## 12. Program set 9 – file handling

## 1) Write text (truncate / create)

#include <fstream>

#include <iostream>

int main() {

std::ofstream out("out1.txt", std::ios::out | std::ios::trunc);

if (!out) {

std::cerr << "open fail\n";

return 1;

}

out << "Hello file!\n";

}

## 2) Append text (preserve existing)

#include <fstream>

#include <iostream>

int main() {

std::ofstream out("out1.txt", std::ios::out | std::ios::app);

if (!out) {

std::cerr << "open fail\n";

return 1;

}

out << "Appending a new line.\n";

}

## 3) Read text with operator>>

#include <fstream>

#include <iostream>

#include <string>

int main() {

std::ifstream in("out1.txt"); // default std::ios::in

if (!in) {

std::cerr << "open fail\n";

return 1;

}

std::string word;

while (in >> word)

std::cout << word << "\n";

}

## 4) Read text line-by-line

#include <fstream>

#include <iostream>

#include <string>

int main() {

std::ifstream in("out1.txt");

if (!in) {

std::cerr << "open fail\n";

return 1;

}

std::string line;

while (std::getline(in, line))

std::cout << line << "\n";

}

## 5) Copy text file (fstream in/out)

#include <fstream>

#include <iostream>

#include <string>

int main() {

std::ifstream in("out1.txt");

if (!in) {

std::cerr << "open source file fail\n";

return 1;

}

std::ofstream out("dst.txt");

if (!out) {

std::cerr << "open dst fail\n";

return 1;

}

std::string line;

while (std::getline(in, line))

out << line << "\n";

}

## 6) Read/write in one stream (update text)

#include <fstream>

#include <iostream>

#include <string>

int main() {

std::fstream f("up6.txt", std::ios::in | std::ios::out | std::ios::trunc);

if (!f) {

std::cerr << "open fail\n";

return 1;

}

f << "Line A\nLine B\n";

f.flush();

f.seekg(0);

std::string s;

while (std::getline(f, s))

std::cout << s << "\n";

}

## 7) Binary write of array

#include <fstream>

#include <iostream>

int main() {

int a[5] = {1,2,3,4,5};

std::ofstream out("data7.bin", std::ios::binary);

if (!out) {

std::cerr << "open fail\n";

return 1;

}

out.write(reinterpret\_cast<const char\*>(a), sizeof(a));

}

## 8) Binary read of array

#include <fstream>

#include <iostream>

int main() {

int a[5] = {0};

std::ifstream in("data7.bin", std::ios::binary);

if (!in) {

std::cerr << "open fail\n";

return 1;

}

in.read(reinterpret\_cast<char\*>(a), sizeof(a));

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

std::cout << a[i] << " ";

std::cout << "\n";

}

## 9) Append binary record

#include <fstream>

#include <iostream>

struct Rec {

int id;

double x;

};

int main() {

Rec r = {42, 3.14};

std::ofstream out("log9.bin", std::ios::binary | std::ios::app);

if (!out) {

std::cerr << "open fail\n";

return 1;

}

out.write(reinterpret\_cast<const char\*>(&r), sizeof(r));

}

## 10) Check state: eof/fail/bad

#include <fstream>

#include <iostream>

#include <string>

int main() {

std::ifstream in("dst.txt");

if (!in) {

std::cerr << "open fail\n";

return 1;

}

std::string s;

while (true) {

if (!(in >> s))

break;

std::cout << s << "\n";

}

std::cerr << "eof=" << in.eof() << " fail=" << in.fail() <<

" bad=" << in.bad() << "\n";

}

## 11) tellg/seekg on text (positions)

#include <fstream>

#include <iostream>

#include <string>

int main() {

std::ifstream in("dst.txt");

if (!in) {

std::cerr << "open fail\n";

return 1;

}

in.seekg(3);

std::streampos p0 = in.tellg();

std::string first;

std::getline(in, first);

std::cout << "First line: " << first << "\n";

in.seekg(p0);

std::string again;

std::getline(in, again);

std::cout << "Again: " << again << "\n";

}

## 12) tellp/seekp on text (overwrite start)

#include <fstream>

#include <iostream>

int main() {

std::ofstream out("dst1.txt", std::ios::out | std::ios::trunc);

if (!out) {

std::cerr << "open fail\n";

return 1;

}

out << "XXXXXXXXXX\n";

out.seekp(0);

out << "HELLO";

}

## 13) Random access in binary (fixed-size records)

#include <fstream>

#include <iostream>

struct Rec {

int id;

double x;

};

int main() {

std::fstream f("db13.bin", std::ios::in | std::ios::out | std::ios::binary | std::ios::trunc);

if (!f) {

std::cerr << "open fail\n";

return 1;

}

Rec r[3] = {{1,1.1},{2,2.2},{3,3.3}};

f.write(reinterpret\_cast<const char\*>(r), sizeof(r));

// update record #2 (index 1)

Rec upd = {2, 9.9};

f.seekp(sizeof(Rec) \* 1, std::ios::beg);

f.write(reinterpret\_cast<const char\*>(&upd), sizeof(upd));

}

## 14) getline with custom delimiter

#include <fstream>

#include <iostream>

#include <string>

int main() {

std::ifstream in("username.csv");

if (!in) {

std::cerr << "open fail\n";

return 1;

}

std::string field;

while (std::getline(in, field, ','))

std::cout << "[" << field << "]\n";

}

## 15) ignore / peek / get / putback

#include <fstream>

#include <iostream>

int main() {

std::ifstream in("dst.txt");

if (!in) {

std::cerr << "open fail\n";

return 1;

}

int c = in.peek(); // look

if (c != EOF)

std::cout << "peek:" << char(c) << "\n";

in.ignore(1); // skip one

char ch;

if (in.get(ch))

std::cout << "get:" << ch << "\n";

in.putback(ch); // return the char

if (in.get(ch))

std::cout << "get again:" << ch << "\n";

}

## 16) Count lines/words/chars (text)

#include <fstream>

#include <iostream>

#include <string>

#include <sstream>

int main() {

std::ifstream in("dst.txt");

if (!in) {

std::cerr << "open fail\n";

return 1;

}

std::string line;

long lines=0, words=0, chars=0;

while (std::getline(in, line)) {

++lines;

chars += line.size() + 1;

std::string w;

std::istringstream iss(line);

while (iss >> w)

++words;

}

std::cout << "L="<<lines<<" W="<<words<<" C~="<<chars<<"\n";

}

## 

## 17) Simple text filter (uppercase copy)

#include <fstream>

#include <iostream>

#include <string>

#include <cctype>

int main() {

std::ifstream in("src.txt");

std::ofstream out("dst17.txt");

if (!in || !out) {

std::cerr << "open fail\n";

return 1;

}

char ch;

while (in.get(ch))

out << char(std::toupper((unsigned char)ch));

}

## 18) Merge two sorted text files into one

#include <fstream>

#include <iostream>

#include <string>

int main() {

std::ifstream a("src.txt"), b("dst.txt");

std::ofstream out("m18.txt");

if (!a || !b || !out) {

std::cerr << "open fail\n";

return 1;

}

std::string s, t;

bool ha = bool(std::getline(a, s)), hb = bool(std::getline(b, t));

while (ha || hb) {

if (!hb || (ha && s <= t)) {

out << s << "\n";

ha = bool(std::getline(a, s));

}

else {

out << t << "\n";

hb = bool(std::getline(b, t));

}

}

}

## 19) CSV parse (very simple, split by comma)

#include <fstream>

#include <iostream>

#include <vector>

#include <string>

int main() {

std::ifstream in("username.csv");

if (!in) {

std::cerr << "open fail\n";

return 1;

}

std::string line;

while (std::getline(in, line)) {

std::vector<std::string> fields;

std::string cur;

for (std::size\_t i=0;i<line.size();++i) {

if (line[i]==','){

fields.push\_back(cur);

cur.clear();

}

else

cur += line[i];

}

fields.push\_back(cur);

for (std::size\_t i=0;i<fields.size();++i)

std::cout << "[" << fields[i] << "]";

std::cout << "\n";

}

}

## 20) Check is\_open and close()

#include <fstream>

#include <iostream>

int main() {

std::ofstream out;

out.open("dst20.txt");

if (!out.is\_open()) {

std::cerr << "open fail\n";

return 1;

}

out << "OK\n";

out.close();

if (out.is\_open()) std::cerr << "still open?\n";

}

## 21) Binary file size via seek/tell

#include <fstream>

#include <iostream>

int main() {

std::ifstream in("data7.bin", std::ios::binary);

if (!in) {

std::cerr << "open fail\n";

return 1;

}

in.seekg(0, std::ios::end);

std::streampos sz = in.tellg();

std::cout << "bytes=" << sz << "\n";

}

## 22) Read fixed-size chunks (buffered copy)

#include <fstream>

#include <iostream>

int main() {

std::ifstream in("big22.bin", std::ios::binary);

std::ofstream out("copy22.bin", std::ios::binary);

if (!in || !out) {

std::cerr << "open fail\n";

return 1;

}

const std::size\_t BUFSZ = 4096;

char buf[BUFSZ];

while (in) {

in.read(buf, BUFSZ);

std::streamsize n = in.gcount();

if (n > 0)

out.write(buf, n);

}

}

## 23) Safe open with mode flags combined

#include <fstream>

#include <iostream>

int main() {

std::fstream f("combo23.txt",

std::ios::in | std::ios::out | std::ios::app);

if (!f) {

std::cerr << "open fail\n";

return 1;

}

f << "Appended line\n";

}

## 24) Create if missing, then read (open twice)

#include <fstream>

#include <iostream>

#include <string>

int main() {

{

std::ofstream out("maybe24.txt", std::ios::app);

} // ensure exists

std::ifstream in("maybe24.txt");

if (!in) {

std::cerr << "open fail\n";

return 1;

}

std::string s;

while (std::getline(in, s))

std::cout << s << "\n";

}

## 25) Read numbers, compute sum/avg

#include <fstream>

#include <iostream>

int main() {

std::ifstream in("nums25.txt");

if (!in) {

std::cerr << "open fail\n";

return 1;

}

long long sum = 0;

long long n = 0;

long long x;

while (in >> x) {

sum += x; ++n;

}

if (n)

std::cout << "sum="<<sum<<" avg="<<(double)sum/n<<"\n";

}

## 26) Write then reopen and append

#include <fstream>

#include <iostream>

int main() {

{

std::ofstream a("log26.txt");

a << "First\n";

}

{

std::ofstream b("log26.txt", std::ios::app);

b << "Second\n";

} // append

}

## 27) Replace a line at known offset (text demo)

#include <fstream>

#include <iostream>

int main() {

std::fstream f("x27.txt", std::ios::in | std::ios::out | std::ios::trunc);

if (!f) {

std::cerr << "open fail\n";

return 1;

}

f << "ABCDEFGHIJ\n";

f.seekp(3); // overwrite from 4th char

f << "xyz";

}

## 28) Binary struct array: read record n

#include <fstream>

#include <iostream>

struct Rec {

int id;

double x;

};

int main() {

std::ifstream in("db13.bin", std::ios::binary);

if (!in) {

std::cerr << "open fail\n";

return 1;

}

Rec r;

std::size\_t n = 2; // read 3rd (0-based)

in.seekg(sizeof(Rec)\*n, std::ios::beg);

if (in.read(reinterpret\_cast<char\*>(&r), sizeof(r)))

std::cout << "id="<<r.id<<" x="<<r.x<<"\n";

}

## 29) Split file into two (odd/even lines)

#include <fstream>

#include <iostream>

#include <string>

int main() {

std::ifstream in("nums25.txt");

std::ofstream odd("odd29.txt"), even("even29.txt");

if (!in || !odd || !even) {

std::cerr << "open fail\n";

return 1;

}

std::string line;

long i=0;

while (std::getline(in, line)) {

(++i % 2 ? odd : even) << line << "\n";

}

}

## 30) Simple log rotate (rename via copy)

#include <fstream>

#include <iostream>

#include <string>

int main() {

// copy current log to backup, then truncate current

{ std::ifstream in("log26.txt");

std::ofstream out("app30.bak");

if (in && out) {

std::string line;

while (std::getline(in, line)) out << line << "\n";

}

}

{ std::ofstream trunc("app30.log", std::ios::trunc);

if (trunc) trunc << "Log rotated.\n";

}

}