

## PLIKOWE OPERACJE WEJŚCIA - WYJŚCIA

Język C/C++ nie ma wbudowanych żadnych instrukcji umożliwiających wykonywanie operacji wejścia-wyjścia ! Służą do tego funkcje biblioteczne.

Omówienie pojęć „zmienna plikowa” oraz „wskaźnik położenia” na przykładzie historycznej / niestandardowej biblioteki `<io.h>`

Dostęp do pliku za pomocą uchwytu (ang. Handle) - operacje niskiego poziomu

1. Funkcje otwierania (zwraca uchwyt pliku) oraz zamknięcia pliku

```
int open ( const char *nazwa_pliku, int tryb_dostepu )
int close ( int handle )
```

2. Funkcje zapisu i odczytu z pliku

```
int write ( int handle, void *adres_bufora, unsigned ilosc_bajtow )
int read ( int handle, void *adres_bufora, unsigned ilosc_bajtow );
```

3. Funkcje pomocnicze

```
int eof ( int handle )           // zwraca 1 gdy „END OF FILE”
long tell ( int handle )         // zwraca pozycję wskaźnika pliku
long filelength ( int handle )  // zwraca długość pliku w bajtach
long lseek ( int handle, long przesuniecie, int względem_czego )
    // przesuwa wskaźnik pliku o zadaną ilość bajtów
    // względem zadanego miejsca:
    SEEK_SET - względem początku pliku
    SEEK_CUR - względem aktualnej pozycji
    SEEK_END - względem końca pliku
```

### Przykład

```
int plik;           // zmienna plikowa / identyfikator pliku
char tekst[ ] = "To jest tekst zapisywany i odczytywany z pliku";
char znak;
plik = open( "test.dat", O_CREAT | O_RDWR );
write( plik, tekst, strlen( tekst ) );           // zapis zawartosci tekstu do pliku
lseek( plik, 0L, SEEK_SET );                     // przesuniecie wskaźnika na poczatek
do
{
    read( plik, &znak, 1);                       // odczyt po jednym znaku aż do napotkania eof
    printf( "%c", znak );                          // wydruk odczytanego znaku na ekranie
} while ( !eof( plik ) );
close( plik );
```

## Podstawowa biblioteka I/O dla języka C: <stdio.h>

Operacje we/wy realizowane za pomocą **strumieni** (ang. stream)

Strumienie reprezentowane są przez zmienne typu **FILE**. Struktura taka stworzona jest automatycznie podczas otwierania strumienia (zawiera informacje o nazwie pliku, trybie otwarcia, itp.). Wszystkie dalsze operacje na strumieniu wymagają podania wskaźnika na tą strukturę.

### Przykład

```
FILE *plik_wej, *wyniki ;           // definicja zmiennych plikowych
```

### 0. Standardowe strumienie wejścia i wyjścia (otwierane automatycznie)

**stdin** – strumień wejściowy (konsola - klawiatura)  
**stdout** – strumień wyjściowy (konsola - monitor)  
**stderr** – strumień komunikatów błędów (konsola)  
**stdprn** – strumień drukarki

### 1. Funkcja otwarcia (zwraca wskaźnik na FILE) oraz zamknięcia pliku

```
FILE * fopen ( char *nazwa_pliku, char *rodzaj_operacji )
```

rodzaj operacji:

<b>r</b>	–	tylko do odczytu
<b>w</b>	–	tylko do zapisu (utworzenie nowego)
<b>a</b>	–	dopisywanie na końcu
<b>+</b>	–	z możliwością aktualizacji
<b>b</b>	–	otwarcie jako plik binarny
<b>t</b>	–	otwarcie jako plik tekstowy

```
int fclose ( FILE *strumien )           // zamknięcie wskazanego strumienia
```

Przykład utworzenie nowego pliku binarnego (z dodaną możliwością aktualizacji)

```
FILE *plik;                           // utworzenie zmiennej plikowej
plik = fopen( "a:\\wyniki.dat", "w+b" ); // otwarcie / utworzenie pliku
if( plik == NULL )                     // kontrola błędów we/wy
{
    printf( "Bład otwarcia pliku. " );
    return -1;
}
...
fclose( plik );                       // zamknięcie / zwolnienie pliku
```

## 2. Zapis danych do strumienia

2.a) Operacje na pliku **tekstowym** (pliku zawierającym kody ASCII, spacje, NL):

```
int fputc ( int znak, FILE *strumien )    // wysłanie pojedynczego znaku  
  
int fprintf ( FILE *strumien, char *format, . . . )  
    // funkcja sformatowanego wyjścia analogiczna do printf()  
  
int fputs ( char *tekst, FILE *strumien )    // wysłanie łańcucha znaków
```

2.b) Operacje na pliku **binarnym** (na ciągu dowolnych bajtów):

```
int fwrite ( void* adres_w_pamieci,  
            size_t rozmiar_bloku, size_t ilosc_blokow,  
            FILE * strumien)  
    // funkcja kopiująca (ilosc_blokow*rozmiar_bloku) bajtów  
    // ze wskazanego obszaru pamięci do strumienia (pliku)
```

### Przykład

```
#include <stdio.h>  
struct T_student  
{  
    char nazwisko[31];  
    char imie[16];  
    int wiek;  
};  
  
void main( void )  
{  
    FILE *strumien;  
    T_student baza_danych[10];  
    if ( (strumien = fopen( "test.bin" , "wb" ) ) != NULL )  
    { // zapis zawartości całej bazy ( tablicy struktur) do pliku binarnego  
      fwrite( baza_danych, sizeof(T_student), 10 , strumien);  
      fclose( strumien );  
    }  
  
    if ( (strumien = fopen( "test.txt" , "wt" ) ) != NULL )  
    { // zapis zawartości całej bazy ( tablicy struktur) do pliku tekstowego  
      for( int i = 0; i < 10; i++ )  
        fprintf ( strumien, "%s %s %d \n", baza_danych[ i ].nazwisko,  
                 baza_danych[ i ].imie, baza_danych [ i ].wiek );  
      fclose( strumien );  
    }  
}
```

Jeżeli jako strumień wyjściowy podamy **stdout** (standardowy strumień wyjściowy) to wtedy wydruk będzie dokonywany na konsolę/ekran.

np. **fprintf**( stdout, "format" , ..... ) ≡ **printf**( "format" , ..... )

### 3. Odczyt danych ze strumienia

3.a) Operacje na pliku **tekstowym** (pliku zawierającym kody ASCII, spacje, NL):

```
int fgetc ( FILE *strumien )           // wczytanie pojedynczego znaku  
int fscanf ( FILE *strumien, char *format, . . . )  
// funkcja sformatowanego wejścia analogiczna do scanf()
```

```
char* fgets ( char *tekst, int dlugosc, FILE *strumien )  
// wczytanie łańcucha składającego się z co najwyżej (dlugosc-1) znaków
```

3.b) Operacje na pliku **binarnym** (na ciągu dowolnych bajtów):

```
int fread ( void* adres_w_pamieci,  
            size_t rozmiar_bloku, size_t ilosc_blokow,  
            FILE * strumien)  
// funkcja odczytująca (ilosc_blokow*rozmiar_bloku) bajtów  
// ze strumienia do wskazanego obszaru pamięci
```

#### Przykład

```
#include <stdio.h>  
struct T_student  
{  
    char nazwisko[31];  
    char imie[16];  
    int wiek;  
};  
  
void main( void )  
{  
    FILE *strumien;  
    T_student baza_danych[10];  
    int ilosc;  
    if ( (strumien = fopen( "test.bin" , "rb" ) ) != NULL )  
        { // wczytanie zawartości bazy ( tablicy struktur) z pliku binarnego  
          ilosc = 0;  
          while( fread( &baza_danych[ilosc],sizeof(T_student),1,strumien)==1)  
              ilosc++;  
          fclose( strumien );  
        }  
  
    if ( (strumien = fopen( "test.txt" , "rt" ) ) != NULL )  
        { // wczytanie zawartości bazy ( tablicy struktur) z pliku tekstowego  
          for( int i = 0; ( ! feof(strumien) ) && ( i < 10); i++)  
              fscanf( strumien, "%s %s %d" , baza_danych [ i ].nazwisko,  
                    baza_danych [ i ].imie, &(baza_danych [ i ].wiek) );  
          fclose( strumien );  
        }  
    }  
}
```

#### 4. Funkcje pomocnicze

```
int feof ( FILE *strumien )           // testowanie osiągnięcia końca pliku
int fseek ( FILE *strumien, long przesuniecie, int wzgleciem)
    // przesuwa wskaźnik pliku o zadaną ilość bajtów
    // względem zadanego miejsca:
    SEEK_SET - względem początku pliku
    SEEK_CUR - względem aktualnej pozycji
    SEEK_END - względem końca pliku

long ftell ( FILE *strumien ) // zwraca aktualną pozycję wskaźnika pliku
int fflush ( FILE *strumien ) // „czyści” bufor wskazanego strumienia
```

#### Przykład zadania zaliczeniowego

```
// funkcja wyznaczająca pozycję maksymalnej liczby double w pliku binarnym
#include <stdio.h>

long Maksimum( char *nazwa_pliku )
{
    FILE *plik_danych;
    long pozycja=0, poz_max = -1;
    double liczba, maksimum;
    if ( (plik_danych = fopen( nazwa_pliku , "rb" ) ) != NULL )
    {
        while( fread( &liczba, sizeof(double), 1, plik_danych) == 1)
        {
            if( pozycja == 0 )
            {
                maksimum = liczba;
                poz_max = 0;
            }
            else
            if( liczba > maksimum )
            {
                maksimum = liczba;
                poz_max = pozycja;
            }
            pozycja++;
        }
        fclose( strumien );
    }
    return poz_max ;
}
```