**Тема16.Класи потоків введення-виведення**

З потоковим введенням-виведенням файлів ми познайомилися в загальних рисах ще на лабораторні роботі №6. Тепер ми поглибимо і систематизуємо ці знання.

**Поняття файлу**

Під час розв’язання задач на комп’ютері часто виникає необхідність у використанні даних, які записані на зовнішніх носіях інформації (дисках) і оформлені у вигляді файлів даних. Незалежно від того, які дані містять файли (числа, символи, рядки, масиви, структури тощо), в мові С++ вони трактуються як потоки даних **(stream),** які являють собою послідовність байтів, що зчитуються або записуються.

За замовчуванням у кожній програмі С++ можна користуватися такими стандартними потоками: стандартного введення **(сіn),** стандартного виведення **(cout)** та виведення помилок **(сеrr).** Щоб користуватися файлами, потоки повинні бути створені і закріплені за цими файлами.

**Використання файлів даних у програмі передбачає виконання таких операцій**:

* створення потоку обміну даними між файлом і пам’яттю комп’ютера;
* зв’язування цього потоку з конкретним ім’ям файлу на диску і відкриття файлу;
* запис даних у файл або читання їх з файлу;
* закриття файлу.

Для реалізації цих операцій існують спеціальні класи, які містять конструктори створення необхідних потоків:

* **ifstream** — для створення потоку читання даних;
* **ofstream** — для створення потоку запису даних у файл;
* **fstream** — використовується як для запису даних у файл, так і їх читання.

**Конструктори для роботи з файлами**

Конструктори записуються як з параметрами, так і без параметрів. Конструктори з параметрами одночасно створюють відповідний потік, зв’язують його з файлом на диску, відкривають файл для роботи і мають такі форми запису:

**ofstream іп (“іф”, ios::out); або ofstream (“іф”);**

**ifstream іп (“іф”, ios::in); або ifstream (” “);**

**fstream іп (“іф”, ios::in | ios::out);**

де **іп** — ім’я потоку, який створюється для роботи з файлом;

**іф** — константа або змінна типу **char[ ],** її значення — ім’я файлу на диску.

Перший з конструкторів використовується для запису даних у файл, другий — для читання даних з файлу, а третій — як для запису, так і для читання даних, наприклад:

**ofstream fout( “myfile.dat”, ios::out);**

Цей запис створює потік з ім’ям **fout**, зв’язує його з файлом на диску, який має ім’я **myfile.dat** і відкриває цей файл для запису даних. Файл **myfile.dat** буде створено у тому ж каталозі, що і програма. Якщо треба створити файл у другому місці, то для запису його імені треба вказати шлях, наприклад, **a:\\pvp\\myfile.dat**. Тепер цей файл буде записано на диску **а:** в каталозі **pvp**.

Зверніть увагу на те, що для запису шляху треба використовувати подвійні зворотні косі риски.

Можна також для роботи з файлами застосувати конструктори без параметрів:

**ofstream іп;**

**ifstream іп;**

**fstream іп;**

де **іп** — ім’я відповідного потоку, тоді для зв’язування потоку з ім’ям файлу на диску і відкриття його для роботи треба додатково використовувати функцію-член відповідного класу, тобто:

**іn.ореn(“іф”, ios :: ознака | ios :: ознака);**

(в цьому запису  **|** означає "або", тобто можливість завдання декількох ознак відкриття файлу).

Наприклад, відкриття файлу для запису до нього даних матиме вигляд:

**ofstream fout;**

**fout.open ( “a:\\pvp\\my file.dat”, ios::out);**

Конструктори як з параметрами, так і без них, виконують однакову роботу, тому яким з них надати перевагу — справа користувача.

***Приклад 1.***  Створити файл на диску і записати до нього масив чисел. Прочитати цей файл і вивести його компоненти на екран.

// Створення файлу та запис до нього масиву

**#include <iostream>**

**#include <fstream>**

**#include <conio.h>**

**#include <Windows.h>**

**using namespace std;**

**int main()**

**{ int i, mas[5];**

**system("color F0");**

**// запис елементів масиву до файлу:**

**ofstream fout("massiv.txt"); /\* створення потоку fout та відкриття файлу з іменем massiv. txt для запису \*/**

**if (! fout) cout <<"Cannot open file\n";**

**for (i = 0; i <5; i++)**

**{**

**cout << " Enter "<< i << " element\n";**

**cin >> mas[i]; // введення елементу масиву з клавіатури**

**fout << mas[i] << " "; //запис елементу до файлу**

**}**

**fout.close();**

**//---------- читання компонентів файлу та виведення на екран**

**ifstream fin("massiv.txt"); /\* створення потоку fin для читання файлу\*/**

**if (!fin) cout << " Cannot open file fo reading\n";**

**cout <<"REZULTAT \n";**

**for (i = 0; i <5; i++)**

**{**

**fin >> mas[i];**

**// читання поточного елементу масиву з файлу**

**cout << "mas[" << i << "]=" << mas[i] << " ";**

**}**

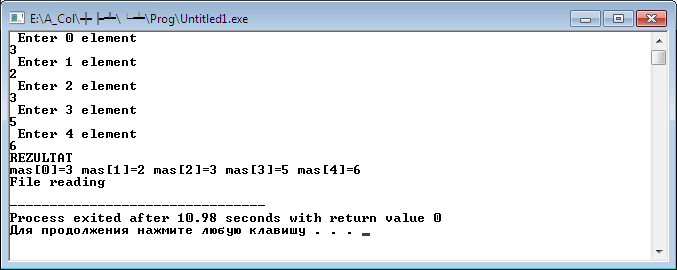
**cout << "\nFile reading\n";**

**fin.close();**

**return 0;**

**}**

Результат розв’язання прикладу має вигляд:



**Ознаки відкриття файлу**

Якщо не використовувати ознаки відкриття файлу для читання чи запису, на диску буде створений новий файл із зазначеним ім'ям, або перезаписаний файл із таким же ім'ям, якщо він вже існує.

Наприклад, для того, щоб записати потрібну інформацію в кінець файлу text1.dat, необхідно скористатися фрагментом програмного коду (тут файлова змінна flags):

ofstream flags(“text1.dat”, ios::app);

Таблиця 1 – Ознаки відкриття файлу

| **Ознака** | **Призначення** |
| --- | --- |
| ios::in | Відкриває файл для читання. Вміст файлу зберігається |
| ios::out | Відкриває файл для запису. Якщо файл не існує, то буде створений |
| ios::app | Відкриває файл для дозапису. Дані будуть записані у кінець файлу |
| ios::trunk | Якщо  файл, який відкривають для запису, вже існує, то його вміст буде видалено. |
| ios::nocreate | Забороняє створювати файл, який відкривають |
| ios::inoreplace | Забороняє перезаписувати існуючий  файл |

Файли класифікують за типом компонентів і за методом доступу до них. За типом компонентів розрізняють текстові та бінарні (двійкові) файли, а за методом доступу – файли послідовного і прямого доступу.

Текстові файли призначені для збереження текстів (наприклад, текстів програм), а бінарні файли використовуються для збереження даних різних типів. Файл бінарний – це лінійна послідовність байтів, що відповідає внутрішньому поданню даних без поділу на рядки. Для завдання бінарного файлу використовується ознака ios:binary (двійковий). Якщо задається декілька ознак (як в прикладі), то вони розділяються вертикальною рискою ( | ).

***Приклад 2.*** Записати у файл 5 прізвищ, потім прочитати їх і вивести на екран.

**/\* *програма запису до файлу та читання з файлу масиву з 5 прізвищ* \*/**

**#include <iostream>**

**#include <fstream>**

**#include <conio.h>**

**#include <Windows.h>**

**using namespace std;**

**int main()**

**{system("color F0");**

**char st[5][15];**

**int i;//------------------ запис до файлу**

**ofstream fout("st\_file.dat"); // відкриття файлу**

**if (!fout) cout << "Cannot open file\n";**

**for (i = 0; i <5; i++)**

**{**

**cout << " Enter " << (i+1) << " name\n";**

**cin.getline(st[i],15); // введення поточного прізвища**

**fout << st[i] << '\n'; // запис прізвища до файлу**

**}**

**fout.close();//------------читання файлу та виведення на екран**

**cout << "\nReading file\n\n";**

**ifstream fin("st\_file.dat");**

**if (!fin)cout << "Cannot open file.dat\n";**

**for (i = 0; i < 5; i++)**

**{**

**fin.getline(st[i],15);**

**cout << st[i] << " ";**

**}**

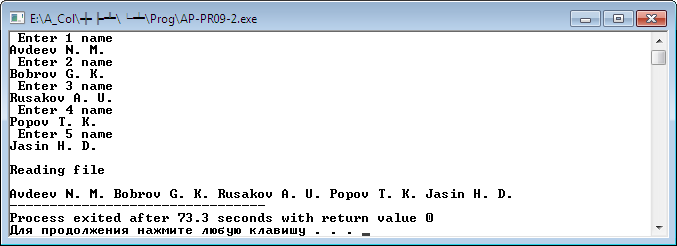
**fin.close();**

**getch();**

**return 0;**

**}**

Результат роботи програми має вигляд:

**.**

У попередніх програмах запис даних у файл та їх читання з файлу здійснювалось послідовно поелементно. Але записати або прочитати декілька даних (наприклад, масив чисел) можна однією операцією. Для цього використовують функції-члени відповідних класів, які мають вигляд:

**in.write((char\*)&p,sizeof(p));** — для запису даних у файл,

**in.read((char\*)&p,sizeof(p));** — для читання даних з файлу,

де **іn** — ім’я потоку введення або виведення;

**р** — змінна будь-якого типу, якщо змінна **р** має тип **char[ ],** то операція її приведення не потрібна.

**Функції обробки файлів та режими роботи з файлами**

У мові С++ використовуються функції роботи з файлами, запозичені з мови програмування C. Загальновживані функції роботи з файлами наведено в табл. 2.

Таблиця 2 – Основні функції роботи з файлами

| **Функція** | **Призначення** |
| --- | --- |
| fopen() | Відкриває потік |
| fclose() | Закриває потік |
| fputc() | Записує символ у потік |
| fgetc() | Зчитує символ з потоку |
| fwrite() | Записує блок даних у потік |
| fread() | Зчитує блок даних з потоку |
| fseek() | Встановлює індикатор позиції файлу на заданий байт у потоці |
| fprintf() | Робить для потоку те, що функція printf() робить для консолі |
| fscanf() | Робить для потоку те, що функція scanf() робить для консолі () |
| feof() | Повертає значення true, якщо досягнуто кінець файлу |
| ferror() | Повертає значення true, якщо виникла помилка |
| rewind() | Встановлює індикатор позиції файлу у початок файлу |
| remove() | Видаляє файл |

**Функції, призначені для відкриття / закриття файлів**

Функція **fopen()** виконує наступні дії:

* відкриває потік;
* пов'язує файл з потоком;
* повертає покажчик типу FILE на цей потік.

Прототип функції fopen():

**FILE \*fopen(const char \* filename, const char \*mode);**

У цьому записі параметр filename вказує на ім'я файлу, що відкривається, параметр mode – на рядок, що містить потрібний режим відкриття файлу. Можливі значення режима відкриття файлу наведені у табл. 3. Параметр filename повинен представляти рядок символів, який визначає ім'я файлу, допустиме у даній операційній системі. Цей рядок може містити специфікацію шляху до файлу, якщо діюче програмне середовище підтримує таку можливість.

Таблиця 3– Режими роботи з файлами

| **mode** | **Призначення** |
| --- | --- |
| "r" | Відкриває текстовий файл для зчитування |
| "w" | Створює текстовий файл для запису |
| "а" | Відкриває текстовий файл для запису у кінець файлу |
| "rb" | Відкриває двійковий файл для зчитування |
| "wb" | Створює двійковий файл для запису |
| "ab" | Відкриває двійковий файл для запису у кінець файлу |
| "r+" | Відкриває текстовий файл для зчитування і запису |
| "w+" | Створює текстовий файл для зчитування і запису |
| " a+ " | Відкриває текстовий файл для зчитування і запису у кінець файлу |
| "r+b" | Відкриває двійковий файл для зчитування і запису |
| "w+b" | Створює двійковий файл для зчитування і запису |
| "a + b" | Відкриває двійковий файл для зчитування і запису у кінець файлу |

У випадку успішного відкриття вказаного файлу функція fopen() повертає покажчик FILE. Цей покажчик ідентифікує файл. Якщо файл не вдається  відкрити, наприклад, якщо файл із заданим іменем відсутній на диску, повертається нульовий покажчик.

**Приклад 3.** Відкриття текстового файлу test для запису**:**

fp = fopen("test", "w");

Функція fopen() має режими роботи як з текстовими, так і з двійковими (бінарними) файлами.

Закриття файлу здійснюється функцією **fclose()**, що має наступний прототип:

**Int fclose (FILE \* Stream);**

У якості параметра функції виступає покажчик на файл, що закривається. Функція повертає 0 якщо файл був успішно закрито, і  значення 1 – в іншому випадку.

Приклад виклику функції закриття файлу:

fclose(p1);

Під час виклику функції  fclose()  звільняється  блок  керування файлом, що  пов'язаний  з  потоком,  що  робить  його  доступним  для  повторного використання.

**Введення/виведення даних**

Функцію **fputc()** використовують для виведення символів у потік, який заздалегідь відкрито для запису за допомогою функції fopen().

Прототип функції fputc():

**int fputc(int ch, FILE \*fp);**

У цьому  записі параметр  fp представляє собою файловий покажчик, що повертається функцією fopen(), а параметр  ch –  символ, що виводиться. Файловий покажчик повідомляє функції fputc(), в який дисковий файл необхідно записати символ. При успішному виконанні операції виведення функція  fputc()  повертає записаний у файл символ, інакше – значення EOF.

Функція  **fgetc()**  використовують  для  зчитування  символів  з потоку, відкритого в режимі  зчитування  за допомогою функції  fopen().

Прототип функції fgetc():

**int fgetc(FILE \*fp);**

У цьому  записі параметр  fp означає файловий покажчик, що повертається fopen(). Під  час  виникненні помилки  або  досягнення кінця файлу функція fgetc() повертає значення EOF. Отже, для того, щоб зчитати весь вміст текстового файлу можна використати фрагмент програмного коду:

**ch = fgetc(fp);**

**while(ch != EOF)**

**{**

**ch= fgetc(fp);**

**}**

Функція **fprintf()** форматує і друкує у вихідний потік набори символів і значень. Кожен аргумент (якщо він є) перетворюється і виводиться згідно заданої специфікації формату.

Прототип функції fprintf():

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

Функція **fscanf()** зчитує дані з поточної позиції потоку в місце, обумовлене значенням аргументів (якщо вони є). Кожен аргумент повинен бути покажчиком на змінну і тип, який відповідають типу, заданому в рядку формату.

Прототип функції fscanf():

**int fprintf (stream, format-string [, argument ...]);**

Таким чином призначення функцій fprintf() та fscanf() аналогічно функціям printf()  і scanf(), за винятком  того, що вони працюють з файлами.

**Приклад 4.** Дано два текстових файли, що містять цілі числа, впорядковані за зростанням. Необхідно об'єднати ці файли в третій, що будуть містити впорядковані за зростанням числа з перших двох файлів.

**#include <iostream>**

**#include <fstream>**

**#include <Windows.h>**

**using namespace std;**

**int main()**

**{system("color F0");**

**FILE \*f1,\*f2,\*r;**

**int a1,a2;**

**f1= fopen("file1.txt","r");**

**if(f1==NULL) return 1;**

**f2 = fopen("file2.txt","r");**

**if(f2==NULL)**

**{**

**fclose(f1);**

**return 1;**

**}**

**r = fopen("result.txt","w");**

**if(r==NULL){fclose(f1);fclose(f2);return 1;}**

**fscanf(f1,"%d",&a1);**

**fscanf(f2,"%d",&a2);**

**cout << a1 << '\t'<< a2 << endl;**

**while(1)**

**{**

**if(a1<a2)**

**{**

**fprintf(r,"%d ",a1);**

**if(fscanf(f1,"%d",&a1)==EOF) break;**

**}**

**else**

**{**

**fprintf(r,"%d ",a2);**

**if(fscanf(f2,"%d",&a2)==EOF) break;**

**}**

**}**

**if(feof(f1))**

**{**

**while(!feof(f2))**

**{**

**fprintf(r,"%d ",a2);**

**fscanf(f2,"%d",&a2);**

**}**

**fprintf(r,"%d ",a2);**

**}**

**else**

**if(feof(f2))**

**{**

**while(!feof(f1))**

**{**

**fprintf(r,"%d ",a1);**

**fscanf(f1,"%d",&a1);**

**}**

**fprintf(r,"%d ",a1);**

**}**

**fclose(f1);**

**fclose(f2);**

**fclose(r);**

**return 0;**

**}**

Функцію **ferror()** використовують  для  визначення факту виникнення помилки у процесі виконання операції з файлом.

Її прототип має такий вигляд:

**int ferror(FILE \*fp);**

У цьому записі параметр  fp дійсний файловий покажчик. Функція ferror() повертає значення  true, якщо у процесі виконання останньої файлової операції відбулася  помилка;  інакше  значення  false.  Оскільки  виникнення  помилки можливе  у  процесі  виконання  будь-якої  операції  з файлом, функцію  ferror() необхідно викликати відразу після кожної функції оброблення файлів; інакше інформацію про помилку можна просто втратити.

Функція  **rewind()**  переміщує  індикатор  позиції  файлу у початок файлу, що задається як аргумент.

Її прототип має такий вигляд:

**void rewind(FILE \*fp);**

У цьому записі параметр fp представляє собою дійсний файловий покажчик.

Функція **fseek()** призначена для встановлення індикатора позиції файла у необхідному місці. Таким чином, система введення-виведення у мові С++ подібно до мови С дає змогу виконувати операції зчитування і запису даних з довільним доступом.

Прототип функції fseek():

**int fseek(FILE \*fp, long numbytes, int origin);**

У цьому  записі параметр  fp означає файловий покажчик, що повертається функцією fopen(), параметр numbytes визначає кількість байтів щодо початкового положення, які задаються параметром origin. Параметр origin може приймати одне наступних макроімен (визначених у заголовку stdio.h):

* SEEK\_SET –константа 0 – пошук з початку файлу;
* SEEK\_CUR– константа 1 – пошук з поточної позиції;
* SEEK\_END– константа 2 –пошук з кінця файлу.

Нульове  значення  результату  функції  свідчить  про  успішне  виконання функції  fseek(),  а  ненульове  –  про  виникнення  збою.  Як  правило,  функцію fseek() не рекомендується використовувати для файлів, відкритих у текстовому режимі, оскільки перетворення символів може призвести до помилкових переміщень індикатора позиції у файлі. Тому краще використовувати цю функцію для файлів, відкритих у двійковому (бінарному) режимі.

**Приклад 5.** Дано бінарний файл, що містить дійсні числа. Поміняти в ньому місцями максимальний і мінімальний елементи**.**

**#include <iostream>**

**#include <fstream>**

**#include <Windows.h>**

**using namespace std;**

**int main()**

**{system("color F0");**

**FILE \*f;**

**int i=0, pmin=0, pmax=0;**

**double max,min,v;**

**f = fopen("file\_digits.dat","r+b");**

**if(f==NULL) return 1;**

**fread(&v,sizeof(double),1,f);**

**max = v;**

**min = v;**

**i=1;**

**while(!feof(f))**

**{**

**fread(&v,sizeof(double),1,f);**

**if(max<v) pmax = i;**

**if(min>v) pmin = i;**

**i++;**

**}**

**fseek(f,pmax\*sizeof(double),0);**

**fwrite(&min,sizeof(double),1,f);**

**fseek(f,pmin\*sizeof(double),0);**

**fwrite(&max,sizeof(double),1,f);**

**fclose(f);**

**return 0;**

**}**

**Введення / виведення бінарних файлів стандартними засобами**

У файловій системі мови С передбачено дві функції  fread()  і  fwrite(), які дають  змогу  зчитувати  і  записувати  блоки  даних. Ці функції подібні С++ функціям read() і write(). Їх прототипи мають такий вигляд:

**size\_t fread(void \*buffer, size\_t num\_bytes, size\_t count, FILE \*fp);**

**size\_t fwrite(const void \*buffer, size\_t num\_bytes size\_t count, FILE \*fp);**

Під  час  виклику  функції  fread()  параметр  buffer  є  покажчиком на область пам'яті, яка призначена для прийому даних, які зчитуються з файлу. Функція fread() повертає кількість зчитаних об'єктів, яка може опинитися менше заданого  значення  count,  якщо  у  процесі  виконання  цієї операції виникла помилка або був досягнуто кінець файлу.

Під час виклику функції  fwrite() параметр buffer є покажчиком на інформацію, яка підлягає запису у файл. Ця функція записує count об'єктів завдовжки um\_bytes у потік, яка адресується файловим покажчиком  fp. Функція fwrite() повертає кількість записаних об'єктів, яка буде дорівнює значенню count, як-що у процесі виконання цієї операції не було помилки.

Якщо під час виклику функцій fread() і fwrite() файл був відкрито для виконання двійкової операції, то вони можуть зчитувати або  записувати  дані будь-якого  типу.

**Приклад 6.** Запис значення типу float у двійковий (бінарний) файл.

#**include "stdafx.h"**

**int main()**

**{**

**#include <iostream>**

**#include <fstream>**

**#include <Windows.h>**

**using namespace std;**

**int main()**

**{system("color F0");**

**FILE \*fp;**

**float f = 12.23F;**

**if((fp=fopen("test","wb"))==NULL)**

**{**

**printf("File is not opened!" );**

**return 1;**

**}**

**fwrite(&f,sizeof(float), 1, fp);**

**fclose(fp);**

**Sleep(2000);**

**return 0;**

**}**

**Приклад 5. Запис масиву у бінарний файл.**

**#include <iostream>**

**#include <fstream>**

**#include <Windows.h>**

**int main()**

**{ system("color F0");**

**int i;**

**FILE \*fp;**

**float balance[100];**

**//Відкриваємо файл для запису.**

**if((fp=fopen("balance","w"))==NULL)**

**{**

**printf("Не вдається відкрити файл");**

**return 1;**

**}**

**for(int i=0; i<100; i++) balance[i] = (float) i;**

**//зберігаємо весь масив balance.**

**fwrite(balance, sizeof balance, 1, fp);**

**fclose(fp);**

**//Обнуляємо масив.**

**for(int i=0; i<100; i++) balance[i] = 0.0;**

**//Відкриваємо файл для зчитування.**

**if((fp=fopen("balance","r"))==NULL)**

**{**

**printf("Не вдається відкрити файл");**

**return 1;**

**}**

**//зчитуємо весь масив balance.**

**fread(balance, sizeof balance, 1, fp);**

**// Відображаємо вміст масиву.**

**for(int i=0; i<100; i++) printf("%f ", balance[i]);**

**fclose(fp);**

**return 0;**

**}**

 Робота з текстовим файлом як потоком

Ці питання ми постійно опрацьовуємо на лабораторних, тому приклади розглядати не будемо.

Оголошення файлової змінної та відкриття текстового файлу для читання має наступний синтаксис:

**ifstream f\_in(“filename.txt”, ознака1|ознака2|…|ознакаN);**

У даному записі f\_in – назва файлової змінної, filename.txt –назва файлу на диску. Ознаки слугують для визначення прав доступу до файлу. Перелік ознак відкриття файлу та їх значень наведено в табл. 1.

Оголошення файлової змінної та відкриття файлу для запису має синтаксис:

**ofstream f\_out(“filename.txt”, ознака1|ознака2|…|ознакаN);**

У даному записі f\_out – назва файлової змінної, filename.txt –  назва файлу на диску.

Наприклад, для того, щоб створити текстовий файл і записати в нього рядок «Робота в С++» необхідно виконати наступні кроки:

1. створити об’єкт класу ofstream, наприклад, з назвою fout:
2. пов’язати об’єкт класу з файлом, в який буде проводитися запис:
3. записати рядок в файл;
4. закрити файл:

**ofstream fout;**

**fout.open("cppstudio.txt"); // це можна було зробити, записавши**

**// ofstream fout(“cppstudio.txt ”, app);**

**fout << “ Робота в С++ ”;**

**fout.close();**

Для того щоб прочитати створений файл, знадобитися виконати аналогічні кроки, що й при запису у файл з невеликими змінами:

1. створити об'єкт класу ifstream і пов'язати його з файлом, який відкриємо в режимі для читання:
2. прочитати файл;
3. закрити файл.

**ifstream fin("cppstudio.txt");**

**fin >> buff;**

**fin.close().**

*Для самостійного вивчення*: Поглибити матеріал лекції за наданою літературою. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

*Література* [1, 4, 6, 11, 12, 18]

*Контрольні запитання*.

1. Які види потоків ви знаєте?
2. Для чого призначені стандартні потоки?
3. Який файл необхідно включати у програму для використання стандартних потоків?
4. Назвіть класи вхідних та вихідних потоків.
5. Який клас є базовим для потоків?
6. Назвіть класи для створення файлових потоків і їх призначення.
7. Які дії необхідно виконати для використання файлів у програмі?
8. Які існують режими відкриття файлів?
9. Для чого призначений метод open() і які він має параметри?
10. Які способи відкриття та закриття файлу ви знаєте?
11. Як записати дані у файл?
12. Як прочитати дані з файлу?