

## OPERACJE WEJŚCIA – WYJŚCIA

Operacje wejścia / wyjścia → odczyt i zapis danych do różnych zewnętrznych urządzeń lub nośników pamięciowych komputera: np. klawiatury, ekranu monitora, dyskietki, czytnika taśmy, drukarki, itp.

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

### Operacje na plikach (niskiego poziomu) → < IO.H >

```
int  open ( char *nazwa_pliku, int tryb_dostepu )
int  close ( int handle )
int  write ( int handle, void *adres_bufora, unsigned ilosc_bajtow )      bin.
int  read ( int handle, void *adres_bufora, unsigned ilosc_bajtow )      bin.
int  eof ( int handle )
long tell ( int handle )
long filelength ( int handle )
long lseek ( int handle, long przesuniecie, int wzgledem_czego )
```

### Proceduralnie za pomocą strumienia → < STDIO.H >

```
FILE * fopen ( char *nazwa_pliku, char *rodzaj_operacji )
int    fclose ( FILE *strumien )
int    fcloseall ( void )
int    fputc ( int znak, FILE *strumien )                                txt
int    fputs ( char *tekst, FILE *strumien )                            txt
int    fprintf ( FILE *strumien, char *format, . . . )                  txt
int    fwrite ( void* adres, size_t rozm_bl, size_t il_blokow, FILE* strumien ) bin
int    fgetc ( FILE *strumien )                                         txt
char*  fgets ( char *tekst, int dlugosc, FILE *strumien )              txt
int    fscanf ( FILE *strumien, char *format, . . . )                  txt
int    fread ( void* adres, size_t rozm_bl, size_t il_blokow, FILE* strumien ) bin
int    feof ( FILE *strumien )
int    fseek ( FILE *strumien, long przesuniecie, int wzgledem)
long   ftell ( FILE *strumien )
int    fflush ( FILE *strumien )
int    flushall ( void )
```

## OBIEKTOWA REALIZACJA OPERACJI WEJŚCIA – WYJŚCIA

W języku C++ możliwa jest obiektowa realizacja operacji we/wy. Podejście obiektowe zakłada, że różne „urządzenia” będą reprezentowane w programie za pomocą różnych **obiektów** modelujących **strumienie** danych wpływające lub wypływające z tych „urządzeń”.

W obiektowych bibliotekach we/wy **zdefiniowano różne klasy** obiektów –strumieni (w zależności od specyficznych cech danego „urządzenia”). Cechy strumienia można odczytać z poszczególne liter nazw klas :

- **i....** – (*in*) – strumienie wejściowe (np. **istream**, **ifstream**, **istrstream**),
- **o....** – (*out*) – strumienie wyjściowe (np. **ostream**, **ofstream**, **ostrstream**),
- **f....** – (*file*) – strumienie plikowe (np. **ifstream**, **ofstream**, **fstream**),
- **str..** – (*string*) – strumienie pamięciowe (np. **istrstream**, **strstream**),

Aby uniknąć wielokrotnego definiowania tych samych operacji (np. dla każdego strumienia musi być funkcja informująca czy wystąpił błąd) klasy strumieni tworzą **wielopoziomą hierarchię**:

### PODSTAWOWĄ KLASĄ JEST KLASA **ios**

Modeluje ona właściwości (tzn. funkcje, zmienne i stałe) wspólne dla wszystkich strumieni. Definicja klasy **ios** jest zawarta w pliku <iostream.h>.

Najważniejsze metody tej klasy:

- **int ios::bad( )** - zwraca wartość różną od zera, jeżeli wystąpił błąd,
- **int ios::good( )** - zwraca wartość różną od zera, jeżeli nie było błędów,
- **int ios::eof( )** - zwraca wartość różną od zera, gdy koniec danych,
- **int ios::width( int )** - steruje szerokością pola wyjściowego (np. ilość cyfr)
- **int ios::precision( int )** - steruje ilością cyfr po przecinku

Stałe trybów otwarcia strumienia:

- **ios::in** - otwórz strumień do **odczytu**,
- **ios::out** - otwórz strumień do **zapisu**,
- **ios::app** - otwórz strumień w trybie **dopisywania** na końcu,
- **ios::trunc** - wyzeruj rozmiar pliku, jeżeli istnieje,
- **ios::binary** - otwórz jako strum. **binarny** (domyślnie → strum. **tekstowy**),

Stałe określające pozycję odniesienia (podczas przesuwania pozycji):

- **ios::beg** - względem początku pliku,
- **ios::cur** - względem pozycji aktualnej,
- **ios::end** - względem końca pliku,

## PODSTAWOWE OPERACJE ODCZYTU → klasa **istream**

Modeluje ona metody wspólne dla wszystkich strumieni wejściowych z których odczytujemy dane (tekstowe lub binarne). Definicja klasy **istream** jest zawarta również w pliku <iostream.h>.

Najważniejsze metody tej klasy:

- **get( char& znak)** - wczytuje jeden znak ze strumienia,
- **getline(char\* bufor, int max\_dlug, char znak\_konca)** - wczytuje linię znaków,
- **read( char\* bufor, int ilość\_bajtów )** - wczytuje ciąg bajtów do bufora,
- **>>** - operator pobrania/odczytu danych ze strumienia tekstowego.

## PODSTAWOWE OPERACJE ZAPISU → klasa **ostream**

Modeluje ona metody wspólne dla wszystkich strumieni wyjściowych do których zapisujemy dane (tekstowe lub binarne). Definicja klasy **ostream** jest zawarta również w pliku <iostream.h>.

Najważniejsze metody tej klasy:

- **put( char& znak)** - wysyła jeden znak do strumienia,
- **write(char\* bufor, int ilość\_bajtów)** - wysyła ciąg bajtów z bufora do strum.
- **<<** - operator wysłania/zapisu danych do strumienia tekstowego.

## STRUMIENIE STANDARDOWE

W programach napisanych w języku C++ można korzystać z czterech predefiniowanych, zawsze otwartych strumieni standardowych:

- cin** - standardowy strumień wejściowy - **klawiatura** - (istream),
- cout** - standardowy strumień wyjściowy - **ekran** - (ostream),
- cerr** - strumień komunikatów błędów - zazwyczaj ekran - (ostream),
- clog** - w pełni buforowany strumień komunikatów błędów,

## PORÓWNANIE WE/WY «proceduralnego» i «obiektowego»

Wczytywanie danych z klawiatury i wydruk na ekranie

<pre>// podejście proceduralne # include &lt;stdio.h&gt; void main( void ) {   char znak;   int x;   long y;   double z;   char tekst[ 20 ];    scanf( "%c", &amp;znak );   scanf( "%d", &amp;x );   scanf( "%ld", &amp;y );   scanf( "%lf", &amp;z );   scanf( "%19s", tekst );    printf( "znak = %c \n", znak );   printf( "int = %d \n", x );   printf( "long = %d \n", y );   printf( "double = %f \n", z );   printf( "tekst = %s \n", tekst ); }</pre>	<pre>// podejście obiektowe # include &lt;iostream.h&gt; void main( void ) {   char znak;   int x;   long y;   double z;   char tekst[ 20 ];    cin &gt;&gt; znak; // cin.get(znak);   cin &gt;&gt; x;   cin &gt;&gt; y;   cin &gt;&gt; z;   cin &gt;&gt; tekst; //cin.getline(tekst,19)    cout &lt;&lt; "znak =" &lt;&lt; znak &lt;&lt; "\n";   cout &lt;&lt; "int =" &lt;&lt; x &lt;&lt; "\n";   cout &lt;&lt; "long =" &lt;&lt; y &lt;&lt; "\n";   cout &lt;&lt; "double =" &lt;&lt; z &lt;&lt; "\n";   cout &lt;&lt; "tekst =" &lt;&lt; tekst &lt;&lt; "\n"; }</pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### STRUMIENIE PLIKOWE → klasa **fstream**

Klasa **fstream** jest klasą pochodną od klas **iostream** (**istream** + **ostream**) oraz **fstreambase**. Jej definicja zawarta jest w pliku <fstream.h>.

Najważniejsze metody tej klasy:

- void **open**( char \*nazwa\_pliku, int tryb\_otwarcia ) - otwarcie pliku,
- void **close**( void ) - zamknięcie pliku skojarzonego ze strumieniem

Oraz wszystkie metody klas pierwotnych (względem **fstream**):

z klasy **ios** → **fail, good, eof, width, precision**  
z klasy **istream** → **get, getline, read, <<**  
z klasy **ostream** → **put, write, >>**

## Kopiowanie plików tekstowych z jednoczesną zamianą liter na duże

<pre> // podejście proceduralne # include &lt;stdio.h&gt; # include &lt;ctype.h&gt; void main( void ) {     char znak;     FILE *wej, *wyj;     wej = fopen( "dane.dat", "rt" );     wyj = fopen( "wyniki.dat", "wt" );     if( (wej!=NULL) &amp;&amp; (wyj!=NULL) )     {         while( !feof(wej) )         {             znak = fgetc(wej);             znak = toupper(znak);             fputc( znak,wyj );         }     }     fclose( wej );     fclose( wyj ); } </pre>	<pre> // podejście obiektowe # include &lt;fstream.h&gt; # include &lt;ctype.h&gt; void main( void ) {     char znak;     fstream wej,wyj;     wej.open( "dane.dat", ios::in );     wyj.open( "wyniki.dat", ios::out );     if( wej.good( ) &amp;&amp; wyj.good( ) )     {         while( ! wej.eof( ) )         {             wej.get( znak );             znak = toupper( znak );             wyj.put( znak );         }     }     wej.close( );     wyj.close( ); } </pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

*// funkcja wyznaczająca pozycję maksymalnej liczby double w pliku binarnym*

```
# include <fstream.h>
```

```
# include <values.h>
```

```
double POZYCJA_MAKSYMUM( char *nazwa_pliku )
```

```

{
    long licznik=0, pozycja=0;    double liczba, max = -MAXDOUBLE;
    fstream plik( nazwa_pliku , ios::in | ios::binary );
    while( plik.good( ) && !plik.eof( ) )
    {
        plik.read( (char*)&liczba, sizeof(double) );
        licznik++;
        if( liczba>max )
        {
            max=liczba;    pozycja=licznik;
        }
    }
    plik.close( );
    return( pozycja );
}

```