Operacje na plikach

Informatyka

Materiały

Operacje wejścia-wyjścia w C/C++:

Podejście proceduralne	Podejście obiektowe
Standardowe strumienie wejścia i wyjścia	
stdin - strumień wejściowy (klawiatura) **stdout** - strumień wyjściowy (ekran) **stderr** - strumień komunikatów błędów (ekran)	**cin** - standardowy strumień wejściowy (klawiatura), **cout** - standardowy strumień wyjściowy (ekran), **cerr** - strumień komunikatów błędów (zazwyczaj ekran)
Operacje plikowe (realizowane są za pomocą strumieni)	
Strumienie reprezentowane są poprzez zmienne typu **FILE** (taka struktura tworzona jest automatycznie podczas otwierania strumienia i zawiera informacje o nazwie pliku, trybie otwarcia, itp.). Wszystkie kolejne operacje na pliku wymagają podania wskaźnika na tą strukturę. np. FILE *fp; //definicja zmiennej plikowej	W języku C++ operacje we/wy można realizować obiektowo. Podejście obiektowe zakłada, że różne „urządzenia” są reprezentowane w programie za pomocą różnych obiektów modelujących strumienie danych, które wpływają lub wypływają z tych „urządzeń”. np. fstream plik; //definicja zmiennej plikowej
Funkcje realizujące operacje na plikach	

Funkcje zawarte w bibliotece <stdio.h>

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

Rodzaj operacji:

- r - tylko do odczytu,
- w - tylko do zapisu,
- a - dopisywanie na końcu,
- + - z możliwością aktualizacji,
- b - otwarcie jako plik binarny,
- t - otwarcie jako plik tekstowy.

`int fclose ( FILE *strumien );` - zamknięcie strumienia

`int fcloseall (void );` - zamknięcie wszystkich strumieni

```
FILE *plik; // utworzenie pliku do odczytu
plik = fopen( "c:\\wyniki.txt", "r" );
if( plik == NULL ) // kontrola błędów
{
        printf( "Blad otwarcia pliku");
        return -1;
}
```

`int feof ( FILE *strumien );` - testowanie osiągnięcia końca pliku

`int fgetc ( FILE *strumien );` - wczytanie pojedynczego znaku

Funkcje zawarte w bibliotece <fstream.h>

`void open( char *nazwa_pliku, int tryb_otwarcia )` - otwarcie pliku,

Lista dostępnych trybów otwarcia strumienia:

- ios::**in** - otwarcie strumienia do **odczytu**,
- ios::**out** - otwarcie strumienia do **zapisu**,
- ios::**app** - otwarcie strumienia w trybie **dopisywania**,
- ios::**trunc** - wyzerowanie rozmiaru pliku, jeżeli plik istnieje,
- ios::**binary** - otwarcie strumienia jako **binarny**,

`void close( void )` - zamknięcie pliku skojarzonego z danym strumieniem

`int ios::bad( )` - zwraca wartość różną od zera, jeżeli pojawił się błąd,

`int ios::good( )` - zwraca wartość różną od zera, jeżeli nie wystąpił błąd,

`int ios::eof( );` - zwraca wartość różną od zera, gdy koniec pliku,

`int ios::width(int);` - szerokość pola wyjściowego (np.ilość cyfr)

`int ios::precision( int )` - steruje ilością cyfr po przecinku

`get( char& znak);` - wczytuje jeden znak ze strumienia,

`char* fgets ( char *tekst, int dlugosc, FILE *strumien );` - wczytanie łańcucha składającego się z co najwyżej (dlugosc-1) znaków `int fscanf ( FILE *strumien, char *format, ...);` - analogiczna do scanf `int fread ( void* adres, size_t rozm_bl, size_t il_blokow, FILE* strumien );` - funkcja odczytująca (ilosc_blokowrozmiar_bloku) bajtów	`getline(char* bufor, int max_dlug, char znak_konca);` - wczytuje linię/łańcuch znaków, >> - operator odczytu danych ze strumienia tekstowego. `read( char* bufor, int ilość_bajtów );` - wczytuje ciąg bajtów do bufora,
`int fputc ( int znak, FILE *strumien );` - wysłanie pojedynczego znaku `int fputs ( char *tekst, FILE *strumien );` - wysłanie łańcucha znaków `int fprintf ( FILE *strumien, char *format, . . . );` - funkcja analogiczna do printf `int fwrite ( void* adres, size_t rozm_bl, size_t il_blokow, FILE* strumien );` - funkcja kopiująca (ilosc_blokowrozmiar_bloku) bajtów spod wskazanego obszaru pamięci do strumienia (pliku)	**put**`( char& znak);` - wysyła jeden znak do strumienia, **<<** - operator wysłania/zapisu danych do strumienia tekstowego. `write(char* bufor, int ilość_bajtów);` - wysyła ciąg bajtów z bufora do strumienia
`int fseek ( FILE *strumien, long przesuniecie, int wzgledem);` - 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	`istream & seekg( streamoff offset, ios_base::seekdir kierunek);` - ustawia wewnętrzny wskaźnik pliku dla funkcji odczytujących dane `ostream & seekp( streamoff offset, ios_base::seekdir kierunek );` - ustawia wewnętrzny wskaźnik pliku dla funkcji zapisujących dane Stałe określające pozycję odniesienia (podczas przesuwania pozycji): · ios::**beg** - względem początku pliku, · ios::**cur** - względem pozycji aktualnej, · ios::**end** - względem końca pliku,

`long ftell ( FILE *strumien );` - zwraca aktualną pozycję wskaźnika pliku

`int fflush ( FILE *strumien );` - „czyści” bufor wskazanego strumienia

`int flushall ( void );` - czyści bufor dla wszystkich buforowanych strumieni

`streampos tellg();` - zwraca aktualną pozycję wewnętrznego wskaźnika pliku od której będzie następowało wczytywanie danych z pliku

`streampos tellp();` - zwraca aktualną pozycję wewnętrznego wskaźnika pliku od której będzie następowało zapisywanie danych do pliku.

Przykład (kopiowanie pliku z jednoczesną zamianą liter na małe)

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

int main()
{
        char c;
        FILE *fin, *fout;
        fin = fopen( "data.txt", "rt" );
        fout = fopen( "results.txt", "wt" );

        if( (fin!=NULL) && (fout!=NULL) ){
                while( !feof(fin) ){
                        c = fgetc(fin);
                        c = tolower(c);
                        fputc(c, fout);
                }
        }
        fclose( fin );
        fclose( fout );
        return 0;
}
```

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

using namespace std;

int main()
{
        char c;
        fstream fin, fout;
        fin.open( "data.txt", ios::in );
        fout.open( "results.txt", ios::out );

        if( fin.good( ) && fout.good( ) ){
                while( ! fin.eof( ) ){
                        fin.get( c );
                        c =tolower( c );
                        fout.put( c );
                }
        }
        fin.close( );
        fout.close( );
        return 0;
```

	}