ЛАБОРАТОРНА РОБОТА №8

Тема : ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ РОБОТИ З ФАЙЛАМИ

Мета роботи : Отримати знання і навички, необхідні для освоєння технології роботи з даними на основі використання файлів за допомогою засобів мови С++**.**

Для цього потрібно:

- Вивчити програмні засоби для роботи з файлами та потоками.
- Дослідити основні функції роботи з файлами та реалізувати найпростіші операції з ними.
- Навчитись застосовувати у своїх програмах вхідні і вихідні текстові і бінарні файли і файлові потоки.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Поняття файлів і потоків

В комп'ютерній системі між програмою і пристроєм знаходиться щось більш загальне, ніж сам пристрій. Такий узагальнений пристрій введення або виведення (пристрій більш високого рівня абстракції) називається ***потоком,*** в той час як конкретний пристрій називається ***файлом*** (файл - теж поняття абстрактне). Потоки бувають двох видів: текстові і двійкові.

Текстовий потік - це послідовність символів. У стандарті С/С++ вважається, що текстовий потік організований у вигляді рядків, кожен з яких закінчується символом нового рядка. Проте, в кінці останнього рядка цей символ не є обов'язковим. В текстовому потоці на вимогу базового середовища можуть відбуватися певні перетворення символів. Тому може і не бути однозначної відповідності між символами, які пишуться (читаються), і тими, які зберігаються в зовнішньому пристрої. Крім того, кількість тих символів, які пишуться (читаються), і тих, які зберігаються в зовнішньому пристрої, може також не співпадати із-за можливих перетворень.

Двійковий потік - це послідовність байтів, яка взаємно однозначно відповідає байтам на зовнішньому пристрої, причому ніякого перетворення символів не відбувається. Крім того, кількість тих байтів, які пишуться (читаються), і тих, які зберігаються на зовнішньому пристрої, однакова. Однак наприкінці двійкового потоку може додаватися визначена додатком кількість нульових байтів (наприклад, для заповнення сектора на диску).

1.2. Файли і функції файлової системи у мові С/С++

У мові С/С++ файлом може бути все що завгодно, починаючи з дискового файлу і закінчуючи терміналом або принтером. Потік пов'язують з певним файлом, виконуючи операцію відкриття.

Як тільки файл відкритий, можна проводити обмін інформацією між ним і програмою. При відкритті файлу покажчик поточної позиції у файлі встановлюється в початок. При читанні з файлу (або записі в нього) кожного символу покажчик поточної позиції збільшується, забезпечуючи просування по файлу.

Файл від'єднується від певного потоку (тобто розривається зв'язок між файлом і потоком) за допомогою операції закриття.

Бібліотека C/C++ підтримує два основні способи роботи з файлами:

- форматоване введення/виведення за допомогою функцій (заголовний файл *stdio.h*);

- потокове введення/виведення (заголовний файл *fstream*).

За першого способу роботи з файлами (за допомогою бібліотечних функцій *stdio.h*) застосовується спеціальна керуюча структура, що містить інформацію про файл та надає тимчасовий буфер для зберігання даних. Дана структура має тип ***FILE.*** Крім тимчасового буферу у керуючій структурі міститься інформація про ідентифікатор файлу, його розташування на диску та покажчик поточної позиції у файлі. У цьому блоці управління файлом ніколи нічого міняти не можна.

Типовий сценарій роботи з файлами:

- відкриття файлу: вказується ім'я файлу, визначається режим доступу (читання, запис, додавання) та тип файлу (текстовий або двійковий);

- читання або запис даних: після того, як відкриття файлу успішно виконано, з нього можна прочитати або записати в нього дані у визначеному форматі (форматоване введення/виведення);

- закриття файлу: для завершення роботи з файлом його необхідно закрити.

Для реалізації даного сценарію в бібліотеці *stdio.h* призначені наступні функції:

- відкриття файлу - функція *fopen();*

- форматоване виведення даних - сімейство функцій *printf() (fprintf());*

- форматоване введення даних - сімейство функцій *scanf() (fscanf());*

- закриття файлу - *fclose().*

Часто використовувані функції файлової системи C/C++ представлені у табл.1.

Щоб оголосити змінну-покажчик файлу, пишуть:

***FILE** *fp;*

Функція *fopen()* відкриває потік і пов'язує з цим потоком певний файл. Потім вона повертає покажчик цього файлу. Вона має наступний синтаксис:

FILE *fopen* ***(const char** * ім 'яфайлу, **const char** * режим);*

Рядок "режим", визначає, яким чином файл буде відкритий.

Таблиця 1 - Функції для роботи з файловою системою

Назва	Що робить
fopen()	Відкриває файл
fclose()	Закриває файл
putc()	Записує символ у файл
fputc()	Те саме, що і putc()
getc()	Читає символ з файлу
fgetc()	Те саме, що і getc()
fgets()	Читає рядок з файлу
fputs()	Записує рядок у файл
fseek()	Встановлює покажчик поточної позиції на певний байт файлу
ftell()	Повертає поточне значення покажчина у файлі
fprintf()	Для файлу те саме, що printf() для консолі
fscanf()	Для файлу те саме, що scanf() для консолі
feof()	Повертає значення true (істина), якщо досягнуто кінець файлу
error()	Повертає значення true, якщо виникла помилка
rewind()	Встановлює покажчик поточної позиції на початок файлу
remove()	Знищує файл
fflush()	Дозапис потоку у файл

У стандарті мови C/C++ серед інших визначено наступні основні режими доступу до файлів:

- **r** - доступ тільки для читання (застосовний тільки для існуючого файлу);

- **w** - доступ для запису: якщо файл існує, його вміст очищується, а якщо файл не існує, то в такому випадку він створюється;

- **a** - доступ для додавання нової інформації: якщо файл вже існує, дані додаються в кінець, а якщо файл не існує, то в такому випадку він створюється.

При визначенні режиму доступу для двійкового файлу використовується тип файлу **b**, а для текстового файлу - **t**.

Рядки, подібні "r+b" можуть бути представлені і у вигляді "rb+" (табл. 2).

Функція *виведення даних у файл fprintf()* має наступний синтаксис:

***int** fprintf **(FILE** * потік, **const char** * формат, ...);*

В якості параметра *формат* може використовуватися довільний рядок, що може містити специфікатори формату наступного вигляду:

%[прапори][ширина][.точність][довжина] специфікатор

Поле *специфікатор* визначає тип та формат даних, що буде записано у файл.

Функція *fscanf()* має наступний синтаксис:

***int** fscanf **(FILE** * stream, **const char** * format, ...);*

Заголовок <*stdio.h*> надає прототипи функцій введення/виведення і визначає наступні три *типи*: *size_t*, *fpos_t* і *FILE*. Типи *size_t* і *fpos_t* представляють собою певні різновиди такого типу, як ціле без знака. А про третій тип, *FILE*, мова йде далі.

Таблия 2 - Режими відкриття файлів

Режим	Що означає
r	Відкрити текстовий файл для читення
w	Створити текстовий файл для запису
a	Додати в кінець текстового файлу. Якщо файл не існує, то він буде створений. Всі нові дані, які записуються в нього, будуть додаватися в кінець файлу.
rb	Відкрити двійковий файл для читання
wb	Створити двійковий файл для запису
ab	Додати в кінець двійкового файла
r+	Відкрити текстовий файл для читання/запису. Вміст залишиться недоторканим. Якщо файлу не існує, то він створений не буде.
w+	Створити текстовий файл для читання /запису. Якщо файл не існує, то він буде створений. Якщо файл вже існує, то відкриття призведе до втрати його вмісту, а в режимі r+ він залишиться недоторканим
a+	Додати в кінець текстового файлу або створити його для читання/запису
r+b	Відкрити двійковий файл для читання /запису
w+b	Створити двійковий файл для читання /запису
a+b	Додати в кінець двійкового файла або створити його для читання /запису

Крім того, в <*stdio.h*> визначається декілька макросів. Серед них:

NULL, EOF, FOPEN_MAX, SEEK_SET, SEEK_CUR і *SEEK_END.*

Макрос *NULL* визначає порожній *(null)* покажчик. Макрос *EOF* часто визначається як -1, і є значенням, що повертається тоді, коли функція введення намагається виконати читання після кінця файлу. *FOPEN_MAX* -ціле значення, рівне максимальному числу одночасно відкритих файлів.

Інші макроси використовуються разом з *fseek(),* що виконує операції прямого доступу до файлу.

Для читання з файлу або запису в файл структури або масиву використовують наступні функції:

***size_t** fread **(void** *ptr, **size_t** size, **size_t** count, **FILE** * stream);*

***size_t** fwrite **(const void** * ptr, **size_t** size, **size_t** count, **FILE** * stream);*

Дані функції мають наступні параметри:

- **ptr** - покажчик на масив, що буде прочитано з файлу (або записано в файл);
- **size** - розмір елементу масиву в байтах;
- **count** - кількість елементів у масиві;
- **stream** - покажчик на структуру FILE.

1.3. Робота з файловими потоками

У мові С++ введення/виведення описується як набір класів, описаний в заголовному файлі *iostream.h.* Аналогами потоків *stdin, stdout, stderr* є класи *cin, cout*

і *cerr*. Ці три потоки відкриваються автоматично. Потік *cin* пов'язаний з клавіатурою, а *cout*, *cerr* - з дисплеєм.

Файл <*fstream.h*> визначає класи *ifstream* і *ofstream*, за допомогою яких програма може виконувати операції файлового введення/виведення.

У такому випадку для відкриття файлу можна використати наступні способи на основі створення потоку та застосування відповідних методів, представлений у загальному вигляді:

*ifstream (**const char** * filename, mode);*

*ofstream (**const char** * filename, mode);*

***void** open (**const char** * filename, mode);*

***void** open (**const char** * filename, mode);*

При цьому використовуються наступні параметри:

- *filename* - ім'я файлу;
- *mode* - режим доступу до файлу:

а) *app* - доступ для додавання нової інформації;
б) *ate* - перемістити вказівник файлу в кінець;
в) *binary* - режим доступу до бінарного файлу;
г) *in* - доступ для читання;
д) *out* - доступ для запису;
е) *trunc* - створити порожній файл для читання та запису.

Для відкриття файлу на введення/виведення оголошують об'єкт типу *ifstream/ ofstream*, передаючи конструктору цього об'єкта ім'я необхідного файлу:

***ofstream** myOutput ("FileOut.EXT");*

***ifstream** mylnput ("Fileln.EXT");*

Після того, як програма відкрила файл для введення або виведення, вона може читати або писати дані, використовуючи оператори: "<<" - для занесення (запису) в потік; ">>" - для вилучення (читання) з потоку.

```
char word [64];
while  (!myInput.eof ())
{      myInput >> word;                    // зчитуємо слово (до пробілу)
cout <<word <<endl;                 // виводимо на екран
myOutput << word;  // записуємо у файл
}
```

Більшість програм читають вміст файлу, поки не зустрінеться *кінець файлу*. Визначити кінець файлу можна за допомогою функції ***eof().***

Для введення або виведення *символів* у файл або з файлу використовую функції ***get()*** і ***put().***

```
char letter;
while  (!myInput.eof ())
    {letter = myInput.get();                    // зчитуємо з файлу
    cout <<letter;                              // виводимо на екран
    myOutput.put(letter);                       // записуємо у файл
    }
```

Для зчитування *цілого рядка* використовують функцію ***getline():***

```
char line [80];
while  (!myInput.eof ())
{      myInput.getline (line,sizeof (line));
cout <<line <<endl;
}
```

Для перевірки помилок можна використовувати функцію ***fail():***

```
if (myInput.fail ())
{                cerr «"Помилка відкриття FileIn.txt" <<endl;
exit (1);
}
```

Якщо програмам необхідно вводити або виводити такі дані, як структури або масиви, можна використовувати методи ***read()*** і ***write().***

```
myInput.read  (buffer,  sizeof (buffer));
myOutput.write (buffer, sizeof (buffer));
```

Якщо програма завершила роботу з файлом, його слід закрити за допомогою функції ***close().***

Для того, щоб операції введення/виведення виконувалися не з початку файлу, можна використовувати інші режими відкриття файлів (табл. 3).

Таблиця 3 - Режими відкриття файлових потоків

Режим відкриття	Призначення
ios::app	Відкриває файл в режимі додавання, встановлюючи файловий покажчик на кінець файлу
ios::ate	Встановлює файловий покажчик на кінець файлу
ios::in	Вказує відкрити файл для введення .
ios::nocreate	Якщо й файл не існує, не створювати файл і повернути помилку
ios::noreplace	Якщо файл існує, операція відкриття повинна бути перервана и повинна повернути помилку
ios::out	Вказує відкрити файл для виведення .
ios::trunc	Перезаписує вміст існуючого файлу
ios::binary	Робота з файлом у двійковому вигляді

Наприклад,

ifstream myFile ("Filename.txt", ios::out | ios::noreplace);

Для читання і запису даних будь-якого типу, тип яких може займати більше 1 байта, у файловій системі мови С є дві функції: ***fread()*** і ***fWrite().***

size_t ***fread****(void *buf,size_t count,size_t k,FILE *pf);*

size_t ***fwrite****(const void *buf,size_t count,size_t k,FILE *pf);*

Функція *fread()* повертає кількість прочитаних елементів. Якщо досягнуто кінець файлу або сталася помилка, то повернуте значення може бути менше, ніж лічильник. А функція *fwrite()* повертає кількість записаних елементів. Якщо помилки не було, то повернений результат буде дорівнює значенню лічильник. Одним з найбільш корисних застосувань функцій *fread()* і *fwrite()* є читання і запис даних типів користувача . Наприклад, якщо визначена структура:

```
struct struct_type
{   float balance;
    char name [80];
} Cust;
```

то наступний оператор записує вміст *Cust* у файл, на який вказує *fp:*

fwrite *(&cust, sizeof (struct struct_type), 1, fp);*

В системі введення/виведення мови С/С++ є функції ***fprintf()*** і ***fscanf():***

int ***fprintf*** *(FILE * pf, const char *str, ...);*

int ***fscanf*** *(FILE * pf, const char *str, ...);*

1.4 Приклади програм для робот из файлами

Приклад 1. Введення інформації з клавіатури і виведення на диск.

```
int main (int argc, char * argv [])
{
   FILE * fp;
   char ch;
   if (argc! = 2)
   {
    printf ("Ви забули ввести ім'я файлу. \n");
    exit (1);
   }
   if ((fp = fopen (argv [1], "w "))== NULL)
   {
    printf ("Помилка при відкритті файлу. \n");
    exit (1);
   }
```

```
   do {    ch = getchar ();
      putc (ch, fp);
      } while (ch! ='$');
   fclose (fp);
   return 0;
}
```

Приклад 2. Програма читає текстовий файл і виводить його на екран

```
int main (int argc, char * argv [])
{
    FILE * fp;
    char ch;
    if (argc! = 2)
    {
      printf ("Ви забули ввести ім'я файлу. \n");
      exit (1);
    }
    if ((fp = fopen (argv [1], "r "))== NULL)
    {
      printf ("Помилка при відкритті файлу. \n");
      exit (1);
    }
    ch = getc (fp); /* читання одного символу */
    while (ch! = EOF)
    {
       putchar (ch); /* виведення на екран */
       ch = getc (fp);
    }
    fclose (fp);
  return 0;
}
```

Приклад 3. Програма вводить рядок з клавіатури, записує у файл, а потім читає файл і виводить його вміст на екран

```
int main (void)
{
  char str[80];
  FILE *fp;
  if ((fp = fopen ("TEST", "w+"))== NULL)
  {
      printf ("Помилка при відкритті файлу. \n");
      exit (1);
  }
  do { printf("Введіть рядок (порожній - для виходу): \n");
       gets(str);
       strcat(str,  "\n"); /* введення роздільник рядків */
```

```
        fputs(str, fp);
    } while(*str != '\n');
       /* Тепер виконується читання і відображення файлу */
    rewind(fp); /* встановити покажчик на початок файлу */
    while (!feof(fp))
    {       fgets (str,79,fp);
            printf (str);
    }
    return 0;
}
```

Приклад 4. Використання форматованого введення-виведення у файл

```
int main (void)
{
     FILE *fp;
     char s[80];
     int t;
     if ((fp = fopen ("test", "w")) == NULL)
     {    printf ("Помилка відкриття файлу. \n");
          exit (1);
     }
     printf ("Введіть рядок і число:");
     fscanf (stdin, "%s%d",s,&t); // читати з клавіатури
     fprintf (fp,  "%s%d",s,t);      // писати в файл
     fclose (fp);
     if ((fp = fopen ("test", "r")) == NULL)
     {    printf ("Помилка при відкритті файлу. \n");
          exit (1);
     }
     fscanf (fp, "%s%d",s,&t);  // читання з файлу
     fprintf (stdout, "%s%d",s,t);   // виведення на екран return 0;
}
```

2. Порядок виконання роботи

1. Ознайомитись з теоретичним матеріалом, з функціями стандартних бібліотек для роботи файлами і файловими потоками.

2. Дослідити процес реалізації завдань прикладів, відлагодити наведені програму на своєму комп'ютері.

3. Розробити власні програми, які реалізує індивідуальне завдання.

4. Підготувати звіт по кожній задачі:

- варіант і текст завдання;
- формалізація змісту;
- алгоритм;
- лістинг програми;

- схему даних (для обох задач) і схему взаємозв'язку функцій (лише для другої задачі);
- роздруківку вхідних і вихідних файлів і результати виконання;
- висновки.

3. Варіанти індивідуальних завдань

Задача 1. Взявши за основу *лабораторну роботу № 2_5*, змінити код програми таким чином, щоб:

- вхідні дані вводилися не з клавіатури, а з файлу,
- результати виконання виводились і на екран, і у файл.

Задача 2, 3. Взявши за основу *лабораторну роботу № 2_6 (задача 1)*, змінити код програми таким чином, щоб:

- вхідні дані (поля структури) вводилися з клавіатури і після введення записувалися у файл (окрема функція);
- програма мала можливість дописувати дані у файл (окрема функція);
- дані з файлу виводились на екран (окрема функція);
- результати виконання другого підпункту виводились на екран і у файл. При цьому виконати завдання задачі №2 у двох варіантах:

1) за допомогою класів потоків *ofstream* та *ifstream;*
2) за допомогою структури *FILE* та функцій роботи з нею.

***Задача 4.** Одним за найпростіших в реалізації та найшвидших методів шифрування є накладання гами - послідовності псевдовипадкових чисел -шляхом виконання операції побітового додавання за модулем 2 (також відомої як exclusive or чи скорочено XOR).

Розробіть програму для зашифрування та розшифрування файлів з довільним розширенням, використовуючи як ключ, на основі якого генерується послідовність псевдовипадкових чисел, номер варіанту.

Контрольні питання

1. Дати поняття потоків і файлів. Який між ними зв'язок?
2. Які види потоків бувають? Пояснити різницю між ними.
3. Охарактеризувати стандартні потоки введення-виведення.
4. Яким чином можна оголосити і відкрити файл?
5. Що таке режими відкриття файлів і які режими ви знаєте?
6. Наведіть приклади форматованого запису у файл і читання з нього.
7. Яким чином можна встановлювати покажчик файлу у задану позицію?
8. Яким чином можна здійснювати запис і зчитування блоками?
9. Чи можна одночасно використовувати файл для запису і читання?

10. Що таке файлові потоки? Які поняття програмування забезпечують роботу з ними? Назвіть їх.

11. Основні функції роботи з файловими потоками.

12. Які режими доступу до файлових потоків ви знаєте?

13. Яким чином можна виводити у файлові потоки і вводити великі обсяги даних (структури, масиви)?

14. Як можна перевірити, чи закінчився файл, і наявність помилок?

15. Наведіть приклади запису і читання по символах, по словах, по рядках та блоками довільного розміру.

16. Алгоритм (сценарій) роботи з файлами.

17. Яким чином реалізується робота з файлами в бібліотеці stdio.h?

18. Які існують переваги бібліотеки stdio.h над бібліотекою fstream?

19. Яким чином реалізується робота з файлами в бібліотеці fstream?

20. Які існують переваги бібліотеки fstream над бібліотекою stdio.h?

21. Які існують методи відкриття файлів?

22. Яким чином отримати доступ у файлі за заданою позицією?

23. Яким чином записати дані у файл?

24. Яким чином задати формат даних, що виводяться у файл?

25. Яким чином закрити файл?

26. Яким чином записати у файл структуру?

27. Яким чином можна прочитати дані з файлу?

28. Яким чином можна прочитати дані з файлу, визначаючи форматування?