

Vstup a výstup

Karel Richta a kol.

Katedra technických studií
Vysoká škola polytechnická Jihlava

© Karel Richta, 2020

Objektově-orientované programování, OOP 02/2020, Lekce 5

<https://moodle.vspj.cz/course/view.php?id=200875>



Vysoká škola
polytechnická
Jihlava

Vstup a výstup

- Je realizován pomocí standardních knihoven (<stdio.h> pro C a <cstdio> nebo <iostream> pro C++).
- Může být neformátovaný (binární) nebo formátovaný (textový).
- **Neformátovaný** vstup/výstup - reprezentace dat v paměti a na vstupu/výstupu je stejná. Čtecí/zápisové operace nezasahují do zpracovávané posloupnosti bytů (slabik).
- **Formátovaný** vstup/výstup - paměťová (binární) reprezentace je odlišná od textové reprezentace na vstupu/výstupu. Zápisové operace provádí konverzi z paměťové (binární) reprezentace na text. Čtecí operace převádějí textovou reprezentaci zpět na paměťovou binární.

Soubory

- Soubory představují informaci uloženou na vnějším médiu.
- Obsah souboru můžeme interpretovat binárně nebo textově.
- Při binárním zpracování interpretujeme obsah souboru jako posloupnost bytů, tj. hodnot v rozsahu 0..255 (`unsigned char`).
- Při textovém zpracování interpretujeme obsah souboru jako posloupnost znaků (`char`). V textových souborech interpretujeme jistou kombinaci bytů jako znak konce řádku. Tato kombinace závisí na systému ovládání souborů v operačním systému:
 - LF (`\n` = 0x0A) UNIX,
 - CR LF (`\r\n` = 0x0D 0x0A) Windows, DOS,
 - CR (`\r` = 0x0D) Mac.
- V paměti je konec řádku vždy LF (`'\n'`) :
 - C/C++ knihovna provádí automatickou konverzi na LF (při čtení nahradí posloupnost CR LF (CR) znakem LF, při zápisu nahradí znak LF znaky CR LF (CR) dle systému.

Příklad (celá čísla)

- Příklad neformátovaného vstupu/výstupu:

Hodnota	v paměti	v souboru
1	00 00 00 01 (4B)	00 00 00 01 (4B)
1000	00 00 03 E8 (4B)	00 00 03 E8 (4B)
1000000	00 0F 42 40 (4B)	00 0F 42 40 (4B)

- Příklad formátovaného vstupu/výstupu:

Hodnota	v paměti	v souboru (např.)
1	00 00 00 01	31 (1B)
1000	00 00 03 E8	31 30 30 30 (4B)
1000000	00 0F 42 40	46 34 32 34 30 (5B)

C++ vstup, výstup, datový proud

- Knihovny funkcí v C lze využívat i v C++ (C – `stdio.h` v C++ - `cstdio`).
- Navíc má C++ nové knihovny pro práci s tzv. datovými proudy (standard):
 - `iostream`,
 - `fstream`.
- **Datový proud** – zprostředkovatel přenosu (stream) - prostředník mezi programem a zdrojem/cílem.
- **Vstupní** – přicházejí data z klávesnice, ze souboru, z jiného programu:
 - Spojuje vstupní zařízení (zdroj) a proud (stream).
 - Spojuje proud s programem.
- **Výstupní** – odcházejí data na obrazovku, na tiskárnu, do souboru, do programu:
 - Spojuje program a proud (stream).
 - Spojuje proud a výstupní zařízení (cíl).
- Fyzická realizace pomocí vyrovnávací paměti (`streambuf`) - při výstupu je po zaplnění uvolněna, při libovolném příkazu vstupu je vyrovnávací paměť uvolněna.

Formátovaný výstup v C++

Knihovna `<iostream>` – operátor `<<`

- operátor `<<` připojí k výstupnímu proudu formátovaná data
- formátování je definováno pro základní typy
- pro uživatelské typy je třeba operátor `<<` přetížit novým významem
- konverze jsou automatické, není nutno je explicitně uvádět
- parametry pro konverze se nastavují pomocí manipulátorů.

Příklad:

```
int a = 10;  
double b = 2.5;  
const char * c = "Hello";  
cout << "a= " << a << " b= " << b << " c= " << c << endl;
```

Formátovaný vstup v C++

Knihovna `<iostream>` – operátor `>>`

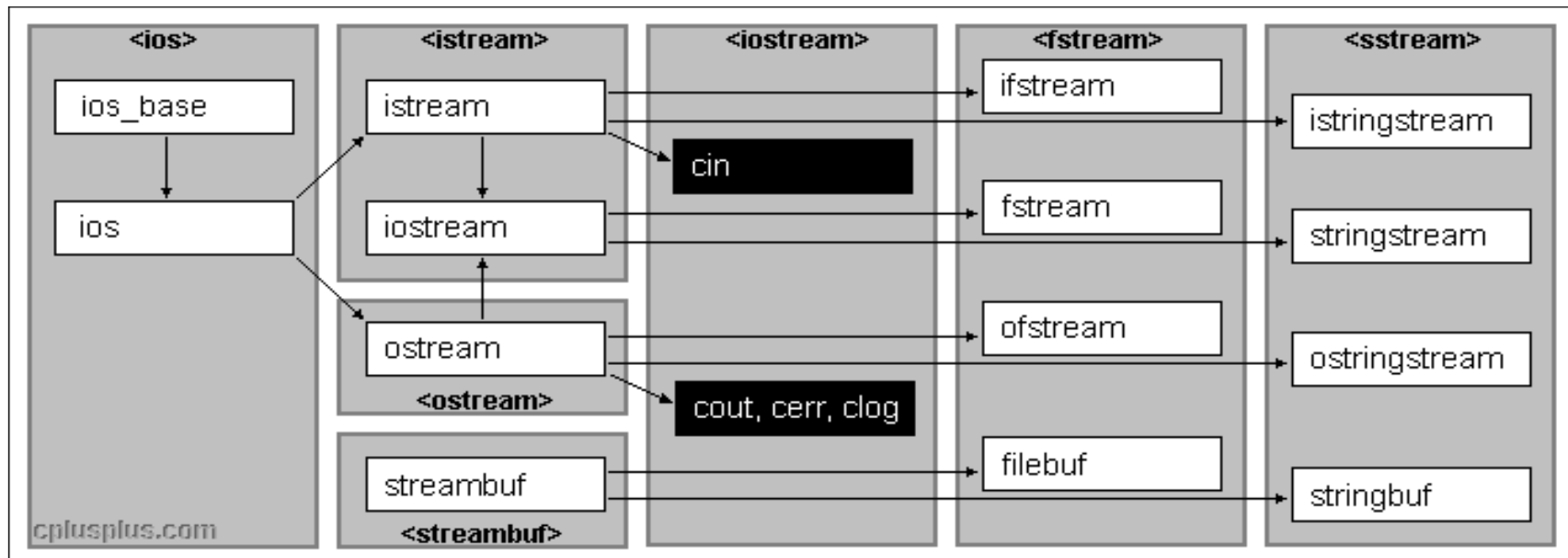
- operátor `>>` přečte ze vstupního proudu formátovaná data
- formátování je definováno pro základní typy
- pro uživatelské typy je třeba operátor `>>` přetížit novým významem
- konverze jsou automatické, není nutno je explicitně uvádět
- parametry pro konverze se nastavují pomocí manipulátorů.

Příklad:

```
int a;  
double b;  
char c[40];  
cin >> a >> b >> c;
```

C++ vstup, výstup, hierarchie tříd

- Abstraktní třída `ios_base` (vlastnosti proudu - otevřený, binární)
 - `ios` – ukazatel na streambuf
 - `istream` – metody vstupu + operator `>>`
 - `ostream` – metody výstupu + operator `<<`
 - `iostream` - `cin`, `cout`, `cerr` (bez vyr. paměti), `clog`.
- Objekt reprezentuje proud (řídí tok) – má atributy o proudu (čís. základ) + adresu vyrovnávací paměti.



Soubory mimo standardní prostředí

- metody: `open(jméno, mód)`, lze volat jako konstruktor; `close()`
- módy: textový (default) nebo binární
- operátory `>>` a `<<` pro formátovaný vstup/výstup (textové soubory)
- metody `get` a `put` pro formátovaný vstup a výstup (textové soubory)
- metody `read` a `write` pro binární soubory
- Příklad: zápis do textového souboru

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    ofstream soubor;
    soubor.open("priklad.txt");
    soubor << "Nějaký text" << endl;
    soubor.close();
    return 0;
}
```

Rozšíření třídy Trojúhelník

- Vytvořte metodu, která zapíše do souboru (název se předá jako parametr) – v textovém režimu – informace o trojúhelníku

Testování stavu souboru

Následující metody vracejí informace o stavu souboru (vracejí typ **bool**):

- **is_open()** soubor byl úspěšně otevřen
- **bad()** předchozí operace čtení nebo zápisu byla neúspěšná
- **fail()** totéž co **bad()**, navíc pokud došlo k formátovací chybě
- **eof()** pozice ve vstupním souboru je na konci souboru
- **good()** vše je O.K.
- **clear()** maže všechny příznaky

Objekt cout (standardní prostředí)

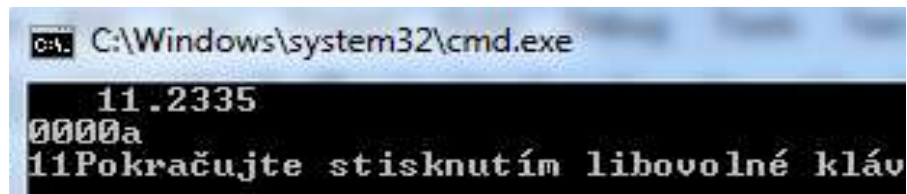
- Objekt třídy `ostream` – umí metody pro převod reprezentace vnitřní do znakové, převod číselných typů na proud znaků.
- `ostream` je přístupný ve jmenném prostoru `std`.
- Přetížený operátor vložení `<<` (připojení k proudu) existuje pro všechny základní datové typy a má profil:
 - `ostream & operator <<(type &)`
- Metody třídy `ostream`:
`cout.put('A') , cout.put(65);`
`cout.write(ret,3); // char *ret = "jak se mas";`
- Vyprázdnění po novém řádku, při očekávání nového vstupu, nebo pomocí `fflush(stdout)`.

Objekt cout - formátování

- Objekty float – implicitně 6 des. míst bez koncových 0.
- Třída ios_base definuje manipulátory pro formátování.
- Číselný základ – dec, hex, oct – volání hex (cout), cout << hex, platí do další změny.
- Členská metoda width – nastavuje šířku oblasti do které bude zapsán následující výstupní parametr. – volání cout.width(10).
- Nevyužité části oblasti vyplňuje mezerou – lze změnit – členskou metodou fill() – volání cout.fill('_'), platí než je změněn.
- Nastavení přesnosti float – počet číslic (implicitně 6), jinak – členská metoda precision() – volání cout.precision(4), platí než je změněna.

Objekt cout - formátování

```
int main() {  
    float f = 11.23345543;  
    int a = 10;  
    cout<<hex;  
    cout.width(10);  
    cout<<f;  
    cout<<endl;  
    cout.fill('0');  
    cout.width(5);  
    cout<<a;  
    cout.precision(2);  
    cout<<endl;  
    cout<<f;  
    return 0;  
}
```



```
C:\Windows\system32\cmd.exe  
11.2335  
0000a  
11Pokračujte stisknutím libovolné klávy
```

Třída `ios_base` - metoda `setf()`

- Obsahuje metodu `setf()` – nastavuje formátování.
- Obsahuje statické konstanty ,které se ovládají `setf()` – `cout.setf(maska)`.
- Konstanty nastavují bitové příznaky – maska.
- Přístup ke konstantám přes rozlišovací operátor a třídu `ios_base`
 - `ios_base::showpoint` – zobrazí des. oddělovač,
 - `ios_base::boolalpha` – bool zobrazí jako TRUE-FALSE,
 - `ios_base::showbase` – pro hexa zobrazí 0x před číslem,
 - `ios_base::uppercase` – pro hexa velká písmena,
 - `ios_base::showpos` – zobrazí plus před kladným číslem.
- Použití - `cout.setf(ios_base::showpoint);`
- Prototyp i s 2 argumenty - `setf(nastavení bit masky, čišění bitu masky);`
- `setf(ios_base::hex, ios_base::basefield)` – uživatelsky nevhodné.

Standardní manipulátory pro cout

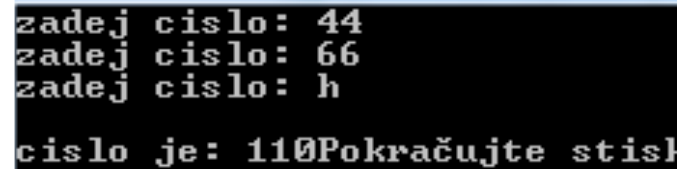
- `setf()` – jednodušší varianta jsou manipulátory – volají metodu i s konstantami, použití :
`cout.showpoint()`, `cout << noshowpoint()`:
 - `boolalpha`
 - `noboolalpha`
 - `showbase`
 - `noshowbase`
 - `showpoint`
 - `noshowpoint`
 - `showpos`
 - `noshowpos`
- **Hlavičkový soubor – `<iomanip>`:**
 - Lze použít jako manipulátory: `setprecision`, `setfill`, `setw`
 - `uppercase`
 - `nouppercase`
 - `left`
 - `right`
 - `dec`
 - `hex`
 - `oct`
 - `fixed`
 - `scientific`

Objekt cin (standardní prostředí)

- Objekt třídy `istream` – umí metody pro převod znakové reprezentace do vnitřní – binární reprezentace.
- Proud `istream` je přístupný ve jmenném prostoru `std`.
- Přetížený operátor získání `>>` (vyjmutí z proudu) existuje pro všechny základní datové typy a má profil:
 - `istream & operator >>(typ &)`
- Operátor `>>` přeskakuje bílé znaky v proudu, při zřetězení načítá až do prvního znaku neodpovídajícímu danému typu:
 - `cin >> cislo >> ret;`
- Chybové stavy proudu: `eofbit`, `failbit`, `badbit`.
- Užití: `cin.good()`, `cin.fail()`, `cin.eof()`.
- Vyčištění všech chybových stavů – `cin.clear()` - jinak nelze operátor `>>` použít.
- Extrakce znaků z proudu : `cin.ignore ()` – jeden znak:
 - `cin.ignore(100, ' ')` – klasicky `'\n'`

Ošetření vstupních hodnot

```
int main()
{
    int cislo,a;
    int sum = 0;
    while (1) {
        cout<<"zadej cislo: ";
        if (!(cin>>cislo))
            break;
        sum+=cislo;
    }
    cout<<"\ncislo je: "<<sum;
    system("pause");
    return 0;
}
```

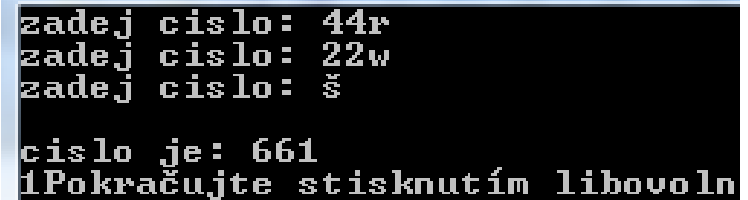


A screenshot of a terminal window showing the execution of the provided C++ code. The user enters three numbers: 44, 66, and h. The program calculates the sum of the first two numbers (110) and then displays the message "cislo je: 110Pokračujte stisk".

```
zadej cislo: 44
zadej cislo: 66
zadej cislo: h
cislo je: 110Pokračujte stisk
```

Ošetření vstupních hodnot

```
int main() {
    int cislo,a;
    int sum = 0;
    while (1) {
        cout<<"zadej cislo: ";
        cin>>cislo;
        cin.ignore(); // pro více znaků - ignore s parametrem
        if (cin.fail()){
            cin.clear();
            cin.ignore(); // pro více znaků ignore s parametrem
            break;
        }
        sum+=cislo;
    }
    cout<<"\ncislo je: "<<sum;
    cin>>a;
    cout<<a;
    system("pause");
    return 0;
}
```



```
zadej cislo: 44r
zadej cislo: 22w
zadej cislo: š
cislo je: 661
Pokračujte stisknutím libovoln
```

Metody iostream

Metoda get:

- `char k; cin.get(k);` jeden znak z proudu, vrací TRUE, když konec souboru vrací FALSE
- `cin.get();` vrací celočíselnou hodnotu, konec souboru EOF
- `istream & get(char *,int, char = '\n');` po načtení nechá '\n'
- `istream & getline(char *,int, char = '\n');` po načtení zruší z proudu '\n'
- `char ret[50]; cin.get(ret,50);`
- přečte-li se více – nastaveno failbit
- `peek()` – podívá se do proudu, ale neodebere znak – pouze na jeden znak
- `read(adr,100);` - čte z proudu 100 B

```
char k;  
while (cin.get(k)){ //telo }  
int k;  
while((k = cin.get())!=EOF){ //telo }
```

Vstup a výstup do souboru

- **Hlavičkový soubor (knihovna) fstream:**
 - ifstream – vstup ze souboru (dědí metody z istream),
 - ofstream – výstup do souboru (dědí metody z ostream),
 - fstream – vstup i výstup.
- **Zápis do souboru:**
 - Vytvořit objekt ofstream.
 - Inicializovat objekt souborem.
 - Používat metody pro zápis.
- **Čtení ze souboru:**
 - Vytvořit objekt ifstream.
 - Inicializovat objekt souborem.
 - Používat metody pro čtení.
- **Ukončení implicitně po skončení platnosti objektu/explicitně – metoda close().**

Vstup a výstup do souboru

Práce se soubory: `open` a `close` na stejném neinicializovaném objektu:

```
int main() {
    char *soubor ="heslo.txt"
    int a;
    ofstream ofile;
        //implicitní konstruktor
    ofile.open(soubor);
    ofile << 14;
    ofile.close();
    ifstream ifile(soubor);
    ifile>>a;
    ifile.close();
    cout<<a;
    return 0;
}
```

Používat metodu `is_open()` pro ověření otevření souboru:

```
int main() {
    char *soubor = "heslo.txt";
    int a;
    ifstream ifile;
    ifile.open(soubor);
    if (!ifile.is_open()) {
        cout << " soubor nelze otevrit";
    }
    else { ifile >> a; cout << a; }
    ifile.close();
    return 0;
}
```

Vstup a výstup do souboru

Lze použít metody:

- `get`, `getline`, `>>`, `read` – blokové čtení i s bílými znaky dle daného počtu,
- `put`, `write` – blokový zápis daného počtu.

Metody otevření souboru - 2 argument konstruktoru – kombinace položek:

- Typ otevření:
 - `ios::in` (implicitní pro `ifstream`)
 - `ios::out` (implicitní pro `ofstream`)
 - `ios::in | ios::out` (implicitní pro `fstream`)
- Způsob otevření:
 - `ios::binary` (binární přístup, implicitní je textový)
 - `ios::app` (přidání na konec, obsah se ponechá)
 - `ios::trunc` (pokud soubor existuje, tak se vymaže)
- Otevření souboru pro přidání:
 - `ofstream fout (file, ios::out | ios::app)`

Kopírování souboru

- Po znacích:

```
int main() {
    char c; // vyrovnací pamet na jeden znak
    ifstream in("text.txt");
    ofstream out("data.lst");
    cout << "Kopie textoveho souboru znak po znaku\n";
    if (!in.is_open()) {
        cout << "Nelze otevrit soubor text.txt\n";
        exit(1);
    }
    if (!out.is_open()) {
        cout << "Nelze vytvorit soubor data.lst\n";
        exit(1);
    }
    while (in.get(c)) //nelze použít >> - zruší mezery
        out.put(c);
    out.close();
    in.close();
    cout << "Konec" << endl;
    return 0;
}
```

- Po řádcích:

```
int main() {
    char radek[100]; // pamet na jeden radek
    while (in.getline(radek,100))
        out << radek << '\n';
    cout << "Konec" << endl;
    return 0;
}
```


Kopírování souborů – po znacích

```
#include <iostream>
#include <fstream>
using namespace std;

int main(int argc, char *argv[]) {
    if (argc != 3) {
        cerr << "Použití: " << argv[0] << " vstup výstup\n";
        return 1;
    }
    ifstream vstup(argv[1]);
    if (!vstup.is_open()) {
        cerr << "Vstupní soubor " << argv[1] << " nelze otevřít\n";
        return 2;
    }
    ofstream vystup(argv[2]);
    if (!vystup.is_open()) {
        cerr << "Výstupní soubor " << argv[2] << " nelze otevřít\n";
        return 3;
    }
    char buffer;
    buffer = vstup.get();
    while (!vstup.eof()) {
        vystup.put(buffer);
        buffer = vstup.get();
    }
    vstup.close();
    vystup.close();
    return 0;
}
```

Binární a textový režim

- Textový režim – vše jako text (i čísla) 1.23e07 => uložení 7 znaků.
- Binární režim – uložení jako 64b hodnota double.
- Převod zajišťuje operátor << .
- Binární uložení a čtení je rychlejší – nejsou konverze, po blocích.
- Textové uložení – přenositelné lépe mezi systémy.
- Binární – pro přenos se musí napsat aplikace, která rozumí vnitřní reprezentaci.
- Příklad pro Osobu:

```
class Osoba{
private:
    string jmeno;
    int vek;
public:
    Osoba(){}
    int getVek() { return vek; }
    string getJmeno() { return jmeno; }
    Osoba(string jm, int ve) : vek(ve) { jmeno = jm; }
    void tisk() { cout << "/n" << jmeno << " " << vek; }
};
```

Hlavní program v textovém režimu

```
int main() {
    char c;
    char jmeno[10];
    int vek;
    Osoba poleout[] = {Osoba("Jan",10),Osoba("Eva",50),Osoba("Jirka",25)};
    Osoba polein[3];
    ofstream out("text.txt");
    for (int i = 0; i<3; i++)
        out << poleout[i].getJmeno() << " " << poleout[i].getVek() << endl;
    out.close();
    ifstream in("text.txt",ios::in);
    int j = 0;
    while (in >> jmeno >> vek) { polein[j] = Osoba(jmeno,vek); j++; }
    polein[0].tisk(); polein[1].tisk(); polein[2].tisk();
    return 0;
}
```

Hlavní program v binárním režimu

```
int main() {
    Osoba poleout[] = {Osoba("Jan",10),Osoba("Eva",50),Osoba("Jirka",25)};
    Osoba pom;
    Osoba polein[3];
    ofstream out("text.dat",ios::out|ios::binary);
    for (int i = 0; i<3; i++)
        out.write((char*)&poleout[i],sizeof (poleout[i]));
    out.close();
    ifstream in("text.dat",ios::in|ios::binary);
    int j = 0;
    while (in.read((char*)&pom,sizeof(pom))) {
        polein[j] = Osoba(pom.getJmeno(),pom.getVek());
        j++;
    }
    polein[0].tisk();
    polein[1].tisk();
    polein[2].tisk();
    return 0;
}
```

Binární vstup a výstup v C++

Neformátovaný vstup a výstup (binární) se provádí pomocí funkcí **read**, **write**

```
#include <iostream>
#include <fstream>
using namespace std;

int main(int argc, char *argv[]) {
    // kontrola parametrů - if (argc != 3) ...
    ifstream in(argv[1],ios::binary);           // nutno nastavit binární mód pro vstup
    ofstream out(argv[2],ios::binary);          // nutno nastavit binární mód pro výstup
    char buffer[1024];                          // vyrovnávací paměť
    int pocet = 0;                              // počet zkopírovaných bloků a posléze bytů
    while (in && out)                            // dokud není chyba na vstupu ani na výstupu
    {
        in.read(buffer,1024);                   // pokus se přečíst blok velikosti 1024
        out.write(buffer,in.gcount());           // zapiš počet skutečně přečtených bytů (gcount) na výstup
        pocet++;                                // přidej blok
    }
    --pocet *= 1024;                            // přepočti bloky na byty až na poslední
    pocet += in.gcount();                       // doplň délku posledního bloku
    cout << "Počet zkopírovaných bytů: " << pocet << endl;
    return 0;
}
```

Kopírování souborů binárně

```
#include <iostream>
#include <fstream>
using namespace std;
#define SIZE 1024

int main(int argc, char *argv[]) {
    if (argc != 3) {
        cerr << "Použití: " << argv[0] << " vstup výstup\n";
        return 1;
    }
    ifstream vstup(argv[1],"rb");
    if (!vstup.is_open()) {
        cerr << "Vstupní soubor " << argv[1] << " nelze otevřít\n";
        return 2;
    }
    ofstream vystup(argv[2],"wb");
    if (!vystup.is_open()) {
        cerr << "Výstupní soubor " << argv[2] << " nelze otevřít\n";
        return 3;
    }
    unsigned char buffer[SIZE];
    int lng = SIZE;
    while (lng = read(buffer,1,SIZE,vstup)) write(buffer,1,lng,vystup);
    vstup.close();
    vystup.close();
    return 0;
}
```

Přímý přístup k souboru

- **Pozice a posun v souboru:**
 - Odkud a kam se bude zapisovat?
 - „Get pointer“ – místo, odkud se bude číst.
 - „Put pointer“ – místo, kam se bude zapisovat.
- **tellg(), tellp()** – získání pozice get/put ukazatele (v C jen **ftell()**).
- **Nastavení pozice pro get (ukazatele pro vstup) seekg (v C jen fseek()):**
 - seekg(offset, pozice_odkud);
 - ios::beg (začátek souboru), ios::cur (aktuální pozice),
 - ios::end (konec souboru).
- **Nastavení pozice pro put (ukazatele pro výstup) seekp:**
 - seekp(offset, pozice_odkud);
- Počáteční pozice ovlivněna módem otevření.

Přímý přístup – délka souboru

```
#include <iostream>
#include <fstream>
#include <stdlib.h>
using namespace std;

int main() {
    ifstream inp("inputFile.txt", ifstream::binary );
    long delka = 0;
    if (inp != NULL) {
        inp.seekg(0, inp.end);
        delka = (long)inp.tellg();
        cout << "Velikost je " << delka << endl;
        inp.close();
    }
    else
        cout << "Soubor se nepodarilo otevrit\n";
    system("pause");
    return 0;
}
```


Ukládání informace do souborů (příklad Menu)

- Chceme aktuální menu uschovat pro příští použití.
- Persistentní úschova znamená uložit aktuální stav menu na nějaké vnější médium – pravděpodobně do nějakého souboru, který přežije ukončení běhu aplikace.
- Pokud má mít taková úschova smysl, musí být možnost menu ze souboru také obnovit.
- Reprezentace menu na vnějším médiu může být rozmanitá, záleží na volbě vývojáře.
- Volby:
 - Binární versus textová reprezentace?
 - Koho pověříme ukládáním a čtením této reprezentace?
 - Jak bude reprezentace strukturována?
 - Jaký formát použijeme (XML, EDI, ...)?
 - Zatím použijeme nejjednodušší formu – text (výhoda – snadná přenosnost, nevýhoda – nelze využít obecné nástroje).

Příklad Menu (pokr.)

```
/* Menu.h - zobrazí volby a přečte volbu */
#ifndef __MENU_H
#define __MENU_H

class Menu {
    static const int MAX_VOLEB = 10;
    string nadpis;
    Volba *volby[MAX_VOLEB];
    int pocetVoleb;
public:
    Menu(string);
    ~Menu();
    bool pridejVolbu(string);
    int vyberVolbu();
    bool ctiSoubor(string); // pridame nacteni ze souboru
    bool pisSoubor(string); // pridame zapis do souboru
};

#endif
```

Načtení menu ze souboru

```
/* Menu.cpp - nacteni menu ze souboru */
#include "menu.h"

bool Menu::ctiSoubor(string jmeno) {
    ifstream vstup;
    char radek[80];

    vstup.open(jmeno, ios::in);
    vstup.getline(radek, 80);
    nadpis = radek;
    pocetVoleb = 0;
    this->pridejVolbu("Konec");
    do {
        vstup.getline(radek, 80);
        this->pridejVolbu(radek);
    } while (!vstup.fail());
    vstup.close();
    return true;
};
```

Vnější reprezentace menu
(textový soubor):

1.řádek = nadpis
2.řádek = 1.volba
3.řádek = 2.volba
4.řádek = 3.volba
...
n.řádek = n-tá volba

Zápis menu do souboru

```
/* Menu.cpp - zapis menu do souboru */
#include "menu.h"

bool Menu::pisSoubor(string jmeno) {
    ofstream vystup;
    char radek[80];

    vystup.open(jmeno, ios::out);
    vystup << nadpis << endl;
    // volbu 0 neukladame
    for (int i=1; i<pocetVoleb; i++)
        vystup << volby[i]->text << endl;
    vystup.close();

    return true;
};
```

Příklad vnější reprezentace menu (textový soubor):

Menu pro modifikaci menu
Načtení menu
Zápis menu
Přidání nové volby
Zrušení existující volby
Oprava existující volby
Zobrazení aktuálního menu

Použití menu uloženého v souboru

```
/* MenuTest.cpp – pouziti menu */
#include "menu.h"

int main() {
    Menu *menuTest;
    char jmeno[81];

    cout << "Cteni menu ze souboru" << endl;
    cout << "Zadej nazev souboru: "; cin.get(c); cin.getline(jmeno,80,'\n');
    if (menuTest->ctiSoubor(jmeno) == false)
        cout << "Chyba: Nelze precist menu ze souboru " << jmeno << endl;

        . . . // práce podle precteneho menu

    cout << "Zapis menu do souboru" << endl;
    cout << "Zadej nazev souboru: "; cin.get(c); cin.getline(jmeno,80,'\n');
    if (menuTest->pisSoubor(jmeno) == false)
        cout << "Chyba: Nelze zapsat menu do souboru " << jmeno << endl;

    system("PAUSE");
    return 0;
}
```

The End