних дужках записано необов'язкові аргументи функції):

```
файлова-змінна.open("ім'я-файлу"[, режим-відкриття-файлу]);
```

де

"ім'я-файлу" – нуль-термінальний літерний рядок, що містить ім'я файлу, записане згідно правил операційної системи.

Наприклад, для вище оголошених файлових змінних, команди відкриття відповідних файлів можуть бути такими:

```
f.open("1.txt");  
fin.open("2.txt");  
fout.open("3.txt");
```

Команди оголошення файлової змінної та відкриття файлу можна поєднати (в квадратних дужках записано необов'язкові аргументи):

```
файловий-потоківий-тип файлова-змінна("ім'я-файлу"[, режим-відкриття-файлу]);
```

Для вищенаведених прикладів альтернативою буде:

```
fstream f("1.txt");  
ifstream fin("2.txt");  
ofstream fout("3.txt");
```

Якщо файл відкрити не вдалося, то файлова змінна буде повертати значення 0:

```
if ( ! f )  
{  
    cout << "Файл не відкрито!" << endl;  
    return;  
}
```

Перевірити успішність відкриття файлу можна також за допомогою функції `is_open()`, яка повертає 1, якщо файлову змінну вдалося пов'язати з відкритим файлом:

```
if ( ! f.is_open() )  
{  
    cout << "Файл не відкрито!" << endl;  
    return;  
}
```

Закриття файлу

Після завершення дій з файлом його необхідно **закрити**, це виконує функція `close()`:

```
файлова-змінна.close();
```

Для файлових змінних, описаних у попередніх прикладах:

```
f.close();  
fin.close();  
fout.close();
```

Коли ми закриваємо файл, то всі дані, які програма записувала у цей файл, скидаються на диск, і операційна система оновлює запис в каталозі для цього файлу.

Якщо файлова змінна – локальна, тобто оголошена у деякому блоці, то при виході з блоку така змінна буде знищена (як і кожна локальна змінна). При цьому автоматично закривається файл, пов'язаний із цією змінною.

Таким чином, замість фрагменту коду

```
{  
    fstream f;  
    f.open("1.txt");  
    ...           // опрацювання файлу  
    ...           // опрацювання файлу  
    f.close();  
}
```

можна використовувати наступний фрагмент:

```
{  
    fstream f("1.txt");  
    ...           // опрацювання файлу  
    ...           // опрацювання файлу  
}
```

Режими відкриття файлу

При відкритті файлу можна вказати другий аргумент – режим відкриття.

Режим відкриття файлу визначається константою або комбінацією констант.

Константи режимів наведені в наступній таблиці:

Режими відкриття та їх призначення

Режим відкриття	Призначення
<code>ios::app</code>	Відкриває файл в режимі додавання, файловий вказівник розміщується в кінці файлу.
<code>ios::ate</code>	Розміщує файловий вказівник в кінці файлу.
<code>ios::binary</code>	Відкрити файл як бінарний.
<code>ios::in</code>	Відкриває файл для введення (зчитування).
<code>ios::nocreate</code>	Якщо вказаний файл не існує – то не створювати нового файлу і повернути помилку.
<code>ios::noreplace</code>	Якщо вказаний файл існує – то не замінювати його, операція відкриття файлу має бути перервана і має повернути помилку.
<code>ios::out</code>	Відкрити файл для виведення (запису).
<code>ios::trunc</code>	Перезаписати вміст існуючого файлу.

Якщо необхідно поєднати кілька констант режимів відкриття файлу, то це слід робити за допомогою бітової (розрядної) операції | «або».

За умовчанням, з файловими потоковими типами пов'язані певні режими:

```

fstream f(const char *name, ios::openmode mode = ios::in | ios::out);

ifstream f(const char *name, ios::openmode mode = ios::in);

ofstream f(const char *name, ios::openmode mode = ios::out | ios::trunc);

```

– тобто, потоковий тип `fstream` дозволяє одночасно читати і писати файл (режим `ios::in | ios::out`); потоковий тип `ifstream` дозволяє лише читати файл (режим `ios::in`), при цьому файл має існувати; а потоковий тип `ofstream` дозволяє лише писати файл (режим `ios::out | ios::trunc`), при цьому, якщо файл не існував, то він буде створений, а якщо існував – то його вміст буде очищено (`ios::trunc`).

Якщо файл потрібно відкрити в режимі до-запису у кінець файлу, то це можна робити так:

```
fstream f("FName.txt", ios::out | ios::app);
```

або так:

```
ofstream f("FName.txt", ios::app);
```

Є різниця між режимами відкриття `ios::ate` та `ios::app`.

Якщо файл відкривається в режимі додавання (`ios::app`), то все виведення у файл буде здійснюватися в позицію, що починається з поточного положення кінця файлу, не залежно від операцій позиціонування файлових вказівників. В режимі `ios::ate` (з англ. «at end» – з кінця) можна буде змінити позицію виведення у файл та здійснювати запис даних, починаючи з цієї позиції.

Для потоків виведення (`ofstream`) режим відкриття за умовчанням визначений як `ios::out | ios::trunc`, тобто можна не вказувати режим відсікання (`ios::trunc`) файлу. Проте для потоків введення-виведення (`fstream`) цей режим необхідно вказувати явно.

Файли, які відкриваються для виведення (`ios::out`), створюються, якщо вони ще не існують.

Визначення кінця файлу

Типовою файловою операцією є зчитування вмісту файлу, поки не буде досягнуто кінець файлу. Для визначення кінця файлу можна використовувати функцію `eof()` файлового потоку. Ця функція повертає значення 0, якщо ще не досягнуто кінця файлу, та 1, якщо досягнуто кінець файлу. Використовуючи цикл `while`, програма може зчитувати вміст файлу, поки не буде досягнуто кінець файлу:

```

while ( ! f.eof() )    // поки не кінець файлу
{
    ...                // опрацювання файлу
}

```

Цикл виконується, поки функція `eof()` повертає значення 0 (хибність).

Проте при використанні функції `eof()` можна допуститись помилок, якщо не розуміти принципів перевірки кінця файлу та механізму дій функції `eof()`.

Багато систем, зокрема, мова C (див. параграф «Перевірка кінця файлу – функція `feof()`»), стан «кінець файлу» зазвичай визначають лише після спроби зчитати дані після досягнення кінця файлу – як сигнал про помилку при зчитуванні. При використанні потоків в мові C++ стан «кінець файлу» визначається за поточною позицією файлового вказівника зчитування.

Розглянемо приклад: сформуємо текстовий файл та запишемо в нього перші п'ять натуральних чисел: 1, 2, 3, 4, 5. Відкриємо цей файл для зчитування та будемо зчитувати дані з нього і виводити їх на екран таким чином:

```
int k;
while ( ! f.eof() )
{
    f >> k;
    cout << k << " ";
}
cout << endl;
```

Якщо файл після останнього числа не містить пробільних символів, тобто, його вміст такий:

1 2 3 4 5<EOF>

або такий:

1<EOLN>
2<EOLN>
3<EOLN>
4<EOLN>
5<EOF>

(де <EOLN> – мітка «кінець рядка», а <EOF> – мітка «кінець файлу»), то результати – очікувані:

1 2 3 4 5

Якщо записати хоча б один пробільний символ після останнього числа, щоб файл став таким:

1 2 3 4 5 <EOF>

або таким:

1<EOLN>
2<EOLN>
3<EOLN>
4<EOLN>
5<EOLN>
<EOF>

то результат несподівано зміниться – останнє число буде повторене:

1 2 3 4 5 5

Бачимо, що результати роботи програми залежать від структури файлу: залежно від того, чи записані після останнього числа якісь пробільні символи (пробіли, символи табуляції або символи нового рядка), чи ні, – результати виконання – різні. Отже, програма складена неправильно: хоча технічних помилок немає, проте програма не відповідає принципу загальності – не може вірно опрацьовувати вхідні файли різної структури.

В першому випадку після останнього числа файл містив мітку «кінець файлу», і після зчитування останнього числа файловий вказівник опинявся на мітці «кінець файлу», умова циклу (поки не кінець файлу) ставала хибною і зайвих ітерацій не відбувалося.

В другому випадку після останнього числа файл містив пробільний символ, а лише тоді мітку «кінець файлу». Причина помилки при такій організації зчитування була наступною: після зчитування останнього числа файловий вказівник ставав налаштований на той пробільний символ, що записаний після числа, і ще не вказував на мітку «кінець файлу». Тому умова циклу (поки не кінець файлу) ще не стала хибною і виконувалася зайва остання ітерація. Спроба зчитати дані в цій ітерації була неуспішною (бо пробільні символи ігноруються командою зчитування `f >> k`), проте змінна `k` зберігала своє значення з попередньої ітерації і наступна команда виведення виводила це значення повторно.

Для усунення вказаних проблем перевірку досягнення кінця файлу при зчитуванні можна організувати наступним чином:

```
int k;
while ( f >> k )
{
    cout << k << " ";
}
cout << endl;
```

В цьому випадку використовуємо команду зчитування в якості умови циклу. Якщо операція зчитування виконалася успішно, то її результат (а точніше, значення потокової змінної `f`) – не нульовий (що трактується в умові як істина, `true`). Якщо ж при зчитуванні виникли помилки, то результат такої операції (значення файлової змінної) – нульовий, що для умови означає хибність, `false`.

Той самий результат можна досягнути і способом, який не потребує перероблення коду, а лише добавляє в тіло циклу одну перевірку:

```
int k;
while ( ! f.eof() )
{
    f >> k;
    if ( ! f )
        break;
    cout << k << " ";
}
cout << endl;
```

– якщо зчитування не вдалося, умова ! f стає істинною і це приводить до виходу із циклу.

Передавання файлів у функції

Файл у функцію можна передати такими способами:

- 1) передати файлову змінну, пов'язану з файлом. Файл відкривається і закривається в блоці, який викликає таку функцію. Цей спосіб використовується, коли з одним і тим самим відкритим потоком функція має виконувати такі дії, як позиціонування та зчитування / запис – щоб не витрачати час і ресурси на відкривання / закривання файлу.

Параметр – файловий потік обов'язково слід передавати як посилання:

```
#include <iostream>
#include <fstream>

using namespace std;

void fprocess(ifstream &fin) // обов'язково - як посилання
{
    ...                       // опрацювання файлу
}

int main()
{
    ifstream fin("1.txt");

    fprocess(fin);

    fin.close();

    return 0;
}
```

- 2) передати оголошену файлову змінну, ще не пов'язану з файлом. Файл відкривається і закривається в цій функції. Можуть бути проблеми, пов'язані із некоректним повторним скануванням файлу, якщо не передбачити явну очистку стану «досягнуто кінець файлу»: хоча файловий вказівник автоматично встановлюється на початок файлу при відкриванні файлу, проте після першого скануванні при досягненні кінця файлу встановлюється цей стан, який перешкодить повторному скануванню. При першому після оголошення файлової змінної відкриванні файлу стан «досягнуто кінець файлу», зрозуміло що не встановлюється (див. приклад «Проблеми при повторному скануванні файлу»).

Параметр – файловий потік обов'язково слід передавати як посилання:

```
// Передавання файлу у функцію
// Спосіб 2: передаємо файловий потік

#include <iostream>
```

```

#include <fstream>

using namespace std;

void fprocess(ifstream &fin) // обов'язково - як посилання
{
    fin.open("1.txt");
    ...                      // опрацювання файлу
}

int main()
{
    ifstream fin;
    fprocess(fin);

    return 0;
}

```

- 3) передати літерний рядок – ім'я файлу. Файл відкривається і закривається в такій функції. Цей спосіб зручний тим, що оскільки файлова змінна оголошується і файл відкривається в тілі функції, то файловий вказівник автоматично встановлюється на початок файлу, і при першому після оголошення файлової змінної відкриванні файлу стан «досягнуто кінець файлу» не встановлюється, цим самим усувається проблема повторного сканування файлу (див. приклад «Проблеми при повторному скануванні файлу»):

```

// Передавання файлу у функцію
// Спосіб 3: передаємо ім'я файлу

#include <iostream>
#include <fstream>

using namespace std;

void fprocess(const char *filename)
{
    ifstream fin(filename);
    ...                      // опрацювання файлу
}

int main()
{
    char filename[] = "1.txt";
    fprocess(filename);

    return 0;
}

```

Опрацювання текстових файлових потоків

Відкриття файлу

Відкриття файлового потоку введення (тобто, відкриття файлу для зчитування):

```
ifstream fin("1.txt");
```


Відкриття файлового потоку виведення (тобто, відкриття файлу для запису):

```
ofstream fout("1.txt");
```

Відкриття файлового потоку виведення в режимі до-запису у кінець файлу (тобто, відкриття файлу для до запису у кінець):

```
ofstream f("1.txt", ios::app);
```

Відкриття файлового потоку введення-виведення (тобто, відкриття файлу для зчитування і запису):

```
fstream fin("1.txt"); // може бути пошкоджена структура текстового файлу
```

Коли файл відкривається в текстовому режимі, відбувається наступне:

- при введенні (зчитуванні) даних з файлу кожна пара символів `'\r' + '\n'` (повернення каретки + переведення рядка) перетворюється у символ *переведення рядка* (`'\n'`);
- при виведенні (запису) даних у файл кожний символ *переведення рядка* (`'\n'`) перетворюється у пару символів `'\r' + '\n'` (повернення каретки + переведення рядка).

Послідовний перегляд та зчитування текстового файлу

Всі команди потокового введення з клавіатури можна застосовувати і для потокового зчитування з файлу. При відкриванні файлу вказівник зчитування встановлюється на початок файлу. При зчитуванні даних з файлу важливою є перевірка досягнення мітки «кінець файлу» – операція зчитування виконується до тих пір, поки не буде досягнуто кінець файлу, при цьому слід уникати можливих помилок визначення кінця файлу, описаних в параграфі «Визначення кінця файлу».

В наступній програмі зчитуються рядки текстового файлу `1.txt`, поки не буде досягнуто кінець файлу. Прочитані рядки виводяться на екран:

```
// Зчитування рядків з текстового файлу
// Версія 1

#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ifstream fin("1.txt");
    char line[80];

    while ( ! fin.eof() )
    {
        fin.getline(line, sizeof(line));
        cout << line << endl;
    }

    return 0;
}
```

Оскільки файлова змінна після невдалої файлової операції отримає значення 0, то в умові циклу можна записати команду зчитування з файлу:

```
// Зчитування рядків з текстового файлу
// Версія 2 – усуває можливі помилки визначення кінця файлу

#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ifstream fin("1.txt");
    char line[80];

    while ( fin.getline(line, sizeof(line)) )
    {
        cout << line << endl;
    }

    return 0;
}
```

Наступна програма зчитує вміст текстового файлу **1.txt** по одному слову:

```
// Зчитування слів з текстового файлу
// Версія 1

#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ifstream fin("1.txt");
    char word[64];

    while ( ! fin.eof() )
    {
        fin >> word;
        cout << word << endl;
    }

    return 0;
}
```

Інший спосіб – команда зчитування як умова циклу:

```
// Зчитування слів з текстового файлу
// Версія 2 – усуває можливі помилки визначення кінця файлу

#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ifstream fin("1.txt");
```

```

    char word[64];

    while ( fin >> word )
    {
        cout << word << endl;
    }

    return 0;
}

```

І нарешті, програма, яка зчитує вміст файлу **1.txt** по одному символу за допомогою функції `get()`:

```

// Зчитування символів з текстового файлу
// Версія 1

```

```

#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ifstream fin("1.txt");
    char letter;

    while ( ! fin.eof() )
    {
        letter = fin.get();
        cout << letter;
    }

    return 0;
}

```

Інша версія – команда зчитування в якості умови циклу:

```

// Зчитування символів з текстового файлу
// Версія 2 – усуває можливі помилки визначення кінця файлу

```

```

#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ifstream fin("1.txt");
    char letter;

    while (( letter = fin.get()) > 0)
    {
        cout << letter;
    }

    return 0;
}

```

В якості опрацювання файлу вище наведені програми виводять прочитані дані на екран. В лабораторних роботах слід виконувати завдання свого варіанту.

Запис у текстовий файл

Подібно до того, як всі команди потокового введення з клавіатури виконуються і для потокового введення (зчитування) з файлу, так і всі команди потокового виведення на екран виконуються і для потокового виведення (запису) у файл. Для управління форматом виведення слід підключити файл заголовків маніпуляторів потоку:

```
#include <iomanip>
```

Опрацювання бінарних файлових потоків

Відкриття бінарного файлу

Для відкриття бінарного файлового потоку слід вказати режим `ios::binary`, наприклад:

```
fstream f("1.dat", ios::binary);
```

Оскільки зазвичай з одним і тим самим бінарним файлом виконують операції і зчитування і запису в одному сеансі роботи, то бінарні файли, як правило, відкривають за допомогою потоку введення-виведення `fstream`.

Зчитування з бінарного файлу

Загальний вигляд команди зчитування даних з бінарного файлу:

```
файлова-змінна.read(адреса-області-пам'яті, sizeof(область-пам'яті));
```

де:

- файлова-змінна* — файлова змінна, пов'язана із відкритим бінарним файлом;
- адреса-області-пам'яті* — вказівник на змінну (адреса змінної), значення якої буде зчитане з файлу;
- `sizeof(область-пам'яті)` — розмір змінної — тої області пам'яті, яка буде заповнена зчитаними з файлу даними; це ж — і розмір зчитаних з файлу даних.

Дані зчитуються з тої компоненти файлу, яка позначена позицією поточного значення файлового вказівника зчитування. При відкриванні файлу вказівник зчитування встановлюється на початок файлу. При зчитуванні даних з файлу важливою є перевірка досягнення мітки «кінець файлу».

Оскільки потік визначається як послідовність символів, що передаються від одного довільного пристрою до іншого, то для зчитування даних із бінарного файлу необхідно привести адресу змінної (якщо це — не символьна змінна) до типу «вказівник на символ», наприклад:

```
int x;
f.read((char*) &x, sizeof(x));
```

Приклад:

```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    struct Employee
    {
        char name[64];
        int age;
        float salary;
    } worker;

    fstream f("EMPLOYEE.DAT", ios::binary);
    f.read((char *) &worker, sizeof(Employee));

    cout << worker.name << endl;
    cout << worker.age << endl;
    cout << worker.salary << endl;

    return 0;
}
```

Запис у бінарний файл

Загальний вигляд команди запису даних у бінарний файл:

```
файлова-змінна.write(адреса-області-пам'яті, sizeof(область-пам'яті));
```

де:

- файлова-змінна* — файлова змінна, пов'язана із відкритим бінарним файлом;
- адреса-області-пам'яті* — вказівник на змінну (адреса змінної), значення якої буде записане у файл;
- `sizeof(область-пам'яті)` — розмір змінної — тої області пам'яті, вміст якої буде записаний у файл.

Дані записуються у ту компоненту файлу, яка позначена позицією поточного значення файлового вказівника запису. Якщо в процесі запису нові дані перекрили мітку «кінець файлу», то розмір файлу автоматично збільшується — при закриванні файлу мітка «кінець файлу» буде автоматично записана після всіх даних. При відкриванні файлу вказівник запису встановлюється на початок файлу. Якщо бінарний файл був відкритий в режимі «відсікання» попереднього вмісту (`ios::trunc`) — то після завершення файлових операцій — при закриванні файлу — в поточну позицію вказівника запису записується мітка «кінець файлу», — це приводить до втрати попереднього вмісту файлу.

Оскільки потік визначається як послідовність символів, що передаються від одного довільного пристрою до іншого, то для запису даних у бінарний файл необхідно привести адресу змінної (якщо це – не символьна змінна) до типу «вказівник на символ», наприклад:

```
int x;
f.write((char*) &x, sizeof(x));
```

Приклад:

```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    struct Employee
    {
        char name[64];
        int age;
        float salary;
    } worker = { "Іваненко Петро", 23, 2500.00 };

    fstream f("EMPLOYEE.DAT", ios::binary);
    f.write((char *) &worker, sizeof(Employee));

    return 0;
}
```

Приклад опрацювання бінарного файлу

```
#include <iostream>
#include <stdio.h>
#include <iostream>
#include <fstream>
using namespace std;

struct Employee
{
    char name[64];
    int age;
    float salary;
};

void enter_save(char* fname);
void load_print(char* fname);

int main()
{
    char fname[61];
    char ch;
    do
    {
        cout << endl;
        cout << "Select:" << endl;
        cout << "[1] - enter & save data;" << endl;
        cout << "[2] - load & print data;" << endl << endl;
        cout << "[0] - exit." << endl << endl;
        cout << "Your choise: "; cin >> ch;
```

```

        switch (ch)
        {
        case '0':
            break;
        case '1':
            cin.get(); // очищуємо буфер клавіатури – щоб не було символу
            cin.sync(); // "кінець рядка", який залишився після вводу числа
            cout << "Enter file name: "; cin.getline(fname, sizeof(fname));
            enter_save(fname);
            break;
        case '2':
            cin.get(); // очищуємо буфер клавіатури – щоб не було символу
            cin.sync(); // "кінець рядка", який залишився після вводу числа
            cout << "Enter file name: "; cin.getline(fname, sizeof(fname));
            load_print(fname);
            break;
        default:
            cout << "Error input! ";
        }
    } while (ch != '0');

    return 0;
}

void enter_save(char* fname)
{
    ofstream f(fname, ios::binary);
    if (!f)
    {
        cerr << "Error opening file '" << fname << "'" << endl;
        return;
    }

    Employee worker;

    char ch;
    do
    {
        cout << endl;
        cout << "name: "; cin.sync();
        cin >> worker.name;
        cout << "age: "; cin >> worker.age;
        cout << "salary: "; cin >> worker.salary;

        if (!f.write((char*)&worker, sizeof(Employee)))
        {
            cerr << "Error writing file." << endl;
        }

        cout << "Continue? (Y/N) "; cin >> ch;
    } while (ch == 'Y' || ch == 'y');
}

void load_print(char* fname)
{
    ifstream f(fname, ios::binary);
    if (!f)
    {
        cerr << "Error opening file '" << fname << "'" << endl;
        return;
    }
}

```

```

Employee worker;

while (f.read((char*)&worker, sizeof(Employee)))
{
    cout << endl;
    cout << "name: " << worker.name << endl;
    cout << "age: " << worker.age << endl;
    cout << "salary: " << worker.salary << endl;
}
}

```

Прямий доступ: навігація по бінарному файлу

Навігація по бінарному файлу здійснюється за допомогою функцій позиціонування (встановлюють позицію у потоці) та функцій отримання позиції (повертають значення поточної позиції).

Функції `seekg()` та `seekp()` використовуються для позиціонування вхідного та вихідного файлового потоку відповідно (тобто, для встановлення позиції файлових вказівників зчитування та запису). Позиція файлового вказівника вказується в байтах.

Прототипи функцій позиціонування:

```

istream& seekg(long pos);
istream& seekg(long offset, seek_dir dir);
ostream& seekp(long pos);
ostream& seekp(long offset, seek_dir dir);

```

де:

`pos` – абсолютна позиція у файлі відносно початку файлу – вказується в байтах;

`offset` – зміщення в байтах у файлі відносно позиції, яка визначається параметром `dir`; якщо `offset > 0`, то зміщення відбувається вперед, в напрямку до кінця файлу; якщо `offset < 0`, то зміщення відбувається назад, в напрямку до початку файлу.

`dir` – визначає позицію, відносно якої виконується зміщення.

Параметр `dir` може набувати значень відповідно до визначення перелічуваного типу `seek_dir` в файлі заголовків `ios`:

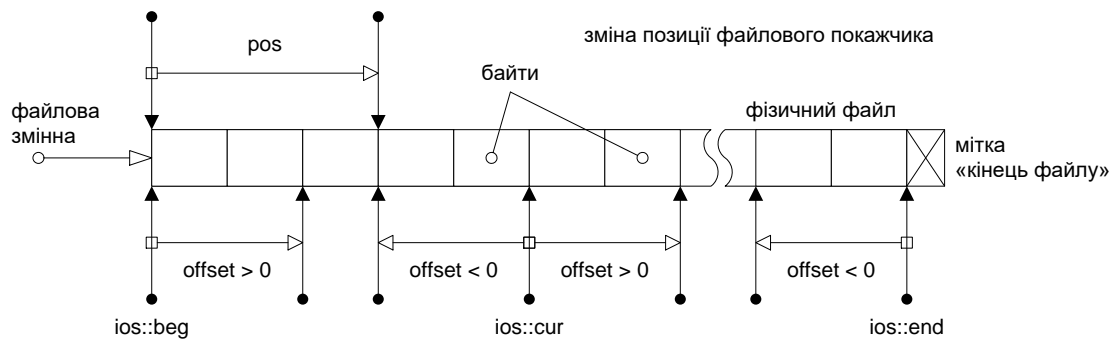
```
enum seek_dir { beg, cur, end };
```

де константи переліку визначають:

`ios::beg` – зміщення виконується від початку файлу;

`ios::cur` – зміщення виконується відносно поточної позиції файлового вказівника;

`ios::end` – зміщення виконується від кінця файлу.



Приклад:

// Навігація по файлу

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    char FileName[61];

    cout << "Enter file name: ";
    cin.getline( FileName, sizeof(FileName) );

    ofstream f1( FileName, ios::binary );

    for(int i=1; i<=100; i++)
        f1.write((char*) &i, sizeof(int));

    f1.close();

    ifstream f2( FileName, ios::binary );

    int i;

    f2.seekg( 0 * (long)sizeof(int));           // 1
    f2.read((char*) &i, sizeof(int));           // 2
    cout << i << endl;

    f2.seekg( 4 * (long)sizeof(int), ios::beg); // 3
    f2.read((char*) &i, sizeof(int));           // 4
    cout << i << endl;

    f2.seekg( 1 * (long)sizeof(int), ios::cur); // 5
    f2.read((char*) &i, sizeof(int));           // 6
    cout << i << endl;

    f2.seekg( -5 * (long)sizeof(int), ios::cur); // 7
    f2.read((char*) &i, sizeof(int));           // 8
    cout << i << endl;

    f2.seekg( -1 * (long)sizeof(int), ios::end); // 9
    f2.read((char*) &i, sizeof(int));           // 10
    cout << i << endl;

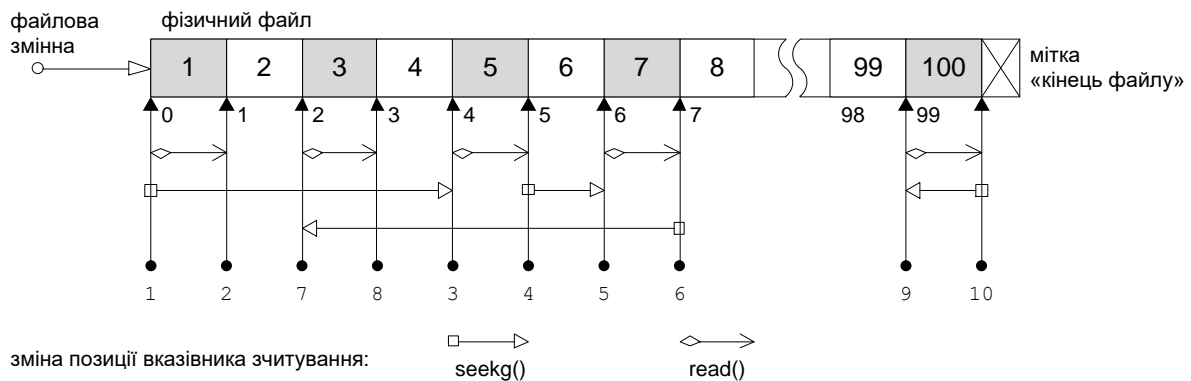
    f2.close();

    return 0;
}
```

Виведення:

```
Enter file name: 1.dat
1
5
7
3
100
```

Програма демонструє навігацію по файлу за допомогою функції встановлення позиції файлового вказівника зчитування `seekg()` та автоматичне переміщення файлового вказівника після виконання операції введення `read()`:



Спочатку (після введення користувачем імені файлу) у вказаний файл записуються 100 цілих чисел, потім цей файл закривається (що забезпечує перенесення даних з буферу у фізичний файл) і повторно відкривається для зчитування.

Перша команда позиціонування (`// 1`) встановлює файловий вказівник зчитування на найпершу компоненту файлу – компоненту з позицією 0 від початку файлу (оскільки після відкриття файлу вказівник встановлюється на початок файлу, то цю команду можна не вказувати – результат буде той самий). Наступна команда (`// 2`) зчитує значення першої компоненти файлу (1), при цьому автоматично переміщується вказівник зчитування – на компоненту з позицією 1.

Команда `// 3` встановлює вказівник зчитування на компоненту з позицією 4 від початку файлу, а наступна команда `// 4` – зчитує значення цієї компоненти (5), при цьому автоматично переміщується вказівник зчитування – на компоненту з позицією 5.

Команда `// 5` переводить вказівник зчитування від поточної позиції на 1 позицію в напрямку «вперед» – до кінця файлу, тобто – на компоненту з позицією 6 ($=5+1$). Потім команда `// 6` зчитує значення цієї компоненти (7), при цьому автоматично переміщується вказівник зчитування – на компоненту з позицією 7.

Команда `// 7` переводить вказівник зчитування від поточної позиції на 5 позицій в напрямку «назад» – до початку файлу, тобто – на компоненту з позицією 2 ($=7-5$). Потім

команда `// 8` зчитує значення цієї компоненти (3), при цьому автоматично переміщується вказівник зчитування – на компоненту з позицією 3.

Команда `// 9` встановлює вказівник зчитування на 1 позицію в напрямку «назад» – від кінця до початку файлу, тобто – на компоненту з позицією 99 ($=100-1$). Потім команда `// 10` зчитує значення цієї компоненти (100), при цьому автоматично переміщується вказівник зчитування – в кінець файлу, тобто в позицію з номером 100.

При навігації по файлу слід номер позиції множити на довжину компоненти – бо функції навігації вказують зміщення в байтах, а не в кількостях компонент.

У цьому прикладі номер позиції множиться на `sizeof(int)` – оскільки розглядається файл цілих чисел. Отримане значення слід привести до типу `long`, за це відповідає конструкція `(long)sizeof(int)`, – бо зміщення вказується в байтах відносно позиції, яка визначається другим аргументом функції `seekg()`.

Функції `tellg()` та `tellp()` дозволяють отримати значення поточної позиції в байтах у файловому потоці введення чи виведення (тобто, значення номеру позиції файлового вказівника зчитування та запису, відповідно). Результат цих функцій – кількість байт від початку файлу до поточної позиції файлового вказівника.

Прототипи цих функцій:

```
long tellg();
long tellp();
```

Наступна програма демонструє можливості позиціонування потоку введення:

```
// Визначення довжини файлу
// Спосіб 1: вказуємо ім'я файлу як
//          параметр командного рядка

#include <iostream>
#include <fstream>
using namespace std;

int main(int argc, char* argv[])
{
    int size = 0;
    if (argc > 1)
    {
        const char *FileName = argv[1];
        ofstream of;
        of.open( FileName, ios::binary );

        for(int i = 0; i<100; i++)
            of.put((char)(i+27));

        of.close();

        ifstream f;
        f.open(FileName, ios::binary );
```

```

        if (f)
        {
            f.seekg(0, ios::end);
            size = f.tellg();

            if (size < 0)
            {
                cerr << "File " << FileName << " not found." << endl;
                return 1;
            }
            cout << FileName << " size = " << size << endl;
        }
    }
    else
        cout << "File name not defined." << endl;

    return 0;
}

```

Програма виводить на екран довжину файлу, ім'я якого визначається параметром командного рядка при запуску програми на виконання.

Версія цієї ж програми, коли ім'я файлу вводить користувач:

```

// Визначення довжини файлу
// Спосіб 2: вводим ім'я файлу під час
// виконання програми

#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    int size = 0;

    char FileName[61];

    cout << "Enter file name: ";
    cin.getline( FileName, sizeof(FileName) );

    ofstream of;
    of.open( FileName, ios::binary );

    for(int i = 0; i<100; i++)
        of.put((char)(i+27));

    of.close();

    ifstream f;
    f.open(FileName, ios::binary );

    if (f)
    {
        f.seekg(0, ios::end);
        size = f.tellg();

        if (size < 0)
        {
            cerr << "File " << FileName << " not found." << endl;

```

```

        return 1;
    }
    cout << FileName << " size = " << size << endl;
}

return 0;
}

```

Спочатку до файлу записуємо 100 символів (за допомогою функції `put()`), потім за допомогою функції навігації встановлюємо позицію вказівника зчитування в кінець файлу (командою `f.seekg(0, ios::end);`) та отримуємо значення цієї позиції в байтах від початку файлу (командою `size = f.tellg();`).

Додаткові можливості опрацювання файлових потоків

Функції управління введенням-виведенням

Розглянемо деякі корисні функції з бібліотеки введення-виведення C++. Більшість з цих функцій використовуються для неформатованого введення-виведення.

`get()`

– зчитує символи з потоку у вигляді неформатованого введення.

Прототипи функцій:

(1) зчитує один символ	(a)	<code>int get();</code>
	(b)	<code>istream& get(char& c);</code>
(2) зчитує рядок символів	(a)	<code>istream& get(char* str, int n);</code>
	(b)	<code>istream& get(char* str, int n, char delim);</code>

(1) Зчитування одного символу

Вилучає один символ із вхідного потоку.

Прототип (a) – повертає код прочитаного символу;

Прототип (b) – прочитаний символ передається як значення вихідного параметру `c`.

(2) Зчитування нуль-термінального рядку символів

Вилучає символи із вхідного потоку та записує їх в нуль-термінальний літерний рядок `str`, до тих пір, поки або не буде прочитано і вилучено $(n-1)$ символів, або не зустрінеться символ завершення введення, тобто або символ переходу до нового рядка (`'\n'`) або символ, визначений параметром `delim` (якщо цей аргумент вказано при виклику функції).

Символ завершення введення при зчитуванні не вилучається із вхідного потоку, він залишається в потоці і буде наступним символом, який буде повертатися потоком при наступній спробі зчитування. Функція `getline()` – альтернативний спосіб зчитування з потоку, вона вилучає символ завершення введення з потоку.

Нуль-символ (`'\0'`) автоматично додається до записаної в змінну `str` послідовності символів, якщо значення `n` більше нуля, навіть якщо з потоку був прочитаний порожній літерний рядок.

Вилучення символів зі вхідного потоку функцією `get()` припиняється також при досягненні мітки «кінець файлу».

Кількість успішно прочитаних функцією `get()` символів можна отримати за допомогою функції `gcount()`.

Параметри

- `c` – посилання на символьну змінну, в яку записується значення символу, зчитаного та вилученого із потоку.
- `str` – вказівник на символьний масив, в який записуються зчитані та вилучені з потоку символи у вигляді нуль-термінального літерного рядка. Якщо функція `get()` не вилучає жодного символу із потоку (або якщо перший зчитаний символ є *символом завершення введення*), та значення `n` більше нуля, то `str` буде порожнім літерним рядком.
- `n` – максимальна кількість символів, яка буде записана в рядок `str` (включно із нуль-завершальним символом). Якщо `n` менше 2, то функція `get()` не вилучатиме жодного символу із потоку.
- `delim` – явно вказаний *символ завершення введення*. Операція успішного вилучення символів з потоку припиняється, як тільки наступний символ в потоці буде символом завершення введення.

Результуюче значення

Функція з першим прототипом повертає код зчитаного символу або значення `-1`, якщо в потоці немає символів, які можна зчитати (коли досягнута мітка «кінець файлу»).

Всі інші функції повертають посилання на екземпляр цього потоку.

Приклад

```
// приклад використання функції get()

#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    char str[256];

    cout << "Enter the name of an existing text file: ";
    cin.get(str, 256);           // вводимо ім'я файлу
```

```

    ifstream f(str);           // відкриваємо файл

    while ( f.good() )         // цикл поки можливе зчитування з файлу
    {
        char c = f.get();       // отримуємо символ з файлу
        if (f.good())
            cout << c;
    }

    f.close();                 // закриваємо файл

    return 0;
}

```

put()

— вставляє символ у потік виведення.

Прототип функції:

```
ostream& put(char ch);
```

getline()

Оскільки функція `get()` не вилучає із вхідного потоку символ завершення введення, вона використовується відносно рідко. Частіше використовується функція `getline()`, яка вилучає із потоку символ завершення введення, але не записує його в буфер.

Прототипи функції:

```
istream& getline(char* str, int n, char delim);
istream& getline(char* str, int n);
```

Параметри мають ті самі значення, що і для функції `get()`.

gcount()

— повертає кількість символів, вилучених з вхідного потоку останньою операцією не форматowanego введення (тобто, функціями `get()`, `getline()` або `read()`).

Прототип функції:

```
int gcount() const;
```

Приклад, в якому використовуються функції `getline()` та `gcount()`:

```

#include <iostream>

using namespace std;

int main()
{
    char *str;
    int len = 100;
    int count = 0;
    str = new char[len];

    cout << "Enter text: ";
    cin.getline(str, len);
}

```

```

count = cin.gcount();

// Зменшуємо значення count на 1, бо
// getline() не заносить символ
// завершення введення до буферу
cout << "\nCount: " << count - 1 << endl;

return 0;
}

```

ignore()

– пропускає кілька символів при введенні.

Прототип функції:

```
istream& ignore(int n = 1, int delim = EOF);
```

Ця функція ігнорує до n символів у вхідному потоці. Пропуск символів припиняється, якщо функція зустрічає символ завершення введення (за умовчанням це – мітка «кінець файлу»). Символ завершення введення вилучається із вхідного потоку.

peek()

– дозволяє «зазирнути вперед» у вхідний потік та взяти наступний символ у потоці.

При цьому цей символ не вилучається із потоку, а залишається в ньому.

Прототип функції:

```
int peek();
```

putback()

– дозволяє повернути прочитаний із потоку введення символ назад у потік.

Прототип функції:

```
istream& putback(char ch);
```

Приклад:

```

#include <iostream>
using namespace std;

int main()
{
    char c[10], c2, c3;

    c2 = cin.get();
    c3 = cin.get();

    cin.putback( c2 );

    cin.getline( &c[0], sizeof(c) );
    cout << c << endl;

    return 0;
}

```


Введення:

qw

Виведення:

q

unget()

– дозволяє повернути прочитані із потоку введення символи назад у потік.

Прототип функції:

```
istream& unget();
```

Приклад:

```
#include <iostream>
using namespace std;

int main()
{
    char c[10], c2;

    cout << "Type 'abc': ";
    c2 = cin.get();

    cin.unget();

    cin.getline( &c[0], sizeof(c) );
    cout << c << endl;

    return 0;
}
```

Введення:

abc

Виведення:

```
Type 'abc': abc
abc
```

flush()

При виконанні операції виведення дані не одразу записуються у файл, а тимчасово зберігаються в пов'язаному з потоком системному буфері – до тих пір, поки цей буфер не заповниться. Функція `flush()` приводить до примусового запису у файл вмісту буферу.

Ця функція неявно використовується маніпулятором `endl`. Цей маніпулятор вставляє у потік символ переведення рядка («новий рядок») і очищує буфер.

Команда

```
cout << endl;
```

еквівалентна наступним:

```
cout << '\n';
cout.flush();
```

Перевірка помилок при виконанні файлових операцій

Важливо запобігати можливим помилкам при виконанні файлових операцій. Наприклад, при відкритті файлу для зчитування слід перевірити, чи існує цей файл. Якщо програма записує дані у файл, слід переконатися, що операція пройшла успішно (наприклад, відсутність місця на диску унеможливить запис даних).

Для контролю за помилками при виконанні файлових операцій можна використовувати функцію `fail()`. Якщо при виконанні файлової операції не було помилок, функція повертає значення `false` (хибність); якщо ж при виконанні трапилася помилка, функція `fail()` поверне значення `true` (істина). Наприклад, якщо програма відкриває файл для зчитування, можна використовувати функцію `fail()` для визначення, чи були помилки при відкритті файлу:

```
ifstream f("FILENAME.DAT");
if ( f.fail() )
{
    cerr << "Помилка при відкритті файлу FILENAME.EXT" << endl;
    exit(1);
}
```

Слід зауважити, що оскільки файловий потік (тобто, файлова змінна) отримує ненульове значення у випадку успішного виконання файлової операції, то значення файлового потоку можна використовувати в якості умови і наведений фрагмент можна переписати у вигляді

```
ifstream f("FILENAME.DAT");
if ( ! f )
{
    cerr << "Помилка при відкритті файлу FILENAME.EXT" << endl;
    exit(1);
}
```

Наведені фрагменти використовують стандартний потік консольного виведення повідомлень про помилки `cerr` (визначений в файлі заголовку `iostream` разом із стандартним потоком консольного виведення `cout` та стандартним потоком консольного введення `cin`).

Наступна програма використовує функцію `fail()` для перевірки різних помилкових ситуацій:

```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    char str[256];
    ifstream f("BOOKINFO.DAT");
    if ( f.fail() )
```

```

        cerr << "Помилка при відкритті файлу BOOKINFO.DAT" << endl;
    else
    {
        while ( ! f.eof() && ! f.fail() )
        {
            f.getline(str, sizeof(str));
            if ( ! f.fail() )
                cout << str << endl;
        }
    }
    return 0;
}

```

Визначення та встановлення стану потоку

Інформація про стан потоку після кожної операції введення-виведення зберігається в змінній типу `iosstate`, який визначається так:

```
typedef int iosstate;
```

Стани потоку – це елементи перелічуваного типу, значення яких наведені в таблиці:

Стани потоку та їх значення:

Стан	Значення
<code>ios::goodbit</code>	Помилки немає
<code>ios::eofbit</code>	Досягнуто кінець файлу
<code>ios::failbit</code>	Помилка форматування або перетворення типу
<code>ios::badbit</code>	Помилка при виконанні операцій введення-виведення

Ці елементи можна комбінувати за допомогою розрядної операції `|` (бітове або).

Для отримання та встановлення стану потоку використовують наступні функції:

Для отримання інформації про стан операцій введення-виведення є два способи:

- 1) Використати функцію `rdstate()`, яка повертає стан операції введення-виведення.

Прототип цієї функції:

```
iosstate rdstate() const;
```

- 2) Використати одну з наступних функцій, прототипи яких – такі:

```

bool good() const;
bool eof() const;
bool fail() const;
bool bad() const;

```

Кожна із цих функцій повертає 1, якщо встановлено відповідний біт стану (точніше кажучи, функція `fail()` повертає 1, якщо встановлено біт `failbit` або `badbit`).

Стан потоку можна очистити за допомогою функції `clear()`, прототип якої наступний:

```
void clear(iosstate state = ios::goodbit);
```

Функція `clear()` спочатку скидає (очищує, записує значення 0) для всіх бітів стану потоку, а потім встановлює (записує значення 1) для тих бітів, які вказані аргументом виклику цієї функції.

Приклад:

```
// clearing errors

#include <iostream>
#include <fstream>

using namespace std;

int main ()
{
    char str[80];
    fstream f;

    f.open("test.txt", ios::in);

    f << "test";
    if ( f.fail() )
    {
        cout << "Error writing to test.txt\n";
        f.clear();
    }

    f.getline(str, sizeof(str));
    cout << str << " successfully read from file.\n";

    return 0;
}
```

В цьому прикладі файл `f` відкривається для зчитування, проте робиться спроба виконати операцію запису, тому встановлюється стан `failbit`. Викликається функція `clear()` для того, щоб очистити цей стан та зробити можливою наступну операцію зчитування за допомогою функції `getline()`.

Встановити необхідний стан можна за допомогою функції `setstate()` з прототипом:

```
void setstate(iostate state);
```

Наступна програма демонструє отримання інформації про стан введення-виведення:

```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    char c;

    char filename[61];
    cout << "Enter file name: ";
    cin >> filename;
```

```

ifstream f(filename);
if ( ! f )
{
    cout << "file not opened\n";
    return 1;
}

while ( f.eof() )
{
    f.get(c);

    // Контроль стану потоку
    if ( f.fail() )
    {
        cout << "Error\n";
        break;
    }
    cout << c;
}
f.close();

return 0;
}

```

Програма зчитує символи із файлу, ім'я якого вводить користувач. Якщо при зчитуванні відбувається помилка, зчитування припиняється та про це виводиться відповідне повідомлення.

Зчитування / запис літерних рядків *string* у бінарний файл

Літерні рядки *string* – це вказівники на динамічні об’єкти в пам’яті. У бінарний файл записуються дані у тому форматі, у якому вони представлені в оперативній пам’яті.

Проте при завершенні роботи функції, яка зчитує літерні рядки *string* з бінарного файлу, виникає помилка у використанні ресурсів – при виділенні та звільненні пам’яті, виділеної цією функцією для літерного рядка *string*.

Приклад:

```
#include <iostream>
#include <fstream>

using namespace std;

void Create(const char* fName);
void Print(const char* fName);

int main()
{
    Create("f.dat");
    Print("f.dat");

    system("pause");
    return 0;
}

void Create(const char* fName)
{
    ofstream f(fName, ios::binary);

    string s;
    char c;
    do
    {
        cout << "s = ? "; cin >> s;
        f.write((char*)&s, sizeof(s));
        cout << "continue ? "; cin >> c;
    } while (c == 'y' || c == 'Y');
}

void Print(const char* fName)
{
    cout << endl << "\nfile \"" << fName << "":\n";

    ifstream f(fName, ios::binary);

    string s;
    while (f.read((char*)&s, sizeof(s)))
        cout << s << endl;

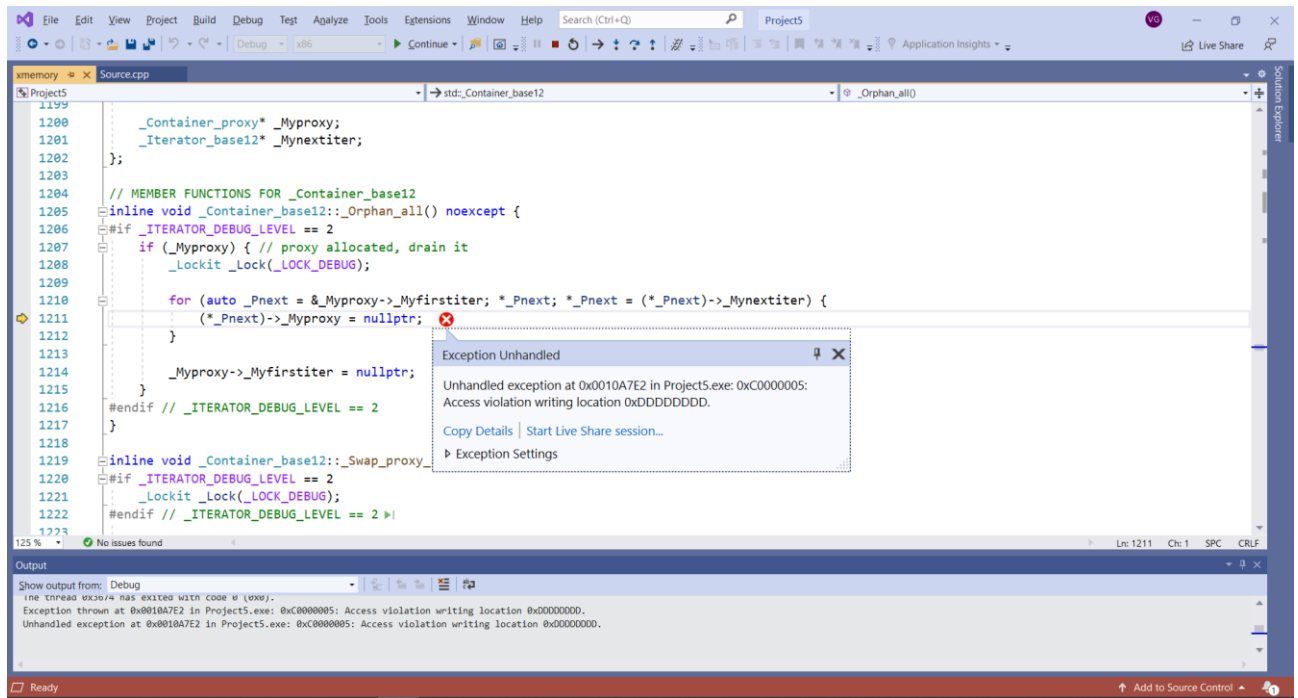
    cout << endl;
}
```

Результат виконання:

```
C:\Users\Biktop\source\repo...
s = ? abc
continue ? y
s = ? 123
continue ? n

file "f.dat":
abc
123
```

Після цього програма аварійно завершує роботу з повідомленням:



Причина помилки – не можна використовувати `sizeof` з `std::string`, бо `sizeof` повертає загальний розмір об'єкта `std::string`, а не реальний розмір літерного рядка всередині `std::string`.

Розглянемо способи вирішення цієї проблеми.

Спосіб 1. Використання рядків в стилі мови C

Згідно цього способу, слід уникати використання літерних рядків `string` в програмах, які здійснюють зчитування / запис з бінарними файлами. Замість цього краще використовувати літерні рядки в стилі мови C (масиви символів `char[]`). Це стосується і всіх типів, до складу яких входять літерні рядки.

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>

using namespace std;
```

```

void Create(const char* fName);
void Print(const char* fName);

int main()
{
    Create("f.dat");
    Print("f.dat");

    system("pause");
    return 0;
}

void Create(const char* fName)
{
    ofstream f(fName, ios::binary);

    char s[100];
    char c;
    do
    {
        cout << "s = ? "; cin >> s;
        f.write((char*)&s, sizeof(s));
        cout << "continue ? "; cin >> c;
    } while (c == 'y' || c == 'Y');
}

void Print(const char* fName)
{
    cout << endl << "\nfile \"" << fName << "":\n";

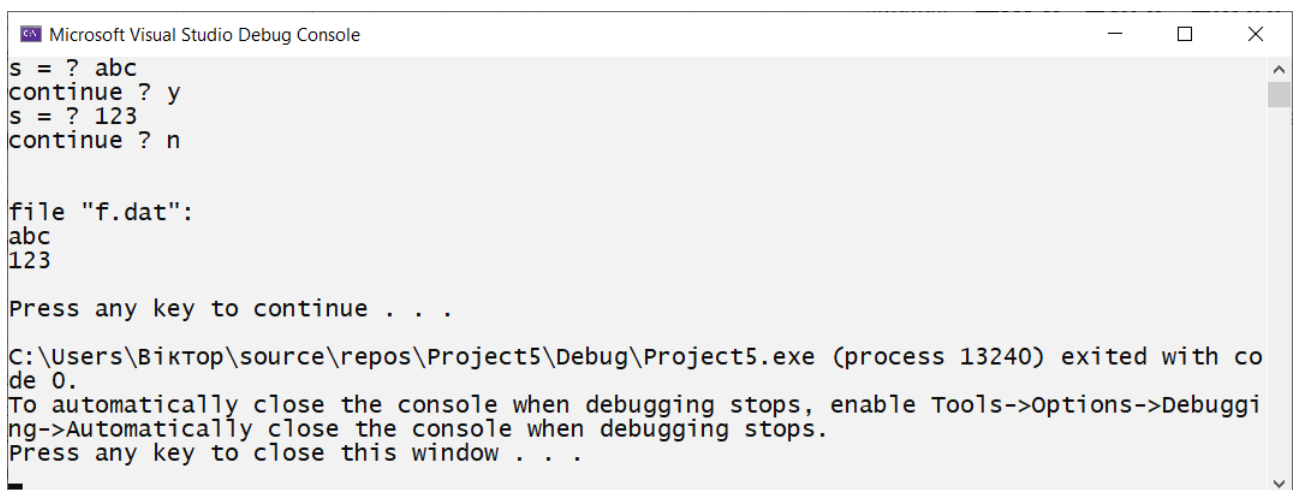
    ifstream f(fName, ios::binary);

    char s[100];
    while (f.read((char*)&s, sizeof(s)))
    {
        cout << s << endl;
    }

    cout << endl;
}

```

Результат виконання:



```

Microsoft Visual Studio Debug Console
s = ? abc
continue ? y
s = ? 123
continue ? n

file "f.dat":
abc
123

Press any key to continue . . .

C:\Users\Віктор\source\repos\Project5\Debug\Project5.exe (process 13240) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```


Спосіб 2. Запис та зчитування довжини рядка `string` та його вмісту

Щоб записати `std::string` у двійковий файл, спочатку потрібно зберегти довжину рядка:

```
std::fstream outfile("1.bin", std::ios::binary);

std::string str("whatever");
size_t len = str.size();           // обчислили довжину рядка
outfile.write((char*)&len, sizeof(len)); // записали довжину у файл
outfile.write(&str[0], len);       // записали вміст рядка
```

Щоб прочитати його, слід «перевернути» процес, спочатку змінивши розмір рядка, щоб для рядка було достатньо місця:

```
std::fstream infile("1.bin", std::ios::binary);

std::string str;
size_t len;
infile.read((char*)&len, sizeof(len)); // прочитали довжину з файлу
str.resize(len);                      // встановили довжину рядка
infile.read(&str[0], len);            // прочитали вміст рядка
```

Оскільки рядки мають змінний розмір, якщо ми не збережемо цей розмір у файлі, то не зможемо отримати його правильно.

Можна спробувати покладатися на маркер `'\0'`, який гарантовано знаходиться в кінці C-рядка або еквівалентного виклику `string::c_str()`, але це не дуже гарна ідея, тому що:

1. Тоді мусимо зчитувати в рядку символ за символом, перевіряючи наявність `'\0'`.
2. `std::string` може законно містити байт `'\0'` (хоча насправді це екзотика, так не повинно бути, бо тоді виклик `c_str()` приведе до помилки – не всі дані рядка `string` будуть перетворені у C-рядок).

Попередній приклад можна переписати, використовуючи цей спосіб:

```
#include <iostream>
#include <fstream>

using namespace std;

void Create(const char* fName);
void Print(const char* fName);

int main()
{
    Create("f.dat");
    Print("f.dat");

    system("pause");
    return 0;
}

void Create(const char* fName)
{
    ofstream f(fName, ios::binary);
```

```

string s;
size_t len;
char c;
do
{
    cout << "s = ? "; cin >> s;
    len = s.size();

    f.write((char*)&len, sizeof(len));
    f.write(&s[0], len);

    cout << "continue ? "; cin >> c;
} while (c == 'y' || c == 'Y');
}

void Print(const char* fName)
{
    cout << endl << "\nfile \"" << fName << "":\n";

    ifstream f(fName, ios::binary);

    string s;
    size_t len;
    while (f.read((char*)&len, sizeof(len)))
    {
        s.resize(len);
        f.read(&s[0], len);

        cout << s << endl;
    }

    cout << endl;
}

```

Тепер все працює правильно:

```

Microsoft Visual Studio Debug Console
s = ? abc
continue ? y
s = ? 123
continue ? n

file "f.dat":
abc
123

Press any key to continue . . .

C:\Users\Віктор\source\repos\Project8\Debug\Project8.exe (process 1252) exited with code 0.
Press any key to close this window . . .

```

Аналогічно слід зчитувати / записувати структури, до складу яких входять літерні рядки `string`.

Операції з фізичними файлами

Для виконання наведених в цьому параграфі файлових операцій слід підключити файл заголовку:

```
#include <stdio.h>
```

remove () – вилучення файлу

Функція remove() вилучає (знищує) фізичний файл. Файл має бути закритий.

Прототип функції:

```
int remove( const char *path );
```

де

path – нуль-термінальний літерний рядок, який містить повне ім'я файлу, записане згідно правил операційної системи.

Якщо вилучення файлу відбулося успішно, функція повертає 0. Якщо вилучити файл не вдалося, функція повертає -1.

Виклик функції:

```
remove( "ім'я-файлу" );
```

Приклад:

```
#include <iostream>
#include <stdio.h>

using namespace std;

int main()
{
    if ( remove( "crt_remove.txt" ) == -1 )
        cerr << "Could not delete 'CRT_REMOVE.TXT'" << endl;
    else
        cout << "Deleted 'CRT_REMOVE.TXT'" << endl;

    return 0;
}
```

rename () – перейменування файлу

Функція rename() перейменовує фізичний файл. Файл має бути закритий.

Прототип функції:

```
int rename( const char *oldname, const char *newname );
```

де

oldname – нуль-термінальний літерний рядок, який містить старе ім'я файлу, записане згідно правил операційної системи;

newname – нуль-термінальний літерний рядок, який містить нове ім'я файлу, записане згідно правил операційної системи.

Якщо перейменування файлу відбулося успішно, функція повертає 0.

Виклик функції:

```
rename( "старе-ім'я-файлу", "нове-ім'я-файлу" );
```

Приклад:

```
#include <iostream>
#include <stdio.h>

using namespace std;

int main()
{
    int result;
    char oldname[] = "OLD_NAME.TXT",
          newname[] = "NEW_NAME.TXT";

    /* Attempt to rename file: */
    result = rename( oldname, newname );

    if ( result != 0 )
        cerr << "Could not rename '" << oldname << "'" << endl;
    else
        cout << "File '" << oldname << "' renamed to '"
              << newname << "'" << endl;

    return 0;
}
```

Приклади

Опрацювання текстового файлу

Дано текстовий файл `"f.txt"`, в якому записано цілі числа.

Послідовно переглядаючи цей файл, записати додатні числа у файл `"g.txt"`, а всі інші – у файл `"h.txt"`.

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ifstream f("f.txt");
    if ( !f )
    {
        cerr << "file \"f.txt\" not found" << endl;
        return 1;
    }

    ofstream g("g.txt");
    ofstream h("h.txt");

    int x;

    while ( f >> x )
    {
        if ( x > 0 )
            g << x << endl;
        else
            h << x << endl;
    }
    f.close();
    g.close();
    h.close();

    return 0;
}
```

Опрацювання файлу виконується в функції `main()`. При виконанні завдань лабораторних робіт опрацювання файлу слід виконувати в окремих функціях.

Опрацювання бінарного файлу

Створити за допомогою генератора випадкових чисел файл `"f.dat"`, який буде містити 100 цілих чисел з діапазону `[-10; 10]`.

Вивести вміст створеного файлу на екран.

Послідовно переглядаючи цей файл, записати додатні числа у файл `"g.dat"`, а всі інші – у файл `"h.dat"`. Вивести вміст результуючих файлів на екран.

Всі дії з файлами виконувати в окремих підпрограмах.

```

#include <iostream>
#include <fstream>
#include <time.h>
using namespace std;

void Create (const char* fName);
void Print (const char* fName);
void Process(const char* fName, const char* gName, const char* hName);

int main()
{
    Create("f.dat");
    Print ("f.dat");

    Process("f.dat", "g.dat", "h.dat");

    Print ("g.dat");
    Print ("h.dat");

    return 0;
}

void Create (const char* fName)
{
    srand((unsigned)time(NULL));

    ofstream f(fName, ios::binary);

    int x;
    for (int i=0; i<100; i++)
    {
        x = -10 + rand() % 21;
        f.write((char*) &x, sizeof(x));
    }
}

void Print (const char* fName)
{
    cout << endl << "\nfile \"" << fName << "":\n";

    ifstream f(fName, ios::binary);

    int x;
    while ( f.read((char*) &x, sizeof(x)) )
        cout << x << endl;

    cout << endl;
}

void Process(const char* fName, const char* gName, const char* hName)
{
    ifstream f(fName, ios::binary);
    ofstream g(gName, ios::binary);
    ofstream h(hName, ios::binary);

    int x;
    while ( f.read((char*) &x, sizeof(x)) )
        if ( x > 0 )
            g.write((char*) &x, sizeof(x));
        else
            h.write((char*) &x, sizeof(x));
}

```

Впорядкування бінарних файлів (рекомендації)

При впорядкуванні масивів та бінарному пошуку у впорядкованих масивах активно використовувалися наступні операції:

- 1) Запис значення змінної x у елемент масиву $a[i]$:

```
a[i] = x;
```

- 2) Зчитування значення елемента масиву $a[i]$ в змінну x :

```
x = a[i];
```

- 3) Обмін значеннями елементів масиву $a[i]$ та $a[j]$:

```
x = a[i];  
y = a[j];  
a[i] = y;  
a[j] = x;
```

Запишемо функції, які реалізують у бінарному файлі вказані операції:

Припустимо, що `Type` – це тип компоненти файлу, тоді

- 1) Запис значення змінної x в i -ту компоненту файлу:

1-й спосіб:

```
void fWrite(fstream& f, const int i, const Type x)  
{  
    f.seekp( i * (long)sizeof(Type)); // встановили вказівник  
    f.write((char*)&x, sizeof(Type)); // записали значення  
}
```

Виклик:

```
fWrite(f, i, x);
```

2-й спосіб:

```
void fWrite(char* fName, const int i, const Type x)  
{  
    ofstream f(fName, ios::binary + ios::nocreate);  
    f.seekp( i * (long)sizeof(Type)); // встановили вказівник  
    f.write((char*)&x, sizeof(Type)); // записали значення  
    f.close();  
}
```

Виклик:

```
fWrite("file.dat", i, x);
```

- 2) Зчитування значення i -тої компоненти файлу в змінну x :

1-й спосіб:

```
Type fRead(fstream& f, const int i)  
{  
    Type x;  
    f.seekg( i * (long)sizeof(Type)); // встановили вказівник  
    f.read ((char*)&x, sizeof(Type)); // прочитали значення  
  
    return x;  
}
```

Виклик:

```
x = fRead(f, i);
```

2-й спосіб:

```
Type fRead(char* fName, const int i)
{
    ifstream f(fName, ios::binary);
    Type x;
    f.seekg( i * (long)sizeof(Type)); // встановили вказівник
    f.read ((char*)&x, sizeof(Type)); // прочитали значення
    f.close();

    return x;
}
```

Виклик:

```
x = fRead("file.dat", i);
```

3) Обмін значеннями i-тої та j-тої компонентів файлу:

1-й спосіб:

```
void fChange(fstream& f, const int i, const int j)
{
    Type x = fRead(f, i);
    Type y = fRead(f, j);
    fWrite(f, i, y);
    fWrite(f, j, x);
}
```

Виклик:

```
fChange(f, i, j);
```

2-й спосіб:

```
void fChange(char* fName, const int i, const int j)
{
    Type x = fRead(fName, i);
    Type y = fRead(fName, j);
    fWrite(fName, i, y);
    fWrite(fName, j, x);
}
```

Виклик:

```
fChange("file.dat", i, j);
```

Вилучення компонент з файлів (рекомендації)

Деякі лабораторні роботи вимагають скласти програму, яка в тому числі буде містити функцію, що вилучає певні компоненти файлу. Принципи створення такої функції розглядаються в цьому параграфі.

Вилучення компонент з текстового файлу

Напишемо програму, яка формує за допомогою генератора випадкових чисел текстовий файл, що містить цілі числа з діапазону $[-10; 10]$, виводить його на екран, вилучає всі від'ємні елементи і виводить на екран модифікований файл.

Функції, що реалізують формування та виведення файлу на екран, – розглядалися раніше.

Функція `Remove()`, яка реалізує вилучення компонент файлу, пояснена в коментарях:

```
#include <iostream>
#include <fstream>
#include <time.h>
#include <stdio.h>

using namespace std;

void Create(char* filename, const int N);
void Print(char* filename);
void Remove(char* filename);

int main(int argc, char* argv[])
{
    srand((unsigned) time(NULL));

    char filename[61];
    cout << "Enter file name : ";
    cin.getline(filename, sizeof(filename));

    int N;
    cout << "Enter count of components: "; cin >> N;

    Create(filename, N);
    Print(filename);

    Remove(filename);
    Print(filename);

    return 0;
}

void Create(char* filename, const int N)
{
    ofstream f(filename);
    for (int i=0; i<N; i++)
        f << (-10 + rand() % 21) << endl;
}

void Print(char* filename)
{
    ifstream f(filename);
    int x;
    while ( f >> x )
        cout << x << endl;
    cout << endl;
}

void Remove(char* filename)
{

```

```

ifstream f(filename);
ofstream t("TMP.TXT");
int x;

while ( f >> x )           // скануємо заданий файл
    if (x >= 0)             // і копіюємо ті компоненти,
        t << x << endl;    // які потрібно залишити

f.close();                 // для вилучення і перейменування
t.close();                 // файли мають бути закриті

remove(filename);          // знищуємо заданий файл
rename("TMP.TXT", filename); // перейменовуємо тимчасовий файл
}

```

Вилучення компонент з бінарного файлу

Напишемо програму, яка формує за допомогою генератора випадкових чисел бінарний файл, що містить цілі числа з діапазону $[-10; 10]$, виводить його на екран, вилучає всі від’ємні елементи і виводить на екран модифікований файл.

Функції, що реалізують формування та виведення файлу на екран, – розглядалися раніше.

Функція Remove(), яка реалізує вилучення компонент файлу, пояснена в коментарях:

```

#include <iostream>
#include <fstream>
#include <time.h>
#include <stdio.h>

using namespace std;

void Create(char* filename, const int N);
void Print(char* filename);
void Remove(char* filename);

int main(int argc, char* argv[])
{
    srand((unsigned) time(NULL));

    char filename[61];
    cout << "Enter file name : ";
    cin.getline(filename, sizeof(filename));

    int N;
    cout << "Enter count of components: "; cin >> N;

    Create(filename, N);
    Print(filename);

    Remove(filename);
    Print(filename);

    return 0;
}

void Create(char* filename, const int N)
{
    ofstream f( filename, ios::binary );

```

```

    int x;
    for (int i=0; i<N; i++)
    {
        x = -10 + rand() % 21;
        f.write((char*) &x, sizeof(x));
    }
}

void Print(char* filename)
{
    ifstream f( filename, ios::binary );
    int x;
    while ( f.read((char*) &x, sizeof(x)) )
        cout << x << endl;
    cout << endl;
}

void Remove(char* filename)
{
    ifstream f( filename, ios::binary );
    ofstream t( "TMP.DAT", ios::binary );
    int x;

    while ( f.read((char*) &x, sizeof(x)) ) // скануємо заданий файл
        if (x >= 0)                          // і копіюємо ті компоненти,
            t.write((char*) &x, sizeof(x)); // які потрібно залишити

    f.close();                               // для вилучення і перейменування
    t.close();                               // файли мають бути закриті

    remove(filename);                        // знищуємо заданий файл
    rename("TMP.DAT", filename);             // перейменовуємо тимчасовий файл
}

```

Проблеми при повторному скануванні файлу

У цьому параграфі розглядаються проблеми, що виникають при спробі повторного сканування файлу, пов'язаному з тою самою файловою змінною, коли при першому скануванні було досягнуто кінця файлу. Ці проблеми стосуються програм, підготовлених в середовищах Microsoft Visual Studio 2005, Microsoft Visual Studio 2008; новіші версії Microsoft Visual Studio (починаючи з Microsoft Visual Studio 2010) цю проблему усувають.

Сканування текстового файлу

```

#include <iostream>
#include <fstream>
using namespace std;

void Write(fstream& f)
{
    f.open("proba.txt", ios::out | ios::trunc);

    for (int i=1; i<=5; i++)
        f << i << endl;

    f.close();
}

```

```

void Read(fstream& f)
{
    f.open("proba.txt", ios::in);

    int i;
    while ( f >> i )
        cout << i << endl;
    cout << endl;

    f.close();
}

int main()
{
    fstream f;

    Write(f);
    Read(f);
    Read(f);

    return 0;
}

```

Виведення:

```

1
2
3
4
5

```

Спочатку функція `Write()` записує до файлу 5 цілих чисел, потім двічі поспіль викликається функція `Read()`, яка виводить на екран вміст файлу. Як бачимо, функція `Read()` коректно спрацювала лише при першому виклику.

Причина помилки в тому, що хоча після оголошення файлової змінної файловий вказівник автоматично встановлюється на початок файлу, проте лише при першому відкриванні файлу стан «досягнуто кінець файлу» не встановлено. При першому виклику функції `Read()` файл сканується до тих пір, поки вказівник зчитування не досягне мітки «кінець файлу», це приведе до встановлення стану «досягнуто кінець файлу», після чого файл закривається. Другий виклик функції `Read()` виконує свою операцію відкривання, яка вже не змінює стану «досягнуто кінець файлу» – він все ще буде встановлений.

Для того, щоб виправити помилку слід (перший спосіб) явно очистити стан «досягнуто кінець файлу»:

```

#include <iostream>
#include <fstream>
using namespace std;

void Write(fstream& f)
{
    f.open("proba.txt", ios::out | ios::trunc);
}

```

```

        for (int i=1; i<=5; i++)
            f << i << endl;

        f.close();
    }

    void Read(fstream& f)
    {
        f.open("proba.txt", ios::in);

        f.clear(); // очистка стану потоку (в т.ч. EOF)

        int i;
        while ( f >> i )
            cout << i << endl;
        cout << endl;

        f.close();
    }

    int main()
    {
        fstream f;

        Write(f);
        Read(f);
        Read(f);

        return 0;
    }

```

Виведення:

```

1
2
3
4
5

1
2
3
4
5

```

Ще один спосіб виправити помилку: використати не другий, а третій спосіб передавання файлів у функції (передавати не оголошену змінну файлового потоку, яка в тілі функції пов'язується із файлом, а ім'я файлу, і вже в тілі функції оголошувати змінну файлового потоку та пов'язувати її з файлом – див. параграф «Передавання файлів у функції»):

```

#include <iostream>
#include <fstream>

using namespace std;

void Write(char* fname)

```

```

{
    fstream f(fname, ios::out | ios::trunc);

    for (int i=1; i<=5; i++)
        f << i << endl;
}

void Read(char* fname)
{
    fstream f(fname, ios::in);

    int i;
    while ( f >> i )
        cout << i << endl;
    cout << endl;
}

int main()
{
    char fname[] = "proba.txt";

    Write(fname);
    Read(fname);
    Read(fname);

    return 0;
}

```

Виведення:

```

1
2
3
4
5

1
2
3
4
5

```

Сканування бінарного файлу

Ті самі проблеми виникають і при скануванні бінарних файлів:

Розглянемо програму:

```

#include <iostream>
#include <fstream>

using namespace std;

void Write(fstream& f)
{
    f.open("proba.dat", ios::out | ios::trunc | ios::binary);

    for (int i=1; i<=5; i++)
        f.write((char*)&i, sizeof(i));

    f.close();
}

```

```

void Read(fstream& f)
{
    f.open("proba.dat", ios::in | ios::binary);
    int i;
    while ( f.read((char*)&i, sizeof(i)) )
        cout << i << endl;
    cout << endl;
    f.close();
}

int main()
{
    fstream f;

    Write(f);
    Read(f);
    Read(f);

    return 0;
}

```

Виведення:

```

1
2
3
4
5

```

Спочатку функція `Write()` записує до файлу 5 цілих чисел, потім двічі поспіль викликається функція `Read()`, яка виводить на екран вміст файлу. Як бачимо, функція `Read()` коректно спрацювала лише при першому виклику.

Причина помилки в тому, що хоча після оголошення файлової змінної файловий вказівник автоматично встановлюється на початок файлу, проте лише при першому відкриванні файлу стан «досягнуто кінець файлу» не встановлено. При першому виклику функції `Read()` файл сканується до тих пір, поки вказівник зчитування не досягне мітки «кінець файлу», це приведе до встановлення стану «досягнуто кінець файлу», після чого файл закривається. Другий виклик функції `Read()` виконує свою операцію відкривання, яка вже не змінює стану «досягнуто кінець файлу» – він все ще буде встановлений.

Для того, щоб виправити помилку слід (перший спосіб) явно очистити стан «досягнуто кінець файлу»:

```

#include <iostream>
#include <fstream>
using namespace std;

void Write(fstream& f)
{
    f.open("proba.dat", ios::out | ios::trunc | ios::binary);

    for (int i=1; i<=5; i++)
        f.write((char*)&i, sizeof(i));
}

```

```

        f.close();
    }

    void Read(fstream& f)
    {
        f.open("proba.dat", ios::in | ios::binary);

        f.clear(); // очистка стану потоку (в т.ч. EOF)

        int i;
        while ( f.read((char*)&i, sizeof(i)) )
            cout << i << endl;
        cout << endl;
        f.close();
    }

    int main()
    {
        fstream f;

        Write(f);
        Read(f);
        Read(f);

        return 0;
    }

```

Виведення:

```

1
2
3
4
5

1
2
3
4
5

```

Ще один спосіб виправити помилку: використати не другий, а третій спосіб передавання файлів у функції (передавати не оголошену змінну файлового потоку, яка в тілі функції пов'язується із файлом, а ім'я файлу, і вже в тілі функції оголошувати змінну файлового потоку та пов'язувати її з файлом – див. параграф «Передавання файлів у функції»):

```

#include <iostream>
#include <fstream>
using namespace std;

void Write(char* fname)
{
    fstream f(fname, ios::out | ios::trunc | ios::binary);

    for (int i=1; i<=5; i++)
        f.write((char*)&i, sizeof(i));
}

```



```

void Read(char* fname)
{
    fstream f(fname, ios::in | ios::binary);

    int i;
    while ( f.read((char*)&i, sizeof(i)) )
        cout << i << endl;
    cout << endl;
}

int main()
{
    char fname[] = "proba.dat";

    Write(fname);
    Read(fname);
    Read(fname);

    return 0;
}

```

Виведення:

```

1
2
3
4
5

```

```

1
2
3
4
5

```

Лабораторний практикум

Оформлення звітів про виконання лабораторних робіт

Вимоги до оформлення звіту про виконання лабораторних робіт №№ 10.1–10.4, 11.1–11.5

Звіти про виконання лабораторних робіт №№ 10.1–10.4, 11.1–11.5 мають містити наступні елементи:

- 1) заголовок;
- 2) умову завдання;
Умова завдання має бути вставлена у звіт як фрагмент зображення (скрін) сторінки посібника.
- 3) структурну схему програми;
Структурна схема програми зображує взаємозв'язки програми та всіх її програмних одиниць: схему вкладеності та охоплення підпрограм, програми та модулів; а також схему звертання одних програмних одиниць до інших.
- 4) текст програми;
Текст програми має бути правильно відформатований: відступами і порожніми рядками слід відображати логічну структуру програми; програма має містити необхідні коментарі – про призначення підпрограм, змінних та параметрів – якщо їх імена не значущі, та про призначення окремих змістовних фрагментів програми. Текст програми слід подавати моноширинним шрифтом (Courier New розміром 10 пт. або Consolas розміром 9,5 пт.) з одинарним міжрядковим інтервалом;
- 5) посилання на git-репозиторій з проектом;
- 6) хоча б для одної функції, яка повертає результат (як результат функції чи як параметр-посилання) – результати unit-тесту: текст програми unit-тесту та скрін результатів її виконання (див. хід виконання Лабораторної роботи № 5.6);
- 7) висновки.

Зразок оформлення звіту про виконання лабораторних робіт №№ 10.1–10.4, 11.1–11.5

ЗВІТ

про виконання лабораторної роботи № < номер >

« назва теми лабораторної роботи »
з дисципліни
«Алгоритмізація та програмування»
студента(ки) групи КН-16
< *Прізвище Ім'я По_батькові* >

Умова завдання:

...

Структурна схема програми:

...

Текст програми:

...

Посилання на git-репозиторій з проектом:

...

Результати unit-тесту:

...

Висновки:

...

10. Текстові файли

Приклад виконання лабораторних завдань

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

void CreateTXT(char* fname) // створення файлу з введених рядків
{
    ofstream fout(fname); // відкрили файл для запису

    char ch; // відповідь користувача - чи продовжувати введення
    string s; // введений користувачем рядок
    do
    {
        cin.get(); // очищуємо буфер клавіатури - щоб не було символу
        cin.sync(); // "кінець рядка", який залишився після вводу числа
        cout << "enter line: "; getline(cin, s); // ввели рядок
        fout << s << endl; // записали його у файл
        cout << "continue? (y/n): "; cin >> ch;
    }
    while (ch == 'y' || ch == 'Y');
    cout << endl;
}

void PrintTXT(char* fname) // виведення файлу на екран
{
    ifstream fin(fname); // відкрили файл для зчитування
    string s; // прочитаний рядок
    while ( getline(fin, s) ) // поки можна прочитати рядок
    {
        cout << s << endl; // виводимо його на екран
    }
    cout << endl;
}

int ProcessTXT1(char* fname) // обчислення загальної кількості символів + - =
{
    ifstream fin(fname); // відкрили файл для зчитування
    string s; // прочитаний рядок
    int k=0; // загальна кількість символів + - =
    while ( getline(fin, s) ) // поки можна прочитати рядок
    {
        // скануємо його і обчислюємо кількість + - =
        for (unsigned i=0; i<s.length(); i++)
            if (s[i] == '+' || s[i] == '-' || s[i] == '=')
                k++;
    }
    return k;
}

int ProcessTXT2(char* fname) // обчислення кількості слів,
// відокремлених пробілами
{
    ifstream fin(fname); // відкрили файл для зчитування
    string s; // прочитане слово
    int k=0; // кількість слів
```

```

while ( fin >> s )          // поки можна прочитати слово
{
    k++;                    // збільшили кількість і вивели його на екран
    cout << k << ": " << s << endl;
}
return k;
}

int ProcessTXT3(char* fname) // обчислення кількості слів,
{                             // відокремлених пробілами та знаками пунктуації
    ifstream fin(fname);      // відкрили файл для зчитування
    char s[100];              // прочитаний рядок
    char* w;                  // виокремлене з рядка слово
    int k=0;                  // кількість слів
    while ( fin.getline(s, sizeof(s)) ) // поки можна прочитати рядок
    {
        w = strtok(s, " .,:;!?'-\""); // початок циклу виокремлення слів
        while ( w != NULL )           // поки можна виокремити слово
        {
            k++;                      // збільшуємо лічильник слів
            cout << k << ": " << w << endl; // виводимо слово
            w = strtok(NULL, " .,:;!?'-\""); // виокремлюємо наступне слово
        }
    }
    return k;
}

void SortTXT(char* fname, char* gname) // сортування рядків текстового файлу
{
    ofstream g(gname);                // відкрили другий файл для запису

    string s, mins, z="";              // s - прочитаний з файлу f рядок
                                        // mins - рядок, який вважається
                                        // найменшим
                                        // z - записаний у файл g рядок
    int k;                             // - вказує, чи є ще рядки, які слід
                                        // записати у файл g
    do                                 // цикл запису мінімального рядка
    {                                  // з тих, які ще не записані у файл g
        k=0;                          // обнуляємо лічильник рядків,
                                        // які слід записати
        ifstream f(fname);            // відкрили перший файл для зчитування
                                        // тепер будемо читати файл з початку

        // цикл початку пошуку мінімального із ще не записаних рядків
        while ( getline(f, s) )       // поки можна зчитувати рядки
        {
            if (s<=z)                 // якщо цей рядок вже записаний у файл g
                continue;             // - пропускаємо його
            mins = s;                 // вважаємо перший ще не записаний рядок
                                        // - мінімальним
            k++;                       // знайшли ще не записаний рядок
                                        // - збільшили лічильник
            break;                    // вийшли з циклу присвоєння змінній
        }                             // mins початкового значення

        // цикл пошуку мінімального із ще не записаних рядків
        while ( getline(f, s) )       // поки можна зчитувати рядки
        {
            if (s<=z)                 // якщо цей рядок вже записаний у файл g
                continue;             // - пропускаємо його
            if (s<mins)                // якщо прочитаний рядок менший

```

```

        {
            mins=s;
            k++;
        }
    }

    // запис мінімального з не записаних рядків у файл g
    z = mins;
    if (k>0)
        g << z << endl;
    f.close();
}
while (k>0);
}

int main()
{
    // text files
    char fname[100];
    cout << "enter file name 1: "; cin >> fname;

    CreateTXT(fname);
    PrintTXT(fname);
    cout << "k(+)= " << ProcessTXT1(fname) << endl;
    cout << "k(word1) = " << ProcessTXT2(fname) << endl;
    cout << "k(word2) = " << ProcessTXT3(fname) << endl;

    char gname[100];
    cout << "enter file name 2: "; cin >> gname;

    SortTXT(fname, gname);
    PrintTXT(gname);

    return 0;
}

```

Лабораторна робота № 10.1. Пошук символів у текстовому файлі

Навчитися опрацьовувати текстові файли.

Питання, які необхідно вивчити та пояснити на захисті

- 1) Загальний синтаксис оголошення текстових файлів.
- 2) Структура текстового файлу.
- 3) Доступ до логічних рядків в текстовому файлі.
- 4) Дії з текстовими файлами.
- 5) Внутрішня реалізація текстових файлів.
- 6) Передавання та опрацювання текстових файлів у підпрограмах.
- 7) Загальна схема послідовного пошуку у текстовому файлі.

Оформлення звіту

Вимоги та зразок оформлення звіту наведені у вступі до лабораторного практикуму.

Варіанти лабораторних завдань

Розроблена програма має опрацьовувати вже існуючий файл, створений за допомогою текстового редактора.

Опрацювання файлів слід виконувати у функціях, які всю інформацію приймають у вигляді параметрів. Використовувати нелокальні змінні не можна.

Кожна функція має виконувати лише одну роль, і ця роль має бути відображена у назві функції.

«Функція, яка повертає / обчислює / шукає ...» — має не виводити ці значення, а повернути їх у місце виклику як результат функції або як відповідний вихідний параметр.

Варіант 1.

Дано текстовий файл *t*.

Вияснити, чи є в цьому файлі пара сусідніх букв “no” або “on” (оформити у вигляді функції).

Варіант 2.

Дано текстовий файл *t*.

Підрахувати загальне число входжень символів “+”, “-”, “=” у цей файл (оформити у вигляді функції).

Варіант 3.

Дано текстовий файл t . Відомо, що серед символів цього файлу є по крайній мірі три коми.

Знайти числа i (номер рядка у файлі) та j (номер позиції у рядку) – такі, що s_{ij} – третя за порядком кома (оформити у вигляді функції).

Варіант 4.

Дано текстовий файл t .

Визначити число входжень у файл групи букв “abc” (оформити у вигляді функції).

Варіант 5.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі такі символи s_{ij} та $s_{i,j+1}$ (i – номер рядка у файлі, j – номер позиції у рядку), що s_{ij} – це кома (“,”), а $s_{i,j+1}$ – тире (“–”), та обчислити їх кількість (оформити у вигляді функції).

Варіант 6.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі пара сусідніх букв “SQ” або “QS” (оформити у вигляді функції).

Варіант 7.

Дано текстовий файл t .

Визначити число входжень у файл групи букв “while” (оформити у вигляді функції).

Варіант 8.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі всі букви, що входять в слово “while” (оформити у вигляді функції).

Варіант 9.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі четвірка сусідніх однакових символів (оформити у вигляді функції).

Варіант 10.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі пара сусідніх букв “aa” або “bb” або “cc” (оформити у вигляді функції).

Варіант 11.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі трійка сусідніх букв “OGO” або “AGA” (оформити у вигляді функції).

Варіант 12.

Дано текстовий файл t .

Підрахувати, скільки разів в цьому файлі зустрічається кожний символ, що входить в слово “BASIC” (оформити у вигляді функції).

Варіант 13.

Дано текстовий файл t .

Вияснити, чи зустрічається в цьому файлі група з трьох знаків оклику “!”, які розташовані поспіль (оформити у вигляді функції).

Варіант 14.

Дано текстовий файл t . Відомо, що в цьому файлі є по крайній мірі три крапки.

Знайти числа i (номер рядка у файлі) та j (номер позиції у рядку) – такі, що s_{ij} – третя за порядком крапка (оформити у вигляді функції).

Варіант 15.

Дано текстовий файл t .

Вияснити, чи зустрічається в цьому файлі група з трьох зірочок “*”, які розташовані поспіль (оформити у вигляді функції).

Варіант 16.

Дано текстовий файл t .

Вияснити, чи є числа i (номер рядка у файлі) та j (номер позиції у рядку) – такі, що $s_{i,j-1}$ – буква “n”, а $s_{i,j+1}$ – буква “o” (оформити у вигляді функції).

Варіант 17.

Дано текстовий файл t .

Підрахувати кількість входжень кожного із символів “+”, “-”, “=” в цей файл (оформити у вигляді функції).

Варіант 18.

Дано текстовий файл t . Відомо, що в цьому файлі є по крайній мірі чотири крапки.

Знайти числа i (номер рядка у файлі) та j (номер позиції у рядку) – такі, що $s_{i,j}$ – четверта від кінця файлу крапка (оформити у вигляді функції).

Варіант 19.

Дано текстовий файл t .

Визначити загальну кількість входжень в цей файл будь-якої групи букв, яка складається із символів “a”, “b”, “c” – тобто, “aaa”, “aab”, “aac”, “aba”, “abb”, “abc”, “aca”, “acb”, “acc”, ... “ccc” (оформити у вигляді функції).

Варіант 20.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі такі елементи $s_{i,j-1}$ та $s_{i,j+1}$ (i – номер рядка у файлі, j – номер позиції у рядку), – що $s_{i,j-1}$ – це кома (“,”), а $s_{i,j+1}$ – тире (“-”), та обчислити їх кількість (оформити у вигляді функції).

Варіант 21.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі пара сусідніх букв “no” або “on” (оформити у вигляді функції).

Варіант 22.

Дано текстовий файл t .

Підрахувати загальне число входжень символів “+”, “-”, “=” у цей файл (оформити у вигляді функції).

Варіант 23.

Дано текстовий файл t . Відомо, що серед символів цього файлу є по крайній мірі три коми.

Знайти числа i (номер рядка у файлі) та j (номер позиції у рядку) – такі, що s_{ij} – третя за порядком кома (оформити у вигляді функції).

Варіант 24.

Дано текстовий файл t .

Визначити число входжень у файл групи букв “abc” (оформити у вигляді функції).

Варіант 25.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі такі символи s_{ij} та $s_{i,j+1}$ (i – номер рядка у файлі, j – номер позиції у рядку), що s_{ij} – це кома (“,”), а $s_{i,j+1}$ – тире (“–”), та обчислити їх кількість (оформити у вигляді функції).

Варіант 26.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі пара сусідніх букв “SQ” або “QS” (оформити у вигляді функції).

Варіант 27.

Дано текстовий файл t .

Визначити число входжень у файл групи букв “while” (оформити у вигляді функції).

Варіант 28.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі всі букви, що входять в слово “while” (оформити у вигляді функції).

Варіант 29.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі четвірка сусідніх однакових символів (оформити у вигляді функції).

Варіант 30.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі пара сусідніх букв “aa” або “bb” або “cc” (оформити у вигляді функції).

Варіант 31.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі трійка сусідніх букв “OGO” або “AGA” (оформити у вигляді функції).

Варіант 32.

Дано текстовий файл t .

Підрахувати, скільки разів в цьому файлі зустрічається кожний символ, що входить в слово “BASIC” (оформити у вигляді функції).

Варіант 33.

Дано текстовий файл t .

Вияснити, чи зустрічається в цьому файлі група з трьох знаків оклику “!”, які розташовані поспіль (оформити у вигляді функції).

Варіант 34.

Дано текстовий файл t . Відомо, що в цьому файлі є по крайній мірі три крапки.

Знайти числа i (номер рядка у файлі) та j (номер позиції у рядку) – такі, що s_{ij} – третя за порядком крапка (оформити у вигляді функції).

Варіант 35.

Дано текстовий файл t .

Вияснити, чи зустрічається в цьому файлі група з трьох зірочок “*”, які розташовані поспіль (оформити у вигляді функції).

Варіант 36.

Дано текстовий файл t .

Вияснити, чи є числа i (номер рядка у файлі) та j (номер позиції у рядку) – такі, що $s_{i,j-1}$ – буква “n”, а $s_{i,j+1}$ – буква “o” (оформити у вигляді функції).

Варіант 37.

Дано текстовий файл t .

Підрахувати кількість входжень кожного із символів “+”, “-”, “=” в цей файл (оформити у вигляді функції).

Варіант 38.

Дано текстовий файл t . Відомо, що в цьому файлі є по крайній мірі чотири крапки.

Знайти числа i (номер рядка у файлі) та j (номер позиції у рядку) – такі, що $s_{i,j}$ – четверта від кінця файлу крапка (оформити у вигляді функції).

Варіант 39.

Дано текстовий файл t .

Визначити загальну кількість входжень в цей файл будь-якої групи букв, яка складається із символів “a”, “b”, “c” – тобто, “aaa”, “aab”, “aac”, “aba”, “abb”, “abc”, “aca”, “acb”, “acc”, ... “ccc” (оформити у вигляді функції).

Варіант 40.

Дано текстовий файл t .

Вияснити, чи є в цьому файлі такі елементи $s_{i,j-1}$ та $s_{i,j+1}$ (i – номер рядка у файлі, j – номер позиції у рядку), – що $s_{i,j-1}$ – це кома (“,”), а $s_{i,j+1}$ – тире (“–”), та обчислити їх кількість (оформити у вигляді функції).

Лабораторна робота № 10.2. Пошук слів у текстовому файлі

Навчитися опрацьовувати текстові файли.

Питання, які необхідно вивчити та пояснити на захисті

- 1) *Загальний синтаксис оголошення текстових файлів.*
- 2) *Структура текстового файлу.*
- 3) *Доступ до логічних рядків в текстовому файлі.*
- 4) *Дії з текстовими файлами.*
- 5) *Внутрішня реалізація текстових файлів.*
- 6) *Передавання та опрацювання текстових файлів у підпрограмах.*
- 7) *Загальна схема послідовного пошуку у текстовому файлі.*

Оформлення звіту

Вимоги та зразок оформлення звіту наведені у вступі до лабораторного практикуму.

Варіанти лабораторних завдань

Розроблена програма має опрацьовувати вже існуючий файл, створений за допомогою текстового редактора.

Опрацювання файлів слід виконувати у підпрограмах, які всю інформацію приймають у вигляді параметрів. Використовувати нелокальні змінні не можна.

Кожна функція має виконувати лише одну роль, і ця роль має бути відображена у назві функції.

«Функція, яка повертає / обчислює / шукає ...» — має не виводити ці значення, а повернути їх у місце виклику як результат функції або як відповідний вихідний параметр.

Варіант 1.

Дано текстовий файл *t*.

Підрахувати найбільшу кількість пробілів, які розташовані поспіль.

Варіант 2.

Дано текстовий файл *t*.

Підрахувати довжину найдовшого слова в цьому файлі (*слово* — це група символів, які розташовані поспіль і відмінні від пробілів та знаків пунктуації).

Варіант 3.

Дано текстовий файл t_1 .

Переписати його вміст у текстовий файл t_2 , при цьому вилучити групи символів, які знаходяться між дужками «(», «)». Самі дужки теж мають бути вилучені. Вважається, що всередині кожної пари дужок нема інших дужок. Якщо всередині деякої пари дужок є пара інших дужок (вкладені дужки), то виводиться повідомлення про помилку.

Варіант 4.

Дано текстовий файл t_1 .

Переписати його вміст у текстовий файл t_2 , при цьому вилучити групи символів, які знаходяться між дужками «(», «)». Самі дужки теж мають бути вилучені. Якщо всередині деякої пари дужок є пара інших дужок (вкладені дужки), то вилучаються всі символи, які знаходяться між самими зовнішніми дужками і самі ці дужки.

Варіант 5.

Дано текстовий файл t_1 .

Відомо, що серед цих символів є хоча б один, відмінний від пробілу.

Переписати його вміст у текстовий файл t_2 , при цьому видалити групи пробілів, якими починається і якими закінчується файл, а також замінити кожну внутрішню групу пробілів одним пробілом. Якщо вказаних груп немає у цьому файлі, то залишити файл без змін.

Варіант 6.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*. Підрахувати кількість слів у цьому файлі.

Варіант 7.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*. Підрахувати кількість букв “а” в останньому слові цього файлу.

Варіант 8.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*. Знайти кількість слів, які починаються з букви “b”.

Варіант 9.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*. Знайти кількість слів, у яких перший і останній символ – однакові.

Варіант 10.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*. Визначити, чи є у файлі будь-яке слово, що починається з букви “a”.

Варіант 11.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*. Знайти перше слово, яке починається з букви “a”.

Варіант 12.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*.

Знайти останнє слово, яке починається з букви “a”.

Варіант 13.

Дано текстовий файл t_1 .

Групи символів, відокремлені пробілами (одним або кількома), відмінні від знаків пунктуації і які не містять пробілів, будемо називати *словами*.

Переписати його вміст у текстовий файл t_2 , при цьому: якщо після слова записано не інше слово, а знак пунктуації, то вилучити всі пробіли, записані між цим словом та наступним за ним знаком пунктуації.

Варіант 14.

Дано текстовий файл t_1 .

Групи символів, відокремлені пробілами (одним або кількома), відмінні від знаків пунктуації і які не містять пробілів, будемо називати *словами*.

Переписати його вміст у текстовий файл t_2 , при цьому: якщо слово записано після крапки – то замінити першу букву в цьому слові відповідною великою буквою.

Варіант 15.

Дано текстовий файл t_1 .

Відомо, що у файлі є хоча б один пробіл. Розглядаються символи, які передують першому пробілу (його координати – номер рядка та номер позиції у рядку – наперед не відомі).

Переписати його вміст у текстовий файл t_2 , при цьому видаливши всі символи, які не є буквами і передують першому пробілу.

Варіант 16.

Дано текстовий файл t_1 .

Відомо, що у файлі є хоча б один пробіл. Розглядаються символи, які передують першому пробілу (його координати – номер рядка та номер позиції у рядку – наперед не відомі).

Переписати його вміст у текстовий файл t_2 , при цьому замінивши всі малі букви, що передують першому пробілу, однойменними великими.

Варіант 17.

Дано текстовий файл t_1 .

Відомо, що у файлі є хоча б один пробіл. Розглядаються символи, які передують першому пробілу (його координати – номер рядка та номер позиції у рядку – наперед не відомі).

Переписати його вміст у текстовий файл t_2 , при цьому видаливши всі символи, які не є буквами або цифрами, що передують першому пробілу, та замінивши кожну велику букву однойменною малою.

Варіант 18.

Дано текстовий файл t .

Підрахувати довжину найкоротшого слова в цьому файлі (*слово* – це група символів, які розташовані поспіль і відмінні від пробілів та знаків пунктуації).

Варіант 19.

Дано текстовий файл t_1 .

Переписати його вміст у текстовий файл t_2 , при цьому видаливши з кожної групи цифр, що розташовані поспіль, в якій більше двох цифр і якій передує крапка, всі цифри, починаючи з третьої (наприклад, $ab+0.1973-1.1$ перетвориться в $ab+0.19-1.1$) – форматування числа методом «заокруглення до сотих шляхом відкидання».

Варіант 20.

Дано текстовий файл t_1 .

Переписати його вміст у текстовий файл t_2 , при цьому вилучивши з кожної групи цифр, що розташовані поспіль і яким не передує крапка, всі початкові нулі (крім останнього, якщо за ним іде крапка – наприклад, $ab+000000.1973-1.0100$ перетвориться в $ab+0.19-1.01$) – форматування числа методом «відкидання незначущих нулів».

Варіант 21.

Дано текстовий файл t .

Підрахувати найбільшу кількість цифр, які розташовані поспіль.

Варіант 22.

Дано текстовий файл t .

Підрахувати найбільшу кількість пробілів, які розташовані поспіль.

Варіант 23.

Дано текстовий файл t .

Підрахувати довжину найдовшого слова в цьому файлі (*слово* – це група символів, які розташовані поспіль і відмінні від пробілів та знаків пунктуації).

Варіант 24.

Дано текстовий файл t_1 .

Переписати його вміст у текстовий файл t_2 , при цьому вилучити групи символів, які знаходяться між дужками «(», «)». Самі дужки теж мають бути вилучені. Вважається, що всередині кожної пари дужок нема інших дужок. Якщо всередині деякої пари дужок є пара інших дужок (вкладені дужки), то виводиться повідомлення про помилку.

Варіант 25.

Дано текстовий файл t_1 .

Переписати його вміст у текстовий файл t_2 , при цьому вилучити групи символів, які знаходяться між дужками «(», «)». Самі дужки теж мають бути вилучені. Якщо всередині деякої пари дужок є пара інших дужок (вкладені дужки), то вилучаються всі символи, які знаходяться між самими зовнішніми дужками і самі ці дужки.

Варіант 26.

Дано текстовий файл t_1 .

Відомо, що серед цих символів є хоча б один, відмінний від пробілу.

Переписати його вміст у текстовий файл t_2 , при цьому видалити групи пробілів, якими починається і якими закінчується файл, а також замінити кожну внутрішню групу пробілів одним пробілом. Якщо вказаних груп немає у цьому файлі, то залишити файл без змін.

Варіант 27.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*. Підрахувати кількість слів в цьому файлі.

Варіант 28.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*. Підрахувати кількість букв “а” в останньому слові цього файлу.

Варіант 29.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*. Знайти кількість слів, які починаються з букви “b”.

Варіант 30.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*. Знайти кількість слів, у яких перший і останній символ – однакові.

Варіант 31.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*. Визначити, чи є у файлі будь-яке слово, що починається з букви “а”.

Варіант 32.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*. Знайти перше слово, яке починається з букви “а”.

Варіант 33.

Дано текстовий файл t .

Групи символів, відокремлені пробілами (одним або кількома) і які не містять пробілів, будемо називати *словами*.

Знайти останнє слово, яке починається з букви “а”.

Варіант 34.

Дано текстовий файл t_1 .

Групи символів, відокремлені пробілами (одним або кількома), відмінні від знаків пунктуації і які не містять пробілів, будемо називати *словами*.

Переписати його вміст у текстовий файл t_2 , при цьому: якщо після слова записано не інше слово, а знак пунктуації, то вилучити всі пробіли, записані між цим словом та наступним за ним знаком пунктуації.

Варіант 35.

Дано текстовий файл t_1 .

Групи символів, відокремлені пробілами (одним або кількома), відмінні від знаків пунктуації і які не містять пробілів, будемо називати *словами*.

Переписати його вміст у текстовий файл t_2 , при цьому: якщо слово записано після крапки – то замінити першу букву в цьому слові відповідною великою буквою.

Варіант 36.

Дано текстовий файл t_1 .

Відомо, що у файлі є хоча б один пробіл. Розглядаються символи, які передують першому пробілу (його координати – номер рядка та номер позиції у рядку – наперед не відомі).

Переписати його вміст у текстовий файл t_2 , при цьому видаливши всі символи, які не є буквами і передують першому пробілу.

Варіант 37.

Дано текстовий файл t_1 .

Відомо, що у файлі є хоча б один пробіл. Розглядаються символи, які передують першому пробілу (його координати – номер рядка та номер позиції у рядку – наперед не відомі).

Переписати його вміст у текстовий файл t_2 , при цьому замінивши всі малі букви, що передують першому пробілу, однойменними великими.

Варіант 38.

Дано текстовий файл t_1 .

Відомо, що у файлі є хоча б один пробіл. Розглядаються символи, які передують першому пробілу (його координати – номер рядка та номер позиції у рядку – наперед не відомі).

Переписати його вміст у текстовий файл t_2 , при цьому видаливши всі символи, які не є буквами або цифрами, що передують першому пробілу, та замінивши кожну велику букву однойменною малою.

Варіант 39.

Дано текстовий файл t .

Підрахувати довжину найкоротшого слова в цьому файлі (*слово* – це група символів, які розташовані поспіль і відмінні від пробілів та знаків пунктуації).

Варіант 40.

Дано текстовий файл t_1 .

Переписати його вміст у текстовий файл t_2 , при цьому видаливши з кожної групи цифр, що розташовані поспіль, в якій більше двох цифр і якій передує крапка, всі цифри, починаючи з третьої (наприклад, $ab+0.1973-1.1$ перетвориться в $ab+0.19-1.1$) – форматування числа методом «заокруглення до сотих шляхом відкидання».

Лабораторна робота № 10.3. Опрацювання текстового файлу

Навчитися опрацьовувати текстові файли.

Питання, які необхідно вивчити та пояснити на захисті

- 1) Загальний синтаксис оголошення текстових файлів.*
- 2) Структура текстового файлу.*
- 3) Доступ до логічних рядків в текстовому файлі.*
- 4) Дії з текстовими файлами.*
- 5) Внутрішня реалізація текстових файлів.*
- 6) Передавання та опрацювання текстових файлів у підпрограмах.*
- 7) Загальна схема послідовного пошуку у текстовому файлі.*

Оформлення звіту

Вимоги та зразок оформлення звіту наведені у вступі до лабораторного практикуму.

Варіанти лабораторних завдань

Кожна програма має містити меню. Необхідно передбачити контроль помилок користувача при введенні даних.

Необхідні програмі дані слід реалізувати за допомогою текстового файлу. Всі дії над даними слід виконувати у файлі – не можна копіювати вміст усього файлу в масив. Ім'я файлу має вводитися з клавіатури.

При розробці програми застосувати технологію низхідного проектування. Логічно закінчені фрагменти оформити у вигляді підпрограм, всі необхідні дані яким передаються через список параметрів. Використовувати глобальні змінні – не можна.

Кожна функція має виконувати лише одну роль, і ця роль має бути відображена у назві функції.

«Функція, яка повертає / обчислює / шукає ...» – має не виводити ці значення, а повернути їх у місце виклику як результат функції або як відповідний вихідний параметр.

Варіант 1.

В текстовому файлі записана звітна відомість результатів екзаменаційної сесії студентської групи, яка для кожного студента містить прізвище, ініціали і оцінки з п'яти предметів.

Скласти програму, за допомогою якої можна створювати, переглядати та поповнювати список і отримувати:

- список всіх студентів;
- список студентів, що склали іспити тільки на «5»;
- список студентів, що мають трійки;
- список студентів, що мають хоч одну двійку;
- список студентів, що мають більш, ніж одну двійку;
- вилучити з файлу дані про студентів, які мають більш ніж одну двійку.

Варіант 2.

Підприємство має місцеву телефонну станцію. В текстовому файлі записано телефонний довідник цього підприємства, який для кожного номера телефону містить номер приміщення і список службовців, що сидять в цьому приміщенні.

Скласти програму, яка:

- створює, переглядає та поповнює базу ;
- за номером телефону видає номер приміщення і список людей, що сидять в ньому;
- за номером приміщення видає номер телефону;
- за прізвищем службовця видає номер телефону і номер приміщення.

Номер телефону – двозначний. У одному приміщенні може знаходитися від одного до чотирьох службовців.

Варіант 3.

У готелі є 15 номерів, з них 5 одномісних і 10 двомісних. Скласти програму, яка створює, переглядає та поповнює текстовий файл, що містить дані про мешканців і за прізвищем визначає номер, де проживає мешканець.

Програма запрошує прізвище мешканця.

- Якщо мешканця з таким прізвищем немає, про це видається повідомлення.
- Якщо мешканець з таким прізвищем в готелі єдиний, програма видає прізвище мешканця і номер помешкання.
- Якщо в готелі проживає два або більше мешканців з таким прізвищем, програма додатково запрошує ініціали.

Варіант 4.

Є текстовий файл, що містить список службовців. Для кожного службовця вказано прізвище і ініціали, назву посади, рік прийому на роботу і оклад (величину заробітної плати).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про службовця, прізвище якого введено з клавіатури.

Варіант 5.

Розклад електричок зберігається у вигляді текстового файлу, що містить сукупність рядків. Кожен рядок містить назву пункту призначення, позначки типу «звичайний», «підвищеного комфорту», «швидкісний експрес» та час відправлення.

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює файл;
- виводить на екран інформації про поїзди, що відходять після введеного часу.

Варіант 6.

Є текстовий файл, що містить список товарів. Для кожного товару вказані його назва, вартість одиниці товару в гривнях, кількість і одиниця вимірювання (наприклад, кількість: 100 шт., одиниця вимірювання: упаковка по 20 кг.).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про товар, назва якого введена з клавіатури;
- здійснює виведення на екран інформації про товари із заданого з клавіатури діапазону вартості.

Варіант 7.

Є текстовий файл, що містить список товарів. Для кожного товару вказані його назва, назва магазину, в якому продається товар, вартість одиниці товару в гривнях і його кількість з вказівкою одиниці вимірювання (наприклад, кількість: 100 шт., одиниця вимірювання: упаковка по 20 кг.).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про товар, назва якого введена з клавіатури;
- здійснює виведення на екран інформації про товари, що продаються в магазині, назва якого введена з клавіатури.

Варіант 8.

Є текстовий файл, що містить список студентської групи. Кожен рядок містить прізвище студента і три екзаменаційні оцінки, причому список ніяк не впорядкований.

Скласти програму, яка створює, переглядає та поповнює список.

Варіант 9.

Є текстовий файл, що містить список товарів. Для кожного товару вказані його назва, назва магазину, в якому продається товар, вартість одиниці товару в гривнях і його кількість з вказівкою одиниці вимірювання (наприклад, кількість: 100 шт., одиниця вимірювання: упаковка по 20 кг.).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про товари, що продаються в магазині, назва якого введена з клавіатури;
- здійснює виведення на екран інформації про товари із заданого з клавіатури діапазону вартості.

Варіант 10.

Є текстовий файл, що містить список маршрутів, кожний рядок якого містить наступні дані:

- назву початкового пункту маршруту;
- назву кінцевого пункту маршруту;
- номер маршруту.

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список маршрутів;
- здійснює виведення на екран інформації про маршрут, номер якого введений з клавіатури; якщо такого маршруту немає, вивести на екран відповідне повідомлення.

Варіант 11.

Є текстовий файл, що містить список маршрутів, кожний рядок якого містить наступні дані:

- назву початкового пункту маршруту;
- назву кінцевого пункту маршруту;
- номер маршруту.

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список маршрутів;
- здійснює виведення на екран інформації про маршрути, які починаються або закінчуються в пункті, назва якого введена з клавіатури; якщо таких маршрутів немає, вивести на екран відповідне повідомлення.

Варіант 12.

Є текстовий файл, що містить список телефонів друзів, кожний рядок якого містить наступні дані:

- прізвище, ім'я;
- номер телефону;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про людину, номер телефону якої введений з клавіатури; якщо такої людини немає, вивести на екран відповідне повідомлення.

Варіант 13.

Є текстовий файл, що містить список телефонів друзів, кожний рядок якого містить наступні дані:

- прізвище, ім'я;
- номер телефону;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про людей, що народилися в тому місяці, значення якого введене з клавіатури; якщо таких немає, вивести на екран відповідне повідомлення.

Варіант 14.

Є текстовий файл, що містить список телефонів друзів, кожний рядок якого містить наступні дані:

- прізвище, ім'я;
- номер телефону;

- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про людину, прізвище якої введене з клавіатури; якщо такої немає, вивести на екран відповідне повідомлення.

Варіант 15.

Є текстовий файл, що містить список друзів, кожний рядок якого містить наступні дані:

- прізвище, ім'я;
- знак Зодіаку;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про людину, чиє прізвище введене з клавіатури; якщо такої людини немає, вивести на екран відповідне повідомлення.

Варіант 16.

Є текстовий файл, що містить список друзів, кожний рядок якого містить наступні дані:

- прізвище, ім'я;
- знак Зодіаку;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про людей, що народилися під знаком, найменування якого введене з клавіатури; якщо таких немає, вивести на екран відповідне повідомлення.

Варіант 17.

Є текстовий файл, що містить список друзів, кожний рядок якого містить наступні дані:

- прізвище, ім'я;
- знак Зодіаку;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;

- здійснює виведення на екран інформації про людей, що народилися в тому місяці, значення якого введено з клавіатури; якщо таких немає, вивести на екран відповідне повідомлення.

Варіант 18.

Є текстовий файл, що містить список товарів, кожний рядок якого містить наступні дані:

- назва товару;
- назва магазину, в якому продається товар;
- вартість товару в гривнях.

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про товар, назва якого введена з клавіатури; якщо таких товарів немає, вивести на екран відповідне повідомлення.

Варіант 19.

Є текстовий файл, що містить список товарів, кожний рядок якого містить наступні дані:

- назва товару;
- назва магазину, в якому продається товар;
- вартість товару в гривнях.

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про товари, що продаються в магазині, назва якого введена з клавіатури; якщо такого магазину немає, вивести на екран відповідне повідомлення.

Варіант 20.

Є текстовий файл, що містить список переказів, кожний рядок якого містить наступні дані:

- розрахунковий рахунок платника;
- розрахунковий рахунок одержувача;
- перерахована сума в гривнях.

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про суму, зняту з розрахункового

рахунку платника, введеного з клавіатури; якщо такого розрахункового рахунку немає, вивести на екран відповідне повідомлення.

Варіант 21.

В текстовому файлі записана звітна відомість результатів екзаменаційної сесії студентської групи, яка для кожного студента містить прізвище, ініціали і оцінки з п'яти предметів.

Скласти програму, за допомогою якої можна створювати, переглядати та поповнювати список і отримувати:

- список всіх студентів;
- список студентів, що склали іспити тільки на «5»;
- список студентів, що мають трійки;
- список студентів, що мають хоч одну двійку;
- список студентів, що мають більш, ніж одну двійку;
- вилучити з файлу дані про студентів, які мають більш ніж одну двійку.

Варіант 22.

Підприємство має місцеву телефонну станцію. В текстовому файлі записано телефонний довідник цього підприємства, який для кожного номера телефону містить номер приміщення і список службовців, що сидять в цьому приміщенні.

Скласти програму, яка:

- створює, переглядає та поповнює базу ;
- за номером телефону видає номер приміщення і список людей, що сидять в ньому;
- за номером приміщення видає номер телефону;
- за прізвищем службовця видає номер телефону і номер приміщення.

Номер телефону – двозначний. У одному приміщенні може знаходитися від одного до чотирьох службовців.

Варіант 23.

У готелі є 15 номерів, з них 5 одномісних і 10 двомісних. Скласти програму, яка створює, переглядає та поповнює текстовий файл, що містить дані про мешканців і за прізвищем визначає номер, де проживає мешканець.

Програма запрошує прізвище мешканця.

- Якщо мешканця з таким прізвищем немає, про це видається повідомлення.

- Якщо мешканець з таким прізвищем в готелі єдиний, програма видає прізвище мешканця і номер помешкання.
- Якщо в готелі проживає два або більше мешканців з таким прізвищем, програма додатково запрошує ініціали.

Варіант 24.

Є текстовий файл, що містить список службовців. Для кожного службовця вказано прізвище і ініціали, назву посади, рік прийому на роботу і оклад (величину заробітної плати).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про службовця, прізвище якого введено з клавіатури.

Варіант 25.

Розклад електричок зберігається у вигляді текстового файлу, що містить сукупність рядків. Кожен рядок містить назву пункту призначення, позначки типу «звичайний», «підвищеного комфорту», «швидкісний експрес» та час відправлення.

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює файл;
- виводить на екран інформації про поїзди, що відходять після введеного часу.

Варіант 26.

Є текстовий файл, що містить список товарів. Для кожного товару вказані його назва, вартість одиниці товару в гривнях, кількість і одиниця вимірювання (наприклад, кількість: 100 шт., одиниця вимірювання: упаковка по 20 кг.).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про товар, назва якого введена з клавіатури;
- здійснює виведення на екран інформації про товари із заданого з клавіатури діапазону вартості.

Варіант 27.

Є текстовий файл, що містить список товарів. Для кожного товару вказані його назва, назва магазину, в якому продається товар, вартість одиниці товару в гривнях і його кількість з

вказівкою одиниці вимірювання (наприклад, кількість: 100 шт., одиниця вимірювання: упаковка по 20 кг.).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про товар, назва якого введена з клавіатури;
- здійснює виведення на екран інформації про товари, що продаються в магазині, назва якого введена з клавіатури.

Варіант 28.

Є текстовий файл, що містить список студентської групи. Кожен рядок містить прізвище студента і три екзаменаційні оцінки, причому список ніяк не впорядкований.

Скласти програму, яка створює, переглядає та поповнює список.

Варіант 29.

Є текстовий файл, що містить список товарів. Для кожного товару вказані його назва, назва магазину, в якому продається товар, вартість одиниці товару в гривнях і його кількість з вказівкою одиниці вимірювання (наприклад, кількість: 100 шт., одиниця вимірювання: упаковка по 20 кг.).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про товари, що продаються в магазині, назва якого введена з клавіатури;
- здійснює виведення на екран інформації про товари із заданого з клавіатури діапазону вартості.

Варіант 30.

Є текстовий файл, що містить список маршрутів, кожний рядок якого містить наступні дані:

- назву початкового пункту маршруту;
- назву кінцевого пункту маршруту;
- номер маршруту.

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список маршрутів;
- здійснює виведення на екран інформації про маршрут, номер якого введений з

клавіатури; якщо такого маршруту немає, вивести на екран відповідне повідомлення.

Варіант 31.

Є текстовий файл, що містить список маршрутів, кожний рядок якого містить наступні дані:

- назву початкового пункту маршруту;
- назву кінцевого пункту маршруту;
- номер маршруту.

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список маршрутів;
- здійснює виведення на екран інформації про маршрути, які починаються або закінчуються в пункті, назва якого введена з клавіатури; якщо таких маршрутів немає, вивести на екран відповідне повідомлення.

Варіант 32.

Є текстовий файл, що містить список телефонів друзів, кожний рядок якого містить наступні дані:

- прізвище, ім'я;
- номер телефону;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про людину, номер телефону якої введений з клавіатури; якщо такої людини немає, вивести на екран відповідне повідомлення.

Варіант 33.

Є текстовий файл, що містить список телефонів друзів, кожний рядок якого містить наступні дані:

- прізвище, ім'я;
- номер телефону;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;

- здійснює виведення на екран інформації про людей, що народилися в тому місяці, значення якого введене з клавіатури; якщо таких немає, вивести на екран відповідне повідомлення.

Варіант 34.

Є текстовий файл, що містить список телефонів друзів, кожний рядок якого містить наступні дані:

- прізвище, ім'я;
- номер телефону;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про людину, прізвище якої введене з клавіатури; якщо такої немає, вивести на екран відповідне повідомлення.

Варіант 35.

Є текстовий файл, що містить список друзів, кожний рядок якого містить наступні дані:

- прізвище, ім'я;
- знак Зодіаку;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про людину, чиє прізвище введене з клавіатури; якщо такої людини немає, вивести на екран відповідне повідомлення.

Варіант 36.

Є текстовий файл, що містить список друзів, кожний рядок якого містить наступні дані:

- прізвище, ім'я;
- знак Зодіаку;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про людей, що народилися під знаком, найменування якого введене з клавіатури; якщо таких немає, вивести на екран

відповідне повідомлення.

Варіант 37.

Є текстовий файл, що містить список друзів, кожний рядок якого містить наступні дані:

- прізвище, ім'я;
- знак Зодіаку;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про людей, що народилися в тому місяці, значення якого введене з клавіатури; якщо таких немає, вивести на екран відповідне повідомлення.

Варіант 38.

Є текстовий файл, що містить список товарів, кожний рядок якого містить наступні дані:

- назва товару;
- назва магазину, в якому продається товар;
- вартість товару в гривнях.

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про товар, назва якого введена з клавіатури; якщо таких товарів немає, вивести на екран відповідне повідомлення.

Варіант 39.

Є текстовий файл, що містить список товарів, кожний рядок якого містить наступні дані:

- назва товару;
- назва магазину, в якому продається товар;
- вартість товару в гривнях.

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про товари, що продаються в магазині, назва якого введена з клавіатури; якщо такого магазину немає, вивести на екран відповідне повідомлення.

Варіант 40.

Є текстовий файл, що містить список переказів, кожний рядок якого містить наступні дані:

- розрахунковий рахунок платника;
- розрахунковий рахунок одержувача;
- перерахована сума в гривнях.

Написати програму, що виконує наступні дії:

- створює, переглядає та поповнює список;
- здійснює виведення на екран інформації про суму, зняту з розрахункового рахунку платника, введеного з клавіатури; якщо такого розрахункового рахунку немає, вивести на екран відповідне повідомлення.

Лабораторна робота № 10.4. Опрацювання рядків текстового файлу

Навчитися опрацьовувати текстові файли.

Питання, які необхідно вивчити та пояснити на захисті

- 1) Загальний синтаксис оголошення текстових файлів.*
- 2) Структура текстового файлу.*
- 3) Доступ до логічних рядків в текстовому файлі.*
- 4) Дії з текстовими файлами.*
- 5) Внутрішня реалізація текстових файлів.*
- 6) Передавання та опрацювання текстових файлів у підпрограмах.*
- 7) Загальна схема послідовного пошуку у текстовому файлі.*

Оформлення звіту

Вимоги та зразок оформлення звіту наведені у вступі до лабораторного практикуму.

Варіанти лабораторних завдань

Кожна програма має містити меню. Необхідно передбачити контроль помилок користувача при введенні даних.

Необхідні програмі дані слід реалізувати за допомогою текстового файлу. Всі дії над даними слід виконувати у файлі – не можна копіювати вміст усього файлу в масив. Ім'я файлу має вводитися з клавіатури.

При розробці програми застосувати технологію низхідного проектування. Логічно закінчені фрагменти оформити у вигляді підпрограм, всі необхідні дані яким передаються через список параметрів. Використовувати глобальні змінні – не можна.

Кожна функція має виконувати лише одну роль, і ця роль має бути відображена у назві функції.

«Функція, яка повертає / обчислює / шукає ...» – має не виводити ці значення, а повернути їх у місце виклику як результат функції або як відповідний вихідний параметр.

Варіант 1.

Створити текстовий файл, кожний рядок якого містить різні слова. Знайти найкоротше слово в кожному рядку файлу, переписати їх в новий текстовий файл, записати останнім рядком файлу кількість слів у файлі.

Кожний рядок нового файлу має містити найкоротше слово відповідного рядка першого файлу; і в останньому рядку – загальну кількість слів першого файлу.

Варіант 2.

Створити текстовий файл, рядок якого містить цифри і символи. В кожному рядку визначити найбільшу послідовність цифр, що йдуть поспіль. Ці значення записати у рядки нового файлу.

Кожний рядок нового файлу має містити довжину найбільшої послідовності цифр відповідного рядка першого файлу, записаних поряд.

Варіант 3.

Створити текстовий файл. Визначити в кожному парному рядку слово найбільшої довжини і записати це слово в наступному непарному рядку нового текстового файлу.

Кожний парний рядок нового файлу має містити такий самий відповідний рядок першого файлу. Кожний непарний рядок нового файлу має містити такий самий відповідний рядок першого файлу, в кінці рядка має бути дописане найдовше слово попереднього парного рядка.

Варіант 4.

Створити текстовий файл, задавши з клавіатури назву фірми, марку автомобіля, його ціну. Визначити максимальну ціну автомобіля, що випускається кожною фірмою і дописати це значення в кінець кожного рядка файлу.

Кожний рядок першого файлу має містити назву фірми та довільну кількість пар «марка автомобіля – його ціна».

Відповідний рядок нового файлу має містити спочатку –ту саму інформацію, потім – максимальну ціну автомобілів відповідної фірми.

Варіант 5.

Ввести з клавіатури рядки і записати їх у текстовий файл. У кожному непарному рядку визначити слово, що має найбільшу кількість голосних. Дописати знайдені слова в кожен рядок нового файлу.

Кожний парний рядок нового файлу має містити такий самий відповідний рядок першого файлу. Кожний непарний рядок нового файлу має містити такий самий відповідний рядок першого файлу, в кінці рядка має бути дописане слово з найбільшою кількістю голосних відповідного рядка першого файлу.

Варіант 6.

Ввести з клавіатури декілька рядків тексту та записати їх у текстовий файл. Визначити в кожному рядку кількість слів і розділових символів. Дописати значення кількості слів на початок кожного рядка, значення кількості розділових символів в кінець рядка нового файлу.

Кожний рядок нового файлу має містити спочатку – кількість слів, потім – вміст відповідного рядка першого файлу, в кінці – кількість розділових символів у відповідному рядку першого файлу.

Варіант 7.

Ввести рядки з клавіатури і записати їх у текстовий файл. Визначити кількість рядків файлу, максимальну довжину рядка, кількість порожніх рядків, рядок, що починається та завершується заданими користувачем символами. Отримані значення записати у новий текстовий файл.

Варіант 8.

Створити текстовий файл, перший рядок якого містить значення n та m , які визначають кількість рядків та стовпців матриці. Наступні n рядків містять по m чисел, що є елементами матриці. Визначити максимальне значення в кожному рядку матриці та записати їх в кожний рядок нового файлу, починаючи з другого.

Перший рядок нового файлу має містити ті самі значення n та m , кожний наступний рядок нового файлу – ті самі значення з відповідного рядка першого файлу та максимальне з цих значень.

Варіант 9.

Створити текстовий файл, рядки якого містять дані про назву фірми, назву товару, ціну в доларах. Перерахувати ціни в гривнях у відповідності з курсом гривні та дописати в кожний рядок нового файлу отримані значення.

Кожний рядок нового файлу має містити вміст відповідного рядка першого файлу, в кінці якого має бути записана ціна товару в гривнях.

Варіант 10.

Створити текстовий файл. Визначити кількість рядків, які починаються з заданого користувачем символу, починаються та закінчуються одним й тим самим символом,

складаються з однакових символів, є пустими рядками. Отримані значення записати в окремий файл з відповідними коментарями.

Варіант 11.

Створити текстовий файл **F**. Увести з клавіатури рядок символів **S**. Отримати всі рядки файлу **F**, що містять у собі рядок **S** і записати їх до нового файлу **G**. Останнім рядком файлу **G** записати кількість знайдених у файлі **F** рядків.

Варіант 12.

Створити текстовий файл, який містить додатні, від'ємні, нульові числа та довільні символи. Визначити кількість додатних, від'ємних, нульових чисел та слів у кожному рядку файлу. Записати отримані значення з відповідними коментарями в інший текстовий файл.

Варіант 13.

Створити текстовий файл, кожний рядок якого містить марку автомобіля, його вартість, кілометраж.

- 1) Упорядкувати рядки за алфавітом марки автомобіля.
- 2) В кожний рядок нового файлу додати середній кілометраж по всім маркам у файлі.
- 3) Останній рядок нового файлу має містити кількість різних марок перерахованих у файлі автомобілів.

Варіант 14.

Створити текстовий файл, який містить числа і символи. Визначити в кожному рядку файлу середнє арифметичне та середнє геометричне чисел рядка. Переписати в новий текстовий файл числа з попереднього файлу та їх середні значення – середнє арифметичне та середнє геометричне.

Варіант 15.

Створити два текстових файлів. Рядки першого файлу містять назву продукту та його ціну. Рядки другого файлу містять назву продукту та значення кількості цього продукту. Створити третій текстовий файл, кожний рядок якого має містити назву продукту, його кількість та ціну. У третьому файлі рядки мають містити назви товарів, що не повторюються.

Варіант 16.

Створити текстовий файл, рядки якого містять числа, що задають координати вершин трикутника. Визначити вид трикутника за значеннями його сторін. Дописати в кожний рядок нового файлу слово, що визначає вид трикутника (прямокутний, рівнобедрений тощо).

Варіант 17.

Створити текстовий файл, рядки якого містять по 3 числа, що задають коефіцієнти рівняння прямої в декартових координатах ($Ax+By+C=0$) на площині. Визначити лінії, що є паралельними, перпендикулярними та ті, що перетинаються. Створити новий текстовий файл, в рядки якого записати по 6 чисел – коефіцієнтів прямих та слово, що визначає їх взаємне розташування.

Варіант 18.

Створити текст, вводячи дані з клавіатури. У кожному рядку тексту знайти слово максимальної довжини і записати знайдені слова та їх довжини у новий текстовий файл.

Варіант 19.

Створити програму, що перевіряє правильність розстановки фігурних дужок у текстовому файлі, що є програмою на C.

Варіант 20.

Створити текстовий файл F. Переписати рядки файлу F до файлу H. При цьому вставити на початку рядка порядковий номер, в кінці рядка – його довжину. Порядок рядків зберігається.

Варіант 21.

Створити текстовий файл. Вилучити з файлу всі слова, довжина яких менше заданої користувачем, та пусті рядки. Записати в останній рядок текстового файлу кількість вилучених слів.

Варіант 22.

Створити два текстових файли, перший рядок кожного з яких містить значення n та m , які визначають кількість рядків та стовпців матриці. Наступні n рядків містять по m чисел, що є елементами матриці. Вивести на екран ці матриці. Визначити добуток двох матриць, результати множення матриць вивести на екран та у новий текстовий файл.

Варіант 23.

Створити текстовий файл. Отримати його копію. У вхідному файлі замінити усі входження одного слова на інше. Слова задаються користувачем з клавіатури.

Варіант 24.

Створити два тестових файли. Визначити та вивести рядки, які збігаються в обох файлах, та кількість рядків, що відрізняються.

Варіант 25.

Створити текстовий файл F. Записати до файлу G рядки файлу F в зворотному порядку, виключивши заданий користувачем символ.

Варіант 26.

Створити текстовий файл, кожний рядок якого містить різні слова. Знайти найкоротше слово в кожному рядку файлу, переписати їх в новий текстовий файл, записати останнім рядком файлу кількість слів у файлі.

Кожний рядок нового файлу має містити найкоротше слово відповідного рядка першого файлу; і в останньому рядку – загальну кількість слів першого файлу.

Варіант 27.

Створити текстовий файл, рядок якого містить цифри і символи. В кожному рядку визначити найбільшу послідовність цифр, що йдуть поспіль. Ці значення записати у рядки нового файлу.

Кожний рядок нового файлу має містити довжину найбільшої послідовності цифр відповідного рядка першого файлу, записаних поряд.

Варіант 28.

Створити текстовий файл. Визначити в кожному парному рядку слово найбільшої довжини і записати це слово в наступному непарному рядку нового текстового файлу.

Кожний парний рядок нового файлу має містити такий самий відповідний рядок першого файлу. Кожний непарний рядок нового файлу має містити такий самий відповідний рядок першого файлу, в кінці рядка має бути дописане найдовше слово попереднього парного рядка.

Варіант 29.

Створити текстовий файл, задавши з клавіатури назву фірми, марку автомобіля, його ціну. Визначити максимальну ціну автомобіля, що випускається кожною фірмою і дописати це значення в кінець кожного рядка файлу.

Кожний рядок першого файлу має містити назву фірми та довільну кількість пар «марка автомобіля – його ціна».

Відповідний рядок нового файлу має містити спочатку –ту саму інформацію, потім – максимальну ціну автомобілів відповідної фірми.

Варіант 30.

Ввести з клавіатури рядки і записати їх у текстовий файл. У кожному непарному рядку визначити слово, що має найбільшу кількість голосних. Дописати знайдені слова в кожний рядок нового файлу.

Кожний парний рядок нового файлу має містити такий самий відповідний рядок першого файлу. Кожний непарний рядок нового файлу має містити такий самий відповідний рядок першого файлу, в кінці рядка має бути дописане слово з найбільшою кількістю голосних відповідного рядка першого файлу.

Варіант 31.

Ввести з клавіатури декілька рядків тексту та записати їх у текстовий файл. Визначити в кожному рядку кількість слів і розділових символів. Дописати значення кількості слів на початок кожного рядка, значення кількості розділових символів в кінець рядка нового файлу.

Кожний рядок нового файлу має містити спочатку – кількість слів, потім – вміст відповідного рядка першого файлу, в кінці – кількість розділових символів у відповідному рядку першого файлу.

Варіант 32.

Ввести рядки з клавіатури і записати їх у текстовий файл. Визначити кількість рядків файлу, максимальну довжину рядка, кількість порожніх рядків, рядок, що починається та завершується заданими користувачем символами. Отримані значення записати у новий текстовий файл.

Варіант 33.

Створити текстовий файл, перший рядок якого містить значення n та m , які визначають кількість рядків та стовпців матриці. Наступні n рядків містять по m чисел, що є елементами матриці. Визначити максимальне значення в кожному рядку матриці та записати їх в кожному рядку нового файлу, починаючи з другого.

Перший рядок нового файлу має містити ті самі значення n та m , кожний наступний рядок нового файлу – ті самі значення з відповідного рядка першого файлу та максимальне з цих значень.

Варіант 34.

Створити текстовий файл, рядки якого містять дані про назву фірми, назву товару, ціну в доларах. Перерахувати ціни в гривнях у відповідності з курсом гривні та дописати в кожному рядку нового файлу отримані значення.

Кожний рядок нового файлу має містити вміст відповідного рядка першого файлу, в кінці якого має бути записана ціна товару в гривнях.

Варіант 35.

Створити текстовий файл. Визначити кількість рядків, які починаються з заданого користувачем символу, починаються та закінчуються одним й тим самим символом, складаються з однакових символів, є пустими рядками. Отримані значення записати в окремий файл з відповідними коментарями.

Варіант 36.

Створити текстовий файл F . Увести з клавіатури рядок символів S . Отримати всі рядки файлу F , що містять у собі рядок S і записати їх до нового файлу G . Останнім рядком файлу G записати кількість знайдених у файлі F рядків.

Варіант 37.

Створити текстовий файл, який містить додатні, від'ємні, нульові числа та довільні символи. Визначити кількість додатних, від'ємних, нульових чисел та слів у кожному рядку файлу. Записати отримані значення з відповідними коментарями в інший текстовий файл.

Варіант 38.

Створити текстовий файл, кожний рядок якого містить марку автомобіля, його вартість, кілометраж.

- 1) Упорядкувати рядки за алфавітом марки автомобіля.
- 2) В кожний рядок нового файлу додати середній кілометраж по всім маркам у файлі.
- 3) Останній рядок нового файлу має містити кількість різних марок перерахованих у файлі автомобілів.

Варіант 39.

Створити текстовий файл, який містить числа і символи. Визначити в кожному рядку файлу середнє арифметичне та середнє геометричне чисел рядка. Переписати в новий текстовий файл числа з попереднього файлу та їх середні значення – середнє арифметичне та середнє геометричне.

Варіант 40.

Створити два текстових файлів. Рядки першого файлу містять назву продукту та його ціну. Рядки другого файлу містять назву продукту та значення кількості цього продукту. Створити третій текстовий файл, кожний рядок якого має містити назву продукту, його кількість та ціну. У третьому файлі рядки мають містити назви товарів, що не повторюються.

11. Бінарні файли

Приклад виконання лабораторних завдань

```
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

void CreateBIN(char* fname) // створення файлу символів з введених рядків
{
    ofstream fout(fname, ios::binary); // відкрили файл для запису

    char ch; // відповідь користувача – чи продовжувати введення
    string s; // введений користувачем рядок
    do
    {
        cin.get(); // очищуємо буфер клавіатури – щоб не було символу
        cin.sync(); // "кінець рядка", який залишився після вводу числа
        cout << "enter line: "; getline(cin, s); // ввели рядок
        for (unsigned i=0; i<s.length(); i++) // скануємо введений рядок
            fout.write((char*)&s[i], sizeof(s[i])); // записали символ у файл
        cout << "continue? (y/n): "; cin >> ch;
    }
    while (ch == 'y' || ch == 'Y');
    cout << endl;
}

void PrintBIN(char* fname) // виведення файлу на екран
{
    ifstream fin(fname, ios::binary); // відкрили файл для зчитування
    char c; // прочитаний символ
    while ( fin.read((char*)&c, sizeof(c)) ) // поки можна прочитати символ
    {
        cout << c << endl; // виводимо його на екран
    }
    cout << endl;
}

void ProcessBIN1(char* fname, char* gname) // переписати із файлу f
// у файл g символи-цифри
{
    ifstream f(fname, ios::binary); // відкрили файл для зчитування
    ofstream g(gname, ios::binary); // відкрили файл для запису

    char c; // прочитаний символ
    while ( f.read((char*)&c, sizeof(c)) ) // поки можна прочитати символ
    {
        if (c >= '0' && c <= '9') // якщо символ є цифрою
            g.write((char*)&c, sizeof(c)); // записуємо його у файл g
    }
}

void fWrite(fstream& f, const int i, const char x)
{
    f.seekp( i * (long)sizeof(char)); // встановили вказівник
    f.write((char*)&x, sizeof(char)); // записали значення
}
```

```

char fRead(fstream& f, const int i)
{
    char x;
    f.seekg( i * (long)sizeof(char)); // встановили вказівник
    f.read ((char*)&x, sizeof(char)); // прочитали значення

    return x;
}

void fChange(fstream& f, const int i, const int j)
{
    char x = fRead(f, i);
    char y = fRead(f, j);
    fWrite(f, i, y);
    fWrite(f, j, x);
}

void SortBIN(char* fname) // сортування файлу - за допомогою
{                          // прямого доступу до файлу
    fstream f(fname, ios::binary | ios::in | ios::out);
    // обов'язково слід вказати режими
    // ios::binary | ios::in | ios::out
    // - бінарний файл, для якого
    // одночасно доступні операції
    // - зчитування та
    // - запису

    f.seekg(0, ios::end);
    int size = f.tellg();

    f.seekg(0, ios::beg);

    for (int i0 = 1; i0<size; i0++)
        for (int i1 = 0; i1<size-i0; i1++)
        {
            char a = fRead(f, i1);
            char b = fRead(f, i1+1);
            if (a>b)
                fChange(f, i1, i1+1);
        }

    f.seekp(0, ios::end);
}

void SortBIN(char* fname, char* gname) // сортування файлу - за допомогою
{                                     // послідовного доступу: зчитуємо
    // з одного файлу і записуємо в інший

    ofstream g(gname, ios::binary); // відкрили файл для зчитування і запису

    char s, mins, z=0;               // s - прочитаний з файлу f символ
    // mins - символ, який вважається
    // найменшим
    // z - записаний у файл g символ
    int k;                           // - вказує, чи є ще символи, які слід
    // записати у файл g
    do                                // цикл запису мінімального символу
    {                                  // з тих, які ще не записані у файл g
        k=0;                          // обнуляємо лічильник символів,
        // які слід записати
        ifstream f(fname, ios::binary); // відкрили перший файл для зчитування
        // тепер будемо читати файл з початку

```

```

// цикл початку пошуку мінімального із ще не записаних символів
while ( f.read((char*)&s, sizeof(char)) ) // поки можна зчитувати
{
    // символи
    if (s<=z) // якщо цей символ вже записаний у файл g
        continue; // - пропускаємо його
    mins = s; // вважаємо перший ще не записаний символ
               // - мінімальним
    k++; // знайшли ще не записаний символ
         // - збільшили лічильник
    break; // вийшли з циклу присвоєння змінній
} // mins початкового значення

// цикл пошуку мінімального із ще не записаних символів
while ( f.read((char*)&s, sizeof(char)) ) // поки можна зчитувати
{
    // символи
    if (s<=z) // якщо цей символ вже записаний у файл g
        continue; // - пропускаємо його
    if (s<mins) // якщо прочитаний символ менший
    {
        // мінімального
        mins=s; // - вважаємо його мінімальним
        k++; // збільшили лічильник ще не записаних
    } // символів
}

// запис мінімального з не записаних рядків у файл g
z = mins; // будемо записувати знайдений
           // мінімальний з не записаних символів
if (k>0) // якщо були знайдені ще не записані
    g.write((char*)&z, sizeof(char)); // символи - записуємо
                                     // мінімальний з них
f.close(); // закрили перший файл
} // щоб потім читати файл з початку
while (k>0); // повторюємо, поки є не записані рядки
}

int main()
{
    // binary files
    char fname[100]; // ім'я першого файлу
    cout << "enter file name 1: "; cin >> fname;

    CreateBIN(fname); // ввели рядки файлу з клавіатури
    PrintBIN(fname); // вивели вміст першого файлу на екран

    char gname[100]; // ім'я другого файлу
    cout << "enter file name 2: "; cin >> gname;

    ProcessBIN1(fname, gname);
    PrintBIN(gname); // вивели вміст другого файлу на екран

    SortBIN(fname, gname); // відсортували символи файлу,
    PrintBIN(gname); // вивели вміст файлу на екран

    SortBIN(fname); // відсортували символи файлу,
    PrintBIN(fname); // вивели вміст файлу на екран

    return 0;
}

```


Лабораторна робота № 11.1. Бінарні файли

Мета роботи

Навчитися опрацювати файли прямого доступу.

Питання, які необхідно вивчити та пояснити на захисті

- 1) Загальний синтаксис оголошення бінарних файлів.
- 2) Доступ до компонентів бінарних файлів.
- 3) Дії з бінарними файлами.
- 4) Внутрішня реалізація бінарних файлів.
- 5) Передавання та опрацювання бінарних файлів у підпрограмах.
- 6) Загальна схема послідовного пошуку у бінарному файлі.

Оформлення звіту

Вимоги та зразок оформлення звіту наведені у вступі до лабораторного практикуму.

Варіанти лабораторних завдань

Опрацювання файлів слід виконувати у функціях, які всю інформацію приймають у вигляді параметрів. Використовувати нелокальні змінні не можна.

Всі дії над даними слід виконувати у файлі – не можна копіювати вміст усього файлу в масив. Ім'я файлу має вводитися з клавіатури.

Всі дії – формування, опрацювання даних та вивід результатів – слід реалізувати окремими функціями, які всю необхідну інформацію отримують за допомогою параметрів. Використання глобальних змінних – не допускається.

Кожна функція має виконувати лише одну роль, і ця роль має бути відображена у назві функції.

«Функція, яка повертає / обчислює / шукає ...» – має не виводити ці значення, а повернути їх у місце виклику як результат функції або як відповідний вихідний параметр.

Варіант 1.

Сформувати файл даних, компонентами якого є символи. Переписати в інший файл даних ті компоненти, які входять в слово “інформатика”.

Вивести на екран вміст результуючого файлу.

Варіант 2.

Сформувати файл даних, компонентами якого є символи. Переписати в інший файл даних ті компоненти, які не являються зображеннями цифр.

Вивести на екран вміст результуючого файлу.

Варіант 3.

Сформувати файл даних, компонентами якого є символи.

Переписати в інший файл даних ті компоненти, які не входять в слово “геометрія”.

Вивести на екран вміст результуючого файлу.

Варіант 4.

Сформувати файл даних, компонентами якого є символи.

Переписати в інший файл даних ті компоненти, які являються буквами латинського алфавіту.

Вивести на екран вміст результуючого файлу.

Варіант 5.

Сформувати файл даних, компонентами якого є цілі числа.

Переписати в інший файл даних ті компоненти, які можуть бути значеннями функції $\cos x$.

Вивести на екран вміст результуючого файлу.

Варіант 6.

Сформувати файл даних, компонентами якого є дійсні числа.

Переписати в інший файл даних ті компоненти, які не можуть бути значеннями функції $\sin x$.

Вивести на екран вміст результуючого файлу.

Варіант 7.

Сформувати файл даних, компонентами якого є дійсні числа.

Переписати в інший файл даних ті компоненти, які можуть бути значеннями функції $\exp x$.

Вивести на екран вміст результуючого файлу.

Варіант 8.

Сформувати файл даних, компонентами якого є дійсні числа

Переписати в інший файл даних ті компоненти, які не можуть бути значеннями функції \sqrt{x} .

Вивести на екран вміст результуючого файлу.

Варіант 9.

Сформувати файл даних, компонентами якого є дійсні числа.

Переписати в інший файл даних ті компоненти, які можуть бути аргументами функції $\lg x$.

Вивести на екран вміст результуючого файлу.

Варіант 10.

Сформувати файл даних, компонентами якого є цілі числа.

Переписати в один файл даних парні компоненти, в інший – непарні.

Вивести на екран вміст результуючих файлів.

Варіант 11.

Сформувати файл даних, компонентами якого є символи.

Переписати в інший файл даних ті компоненти, після яких йде зображення цифри.

Вивести на екран вміст результуючого файлу.

Варіант 12.

Сформувати файл даних, компонентами якого є цілі числа.

Записати в інший файл даних максимальну непарну компоненту і мінімальну парну компоненту.

Вивести на екран вміст результуючого файлу.

Варіант 13.

Сформувати файл даних, компонентами якого є числа.

Записати в інший файл даних значення максимальної і мінімальної компоненти.

Вивести на екран вміст результуючого файлу.

Варіант 14.

Сформувати файл даних, компонентами якого є числа.

Записати в інший файл даних середнє арифметичне максимальної і мінімальної компоненти і середнє арифметичне всіх компонент.

Вивести на екран вміст результуючого файлу.

Варіант 15.

Сформувати файл даних, компонентами якого є числа.

Записати в інший файл даних середнє арифметичне від'ємних компонент і середнє арифметичне додатних компонент.

Вивести на екран вміст результуючого файлу.

Варіант 16.

Сформувати файл даних, компонентами якого є цілі числа.

Записати в інший файл даних середнє арифметичне непарних компонент і середнє арифметичне парних компонент.

Вивести на екран вміст результуючого файлу.

Варіант 17.

Сформувати файл даних, компонентами якого є числа.

Записати в інший файл даних значення найбільшої від'ємної компоненти і найменшої додатної компоненти.

Вивести на екран вміст результуючого файлу.

Варіант 18.

Сформувати файл даних, компонентами якого є числа.

Записати в інший файл даних величини, які рівні: першій від'ємній компоненті; сумі перших двох від'ємних компонент; сумі перших трьох від'ємних компонент і т.д.

Вивести на екран вміст результуючого файлу.

Варіант 19.

Сформувати файл даних, компонентами якого є цілі числа.

Записати в інший файл даних величини, які рівні: першій непарній компоненті; сумі перших двох непарних компонент; сумі перших трьох непарних компонент і т.д.

Вивести на екран вміст результуючого файлу.

Варіант 20.

Сформувати файл даних, компонентами якого є цілі числа.

Записати в інший файл даних величини, які рівні: першій компоненті; сумі перших двох компонент; сумі перших трьох компонент і т.д.

Вивести на екран вміст результуючого файлу.

Варіант 21.

Сформувати файл даних, компонентами якого є символи. Переписати в інший файл даних ті компоненти, які входять в слово “інформатика”.

Вивести на екран вміст результуючого файлу.

Варіант 22.

Сформувати файл даних, компонентами якого є символи. Переписати в інший файл даних ті компоненти, які не являються зображеннями цифр.

Вивести на екран вміст результуючого файлу.

Варіант 23.

Сформувати файл даних, компонентами якого є символи.

Переписати в інший файл даних ті компоненти, які не входять в слово “геометрія”.

Вивести на екран вміст результуючого файлу.

Варіант 24.

Сформувати файл даних, компонентами якого є символи.

Переписати в інший файл даних ті компоненти, які являються буквами латинського алфавіту.

Вивести на екран вміст результуючого файлу.

Варіант 25.

Сформувати файл даних, компонентами якого є цілі числа.

Переписати в інший файл даних ті компоненти, які можуть бути значеннями функції $\cos x$.

Вивести на екран вміст результуючого файлу.

Варіант 26.

Сформувати файл даних, компонентами якого є дійсні числа.

Переписати в інший файл даних ті компоненти, які не можуть бути значеннями функції $\sin x$.

Вивести на екран вміст результуючого файлу.

Варіант 27.

Сформувати файл даних, компонентами якого є дійсні числа.

Переписати в інший файл даних ті компоненти, які можуть бути значеннями функції $\exp x$.

Вивести на екран вміст результуючого файлу.

Варіант 28.

Сформувати файл даних, компонентами якого є дійсні числа

Переписати в інший файл даних ті компоненти, які не можуть бути значеннями функції \sqrt{x} .

Вивести на екран вміст результуючого файлу.

Варіант 29.

Сформувати файл даних, компонентами якого є дійсні числа.

Переписати в інший файл даних ті компоненти, які можуть бути аргументами функції $\lg x$.

Вивести на екран вміст результуючого файлу.

Варіант 30.

Сформувати файл даних, компонентами якого є цілі числа.

Переписати в один файл даних парні компоненти, в інший – непарні.

Вивести на екран вміст результуючих файлів.

Варіант 31.

Сформувати файл даних, компонентами якого є символи.

Переписати в інший файл даних ті компоненти, після яких йде зображення цифри.

Вивести на екран вміст результуючого файлу.

Варіант 32.

Сформувати файл даних, компонентами якого є цілі числа.

Записати в інший файл даних максимальну непарну компоненту і мінімальну парну компоненту.

Вивести на екран вміст результуючого файлу.

Варіант 33.

Сформувати файл даних, компонентами якого є числа.

Записати в інший файл даних значення максимальної і мінімальної компоненти.

Вивести на екран вміст результуючого файлу.

Варіант 34.

Сформувати файл даних, компонентами якого є числа.

Записати в інший файл даних середнє арифметичне максимальної і мінімальної компоненти і середнє арифметичне всіх компонент.

Вивести на екран вміст результуючого файлу.

Варіант 35.

Сформувати файл даних, компонентами якого є числа.

Записати в інший файл даних середнє арифметичне від'ємних компонент і середнє арифметичне додатних компонент.

Вивести на екран вміст результуючого файлу.

Варіант 36.

Сформувати файл даних, компонентами якого є цілі числа.

Записати в інший файл даних середнє арифметичне непарних компонент і середнє арифметичне парних компонент.

Вивести на екран вміст результуючого файлу.

Варіант 37.

Сформувати файл даних, компонентами якого є числа.

Записати в інший файл даних значення найбільшої від'ємної компоненти і найменшої додатної компоненти.

Вивести на екран вміст результуючого файлу.

Варіант 38.

Сформувати файл даних, компонентами якого є числа.

Записати в інший файл даних величини, які рівні: першій від'ємній компоненті; сумі перших двох від'ємних компонент; сумі перших трьох від'ємних компонент і т.д.

Вивести на екран вміст результуючого файлу.

Варіант 39.

Сформувати файл даних, компонентами якого є цілі числа.

Записати в інший файл даних величини, які рівні: першій непарній компоненті; сумі перших двох непарних компонент; сумі перших трьох непарних компонент і т.д.

Вивести на екран вміст результуючого файлу.

Варіант 40.

Сформувати файл даних, компонентами якого є цілі числа.

Записати в інший файл даних величини, які рівні: першій компоненті; сумі перших двох компонент; сумі перших трьох компонент і т.д.

Вивести на екран вміст результуючого файлу.

Лабораторна робота № 11.2. Послідовний пошук у бінарному файлі

Мета роботи

Навчитися опрацьовувати бінарні файли.

Питання, які необхідно вивчити та пояснити на захисті

- 1) Загальний синтаксис оголошення бінарних файлів.
- 2) Доступ до елементів бінарних файлів.
- 3) Дії з бінарними файлами.
- 4) Внутрішня реалізація бінарних файлів.
- 5) Передавання та опрацювання бінарних файлів у підпрограмах.
- 6) Загальна схема послідовного пошуку у файлі.

Оформлення звіту

Вимоги та зразок оформлення звіту наведені у вступі до лабораторного практикуму.

Варіанти лабораторних завдань

Рівень А). Сформувати файл структур, що містять інформацію про: прізвище студента, курс, спеціальність (для представлення спеціальності використовувати переліки, а для представлення курсу – цілі числа) та оцінки з фізики, математики, інформатики.

Рівень В). Сформувати файл структур з об'єднаннями, що містять інформацію про: прізвище студента, курс, спеціальність (для представлення спеціальності використовувати переліки, а для представлення курсу – цілі числа) та оцінки з фізики, математики; якщо спеціальність – «Комп'ютерні науки», то третя оцінка – з програмування; якщо спеціальність – «Інформатика», то третя оцінка – з чисельних методів; для всіх інших спеціальностей: «Математика та економіка», «Фізика та інформатика», «Трудове навчання» – третя оцінка – з педагогіки. Для представлення третьої оцінки використовувати об'єднання.

Сформований файл вивести на екран у вигляді таблиці, кожний рядок якої містить:

Рівень А) – сім клітинок: (1) – порядковий номер студента у групі, (2) – прізвище, (3) – курс, (4) – спеціальність, оцінки з (5) – фізики, (6) – математики, (7) – інформатики;

Рівень В) – дев'ять клітинок: (1) – порядковий номер студента у групі, (2) – прізвище, (3) – курс, (4) – спеціальність, оцінки з (5) – фізики, (6) – математики (клітинки цих оцінок заповнені для студентів всіх спеціальностей), оцінки з (7) – програмування, (8) –

чисельних методів, (9) – педагогіки (заповнена лише одна з клітинок цих оцінок – залежно від спеціальності цього студента).

Таблиця має містити заголовок і шапку.

Всі дії над даними слід виконувати у файлі – не можна копіювати вміст усього файлу в масив. Ім'я файлу має вводитися з клавіатури.

Всі дії – формування, опрацювання даних та вивід результатів – слід реалізувати окремими функціями, які всю необхідну інформацію отримують за допомогою параметрів. Використання глобальних змінних – не допускається.

Кожна функція має виконувати лише одну роль, і ця роль має бути відображена у назві функції.

«Функція, яка повертає / обчислює / шукає ...» – має не виводити ці значення, а повернути їх у місце виклику як результат функції або як відповідний вихідний параметр.

Для зарахування лабораторної роботи достатньо виконати одне з двох завдань, кожне з яких оцінюється у 50% від максимальної кількості балів.

Варіант 1.

1. Вивести прізвища студентів, які вчаться на «відмінно».
2. Обчислити процент студентів, у яких середній бал більший за 4,5.

Варіант 2.

1. Обчислити кількість студентів, які вчаться без трійок (на «відмінно» і «добре»).
2. Обчислити процент студентів, у яких середній бал менший 4.

Варіант 3.

1. Вивести прізвища студентів, які вчаться без трійок (на «відмінно» і «добре»).
2. Обчислити кількість студентів, які отримали з фізики оцінку «5».

Варіант 4.

1. Обчислити кількість студентів, які вчаться на «відмінно».
2. Обчислити процент студентів, які отримали з фізики оцінку «5».

Варіант 5.

1. Обчислити процент студентів, які вчаться на «відмінно».
2. Вивести прізвища студентів, які отримали з фізики оцінку «5».

Варіант 6.

1. Обчислити процент студентів, які вчаться без трійок (на «відмінно» і «добре»).
2. Вивести прізвища студентів, які отримали з фізики оцінки «5» або «4».

Варіант 7.

1. Для кожного студента вивести: прізвище і середній бал.
2. Обчислити процент студентів, які отримали з фізики оцінки «5» або «4».

Варіант 8.

1. Для кожного предмету обчислити середній бал.
2. Обчислити кількість студентів, які отримали з фізики оцінки «5» або «4».

Варіант 9.

1. Обчислити кількість оцінок «відмінно» з кожного предмету.
2. Обчислити кількість студентів, які отримали з фізики і математики оцінки «5».

Варіант 10.

1. Обчислити кількість оцінок «добре» з кожного предмету.
2. Обчислити процент студентів, які отримали і з фізики і з математики оцінку «5».

Варіант 11.

1. Обчислити кількість оцінок «задовільно» з кожного предмету.
2. Вивести прізвища студентів, які отримали і з фізики і з математики оцінки «5» або «4».

Варіант 12.

1. Обчислити кількість кожної з оцінок «5», «4», «3» з фізики.
2. Обчислити кількість студентів, які отримали і з фізики і з математики оцінки «4» або «5».

Варіант 13.

1. Обчислити кількість кожної з оцінок «5», «4», «3» з математики.
2. Вивести прізвища студентів, які отримали і з фізики і з математики оцінку «5».

Варіант 14.

1. Обчислити кількість кожної з оцінок «5», «4», «3» з програмування.

2. Обчислити процент студентів, які отримали і з фізики і з математики оцінки «4» або «5».

Варіант 15.

1. Обчислити найбільший середній бал (порівнюючи середні бали для кожного студента).
2. Обчислити процент студентів, які отримали з фізики оцінки «5» або «4».

Варіант 16.

1. Обчислити кількість студентів, середній бал яких вищий за 4,5.
2. Порівнюючи середні бали для кожного предмету, визначити предмет, середній бал якого найбільший.

Варіант 17.

1. Обчислити кількість студентів, середній бал яких менший 4.
2. Порівнюючи середні бали для кожного предмету, визначити предмет, середній бал якого найменший.

Варіант 18.

1. Обчислити найменший середній бал (порівнюючи середні бали для кожного студента).
2. Обчислити кількість оцінок «добре» з кожного предмету.

Варіант 19.

1. Вивести прізвище студента, у якого найбільший середній бал.
2. Обчислити процент студентів, які вчаться на «відмінно».

Варіант 20.

1. Вивести прізвище студента, у якого найменший середній бал.
2. Обчислити кількість оцінок «задовільно» з кожного предмету.

Варіант 21.

1. Вивести прізвища студентів, які вчаться на «відмінно».
2. Обчислити процент студентів, у яких середній бал більший за 4,5.

Варіант 22.

1. Обчислити кількість студентів, які вчаться без трійок (на «відмінно» і «добре»).
2. Обчислити процент студентів, у яких середній бал менший 4.

Варіант 23.

1. Вивести прізвища студентів, які вчаться без трійок (на «відмінно» і «добре»).
2. Обчислити кількість студентів, які отримали з фізики оцінку «5».

Варіант 24.

1. Обчислити кількість студентів, які вчаться на «відмінно».
2. Обчислити процент студентів, які отримали з фізики оцінку «5».

Варіант 25.

1. Обчислити процент студентів, які вчаться на «відмінно».
2. Вивести прізвища студентів, які отримали з фізики оцінку «5».

Варіант 26.

1. Обчислити процент студентів, які вчаться без трійок (на «відмінно» і «добре»).
2. Вивести прізвища студентів, які отримали з фізики оцінки «5» або «4».

Варіант 27.

1. Для кожного студента вивести: прізвище і середній бал.
2. Обчислити процент студентів, які отримали з фізики оцінки «5» або «4».

Варіант 28.

1. Для кожного предмету обчислити середній бал.
2. Обчислити кількість студентів, які отримали з фізики оцінки «5» або «4».

Варіант 29.

1. Обчислити кількість оцінок «відмінно» з кожного предмету.
2. Обчислити кількість студентів, які отримали з фізики і математики оцінки «5».

Варіант 30.

1. Обчислити кількість оцінок «добре» з кожного предмету.
2. Обчислити процент студентів, які отримали і з фізики і з математики оцінку «5».

Варіант 31.

1. Обчислити кількість оцінок «задовільно» з кожного предмету.
2. Вивести прізвища студентів, які отримали і з фізики і з математики оцінки «5» або «4».

Варіант 32.

1. Обчислити кількість кожної з оцінок «5», «4», «3» з фізики.
2. Обчислити кількість студентів, які отримали і з фізики і з математики оцінки «4» або «5».

Варіант 33.

1. Обчислити кількість кожної з оцінок «5», «4», «3» з математики.
2. Вивести прізвища студентів, які отримали і з фізики і з математики оцінку «5».

Варіант 34.

1. Обчислити кількість кожної з оцінок «5», «4», «3» з програмування.
2. Обчислити процент студентів, які отримали і з фізики і з математики оцінки «4» або «5».

Варіант 35.

1. Обчислити найбільший середній бал (порівнюючи середні бали для кожного студента).
2. Обчислити процент студентів, які отримали з фізики оцінки «5» або «4».

Варіант 36.

1. Обчислити кількість студентів, середній бал яких вищий за 4,5.
2. Порівнюючи середні бали для кожного предмету, визначити предмет, середній бал якого найбільший.

Варіант 37.

1. Обчислити кількість студентів, середній бал яких менший 4.
2. Порівнюючи середні бали для кожного предмету, визначити предмет, середній бал якого найменший.

Варіант 38.

1. Обчислити найменший середній бал (порівнюючи середні бали для кожного студента).
2. Обчислити кількість оцінок «добре» з кожного предмету.

Варіант 39.

1. Вивести прізвище студента, у якого найбільший середній бал.
2. Обчислити процент студентів, які вчаться на «відмінно».

Варіант 40.

1. Вивести прізвище студента, у якого найменший середній бал.
2. Обчислити кількість оцінок «задовільно» з кожного предмету.

Лабораторна робота № 11.3. Впорядкування та бінарний пошук у бінарному файлі

Мета роботи

Навчитися впорядковувати файл структур з об'єднаннями. Навчитися здійснювати фізичне та індексне впорядкування. Навчитися здійснювати бінарний пошук у фізично чи індексно впорядкованому файлі.

Питання, які необхідно вивчити та пояснити на захисті

- 1) *Загальний синтаксис оголошення файлів записів.*
- 2) *Доступ до компонентів файлу записів.*
- 3) *Дії з файлами записів.*
- 4) *Внутрішня реалізація файлів записів.*
- 5) *Передавання та опрацювання файлів записів у підпрограмах.*
- 6) *Загальна схема фізичного впорядкування файлу.*
- 7) *Загальна схема індексного впорядкування файлу.*
- 8) *Загальна схема послідовного пошуку у файлі.*
- 9) *Загальна схема бінарного пошуку у файлі.*

Оформлення звіту

Вимоги та зразок оформлення звіту наведені у вступі до лабораторного практикуму.

Варіанти лабораторних завдань

Рівень А). Сформувати файл структур, що містять інформацію про: прізвище студента, курс, спеціальність (для представлення спеціальності використовувати переліки, а для представлення курсу – цілі числа) та оцінки з фізики, математики, інформатики.

Рівень В). Сформувати файл структур з об'єднаннями, що містять інформацію про: прізвище студента, курс, спеціальність (для представлення спеціальності використовувати переліки, а для представлення курсу – цілі числа) та оцінки з фізики, математики; якщо спеціальність – «Комп'ютерні науки», то третя оцінка – з програмування; якщо спеціальність – «Інформатика», то третя оцінка – з чисельних методів; для всіх інших спеціальностей: «Математика та економіка», «Фізика та інформатика», «Трудове навчання» – третя оцінка – з педагогіки. Для представлення третьої оцінки використовувати об'єднання.

Сформований файл вивести на екран у вигляді таблиці, кожний рядок якої містить:

Рівень А) – сім клітинок: (1) – порядковий номер студента у групі, (2) – прізвище, (3) – курс, (4) – спеціальність, оцінки з (5) – фізики, (6) – математики, (7) – інформатики;

Рівень В) – дев'ять клітинок: (1) – порядковий номер студента у групі, (2) – прізвище, (3) – курс, (4) – спеціальність, оцінки з (5) – фізики, (6) – математики (клітинки цих оцінок заповнені для студентів всіх спеціальностей), оцінки з (7) – програмування, (8) – чисельних методів, (9) – педагогіки (заповнена лише одна з клітинок цих оцінок – залежно від спеціальності цього студента).

Таблиця має містити заголовок і шапку.

Програма має містити меню. Необхідно передбачити контроль помилок користувача при введенні даних.

Всі дії – формування, опрацювання файлу даних та виведення результатів – слід реалізувати окремими підпрограмами, які всю необхідну інформацію отримують за допомогою параметрів. Використання глобальних змінних – не допускається.

Всі дії над даними слід виконувати у файлі – не можна копіювати вміст усього файлу в масив. Ім'я файлу має вводитися з клавіатури.

Кожна функція має виконувати лише одну роль, і ця роль має бути відображена у назві функції.

«Функція, яка повертає / обчислює / шукає ...» – має не виводити ці значення, а повернути їх у місце виклику як результат функції або як відповідний вихідний параметр.

Для зарахування лабораторної роботи достатньо виконати одне з трьох завдань, кожне з яких оцінюється незалежно від інших.

Варіант 1.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за номером курсу, в останню чергу (для однакових спеціальностей і курсів) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за номером курсу, в останню чергу (для однакових спеціальностей і курсів) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності на вказаному курсі.

Варіант 2.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за назвою спеціальності, в останню чергу (для однакових курсів і спеціальностей) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за назвою спеціальності, в останню чергу (для однакових курсів і спеціальностей) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі на вказаній спеціальності.

Варіант 3.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за значенням середнього балу, в останню чергу (для однакових спеціальностей і середніх балів) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за значенням середнього балу, в останню чергу (для однакових спеціальностей і середніх балів) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності та вказаним середнім балом.

Варіант 4.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за значенням середнього балу, в останню чергу (для однакових курсів і середніх балів) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за значенням середнього балу, в останню чергу (для однакових курсів і середніх балів) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та вказаним середнім балом.

Варіант 5.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за значенням оцінки з математики, в останню чергу (для однакових спеціальностей і оцінок з математики) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за значенням оцінки з математики, в останню чергу (для однакових спеціальностей і оцінок з математики) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності та вказаною оцінкою з математики.

Варіант 6.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за значенням оцінки з математики, в останню чергу (для однакових курсів і оцінок з математики) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за значенням оцінки з математики, в останню чергу (для однакових курсів і оцінок з математики) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та вказаною оцінкою з математики.

Варіант 7.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за значенням оцінки з третього - профільного - предмету, в останню чергу (для однакових спеціальностей і оцінок з третього предмету) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за значенням оцінки з третього – профільного –

предмету, в останню чергу (для однакових спеціальностей і оцінок з третього предмету) – за прізвищем.

3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності та вказаною оцінкою з третього предмету.

Варіант 8.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за значенням оцінки з третього – профільного – предмету, в останню чергу (для однакових курсів і оцінок з третього предмету) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за значенням оцінки з третього – профільного – предмету, в останню чергу (для однакових курсів і оцінок з третього предмету) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та вказаною оцінкою з третього предмету.

Варіант 9.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за назвою спеціальності, в останню чергу (для однакових середніх балів і спеціальностей) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за назвою спеціальності, в останню чергу (для однакових середніх балів і спеціальностей) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності та вказаним середнім балом.

Варіант 10.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) –

за номером курсу, в останню чергу (для однакових середніх балів і курсів) – за прізвищем за спаданням – в зворотному до алфавітного порядку.

2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за номером курсу, в останню чергу (для однакових середніх балів і курсів) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та вказаним середнім балом.

Варіант 11.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням оцінки з фізики, в другу чергу (для однакових оцінок з фізики) – за назвою спеціальності, в останню чергу (для однакових оцінок з фізики і спеціальностей) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням оцінки з фізики, в другу чергу (для однакових оцінок з фізики) – за назвою спеціальності, в останню чергу (для однакових оцінок з фізики і спеціальностей) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності та вказаною оцінкою з фізики.

Варіант 12.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням оцінки з фізики, в другу чергу (для однакових оцінок з фізики) – за номером курсу, в останню чергу (для однакових оцінок з фізики і курсів) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням оцінки з фізики, в другу чергу (для однакових оцінок з фізики) – за номером курсу, в останню чергу (для однакових оцінок з фізики і курсів) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та вказаною оцінкою з фізики.

Варіант 13.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням оцінки з третього – профільного – предмету, в другу чергу (для однакових оцінок з третього предмету) – за назвою спеціальності, в останню чергу (для однакових оцінок з третього предмету і спеціальностей) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням оцінки з третього – профільного – предмету, в другу чергу (для однакових оцінок з третього предмету) – за назвою спеціальності, в останню чергу (для однакових оцінок з третього предмету і спеціальностей) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності та вказаною оцінкою з третього предмету.

Варіант 14.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням оцінки з третього – профільного – предмету, в другу чергу (для однакових оцінок з третього предмету) – за номером курсу, в останню чергу (для однакових оцінок з третього предмету і курсів) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням оцінки з третього – профільного – предмету, в другу чергу (для однакових оцінок з третього предмету) – за номером курсу, в останню чергу (для однакових оцінок з третього предмету і курсів) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та вказаною оцінкою з третього предмету.

Варіант 15.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за значенням оцінки з математики, в останню чергу (для однакових середніх балів і оцінок з математики) – за прізвищем за зростанням – в алфавітному порядку.

2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за значенням оцінки з математики, в останню чергу (для однакових середніх балів і оцінок з математики) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем та з вказаним середнім балом та вказаною оцінкою з математики.

Варіант 16.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за значенням оцінки з третього предмету, в останню чергу (для однакових середніх балів і оцінок з третього предмету) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за значенням оцінки з третього предмету, в останню чергу (для однакових середніх балів і оцінок з третього предмету) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем та з вказаним середнім балом і вказаною оцінкою з третього предмету.

Варіант 17.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням оцінки з третього – профільного – предмету, в другу чергу (для однакових оцінок з третього предмету) – за значенням середнього балу, в останню чергу (для однакових оцінок з третього предмету і середніх балів) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням оцінки з третього – профільного – предмету, в другу чергу (для однакових оцінок з третього предмету) – за значенням середнього балу, в останню чергу (для однакових оцінок з третього предмету і середніх балів) – за прізвищем.

3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем та з вказаним середнім балом і вказаною оцінкою з третього предмету.

Варіант 18.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за прізвищем за спаданням – в зворотному до алфавітного порядку, в останню чергу (для однакових спеціальностей і прізвищ) – за номером курсу.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за прізвищем, в останню чергу (для однакових спеціальностей і прізвищ) – за номером курсу.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та на вказаній спеціальності.

Варіант 19.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за прізвищем за зростанням – в алфавітному порядку, в останню чергу (для однакових курсів і прізвищ) – за назвою спеціальності.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за прізвищем, в останню чергу (для однакових курсів і прізвищ) – за назвою спеціальності.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та на вказаній спеціальності.

Варіант 20.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за прізвищем за спаданням – в зворотному до алфавітного порядку, в останню чергу (для однакових середніх балів і прізвищ) – за номером курсу.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням середнього балу, в другу чергу (для

однакових середніх балів) – за прізвищем, в останню чергу (для однакових середніх балів і прізвищ) – за номером курсу.

3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та з вказаним середнім балом.

Варіант 21.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за номером курсу, в останню чергу (для однакових спеціальностей і курсів) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за номером курсу, в останню чергу (для однакових спеціальностей і курсів) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності на вказаному курсі.

Варіант 22.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за назвою спеціальності, в останню чергу (для однакових курсів і спеціальностей) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за назвою спеціальності, в останню чергу (для однакових курсів і спеціальностей) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі на вказаній спеціальності.

Варіант 23.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за значенням середнього балу, в останню чергу (для однакових спеціальностей і середніх балів) – за прізвищем за зростанням – в алфавітному порядку.

2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за значенням середнього балу, в останню чергу (для однакових спеціальностей і середніх балів) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності та вказаним середнім балом.

Варіант 24.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за значенням середнього балу, в останню чергу (для однакових курсів і середніх балів) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за значенням середнього балу, в останню чергу (для однакових курсів і середніх балів) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та вказаним середнім балом.

Варіант 25.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за значенням оцінки з математики, в останню чергу (для однакових спеціальностей і оцінок з математики) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за значенням оцінки з математики, в останню чергу (для однакових спеціальностей і оцінок з математики) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності та вказаною оцінкою з математики.

Варіант 26.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за значенням оцінки

- з математики, в останню чергу (для однакових курсів і оцінок з математики) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за значенням оцінки з математики, в останню чергу (для однакових курсів і оцінок з математики) – за прізвищем.
 3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та вказаною оцінкою з математики.

Варіант 27.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за значенням оцінки з третього - профільного - предмету, в останню чергу (для однакових спеціальностей і оцінок з третього предмету) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за значенням оцінки з третього – профільного – предмету, в останню чергу (для однакових спеціальностей і оцінок з третього предмету) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності та вказаною оцінкою з третього предмету.

Варіант 28.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за значенням оцінки з третього – профільного – предмету, в останню чергу (для однакових курсів і оцінок з третього предмету) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за значенням оцінки з третього – профільного – предмету, в останню чергу (для однакових курсів і оцінок з третього предмету) – за прізвищем.

3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та вказаною оцінкою з третього предмету.

Варіант 29.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за назвою спеціальності, в останню чергу (для однакових середніх балів і спеціальностей) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за назвою спеціальності, в останню чергу (для однакових середніх балів і спеціальностей) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності та вказаним середнім балом.

Варіант 30.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за номером курсу, в останню чергу (для однакових середніх балів і курсів) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за номером курсу, в останню чергу (для однакових середніх балів і курсів) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та вказаним середнім балом.

Варіант 31.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням оцінки з фізики, в другу чергу (для однакових оцінок з фізики) – за назвою спеціальності, в останню чергу (для однакових оцінок з фізики і спеціальностей) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням оцінки з фізики, в другу чергу (для

однакових оцінок з фізики) – за назвою спеціальності, в останню чергу (для однакових оцінок з фізики і спеціальностей) – за прізвищем.

3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності та вказаною оцінкою з фізики.

Варіант 32.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням оцінки з фізики, в другу чергу (для однакових оцінок з фізики) – за номером курсу, в останню чергу (для однакових оцінок з фізики і курсів) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням оцінки з фізики, в другу чергу (для однакових оцінок з фізики) – за номером курсу, в останню чергу (для однакових оцінок з фізики і курсів) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та вказаною оцінкою з фізики.

Варіант 33.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням оцінки з третього – профільного – предмету, в другу чергу (для однакових оцінок з третього предмету) – за назвою спеціальності, в останню чергу (для однакових оцінок з третього предмету і спеціальностей) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням оцінки з третього – профільного – предмету, в другу чергу (для однакових оцінок з третього предмету) – за назвою спеціальності, в останню чергу (для однакових оцінок з третього предмету і спеціальностей) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаній спеціальності та вказаною оцінкою з третього предмету.

Варіант 34.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням оцінки з третього – профільного – предмету, в другу чергу (для

однакових оцінок з третього предмету) – за номером курсу, в останню чергу (для однакових оцінок з третього предмету і курсів) – за прізвищем за спаданням – в зворотному до алфавітного порядку.

2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням оцінки з третього – профільного – предмету, в другу чергу (для однакових оцінок з третього предмету) – за номером курсу, в останню чергу (для однакових оцінок з третього предмету і курсів) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та вказаною оцінкою з третього предмету.

Варіант 35.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за значенням оцінки з математики, в останню чергу (для однакових середніх балів і оцінок з математики) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за значенням оцінки з математики, в останню чергу (для однакових середніх балів і оцінок з математики) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем та з вказаним середнім балом та вказаною оцінкою з математики.

Варіант 36.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за значенням оцінки з третього предмету, в останню чергу (для однакових середніх балів і оцінок з третього предмету) – за прізвищем за спаданням – в зворотному до алфавітного порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за значенням оцінки з третього предмету, в останню чергу (для однакових середніх балів і оцінок з третього предмету) – за прізвищем.

3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем та з вказаним середнім балом і вказаною оцінкою з третього предмету.

Варіант 37.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням оцінки з третього – профільного – предмету, в другу чергу (для однакових оцінок з третього предмету) – за значенням середнього балу, в останню чергу (для однакових оцінок з третього предмету і середніх балів) – за прізвищем за зростанням – в алфавітному порядку.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням оцінки з третього – профільного – предмету, в другу чергу (для однакових оцінок з третього предмету) – за значенням середнього балу, в останню чергу (для однакових оцінок з третього предмету і середніх балів) – за прізвищем.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем та з вказаним середнім балом і вказаною оцінкою з третього предмету.

Варіант 38.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за прізвищем за спаданням – в зворотному до алфавітного порядку, в останню чергу (для однакових спеціальностей і прізвищ) – за номером курсу.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за назвою спеціальності, в другу чергу (для однакових спеціальностей) – за прізвищем, в останню чергу (для однакових спеціальностей і прізвищ) – за номером курсу.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та на вказаній спеціальності.

Варіант 39.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за прізвищем за

зростанням – в алфавітному порядку, в останню чергу (для однакових курсів і прізвищ) – за назвою спеціальності.

2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за номером курсу, в другу чергу (для однакових курсів) – за прізвищем, в останню чергу (для однакових курсів і прізвищ) – за назвою спеціальності.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та на вказаній спеціальності.

Варіант 40.

1. Програма має дати користувачеві можливість фізично впорядкувати файл в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за прізвищем за спаданням – в зворотному до алфавітного порядку, в останню чергу (для однакових середніх балів і прізвищ) – за номером курсу.
2. Програма має будувати індексний файл, який забезпечує наступне індексне впорядкування: в першу чергу – за значенням середнього балу, в другу чергу (для однакових середніх балів) – за прізвищем, в останню чергу (для однакових середніх балів і прізвищ) – за номером курсу.
3. За допомогою бінарного пошуку визначити, чи навчається студент із вказаним користувачем прізвищем на вказаному курсі та з вказаним середнім балом.

Лабораторна робота № 11.4. Опрацювання бінарного файлу

Мета роботи

Навчитися опрацьовувати бінарні файли – файли записів.

Питання, які необхідно вивчити та пояснити на захисті

- 1) Загальний синтаксис оголошення бінарних файлів (файлів записів).*
- 2) Доступ до елементів бінарних файлів (файлів записів).*
- 3) Дії з бінарними файлами (файлами записів).*
- 4) Внутрішня реалізація бінарних файлів (файлів записів).*
- 5) Передавання та опрацювання бінарних файлів (файлів записів) у підпрограмах.*
- 6) Загальна схема фізичного впорядкування файлу.*
- 7) Загальна схема індексного впорядкування файлу.*
- 8) Загальна схема послідовного пошуку у файлі.*
- 9) Загальна схема бінарного пошуку у файлі.*

Оформлення звіту

Вимоги та зразок оформлення звіту наведені у вступі до лабораторного практикуму.

Варіанти лабораторних завдань

Кожна програма має містити меню. Необхідно передбачити контроль помилок користувача при введенні даних.

Необхідні програмі дані слід реалізувати за допомогою типізованого файлу (файлу записів). Всі дії над даними слід виконувати у файлі – не можна копіювати вміст усього файлу в масив. Ім'я файлу має вводитися з клавіатури.

При розробці програми застосувати технологію низхідного проектування. Логічно закінчені фрагменти оформити у вигляді підпрограм, всі необхідні дані яким передаються через список параметрів. Використовувати глобальні змінні – не можна.

Кожна функція має виконувати лише одну роль, і ця роль має бути відображена у назві функції.

«Функція, яка повертає / обчислює / шукає ...» – має не виводити ці значення, а повернути їх у місце виклику як результат функції або як відповідний вихідний параметр.

Варіант 1.

Звітна відомість результатів екзаменаційної сесії студентської групи для кожного студента містить прізвище, ініціали і оцінки з п'яти предметів.

Скласти програму, за допомогою якої можна коректувати список (добавляти, вилучати, редагувати інформацію) і отримувати:

- список всіх студентів;
- список студентів, що склали іспити тільки на «5»;
- список студентів, що мають трійки;
- список студентів, що мають двійки. При цьому студент, що має більш ніж одну двійку, виключається із списку.

Варіант 2.

Підприємство має місцеву телефонну станцію. Телефонний довідник цього підприємства для кожного номера телефону містить номер приміщення і список службовців, що сидять в цьому приміщенні.

Скласти програму, яка:

- коректує базу (добавляє, вилучає, редагує інформацію);
- за номером телефону видає номер приміщення і список людей, що сидять в ньому;
- за номером приміщення видає номер телефону;
- за прізвищем службовця видає номер телефону і номер приміщення.

Номер телефону – двозначний. У одному приміщенні може знаходитися від одного до чотирьох службовців.

Варіант 3.

У готелі є 15 номерів, з них 5 одномісних і 10 двомісних. Скласти програму, яка заповнює та корегує дані про мешканців (добавляє, вилучає, редагує інформацію) і за прізвищем визначає номер, де проживає мешканець.

Програма запрошує прізвище мешканця.

- Якщо мешканця з таким прізвищем немає, про це видається повідомлення.
- Якщо мешканець з таким прізвищем в готелі єдиний, програма видає прізвище мешканця і номер помешкання.
- Якщо в готелі проживає два або більше мешканців з таким прізвищем, програма додатково запрошує ініціали.

Варіант 4.

Є список службовців. Для кожного службовця вказано прізвище і ініціали, назву посади, рік прийому на роботу і оклад (величину заробітної плати).

Написати програму, що виконує наступні дії:

- корегування списку з клавіатури (добавлення, вилучення, редагування інформації);
- сортування за прізвищем, окладом або роком прийому на роботу;
- виведення на екран інформації про службовця, прізвище якого введене з клавіатури.

Варіант 5.

Розклад електричок зберігається у вигляді сукупності записів. Кожен запис містить назву пункту призначення, позначки типу «звичайний», «підвищеного комфорту», «швидкісний експрес» та час відправлення.

Написати програму, що виконує наступні дії:

- коректування розкладу з клавіатури (добавлення, вилучення, редагування інформації);
- сортування за станцією призначення або за часом відправлення;
- виведення на екран інформації про поїзди, що відходять після введеного часу.

Варіант 6.

Є список товарів. Для кожного товару вказані його назва, вартість одиниці товару в гривнях, кількість і одиниця вимірювання (наприклад, кількість: 100 шт., одиниця вимірювання: упаковка по 20 кг.).

Написати програму, що виконує наступні дії:

- коректування списку з клавіатури (добавлення, вилучення, редагування інформації);
- сортування по назві товару або за загальною вартістю;
- виведення на екран інформації про товар, назва якого введена з клавіатури;
- виведення на екран інформації про товари із заданого з клавіатури діапазону вартості.

Варіант 7.

Є список товарів. Для кожного товару вказані його назва, назва магазину, в якому продається товар, вартість одиниці товару в гривнях і його кількість з вказівкою одиниці вимірювання (наприклад, кількість: 100 шт., одиниця вимірювання: упаковка по 20 кг.).

Написати програму, що виконує наступні дії:

- коректування списку з клавіатури (добавлення, вилучення, редагування інформації);
- сортування за назвою товару або за назвою магазину;
- виведення на екран інформації про товар, назва якого введена з клавіатури;
- виведення на екран інформації про товари, що продаються в магазині, назва якого введена з клавіатури.

Варіант 8.

Є список студентської групи. Кожен рядок списку містить прізвище студента і три екзаменаційні оцінки, причому список ніяк не впорядкований.

Скласти програму, яка коректує список (добавляє, вилучає, редагує інформацію) і сортує його або за середнім балом, або за прізвищами – в алфавітному порядку, або за оцінками із заданого предмету.

Варіант 9.

Є список товарів. Для кожного товару вказані його назва, назва магазину, в якому продається товар, вартість одиниці товару в гривнях і його кількість з вказівкою одиниці вимірювання (наприклад, кількість: 100 шт., одиниця вимірювання: упаковка по 20 кг.).

Написати програму, що виконує наступні дії:

- коректування списку з клавіатури (добавлення, вилучення, редагування інформації);
- сортування за назвою магазину або за загальною вартістю;
- виведення на екран інформації про товари, що продаються в магазині, назва якого введена з клавіатури;
- виведення на екран інформації про товари із заданого з клавіатури діапазону вартості.

Варіант 10.

Описати структуру з ім'ям **Route**, що містить наступні поля:

- назву початкового пункту маршруту;

- назву кінцевого пункту маршруту;
- номер маршруту.

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу **Route**;
- впорядкування записів за номерами маршрутів;
- виведення на екран інформації про маршрут, номер якого введений з клавіатури; якщо такого маршруту немає, вивести на екран відповідне повідомлення.

Варіант 11.

Описати структуру з ім'ям **Route**, що містить наступні поля:

- назву початкового пункту маршруту;
- назву кінцевого пункту маршруту;
- номер маршруту.

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу **Route**;
- впорядкування записів за номерами маршрутів;
- виведення на екран інформації про маршрути, які починаються або закінчуються в пункті, назва якого введена з клавіатури; якщо таких маршрутів немає, вивести на екран відповідне повідомлення.

Варіант 12.

Описати структуру з ім'ям **Note**, що містить наступні поля:

- прізвище, ім'я;
- номер телефону;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу **Note**;
- впорядкування записів за датами днів народження;
- виведення на екран інформації про людину, номер телефону якої введений з клавіатури; якщо такої людини немає, вивести на екран відповідне повідомлення.

Варіант 13.

Описати структуру з ім'ям **Note**, що містить наступні поля:

- прізвище, ім'я;

- номер телефону;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу **Note**;
- впорядкування записів за алфавітом;
- виведення на екран інформації про людей, що народилися в тому місяці, значення якого введене з клавіатури; якщо таких немає, вивести на екран відповідне повідомлення.

Варіант 14.

Описати структуру з ім'ям **Note**, що містить наступні поля:

- прізвище, ім'я;
- номер телефону;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу **Note**;
- впорядкування записів за телефонними номерами;
- виведення на екран інформації про людину, прізвище якої введене з клавіатури; якщо такої немає, вивести на екран відповідне повідомлення.

Варіант 15.

Описати структуру з ім'ям **Zodiac**, що містить наступні поля:

- прізвище, ім'я;
- знак Зодіаку;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу **Zodiac**;
- впорядкування записів за датами днів народження.
- виведення на екран інформації про людину, чиє прізвище введене з клавіатури; якщо такої людини немає, вивести на екран відповідне повідомлення.

Варіант 16.

Описати структуру з ім'ям **Zodiac**, що містить наступні поля:

- прізвище, ім'я;
- знак Зодіаку;

- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу `Zodiac`;
- впорядкування записів за прізвищами.
- виведення на екран інформації про людей, що народилися під знаком, найменування якого введене з клавіатури; якщо таких немає, вивести на екран відповідне повідомлення.

Варіант 17.

Описати структуру з ім'ям `Zodiac`, що містить наступні поля:

- прізвище, ім'я;
- знак Зодіаку;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу `Zodiac`;
- впорядкування записів за знаками Зодіаку;
- виведення на екран інформації про людей, що народилися в тому місяці, значення якого введене з клавіатури; якщо таких немає, вивести на екран відповідне повідомлення.

Варіант 18.

Описати структуру з ім'ям `Price`, що містить наступні поля:

- назва товару;
- назва магазину, в якому продається товар;
- вартість товару в гривнях.

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу `Price`;
- впорядкування записів за алфавітним порядком назв магазинів;
- виведення на екран інформації про товар, назва якого введена з клавіатури; якщо таких товарів немає, вивести на екран відповідне повідомлення.

Варіант 19.

Описати структуру з ім'ям `Price`, що містить наступні поля:

- назва товару;
- назва магазину, в якому продається товар;

- вартість товару в гривнях.

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу Price;
- впорядкування записів за алфавітним порядком назв товарів;
- виведення на екран інформації про товари, що продаються в магазині, назва якого введена з клавіатури; якщо такого магазину немає, вивести на екран відповідне повідомлення.

Варіант 20.

Описати структуру з ім'ям **Bill**, що містить наступні поля:

- розрахунковий рахунок платника;
- розрахунковий рахунок одержувача;
- перерахована сума в гривнях.

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу **Bill**;
- впорядкування записів за алфавітним порядок розрахункових рахунків платників;
- виведення на екран інформації про суму, зняту з розрахункового рахунку платника, введеного з клавіатури; якщо такого розрахункового рахунку немає, вивести на екран відповідне повідомлення.

Варіант 21.

Звітна відомість результатів екзаменаційної сесії студентської групи для кожного студента містить прізвище, ініціали і оцінки з п'яти предметів.

Скласти програму, за допомогою якої можна коректувати список (добавляти, вилучати, редагувати інформацію) і отримувати:

- список всіх студентів;
- список студентів, що склали іспити тільки на «5»;
- список студентів, що мають трійки;
- список студентів, що мають двійки. При цьому студент, що має більш ніж одну двійку, виключається із списку.

Варіант 22.

Підприємство має місцеву телефонну станцію. Телефонний довідник цього підприємства для кожного номера телефону містить номер приміщення і список службовців, що сидять в цьому приміщенні.

Скласти програму, яка:

- коректує базу (добавляє, вилучає, редагує інформацію);
- за номером телефону видає номер приміщення і список людей, що сидять в ньому;
- за номером приміщення видає номер телефону;
- за прізвищем службовця видає номер телефону і номер приміщення.

Номер телефону – двозначний. У одному приміщенні може знаходитися від одного до чотирьох службовців.

Варіант 23.

У готелі є 15 номерів, з них 5 одномісних і 10 двомісних. Скласти програму, яка заповнює та корегує дані про мешканців (добавляє, вилучає, редагує інформацію) і за прізвищем визначає номер, де проживає мешканець.

Програма запрошує прізвище мешканця.

- Якщо мешканця з таким прізвищем немає, про це видається повідомлення.
- Якщо мешканець з таким прізвищем в готелі єдиний, програма видає прізвище мешканця і номер помешкання.
- Якщо в готелі проживає два або більше мешканців з таким прізвищем, програма додатково запрошує ініціали.

Варіант 24.

Є список службовців. Для кожного службовця вказано прізвище і ініціали, назву посади, рік прийому на роботу і оклад (величину заробітної плати).

Написати програму, що виконує наступні дії:

- корегування списку з клавіатури (добавлення, вилучення, редагування інформації);
- сортування за прізвищем, окладом або роком прийому на роботу;
- виведення на екран інформації про службовця, прізвище якого введене з клавіатури.

Варіант 25.

Розклад електричок зберігається у вигляді сукупності записів. Кожен запис містить назву пункту призначення, позначки типу «звичайний», «підвищеного комфорту», «швидкісний експрес» та час відправлення.

Написати програму, що виконує наступні дії:

- коректування розкладу з клавіатури (добавлення, вилучення, редагування інформації);
- сортування за станцією призначення або за часом відправлення;
- виведення на екран інформації про поїзди, що відходять після введеного часу.

Варіант 26.

Є список товарів. Для кожного товару вказані його назва, вартість одиниці товару в гривнях, кількість і одиниця вимірювання (наприклад, кількість: 100 шт., одиниця вимірювання: упаковка по 20 кг.).

Написати програму, що виконує наступні дії:

- коректування списку з клавіатури (добавлення, вилучення, редагування інформації);
- сортування по назві товару або за загальною вартістю;
- виведення на екран інформації про товар, назва якого введена з клавіатури;
- виведення на екран інформації про товари із заданого з клавіатури діапазону вартості.

Варіант 27.

Є список товарів. Для кожного товару вказані його назва, назва магазину, в якому продається товар, вартість одиниці товару в гривнях і його кількість з вказівкою одиниці вимірювання (наприклад, кількість: 100 шт., одиниця вимірювання: упаковка по 20 кг.).

Написати програму, що виконує наступні дії:

- коректування списку з клавіатури (добавлення, вилучення, редагування інформації);
- сортування за назвою товару або за назвою магазину;
- виведення на екран інформації про товар, назва якого введена з клавіатури;
- виведення на екран інформації про товари, що продаються в магазині, назва якого введена з клавіатури.

Варіант 28.

Є список студентської групи. Кожен рядок списку містить прізвище студента і три екзаменаційні оцінки, причому список ніяк не впорядкований.

Скласти програму, яка коректує список (добавляє, вилучає, редагує інформацію) і сортує його або за середнім балом, або за прізвищами – в алфавітному порядку, або за оцінками із заданого предмету.

Варіант 29.

Є список товарів. Для кожного товару вказані його назва, назва магазину, в якому продається товар, вартість одиниці товару в гривнях і його кількість з вказівкою одиниці вимірювання (наприклад, кількість: 100 шт., одиниця вимірювання: упаковка по 20 кг.).

Написати програму, що виконує наступні дії:

- коректування списку з клавіатури (добавлення, вилучення, редагування інформації);
- сортування за назвою магазину або за загальною вартістю;
- виведення на екран інформації про товари, що продаються в магазині, назва якого введена з клавіатури;
- виведення на екран інформації про товари із заданого з клавіатури діапазону вартості.

Варіант 30.

Описати структуру з ім'ям **Route**, що містить наступні поля:

- назву початкового пункту маршруту;
- назву кінцевого пункту маршруту;
- номер маршруту.

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу **Route**;
- впорядкування записів за номерами маршрутів;
- виведення на екран інформації про маршрут, номер якого введений з клавіатури; якщо такого маршруту немає, вивести на екран відповідне повідомлення.

Варіант 31.

Описати структуру з ім'ям **Route**, що містить наступні поля:

- назву початкового пункту маршруту;
- назву кінцевого пункту маршруту;

- номер маршруту.

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу `Route`;
- впорядкування записів за номерами маршрутів;
- виведення на екран інформації про маршрути, які починаються або закінчуються в пункті, назва якого введена з клавіатури; якщо таких маршрутів немає, вивести на екран відповідне повідомлення.

Варіант 32.

Описати структуру з ім'ям `Note`, що містить наступні поля:

- прізвище, ім'я;
- номер телефону;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу `Note`;
- впорядкування записів за датами днів народження;
- виведення на екран інформації про людину, номер телефону якої введений з клавіатури; якщо такої людини немає, вивести на екран відповідне повідомлення.

Варіант 33.

Описати структуру з ім'ям `Note`, що містить наступні поля:

- прізвище, ім'я;
- номер телефону;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу `Note`;
- впорядкування записів за алфавітом;
- виведення на екран інформації про людей, що народилися в тому місяці, значення якого введене з клавіатури; якщо таких немає, вивести на екран відповідне повідомлення.

Варіант 34.

Описати структуру з ім'ям `Note`, що містить наступні поля:

- прізвище, ім'я;

- номер телефону;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу `Note`;
- впорядкування записів за телефонними номерами;
- виведення на екран інформації про людину, прізвище якої введене з клавіатури; якщо такої немає, вивести на екран відповідне повідомлення.

Варіант 35.

Описати структуру з ім'ям `Zodiac`, що містить наступні поля:

- прізвище, ім'я;
- знак Зодіаку;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу `Zodiac`;
- впорядкування записів за датами днів народження.
- виведення на екран інформації про людину, чий прізвище введене з клавіатури; якщо такої людини немає, вивести на екран відповідне повідомлення.

Варіант 36.

Описати структуру з ім'ям `Zodiac`, що містить наступні поля:

- прізвище, ім'я;
- знак Зодіаку;
- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу `Zodiac`;
- впорядкування записів за прізвищами.
- виведення на екран інформації про людей, що народилися під знаком, найменування якого введене з клавіатури; якщо таких немає, вивести на екран відповідне повідомлення.

Варіант 37.

Описати структуру з ім'ям `Zodiac`, що містить наступні поля:

- прізвище, ім'я;
- знак Зодіаку;

- день народження (масив з трьох чисел).

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу **Zodiac**;
- впорядкування записів за знаками Зодіаку;
- виведення на екран інформації про людей, що народилися в тому місяці, значення якого введене з клавіатури; якщо таких немає, вивести на екран відповідне повідомлення.

Варіант 38.

Описати структуру з ім'ям **Price**, що містить наступні поля:

- назва товару;
- назва магазину, в якому продається товар;
- вартість товару в гривнях.

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу **Price**;
- впорядкування записів за алфавітним порядком назв магазинів;
- виведення на екран інформації про товар, назва якого введена з клавіатури; якщо таких товарів немає, вивести на екран відповідне повідомлення.

Варіант 39.

Описати структуру з ім'ям **Price**, що містить наступні поля:

- назва товару;
- назва магазину, в якому продається товар;
- вартість товару в гривнях.

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу **Price**;
- впорядкування записів за алфавітним порядком назв товарів;
- виведення на екран інформації про товари, що продаються в магазині, назва якого введена з клавіатури; якщо такого магазину немає, вивести на екран відповідне повідомлення.

Варіант 40.

Описати структуру з ім'ям **Bill**, що містить наступні поля:

- розрахунковий рахунок платника;
- розрахунковий рахунок одержувача;

- перерахована сума в гривнях.

Написати програму, що виконує наступні дії:

- введення даних з клавіатури у файл, що складається з елементів типу **Bill**;
- впорядкування записів за алфавітним порядком розрахункових рахунків платників;
- виведення на екран інформації про суму, зняту з розрахункового рахунку платника, введеного з клавіатури; якщо такого розрахункового рахунку немає, вивести на екран відповідне повідомлення.

Лабораторна робота № 11.5. Структури, об'єднання та бінарні файли

Мета роботи

Навчитися опрацьовувати бінарні файли.

Питання, які необхідно вивчити та пояснити на захисті

- 1) Загальний синтаксис оголошення бінарних файлів (файлів записів).*
- 2) Доступ до елементів бінарних файлів (файлів записів).*
- 3) Дії з бінарними файлами (файлами записів).*
- 4) Внутрішня реалізація бінарних файлів (файлів записів).*
- 5) Передавання та опрацювання бінарних файлів (файлів записів) у підпрограмах.*
- 6) Загальна схема фізичного впорядкування файлу.*
- 7) Загальна схема індексного впорядкування файлу.*
- 8) Загальна схема послідовного пошуку у файлі.*
- 9) Загальна схема бінарного пошуку у файлі.*

Оформлення звіту

Вимоги та зразок оформлення звіту наведені у вступі до лабораторного практикуму.

Варіанти лабораторних завдань

Кожна програма має містити меню. Необхідно передбачити контроль помилок користувача при введенні даних.

Необхідні програмі дані слід реалізувати за допомогою типізованого файлу (файлу записів). В завданні вказано, які дії над даними слід виконувати у файлі; якщо деяке завдання допускає різне тлумачення (чи дії виконуються з файлом, чи з масивом) – студент сам вибирає, як саме його реалізувати. Ім'я файлу має вводитися з клавіатури.

Для всіх варіантів слід реалізувати дії: запису масиву у файл, зчитування масиву з файлу, доповнити файл, замінити дані в файлі, видалити дані з файлу.

При розробці програми застосувати технологію низхідного проектування. Логічно закінчені фрагменти оформити у вигляді підпрограм, всі необхідні дані яким передаються через список параметрів. Використовувати глобальні змінні – не можна.

Всі дії над даними слід виконувати у файлі – не можна копіювати вміст усього файлу в масив. Ім'я файлу має вводитися з клавіатури.

Кожна функція має виконувати лише одну роль, і ця роль має бути відображена у назві функції.

«Функція, яка повертає / обчислює / шукає ...» – має не виводити ці значення, а повернути їх у місце виклику як результат функції або як відповідний вихідний параметр.

Варіант 1.

Створити масив структур. Кожна структура складається з таких елементів: факультет, курс, група, предмет, студент. Для предмета задається назва та кількість кредитів. Для студента задається прізвище та екзаменаційні оцінки. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити, визначивши: 1) прізвища студентів, які мають дві та більше двійок за сесію на факультеті ІФМЕІТ і вилучити їх; 2) факультет, який на першому курсі має найбільшу кількість відмінників; 3) курс, на якому виключено найбільшу кількість студентів.

Варіант 2.

Створити масив структур. Кожна структура складається з таких елементів: фірма, робітник, посада. Для робітника задається прізвище та заробітна платня. Для посади задається кількість вакантних місць, вимоги до претендентів на кожну посаду, Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити фірми з найбільшою кількістю вакансій на задану посаду; 2) вивести список вимог до кандидатів на задану посаду по кожній фірмі; 3) створити новий масив із структурою: вакантна посада, заробітна платня. Вакансії вибирати з початкового масиву, виключаючи повторення однакових вакантних посад. На випадок різної платні на однакових посадах у різних фірмах включити у масив середню заробітну платню.

Варіант 3.

Створити масив структур. Кожна структура складається з таких елементів: факультет і студент. Для факультету задати його назву, курс і масив груп. Для студента задати прізвище та масив екзаменаційних оцінок. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити факультет і курс з максимальною кількістю відмінників; 2) вивести список відмінників; 3) знайти групу, де немає двісчників.

Варіант 4.

Створити масив структур. Кожна структура складається з таких елементів: інститут та студент. Для інституту задати його назву і масив факультетів. Для студента задати курс, групу, прізвище та середній бал. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити, визначивши: 1) факультет і курс, на якому середній бал не менше 3.5; 2) прізвища студентів із , що навчаються на вказаному курсі та групі; 3) факультет і групу, де найбільше відмінників.

Варіант 5.

Створити масив структур. Кожна структура складається з таких елементів: місто, інститут, факультет. Для інституту задається план прийому на перший курс. Для факультету задається список спеціальностей. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити, визначивши: 1) список спеціальностей, що містять у своїй назві слова "комп'ютерний" або "автоматизований"; 2) факультети й інститути, де є задана спеціальність; 3) факультет, інститут і місто, де на вказану користувачем спеціальність виділено найбільше місць.

Варіант 6.

Створити масив структур, кожна з яких складається з таких елементів: факультет, курс, студент. Для факультету задати його назву та номер групи. Для студента задати прізвище студентів, екзаменаційні оцінки. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити прізвища студентів, групу і факультет, де середній бал студентів 4.5; 2) вилучити прізвища студентів першого курсу, які мають три двійки; 3) відсортувати назви факультетів за умови зростання успішності студентів.

Варіант 7.

Створити масив структур. Кожна структура складається з таких елементів: марка автомобіля, тип (вантажний або легковий), номерний знак, строк служби. Для легкових вказується колір, для вантажних – вантажність. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити номерні знаки автомобілів, вантажність котрих не

менше за задану; 2) вивести марки автомобілів заданого типу і вказаного кольору; 3) вилучити з файлу відомості про автомобілі, строк служби яких перевищує п'ять років.

Варіант 8.

Створити масив структур. Кожна структура складається з таких елементів: абонент і телефон. Для абонента задається прізвище, адреса, заборгованість по оплаті. Для телефону задається номер і оператор (МТС, Київстар тощо). Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити прізвища абонентів, що мають заборгованість по оплаті більше заданої користувачем; 2) вилучити прізвища абонентів, адреса яких змінилася; 3) замінити номер телефону заданого абонента.

Варіант 9.

Створити масив структур. Кожна структура складається з таких елементів: фірма, товар. Для фірми задається назва, кількість товарів. Для товару задається його найменування, вартість, термін поставки. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити фірми, що постачають заданий товар у дводенний строк; 2) визначити назву товару в заданій фірмі, вартість якого не перевищує заданого значення; 3) впорядкувати масив за умови зростання термінів постачання товару.

Варіант 10.

Створити масив структур. Кожна структура складається з таких елементів: інститут, факультет. Для інституту задати назву та план прийому на перший курс. Для факультету задати список спеціальностей та прохідний бал. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) скласти список спеціальностей по факультетах із вказаним прохідним балом; 2) впорядкувати масив за прохідним балом і планом прийому; 3) визначити інститут із вказаним середнім прохідним балом.

Варіант 11.

Створити масив структур. Кожна структура складається з таких елементів: літак і рейс. Для літака задати тип, кількість вільних місць, вартість квитка. Для рейсу задати номер, дата польоту. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) скласти список номерів авіаційних рейсів, на які є вільні місця; 2) вивести типи літаків і

відповідні номери авіаційних рейсів на задану дату; 3) упорядкувати масив за умови зростання вартості квитків по кожній даті.

Варіант 12.

Створити масив структур. Кожна структура складається з таких елементів: напрям польоту та літак. Для напрямку польоту задати льотні дні на тиждень і номер авіаційного рейсу, Для літака задати тип, кількість посадкових місць, вартість квитка. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити номери авіаційних рейсів заданого напрямку з мінімальною вартістю квитків; 2) визначити типи літаків і сумарну кількість посадкових місць у заданому напрямку по днях тижня; 3) вилучити з масиву відомості про рейси, якщо кількість польотів на тиждень менше двох.

Варіант 13.

Створити масив структур. Кожна структура складається з таких елементів : виробник, товар, склад. Для виробника задати назву та адресу фірми. Для товару – назву товару, вартість, об'єм партії товару, що постачається. Для складу задати термін прийняття товару та термін його зберігання. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані у файлі, видалити дані з файлу. Реалізувати запити, визначивши: 1) виробника, який постачає заданий товар в заданий термін.; 2) товар, що може зберігатися найдовше на складі; 3) товар об'єм партії якого найбільший.

Варіант 14.

Створити масив структур. Кожна структура складається з таких елементів: абонент та телефон. Для абонента задати його прізвище та адресу. Для телефону задати його номер, вид оплати (щохвилинна чи абонементна), вартість оплати. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати такі операції: 1) доповнити масив прізвищами абонентів, номери телефонів яких починаються на задані цифри (імітувати введення нової АТС); 2) видати список абонентів з щохвилинною оплатою, сума оплати у яких нижче заданого рівня; 3) визначити номер телефону абонента за заданою адресою.

Варіант 15.

Створити масив структур. Кожна структура складається з таких елементів: напої та шоколадні вироби; їх калорійність, вартість. Для напоїв вказати міцність, для шоколадних виробів - вагу, начинку. Створений масив структур записати до файлу. Передбачити операції

додавання до файлу записів і їх редагування. Реалізувати запити: 1) визначити список шоколадних виробів, що мають максимальну вагу і задану начинку; 2) вилучити з масиву інформацію про алкогольні напої; 3) скласти меню з напоїв і шоколадних виробів, що відповідають заданій калорійності та вартості.

Варіант 16.

Створити масив структур. Кожна структура складається з таких елементів: споживач і продукт. Для споживача задати його прибуток. Для продукту задати ціну, рівень корисності продукту (функцію корисності задати у довільному вигляді). Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити, визначивши: 1) продукти заданого рівня корисності, які може придбати споживач при заданому прибутку; 2) сумарну вартість продуктів з максимальним рівнем корисності; 3) споживачів, прибутків яких не вистачає для придбання продуктів за заданою ціною і рівнем корисності.

Варіант 17.

Створити масив структур. Кожна структура складається з таких елементів: інститут, факультет. Для інституту задати його назву, місто, де він розташований, план прийому на перший курс. Для факультету задати його назву, масив спеціальностей, прохідний бал. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити, визначивши: 1) список міст, де знаходяться політехнічні інститути; 2) інститути і факультети, де приймають студентів на перший курс за заданою користувачем спеціальністю; 3) факультет, інститут і місто, де на вказану користувачем спеціальність існує найвищий прохідний бал.

Варіант 18.

Створити масив структур. Кожна структура складається з таких елементів: факультет, прізвища студентів, прізвища викладачів. Для студентів вказують стипендію і середній бал. Для викладачів - посаду і заробітну плату. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити, визначивши: 1) список студентів по факультетах з найнижчим середнім балом і стипендією; 2) кількість викладачів-доцентів на заданому факультеті; 3) сумарний фонд стипендії і зарплати на заданому факультеті.

Варіант 19.

Створити масив структур. Кожна структура складається з таких елементів: фірма, товар, що продає фірма. Для фірми задати назву, регіон, в якому товар продається. Для товару задати вид (комп'ютери і програмне забезпечення), вартість продажу, термін постачання. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити, визначивши: 1) список комп'ютерів, що продаються у заданому регіоні заданою фірмою; 2) вартість проданого програмного забезпечення у задані терміни; 3) найбільш рентабельні фірми (з найбільшою вартістю продажів).

Варіант 20.

Створити масив структур. Кожна структура складається з таких елементів: прізвище студента, курс, середній бал. Для студентів контрактної форми навчання вказується вартість контракту, а для студентів, що навчаються за бюджетною формою – розмір стипендії. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати такі операції: 1) визначити прізвища студентів із середнім балом вище четвірки, надрукувати окремо студентів контрактної та бюджетної форм навчання; 2) вилучити дані про студентів бюджетної форми з файлу, якщо їх середній бал нижче трійки; 3) вивести список студентів контрактної форми і суму їх контрактів.

Варіант 21.

Створити масив структур. Кожна структура складається з таких елементів: книги, журнали. Для книжок задають назву, прізвище автора, рік видання. Для журналів – назву, номер, рік видання, назву статті, автора. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) вивести авторів, які друкувалися у вказаному році та їх праці; 2) визначити журнал, рік видання вказаної статті; 3) вивести кількість статей заданого автора і журнали, у яких він друкувався.

Варіант 22.

Створити масив структур. Кожна структура складається з таких елементів: прізвище викладача, дисципліна, прізвища студентів, що вивчають конкретну дисципліну. Для викладача задають посаду, заробітну плату, для студента – середній бал і стипендію. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл,

замінити дані в файлі, видалити дані з файлу. Реалізувати такі запити: 1) визначити викладачів і дисципліни з середніми балами менше заданого; 2) вилучити студентів, середній бал яких нижче за трійку; 3) визначити стипендіальний фонд студентів (кількість студентів, що отримують стипендію, помножену на розмір стипендії), які вивчають задану дисципліну.

Варіант 23.

Створити масив структур. Кожна структура складається з таких елементів: книги, читачі. Для книги задати її назву, прізвище автора, рік видання. Для читача – прізвище, адреса, термін повернення книги. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити, визначивши: 1) прізвища читачів, які читають задані книги; 2) книгу заданого автора, що має повернутися в заданий термін; 3) прізвища читачів-боржників (тих, хто в заданий термін книги не повернули).

Варіант 24.

Створити масив структур. Кожна структура складається з таких елементів: пасажир, багаж. Для пасажирів задати його прізвище та вагу багажу. Для багажу задати кількість та вартість речей. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити, визначивши: 1) пасажирів з найбільшою вагою багажу; 2) багаж заданого пасажирів; 3) упорядкувати масив структур за умови зростання вартості багажу.

Варіант 25.

Створити масив структур. Кожна структура складається з таких елементів: фірма, продукт, що виробляє фірма. Для фірми задати її назву, назву міста. Для продукту задати його назву, кількість, ціну одиниці продукту. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити місцезнаходження фірми, що виробляє заданий продукт; 2) визначити сумарну вартість продуктів, що виробляються у даному місті; 3) упорядкувати список продуктів за ціною для даного міста.

Варіант 26.

Створити масив структур. Кожна структура складається з таких елементів: факультет, курс, група, предмет, студент. Для предмета задається назва та кількість кредитів. Для студента задається прізвище та екзаменаційні оцінки. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити

дані з файлу. Реалізувати запити, визначивши: 1) прізвища студентів, які мають дві та більше двійок за сесію на факультеті ІФМЕІТ і вилучити їх; 2) факультет, який на першому курсі має найбільшу кількість відмінників; 3) курс, на якому виключено найбільшу кількість студентів.

Варіант 27.

Створити масив структур. Кожна структура складається з таких елементів: фірма, робітник, посада. Для робітника задається прізвище та заробітна платня. Для посади задається кількість вакантних місць, вимоги до претендентів на кожну посаду, Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити фірми з найбільшою кількістю вакансій на задану посаду; 2) вивести список вимог до кандидатів на задану посаду по кожній фірмі; 3) створити новий масив із структурою: вакантна посада, заробітна платня. Вакансії вибирати з початкового масиву, виключаючи повторення однакових вакантних посад. На випадок різної платні на однакових посадах у різних фірмах включити у масив середню заробітну платню.

Варіант 28.

Створити масив структур. Кожна структура складається з таких елементів: факультет і студент. Для факультету задати його назву, курс і масив груп. Для студента задати прізвище та масив екзаменаційних оцінок. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити факультет і курс з максимальною кількістю відмінників; 2) вивести список відмінників; 3) знайти групу, де немає двісчників.

Варіант 29.

Створити масив структур. Кожна структура складається з таких елементів: інститут та студент. Для інституту задати його назву і масив факультетів. Для студента задати курс, групу, прізвище та середній бал. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити, визначивши: 1) факультет і курс, на якому середній бал не менше 3.5; 2) прізвища студентів із , що навчаються на вказаному курсі та групі; 3) факультет і групу, де найбільше відмінників.

Варіант 30.

Створити масив структур. Кожна структура складається з таких елементів: місто, інститут, факультет. Для інституту задається план прийому на перший курс. Для факультету задається список спеціальностей. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити, визначивши: 1) список спеціальностей, що містять у своїй назві слова "комп'ютерний" або "автоматизований"; 2) факультети й інститути, де є задана спеціальність; 3) факультет, інститут і місто, де на вказану користувачем спеціальність виділено найбільше місць.

Варіант 31.

Створити масив структур, кожна з яких складається з таких елементів: факультет, курс, студент. Для факультету задати його назву та номер групи. Для студента задати прізвище студентів, екзаменаційні оцінки. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити прізвища студентів, групу і факультет, де середній бал студентів 4.5; 2) вилучити прізвища студентів першого курсу, які мають три двійки; 3) відсортувати назви факультетів за умови зростання успішності студентів.

Варіант 32.

Створити масив структур. Кожна структура складається з таких елементів: марка автомобіля, тип (вантажний або легковий), номерний знак, строк служби. Для легкових вказується колір, для вантажних – вантажність. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити номерні знаки автомобілів, вантажність котрих не менше за задану; 2) вивести марки автомобілів заданого типу і вказаного кольору; 3) вилучити з файлу відомості про автомобілі, строк служби яких перевищує п'ять років.

Варіант 33.

Створити масив структур. Кожна структура складається з таких елементів: абонент і телефон. Для абонента задається прізвище, адреса, заборгованість по оплаті. Для телефону задається номер і оператор (МТС, Київстар тощо). Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити прізвища абонентів, що мають заборгованість по

оплаті більше заданої користувачем; 2) вилучити прізвища абонентів, адреса яких змінилася; 3) замінити номер телефону заданого абонента.

Варіант 34.

Створити масив структур. Кожна структура складається з таких елементів: фірма, товар. Для фірми задається назва, кількість товарів. Для товару задається його найменування, вартість, термін поставки. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити фірми, що постачають заданий товар у дводенний строк; 2) визначити назву товару в заданій фірмі, вартість якого не перевищує заданого значення; 3) впорядкувати масив за умови зростання термінів постачання товару.

Варіант 35.

Створити масив структур. Кожна структура складається з таких елементів: інститут, факультет. Для інституту задати назву та план прийому на перший курс. Для факультету задати список спеціальностей та прохідний бал. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) скласти список спеціальностей по факультетах із вказаним прохідним балом; 2) впорядкувати масив за прохідним балом і планом прийому; 3) визначити інститут із вказаним середнім прохідним балом.

Варіант 36.

Створити масив структур. Кожна структура складається з таких елементів: літак і рейс. Для літака задати тип, кількість вільних місць, вартість квитка. Для рейсу задати номер, дата польоту. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) скласти список номерів авіаційних рейсів, на які є вільні місця; 2) вивести типи літаків і відповідні номери авіаційних рейсів на задану дату; 3) упорядкувати масив за умови зростання вартості квитків по кожній даті.

Варіант 37.

Створити масив структур. Кожна структура складається з таких елементів: напрям польоту та літак. Для напрямку польоту задати льотні дні на тиждень і номер авіаційного рейсу, Для літака задати тип, кількість посадкових місць, вартість квитка. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати запити: 1) визначити номери авіаційних рейсів

заданого напрямку з мінімальною вартістю квитків; 2) визначити типи літаків і сумарну кількість посадкових місць у заданому напрямку по днях тижня; 3) вилучити з масиву відомості про рейси, якщо кількість польотів на тиждень менше двох.

Варіант 38.

Створити масив структур. Кожна структура складається з таких елементів : виробник, товар, склад. Для виробника задати назву та адресу фірми. Для товару – назву товару, вартість, об'єм партії товару, що постачається. Для складу задати термін прийняття товару та термін його зберігання. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані у файлі, видалити дані з файлу. Реалізувати запити, визначивши: 1) виробника, який постачає заданий товар в заданий термін.; 2) товар, що може зберігатися найдовше на складі; 3) товар об'єм партії якого найбільший.

Варіант 39.

Створити масив структур. Кожна структура складається з таких елементів: абонент та телефон. Для абонента задати його прізвище та адресу. Для телефону задати його номер, вид оплати (щохвилинна чи абонементна), вартість оплати. Створений масив записати до бінарного файлу. Передбачити можливість доповнити файл, замінити дані в файлі, видалити дані з файлу. Реалізувати такі операції: 1) доповнити масив прізвищами абонентів, номери телефонів яких починаються на задані цифри (імітувати введення нової АТС); 2) видати список абонентів з щохвилинною оплатою, сума оплати у яких нижче заданого рівня; 3) визначити номер телефону абонента за заданою адресою.

Варіант 40.

Створити масив структур. Кожна структура складається з таких елементів: напої та шоколадні вироби; їх калорійність, вартість. Для напоїв вказати міцність, для шоколадних виробів - вагу, начинку. Створений масив структур записати до файлу. Передбачити операції додавання до файлу записів і їх редагування. Реалізувати запити: 1) визначити список шоколадних виробів, що мають максимальну вагу і задану начинку; 2) вилучити з масиву інформацію про алкогольні напої; 3) скласти меню з напоїв і шоколадних виробів, що відповідають заданій калорійності та вартості.

Питання та завдання для контролю знань

Текстові файли

1. Знайдіть помилки в наступному фрагменті програми, якщо `InFile` – файлова змінна текстового файлу, а `Next` – змінна типу `char`. Який файл тут є вхідним, а який вихідним і який з файлів слід тестувати за допомогою функції `eof()`?

```
fstream open("MYDATA.TXT", InFile);
InFile.open("MYDATA.TXT", ios::out | ios::trunc);
Input.open("1.TXT", ios::in);
while ( ! eof() )
{
    InFile >> Next;
    cout << Next;
    Input << Next;
}
```

2. Виправте помилку в наступному фрагменті програми
`Number.open("1.TXT", ios::in);`
`while (! eof(InFile))`
`InFile >> Number)`
3. Напишіть цикл, який зчитує не більше 10 цілих чисел з текстового файлу і виводить їх на екран. Якщо у файлі міститься більше 10 цілих чисел, то після зчитування останнього десятого числа на екран має бути виведено повідомлення "Зчитування завершено".
4. Припустимо, є наступний фрагмент програми.

```
int x = 12;
int y = 34;
OutFile << x;
OutFile << y;
```

Чи можна буде потім зчитати ці два числа, якщо `OutFile` – це текстовий файл?

Відповідь обґрунтуйте.

6. Дані яких типів можуть бути зчитані з текстового файлу або записані у текстовий файл?
7. Чи допустимо передавати файл у функцію через параметр-посилання?
8. Чи допустимо передавати файл у функцію через параметр-значення?
9. Чи вірне наступне твердження:
Ім'я файлової змінної, пов'язаної з текстовим файлом має збігатися з іменем відповідного файлу.
10. Доповнити твердження.

1. Текстовий файл складається із _____ змінної довжини.
2. Для того щоб додати рядки до текстового файлу, його спочатку необхідно

відкрити функцією _____

11. Визначити істинні твердження

1. Усі процедури запису даних до текстових файлів можна застосувати і до бінарних файлів.
2. Бінарні і текстові файли можуть створюватися з використанням будь-якого текстового редактора.

Бінарні файли

1. Означити поняття логічного та фізичного файлу.
2. Чим файли, масиви та записи схожі? Чим вони відрізняються?
3. Чим зумовлена потреба у відкритті та закритті файлів?
4. Як здійснюється навігація по файлу?
5. Як зв'язати логічний файл із фізичним?
6. Які існують характеристики бінарного файлу?
7. Якого типу дані можуть зчитуватися із бінарного файлу, або записуватися в нього?
8. Припустимо, є наступний фрагмент програми.

```
int x = 12;  
int y = 34;  
OutFile.write((char*) &x, sizeof(x));  
OutFile.write((char*) &y, sizeof(x));
```

Чому ці два числа потім можна буде зчитати, якщо OutFile – це бінарний файл?

9. Доповнити твердження.

1. Бінарний файл – це сукупність_____.
2. Кількість компонентів бінарного файлу не є_____.
3. Функції_____відкривають бінарний файл для запису і зчитування.

10. Визначити істинні твердження

1. Компоненти бінарного файлу мають один і той самий тип.
2. Розмір файлу визначається під час оголошення файлової змінної.
3. Після закриття файлу зв'язок файлової змінної із фізичним файлом не переривається.
4. Один фізичний файл можна зв'язати лише з однією файловою змінною.

11. Які значення до файлу **proba.dat** запише наведена далі програма?

```
#include <iostream>  
#include <fstream>  
  
using namespace std;
```

```

int main()
{
    ofstream f("proba.dat", ios::binary);

    int i=1;

    if ( f.eof() )
        f.write((char*) &i, sizeof(i));
    else
    {
        i = i+1;
        f.write((char*) &i, sizeof(i));
    }

    if ( f.eof() )
    {
        i = i+1;
        f.write((char*) &i, sizeof(i));
    }
    else
    {
        i = i+2;
        f.write((char*) &i, sizeof(i));
    }

    f.close();

    return 0;
}

```

Варіанти відповідей:

- 1 2
- 1 3
- 2 3
- 2 4

під час виконання програми виникне помилка.

12. Які дії виконує а) функція seekp(), б) функція seekg()?

1. Обчислює кількість компонентів файлу.
2. Переміщує файловий вказівник зчитування на компоненту із заданим номером.
3. Повертає номер компоненти, на яку вказує файловий вказівник запису.
4. Переміщує файловий вказівник запису на компоненту із заданим номером.
5. Повертає номер компоненти, на яку вказує файловий вказівник зчитування.
6. Переміщує файловий вказівник зчитування на байт із заданим номером.
7. Повертає номер байту, на який вказує файловий вказівник запису.
8. Переміщує файловий вказівник запису на байт із заданим номером.
9. Повертає номер байту, на який вказує файловий вказівник зчитування.
10. Видаляє частину файлу.

13. Якщо файловий вказівник посилається на першу / останню компоненту файлу, то виклик

а) функції read(), б) функції write() приведе до таких результатів:

1. Буде зчитана перша компонента.

2. Буде зчитана остання компонента.
 3. Буде записана перша компонента.
 4. Буде записана остання компонента.
 5. Буде перезаписана перша компонента.
 6. Буде перезаписана остання компонента.
 7. Буде виведене повідомлення про помилку.
15. Як зчитувати та записувати дані в бінарні файли?
16. Як відкрити бінарні файли?
17. Якщо файловий вказівник посилається на останню компоненту файлу, то виклик функції `read()` приведе до таких результатів:
1. Буде зчитана перша компонента.
 2. Буде зчитана остання компонента.
 3. Буде записана перша компонента.
 4. Буде записана остання компонента.
 5. Буде виведене повідомлення про помилку.
18. Якщо файловий вказівник посилається на останню компоненту файлу, то виклик функції `write()` приведе до таких результатів:
1. Буде записана нова компонента у кінець файлу.
 2. Буде зчитана остання компонента.
 3. Буде перезаписана перша компонента.
 4. Буде перезаписана остання компонента.
 5. Буде виведене повідомлення про помилку.
19. Якщо файловий вказівник посилається на кінець файлу, то виклик функції `write()` приведе до таких результатів:
1. Буде записана нова компонента у кінець файлу.
 2. Буде зчитана остання компонента.
 3. Буде перезаписана перша компонента.
 4. Буде перезаписана остання компонента.
 5. Буде виведене повідомлення про помилку.

Файли

1	Поставте у відповідність елементи лівого та правого списків (пронумеруйте відповідні елементи правого списку).			
	1	Масиви		Структурні типи
	2	Структури		Прості типи

	3	Файли		Значення цих типів – набори елементів простішої структури				
				Значення цих типів – атомарні (неділимі)				
				Складаються із фіксованої кількості компонентів				
				Складаються із наперед не визначеної кількості компонентів				
				Всі компоненти мають бути одного і того самого типу				
				Компоненти можуть бути різних типів				
				Оголошуються за допомогою []				
				Оголошуються за допомогою ключового слова struct				
				Оголошуються за допомогою ключових слів FILE* , fstream				
				Доступ до компонентів: <ім'я> [<індекс>]				
				Доступ до компонентів: <ім'я>.<поле>				
			2	Поставте у відповідність елементи лівого та правого списків (пронумеруйте відповідні елементи правого списку).				
1	Бінарні файли			Складаються із фіксованої кількості компонентів				
2	Текстові файли			Всі компоненти мають бути одного і того самого типу				
				Складаються із наперед не визначеної кількості компонентів				
				Компоненти можуть бути різних типів				
				Компонентами є величини (записи) одного і того самого типу				
				Компонентами є літерні рядки				
				Компонентами є символи				
				Компонентами є символи, об'єднані у логічні рядки				
				Компонентами є послідовності байтів вказаної довжини				
3	Нехай дано оголошення: double X; int N; char C; string S; Які будуть значення цих змінних після виконання кожної з команд fin >> , якщо припустити, що текстовий файл fin містить наступні рядки (<eoln> – мітка «кінець рядка»), у кожному із завдань зчитування відбувається з початку файлу: 123 3.145 XYZ<eoln> 35 Z<eoln>							
		X	N	C	S			
	fin >> N >> X; fin >> C;							
	fin >> N >> X >> C;							
	fin >> N >> X >> C >> C;							
	fin >> N; fin >> C;							
	fin >> S >> X; fin >> C;							
	fin >> S >> X >> C;							
	fin >> S >> X >> N >> C;							
	fin >> S; fin >> C;							
	fin >> C >> X; fin >> C;							
	fin >> C >> S >> C;							
	fin >> C >> C >> C >> C;							
	fin >> C; fin >> S;							

Зауваження: в задачах 4 – 12 вміст файлу лише аналізується, в інших задачах створюється новий файл або змінюється вміст даного.

4. Написати програму для визначення скільки разів у файлі `f.txt` зустрічається символ `'A'`.
5. Написати програму для визначення скільки разів у файлі `g.txt` зустрічається рядок `"UNIX"`.
6. Написати програму, яка виводить на екран всі рядки заданого файлу, що містять заданий літерний рядок в якості під-рядка. Ім'я файлу та літерний рядок вводяться користувачем.
7. Написати програму для визначення символу, який найчастіше зустрічається в заданому файлі. Ім'я файлу вводиться користувачем.
8. Написати програму для визначення скільки рядків, що складаються із одного, двох, трьох і т.д. символів, міститься в заданому файлі. Вважати, що довжина кожного рядка – не більша 80 символів. Ім'я файлу вводиться користувач.
9. Написати програму для визначення, який рядок – найдовший в заданому файлі. Якщо таких рядків – кілька, то в якості результату вивести перший з них. Ім'я файлу вводиться користувач.
10. Дано два не порожніх файли. Написати програму для визначення номеру рядка і номеру символу в цьому рядку, що відповідають першому символу, яким вміст одного файлу відрізняється від іншого. Якщо вміст файлів повністю однаковий, то результатом має бути 0, 0 та відповідне повідомлення; якщо один з файлів – збігається з початком іншого, то результатом буде $n+1$, 1 та відповідне повідомлення, де n – кількість рядків в короткому файлі. Імена файлів вводиться користувач.
11. Написати програму, яка опрацьовує текстовий файл, що містить цілі числа (ціле число – це не порожня послідовність десяткових цифр, яка може починатися знаком `+` або `-`). Ім'я файлу вводиться користувач.
 - а) знайти найбільше з цих чисел;
 - б) визначити, скільки парних чисел міститься у файлі;
 - в) визначити, чи утворюють ці числа арифметичну прогресію;
 - г) визначити, чи утворюють ці числа зростаючу послідовність;
 - д) визначити, скільки чисел цієї послідовності є точними квадратами.
12. Написати програму, яка визначає рядок, що найчастіше зустрічається в заданому файлі. Ім'я файлу вводиться користувач.
13. Написати програму, яка копіює заданий файл. Імена файлів (оригіналу та новоствореної копії) вводиться користувач.

14. Написати програму, яка створює файл, що є конкатенацією інших файлів. Користувач вводить N – кількість вхідних файлів; f_1, \dots, f_N – імена вхідних файлів; f_{res} – ім'я результуючого файлу.
15. Дано файл f . Написати програму, яка створює файл g , замінюючи всі великі латинські літери відповідними малими.
16. Дано файл f . Написати програму, яка створює два файли f_1 та f_2 : у файл f_1 записує всі рядки із файлу f , які містять лише латинські літери (малі та великі); у файл f_2 – рядки файлу f , які містять лише цифри; всі інші рядки файлу f не записуються до жодного з файлів f_1 та f_2 .
17. Написати програму, яка в кінець файлу f дописує рядок "FINISH".
18. Написати програму, яка в кінець файлу f дописує вміст файлу g .
19. Дано два файли, рядки яких впорядковані за алфавітом. Написати програму, яка виконує злиття цих двох файлів у третій, рядки якого теж будуть впорядковані за алфавітом. Імена файлів вводить користувач.
20. Дано файл та два рядки. Написати програму, яка всі входження першого рядка у файл замінює другим рядком (входження першого рядка в якості підрядка не розглядати). Ім'я файлу та рядки вводить користувач.
21. Дано файл та два рядки. Написати програму, яка всі входження першого рядка у файл (в тому числі і в якості підрядка) замінює другим рядком. Ім'я файлу та рядки вводить користувач.
22. Написати програму, яка в заданому файлі впорядковує символи кожного рядка за алфавітом. Ім'я файлу вводить користувач.
23. Написати програму, яка впорядковує за алфавітом рядки заданого файлу. Ім'я файлу вводить користувач.
24. Написати програму, яка впорядковує рядки заданого файлу за зростанням їх довжин. Ім'я файлу та максимальну довжину рядка вводить користувач.
25. Дано файл, який містить не порожню послідовність цілих чисел, що є числами Фібоначчі ($n_0 = 1$; $n_1 = 1$; $n_{k+1} = n_{k-1} + n_k$; $k = 1, 2, 3, \dots$). Дописати у файл ще одне, чергове число Фібоначчі.
26. Написати програму, яка опрацьовує файл, що містить непорожню послідовність цілих чисел (ціле число – це не порожня послідовність десяткових цифр, яка може починатися знаком $+$ або $-$). Створити новий файл, в якому (імена файлів вводить користувач):
 - а) всі від'ємні числа замінені нулями;
 - б) мінімальний елемент послідовності переміщений в її початок, а максимальний – в кінець;

- в) переставлені місцями максимальний та мінімальний елементи цієї послідовності;
- г) вилучені всі елементи, які є повними квадратами.

Файлові потоки

1. Що таке «файлові потоки введення-виведення»?
2. Які ще бувають потоки, крім файлових?
3. Який файл заголовку необхідно підключити для використання файлових потоків?
4. Які типи файлових потоків існують? (вказіть ідентифікатори потокових типів)
5. Для яких цілей призначено кожний тип файлового потоку?
6. Який режим відкривання за умовчанням має кожний тип файлового потоку?
7. В який момент (в якому місці програми) найкраще відкривати та закривати кожний файловий потік, що використовується програмою?
8. Вкажіть способи відкривання файлових потоків.
9. Яка різниця між різними способами відкривання файлових потоків?
10. В яких випадках який спосіб відкривання файлового потоку краще використовувати?
11. Скільки аргументів може бути вказано при відкриванні файлового потоку?
12. За що відповідає кожний з аргументів, що вказуються при відкриванні файлового потоку?
13. Які значення може мати кожний з аргументів, що вказуються при відкриванні файлового потоку?
14. Чи можна файловий потік для запису даних у файл відкривати в режимі, що визначається комбінацією (в квадратних дужках – необов'язкові параметри) `ios::in | ios::out [| ...]`? Якщо можна, то для яких типів файлових потоків можна вказувати цей режим? Яка особливість такого режиму відкривання?
15. Які ще бувають режими відкривання файлів? В яких комбінаціях і для яких типів потоків вони можуть застосовуватися? Який дію вони будуть виконувати?
16. Як засобами потоку перевірити існування файлу із вказаним іменем, наприклад, для прийняття рішення про необхідність виведення користувачеві запиту на перезапис цього файлу, якщо планується здійснювати запис даних до нього? Потоки яких типів для цього можна використовувати?
17. Чи завжди операція відкриття файлового потоку буде виконана успішно?
18. Які можуть бути причини невдачі при відкритті файлового потоку?
19. Як перевірити результат виконання операцій відкриття файлового потоку?
20. Які засоби є для перевірки результату виконання операції відкриття файлового потоку? Які варіанти запису такої перевірки допустимі?

Стан потоку

1. Що таке «стан потоку»?
2. Із яких складових частин складається стан потоку?
3. В результаті чого змінюється стан потоку? Коли і які значення мають складові частини стану потоку?
4. На що впливає стан потоку?
5. Як отримати значення стану потоку та окремих його складових частин?
6. Як змінити значення всіх чи окремих складових частин стану потоку? Якими способами це можна зробити?
7. Як за одну дію (в одній команді) скинути значення одного чи кількох певних бітів стану потоку, залишивши без змін інші біти?
8. Як за одну дію (в одній команді) скинути значення всіх, крім одного чи кількох певних бітів, стану потоку, залишивши цей один чи кілька бітів без змін?
9. Як за одну дію (в одній команді) встановити один чи кілька бітів стану потоку, залишивши без змін інші біти?
10. До чого приведе виклик `f.clear(ios::failbit);` ?
11. До чого приведе виклик `f.setstate(ios::goodbit);` ?
12. Вкажіть засоби файлових потоків для зчитування даних різних типів із текстових файлів; для запису даних різних типів у тестові файли.
13. Для чого призначена операція вилучення із потоку? Для роботи з файлами якого формату вона використовується?
14. Як відбувається процес зчитування даних різних типів за допомогою операції вилучення з потоку? Що при цьому відбувається з вказівником потоку? Пояснити на прикладах.
15. Як можна перевірити (в найпростішому випадку) результат виконання операції зчитування з потоку (запису у потік)? Якими різними способами це можна зробити і які варіанти запису цих способів – допустимі?
16. В чому можуть бути причини неуспішності виконання операцій зчитування даних з текстового файлу за допомогою потоку?

Буфер потоку

1. Що таке «буфер потоку»?
2. Навіщо він взагалі потрібний?
3. Які задачі можна вирішувати за його допомогою?

4. Як отримати до нього доступ?
5. Як очистити вміст буферу потоку (видалити з нього всі символи)?
6. Як пропустити (без видалення) всі символи, що знаходяться в буфері потоку?

Вказівник файлового потоку (файловий вказівник)

1. Що таке «вказівник потоку»?
2. Які види вказівників у яких типів потоків бувають?
3. Які операції з потоком змінюють позицію файлового вказівника та як саме?
4. В яких одиницях вказується позиція файлового вказівника?
5. Де знаходиться файловий вказівник після відкриття потоку для більшості режимів відкривання? Які режими є винятками?
6. Якими способами можна перемістити вказівник на початок файлу? В довільне місце? Які для цього є засоби потоку, варіанти їх реалізації та застосування: які команди та функції, в якій послідовності, з якими аргументами та чому?
7. Для чого призначена функція потоку `fail()` ?
8. Як пов'язані функції `fail()` та `eof()` з станом потоку?
9. Чим відрізняються функції `peek()` та `get()` ?
10. Для чого призначена функція `unget()` ? Як, іншим способом, можна виконати ту саму дію, що виконує ця функція?

Опрацювання двійкових файлів за допомогою потоків

1. Чим відрізняються текстові файли від двійкових? Що є вмістом кожного з цих файлів?
2. Що можна сказати про компактність зберігання даних різних типів в текстових та бінарних файлах?
3. В яких випадках який формат файлу (текстовий чи бінарний) є кращим?
4. При збереженні за допомогою файлового потоку деяких даних у файл, чим саме визначається, в якому форматі ці дані будуть збережені: у текстовому чи в бінарному?
5. За допомогою яких засобів файлового потоку здійснюється зчитування та запис двійкових даних? Як слід використовувати ці засоби? Навести приклади.
6. Як реалізується по-елементне зчитування та зчитування усього масиву одразу за допомогою файлового потоку?
7. Як реалізується по-елементний запис та запис усього масиву одразу за допомогою файлового потоку?
8. За що відповідає режим `ios::binary`? В яких випадках його можна / потрібно вказувати?

9. Як можна визначити кількість однотипних компонентів в двійковому файлі?
10. За допомогою яких засобів потоку та як саме ці способи реалізуються?
11. Чим ці способи відрізняються з точки зору надійності та швидкодії?
12. Як надійність цих способів залежить від використання режиму `ios::binary`?

Предметний покажчик

B		R	
bad(), 91		rdstate(), 91	
C		read(), 34, 76	
clear(), 91		remove(), 99	
close(), 31, 66		rename(), 99	
E		rewind(), 52	
eof(), 33, 68, 91		S	
F		seekg(), 36, 80	
fail(), 90, 91		seekp(), 36, 80	
fclose(), 25, 47		setstate(), 92	
feof(), 26, 50		T	
ferror(), 49, 60		tellg(), 36, 83	
fflush(), 60		tellp(), 36, 83	
fgetc(), 26, 48		U	
fgets(), 27, 51		unget(), 89	
flush(), 89		W	
fopen(), 23, 45		write(), 35, 77	
fprintf(), 61		Б	
fputc(), 25, 48		Бінарні файли, 23, 41	
fputs(), 27, 51		опрацювання мовою C, 27, 53	
fread(), 27, 53		опрацювання мовою C++, 34, 76	
freopen(), 62		Буфер обміну, 38	
fscanf(), 61		В	
fseek(), 28, 57		Введення з файлу, 38	
fstream, 29, 64		Виведення у файл, 38	
ftell(), 60		Відкриття файлу, 38	
fwrite(), 27, 53		Відкриття файлу мовою C, 23, 45	
G		режими відкриття, 23, 45	
gcount(), 87		функція fopen(), 23, 45	
get(), 75, 85		Відкриття файлу мовою C++, 30	
getc(), 26, 48		режими відкриття, 31	
getline(), 73, 74, 87		Вказівник введення, 40	
good(), 91		Вказівник виведення, 22, 40	
I		Вказівник поточної позиції файлу, 22, 40	
ifstream, 29, 64		Д	
ignore(), 88		Двійкові файли, 23	
ios		З	
binary, 76		Загальна схема опрацювання файлу, 22, 39	
iostate, 91		Закриття файлу, 39	
O		Закриття файлу мовою C, 25, 47	
ofstream, 29, 64		функція fclose(), 25, 47	
open(), 30, 66		Закриття файлу мовою C++, 31	
P		Запис у файл, 38	
peek(), 88		Зчитування з файлу, 38	
put(), 87			
putback(), 88			
putc(), 25, 48			

Л

Логічний файл, 38

М

Мітка <кінець рядка>, 23
Мітка <кінець файлу>, 23

П

Потоки та файли
мова С, 42

Т

Текстові файли, 23, 40
опрацювання мовою С, 25, 48

опрацювання мовою С++, 33, 72

Ф

Файл
загальна схема роботи з файлом, 39
опрацювання файлу, 40
поняття, 38
Файл операційної системи, 38
Файлова змінна, 38
мова С, 45
Файлова система
мова С, 44
Файловий буфер, 38
Файловий вказівник, 22, 40
Файловий вказівник запису, 22, 40
Файловий вказівник зчитування, 22, 40
Фізичний файл, 38

Література

Основна

1. Ковалюк Т.В. Основи програмування. – К. ВНУ, 2005. – 384 с.
2. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир, 1985.
3. Павловская Т.А. С/С++. Программирование на языке высокого уровня. СПб.: Питер, 2007. – 461 с.
4. Павловская Т.А., Щупак Ю.А. С/С++. Структурное программирование: Практикум. СПб.: Питер, 2005. – 239 с.
5. Прата Стивен. Язык программирования С++. Лекции и упражнения. Учебник: Пер. С англ./Стивен Прата – СПб.: ООО «ДиаСофтЮП», 2003. – 1194 с.
6. Шилдт Г. Полный справочник по С. Электронный ресурс
http://cpp.com.ru/shildt_spr_po_c/index.html
7. Справочник по языку С. Электронный ресурс.
<https://msdn.microsoft.com/uk-ua/library/fw5abdx6.aspx>

Додаткова

8. Кнут, Дональд, Эрвин Искусство программирования. 3-е издание. Том1. М.: Вильямс, 2001.
9. Кнут, Дональд, Эрвин Искусство программирования. 3-е издание. Том2. М.: Вильямс, 2001.
10. Кнут, Дональд, Эрвин Искусство программирования. 3-е издание. Том3. М.: Вильямс, 2001.
11. Брудно А.Л., Каплан Л.И. Московские олимпиады по программированию. – М.: Наука, 1990.
12. Вирт Н. Систематическое программирование. Введение. – М., Мир, 1977.
13. Ахо А.В., Хопкрофт Дж., Ульман Дж. Д. Структуры данных и алгоритмы. Пер. с англ.: М.: «Вильямс», 2001.– 384 с.: ил.
14. Себест Р. Основные концепции языков программирования. 5-е издание. М.: Вильямс, 2001.