MỐI LIÊN HỆ & LẬP TRÌNH TẬP TIN

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

GVHD: Trương Toàn Thịnh

NỘI DUNG

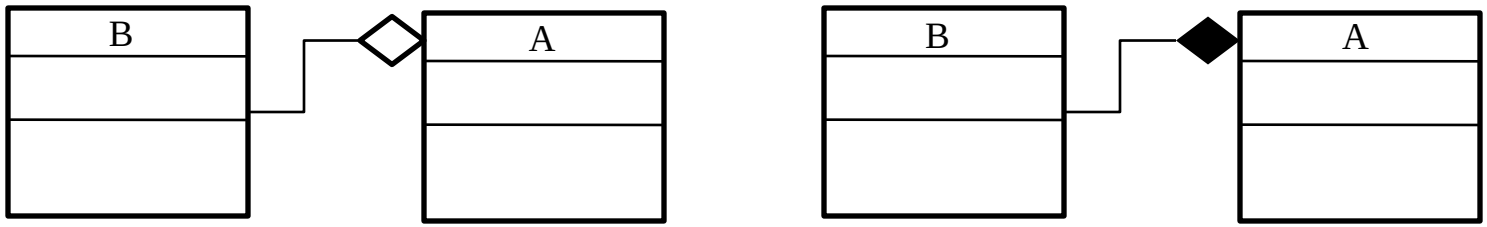
- Các mối liên hệ đối tượng
- Bàn luận về tính chất kế thừa
- Hệ thống nhập/xuất trong C++
- Lập trình trên tập tin
  - Đọc/ghi tập tin
  - Đọc/ghi trên cùng tập tin
- Lập trình tập tin văn bản dạng mở rộng

CÁC MỐI LIÊN HỆ ĐỐI TƯỢNG

- Ngoài dữ liệu & phương thức, mối liên hệ giữa các đối tượng cũng rất quan trọng
- Có các loại chính
  - Bao hàm/bộ phận (has-a, part-of)
    - Phụ thuộc
    - Độc lập
  - Tổng quát/đặc biệt (is-a)
  - Phụ thuộc
  - Bạn

CÁC MỐI LIÊN HỆ ĐỐI TƯỢNG

- Bao hàm/bộ phận (has-a, part-of)
  - Lớp A chứa đối tượng thuộc lớp B
  - Nếu đối tượng A bị hủy và các đối tượng B trong A cũng bị hủy → bao hàm phụ thuộc (composition)
  - Nếu đối tượng A bị hủy và các đối tượng B trong A không bị hủy → bao hàm độc lập (aggregation)

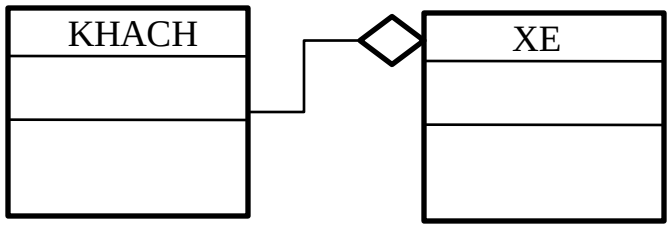


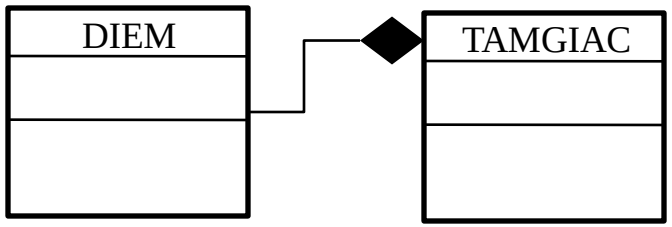
CÁC MỐI LIÊN HỆ ĐỐI TƯỢNG

- Cài đặt quan hệ bao hàm/bộ phận

```
class Xe {
  private:
    Khach* arr;
    int n;
  public:
    Xe();
    ~Xe();
};
```

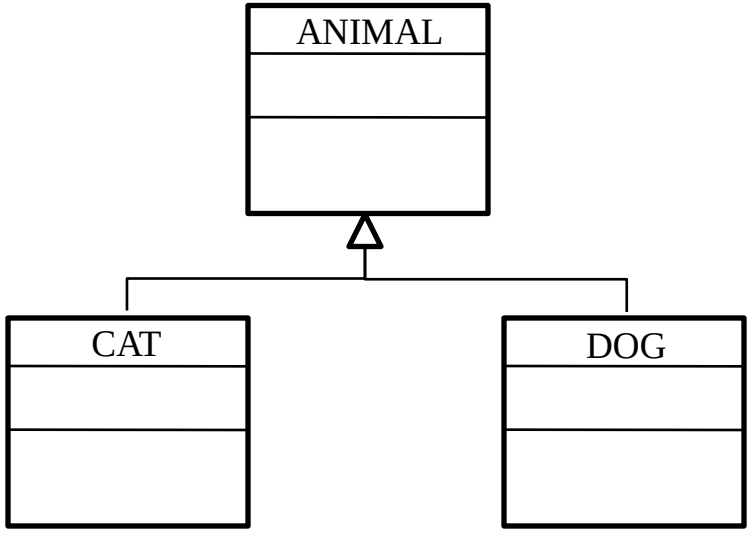
```
class DIEM{
  private:
    double x, y;
    //…
};
class TAMGIAC{
  private:
    DIEM arr[3];
  //…
}
```





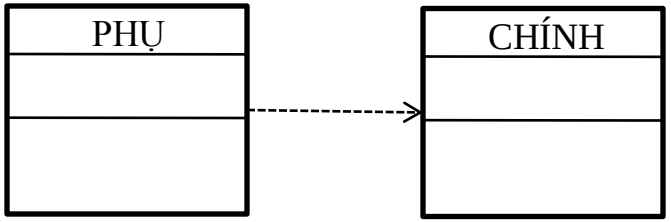
CÁC MỐI LIÊN HỆ ĐỐI TƯỢNG

- Đặc biệt/tổng quát (is-a)
  - Lớp CAT & DOG kế thừa ANIMAL
  - CAT & DOG **là** ANIMAL
  - **Chỉ dùng** public để cài đặt quan hệ kế thừa



CÁC MỐI LIÊN HỆ ĐỐI TƯỢNG

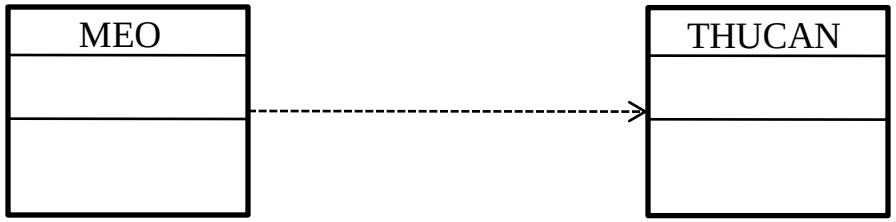
- Quan hệ phụ thuộc
  - Thay đổi trên lớp chính ảnh hưởng tới lớp phụ
  - Thay đổi trên lớp phụ không ảnh hưởng tới lớp chính
  - Một số tình huống nhận biết
    - Lớp phụ dùng biến toàn cục/cục bộ thuộc lớp chính
    - Lớp phụ chứa phương thức có tham số thuộc lớp chính
    - Lớp phụ gửi thông điệp tới lớp chính



CÁC MỐI LIÊN HỆ ĐỐI TƯỢNG

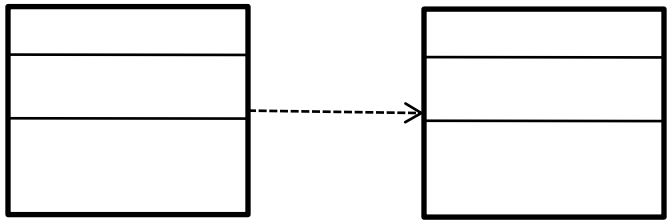
- Cài đặt quan hệ phụ thuộc

```
class THUCAN{
  private:
    double khoiluong;
  public:
    double layKhoiLuong(){return khoiluong;}
};
class MEO{
  private:
    double nangluong;
  public:
    void An(const THUCAN& thucan){
      nangluong+=thucan.layKhoiLuong()*10;
    }
}
```



CÁC MỐI LIÊN HỆ ĐỐI TƯỢNG

- Quan hệ bạn
  - Cho phép lớp bạn truy xuất đến toàn bộ nội dung, kể cả trường private



  - Cài đặt quan hệ bạn

```
class A{
  public:
    friend class B;
  private:
    int a;
};
```

```
class B{
  private:
    A test;
  public:
    void TruyXuat(){test.a = 7;}
}
```

NỘI DUNG

- Các mối liên hệ đối tượng
- Bàn luận về tính chất kế thừa
- Hệ thống nhập/xuất trong C++
- Lập trình trên tập tin
  - Đọc/ghi tập tin
  - Đọc/ghi trên cùng tập tin
- Lập trình tập tin văn bản dạng mở rộng

BÀN LUẬN VỀ TÍNH CHẤT KẾ THỪA

- Ngoài kế thừa public, còn có kế thừa private & protected

```
class A{
  public:
    //…
  private:
    //…
}
class B: private A{
  public:
    //…
  private:
    //…
}
```

```
class A{
  public:
    //…
  private:
    //…
}
class B: protected A{
  public:
    //…
  private:
    //…
}
```

BÀN LUẬN VỀ TÍNH CHẤT KẾ THỪA

- **Kế thừa private** thường dùng để cài đặt mối quan hệ **bao hàm/bộ phận**.
- Lớp con kế thừa kiểu private có thể dùng các trường public/protected trong lớp cha
- Lớp con kế thừa kiểu private có thể viết chồng các phương thức virtual trong lớp cha
- Lập trình viên không thích dùng kế thừa private để cài đặt mối quan hệ bao hàm/bộ phận.

BÀN LUẬN VỀ TÍNH CHẤT KẾ THỪA

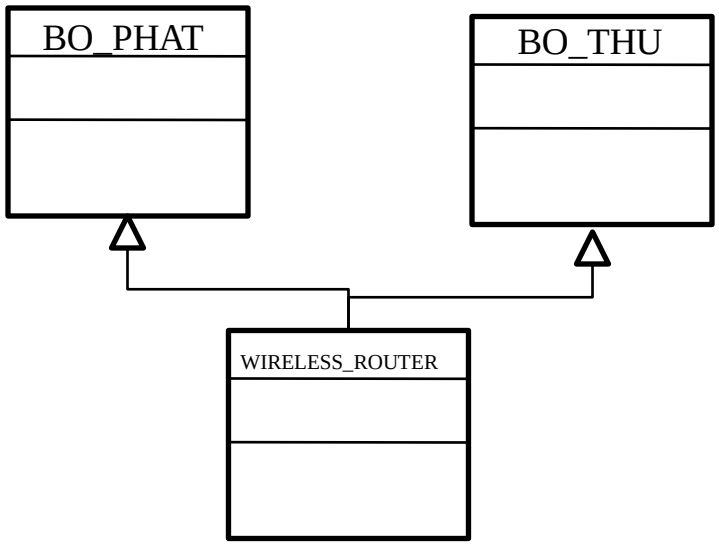
- Ví dụ:

```
class List{
  public:
    //…
  private:
    //…
}
class NhaGiuXe: private List{
  public:
    //…
  private:
    //…
}
```

- Các phương thức **public/protected trong List** sẽ thành **private trong NhaGiuXe** & không được phép truy xuất từ bên ngoài

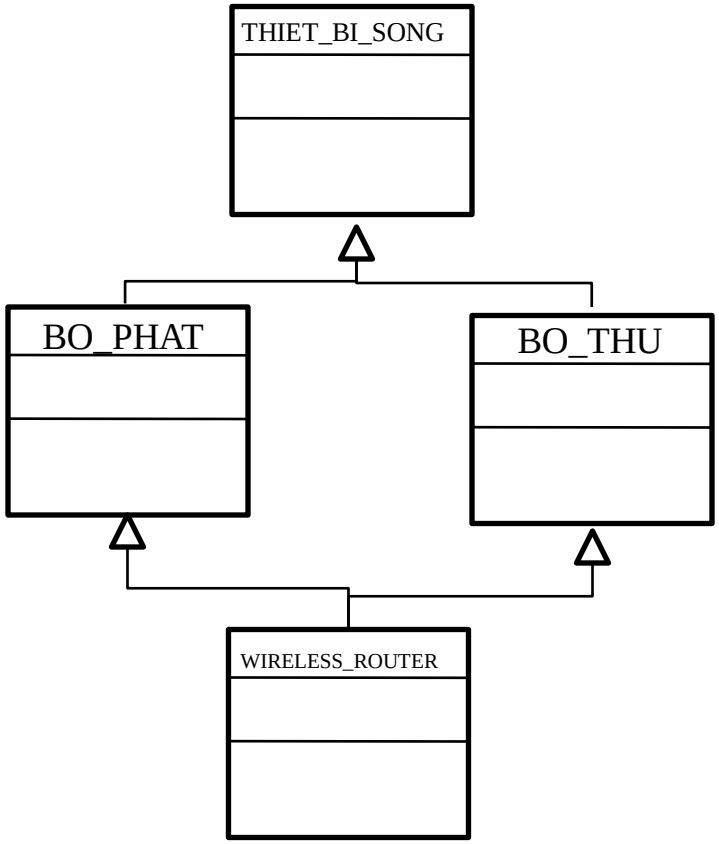
BÀN LUẬN VỀ TÍNH CHẤT KẾ THỪA

- Có tình huống đa kế thừa trong thực tế
- Xét lớp wireless_router, dễ thấy trong thực tế đối tượng wireless_router có thể vừa là thiết bị phát sóng hoặc là thiết bị thu sóng



BÀN LUẬN VỀ TÍNH CHẤT KẾ THỪA

- Thiết bị phát sóng hoặc thiết bị thu sóng đều là thiết bị sóng



BÀN LUẬN VỀ TÍNH CHẤT KẾ THỪA

- Vấn đề: lớp wireless router có hai đối tượng cơ sở đều thuộc kiểu THIET_BI_SONG → nhập nhằng khi biên dịch
- Giải pháp: dùng kế thừa ảo để gom hai lớp thành một

```
class BoPhat: public virtual ThietBiSong{
  //…
}
class BoThu: public virtual ThietBiSong{
  //…
}
class WirelessRouter: public BoPhat, public BoThu{
  //…
}
```

NỘI DUNG

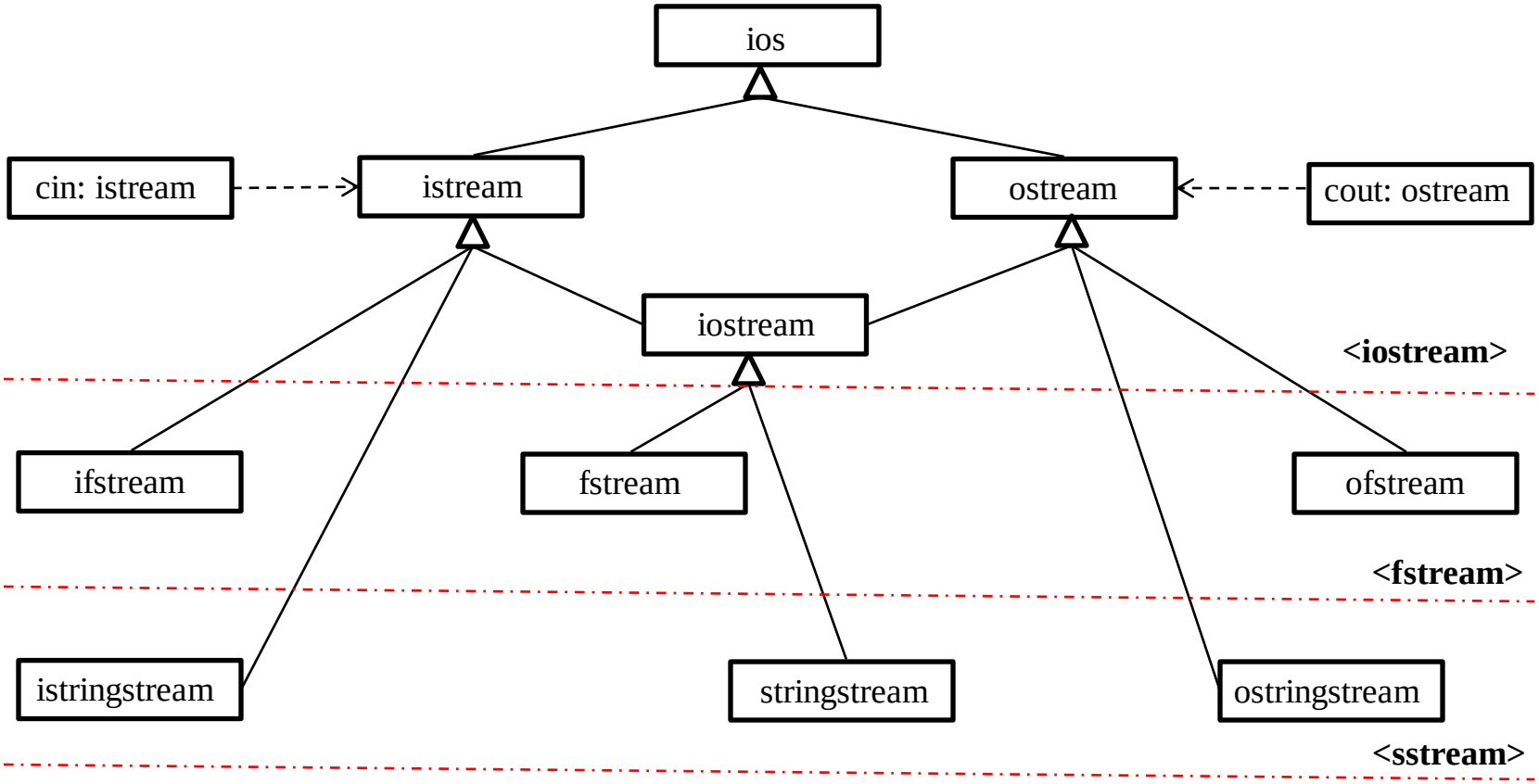
- Các mối liên hệ đối tượng
- Bàn luận về tính chất kế thừa
- Hệ thống nhập/xuất trong C++
- Lập trình trên tập tin
  - Đọc/ghi tập tin
  - Đọc/ghi trên cùng tập tin
- Lập trình tập tin văn bản dạng mở rộng

HỆ THỐNG NHẬP/XUẤT TRONG C++

- Thực hiện nhập/xuất thông qua đối tượng luồng dữ liệu (data stream)
- Các đối tượng nhập/xuất đại diện cho các thiết bị nhập/xuất vật lý như bàn phím, màn hình
- Hệ thống Nhập/xuất của C++ gồm hai phiên bản:
  - Phiên bản pre-standard C++
  - Phiên bản ANSI/ISO standard C++

HỆ THỐNG NHẬP/XUẤT TRONG C++

- Cây kế thừa



HỆ THỐNG NHẬP/XUẤT TRONG C++

- Toán tử ‘<<’ để xuất dữ liệu ra đối tượng có kiểu ostream hay lớp con của nó
- Toán tử ‘>>’ để nhập/nhận dữ liệu ra đối tượng kiểu istream hay lớp con của nó
- C++ I/O định nghĩa sẵn bốn đối tượng
  - cin: nhập dữ liệu từ thiết bị nhập (mặc định bàn phím)
  - cout, cerr, clog: xuất dữ liệu ra thiết bị xuất (mặc định là màn hình console)

HỆ THỐNG NHẬP/XUẤT TRONG C++

- Ví dụ chuyển luồng xuất dữ liệu

```
#include <iostream>
#include <fstream>
using namespace std;
void main(){
  int x;
  cout << “Nhap so nguyen: ”;
  cin>>x;
  if(cin.fail()){
    ofstream errFile(“error.txt”);
    streambuf* b = cerr.rdbuf();
    cerr.rdbuf(errFile.rdbuf());
    cerr<<“Error…”<<endl;
    cerr.rdbuf(b);
  }
}
```

HỆ THỐNG NHẬP/XUẤT TRONG C++

- Có hai loại tập tin: văn bản & nhị phân
- Đặc điểm của tập tin văn bản:
  - Cấu trúc đơn giản (tập các dòng)
  - Có thể xem và sửa bằng các lệnh của HĐH và phần mềm soạn thảo
  - Có hai loại tập tin văn bản: ANSI & Unicode
  - Mỗi kí tự trong tập tin ANSI kích thước 8-bit
  - Tập tin ANSI kết thúc bằng mã 26 (0x1A)
  - Mỗi kí tự trong tập tin Unicode có kích thước 16-bit (UTF-16), 32-bit (UTF-32) hay dạng UTF-8
  - Có thể tự qui định cấu trúc riêng (ví dụ: *.html, *tex)
  - Có hai cách xử lý tập tin văn bản: xử lý byte hay theo dòng

HỆ THỐNG NHẬP/XUẤT TRONG C++

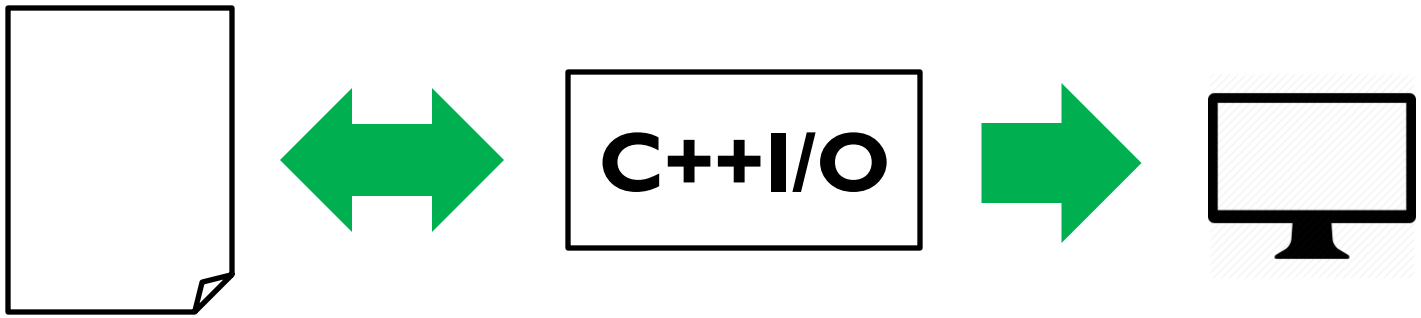
- Đặc điểm của tập tin nhị phân:
  - Có thể có cấu trúc hay không có cấu trúc
  - Nếu có cấu trúc sẽ phụ thuộc vào phần mềm đã tạo ra tập tin đó
  - Xử lý theo byte hoặc theo record
  - Ví dụ: *.bmp, *.pcx, *.jpg…

NỘI DUNG

- Các mối liên hệ đối tượng
- Bàn luận về tính chất kế thừa
- Hệ thống nhập/xuất trong C++
- Lập trình trên tập tin
  - Đọc/ghi tập tin
  - Đọc/ghi trên cùng tập tin
- Lập trình tập tin văn bản dạng mở rộng

LẬP TRÌNH TRÊN TẬP TIN

- Các thao tác trên tập tin
  - Mở tập tin
  - Xử lý đọc/ghi tập tin
  - Đóng tập tin



LẬP TRÌNH TRÊN TẬP TIN

- Trạng thái của tập tin
  - Để mở tập tin ta sẽ dùng đối tượng kiểu ifstream hoặc fstream
  - Quá trình xử lý tập tin có thể thành công hay thất bại, được phản ánh qua bốn bit sau
    - *eofbit*: là true nếu tới cuối tập tin
    - *failbit*: là true nếu có lỗi trầm trọng
    - *badbit*: là true nếu có lỗi không trầm trọng
    - *goodbit*: là true nếu các bit trên là false

LẬP TRÌNH TRÊN TẬP TIN

- Ví dụ trạng thái của tập tin

```
void main(){
  fstream f("Data.txt", ios::out);
  f.close();
  f<<2<<endl;
  cout<<f.bad()<<endl;
}
```

badbit = true

```
void main(){
  fstream f("Data.txt", ios::in);
  f.close();
  int a;
  f>>a;
  cout<<f.fail()<<endl;
}
```

failbit = true

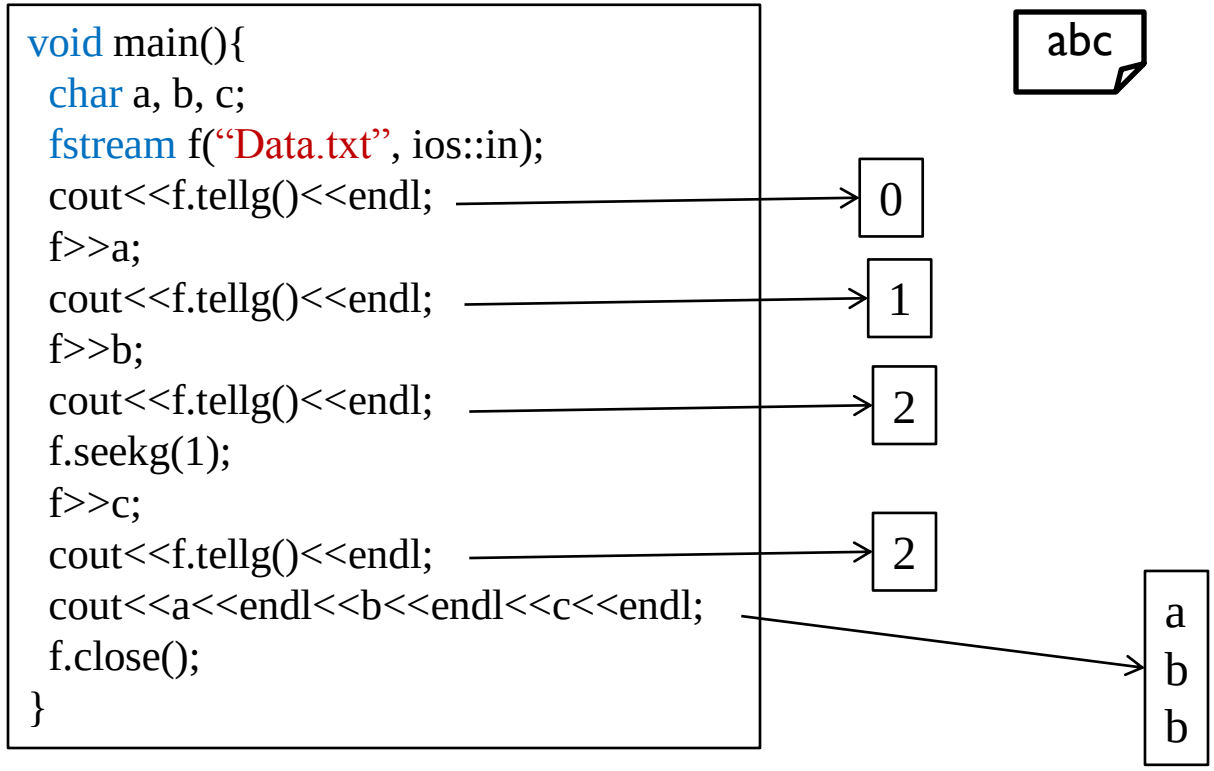
LẬP TRÌNH TRÊN TẬP TIN

- Vị trí đọc/ghi của tập tin
  - Khi đọc/ghi thì con trỏ đọc/ghi sẽ nhảy lên một vị trí: tellg() hay tellp()
  - Phân biệt con trỏ đọc và con trỏ ghi
  - Có thể kéo con trỏ đọc/ghi tới các vị trí mong muốn: seekg() hay seekp()
  - Có thể kiểm tra vị trí của con trỏ đọc/ghi: phương thức eof()

LẬP TRÌNH TRÊN TẬP TIN

- Ví dụ vị trí đọc của tập tin

```
void main(){
  char a, b, c;
  fstream f("Data.txt", ios::in);
  cout<<f.tellg()<<endl;
  f>>a;
  cout<<f.tellg()<<endl;
  f>>b;
  cout<<f.tellg()<<endl;
  f.seekg(1);
  f>>c;
  cout<<f.tellg()<<endl;
  cout<<a<<endl<<b<<endl<<c<<endl;
  f.close();
}
```



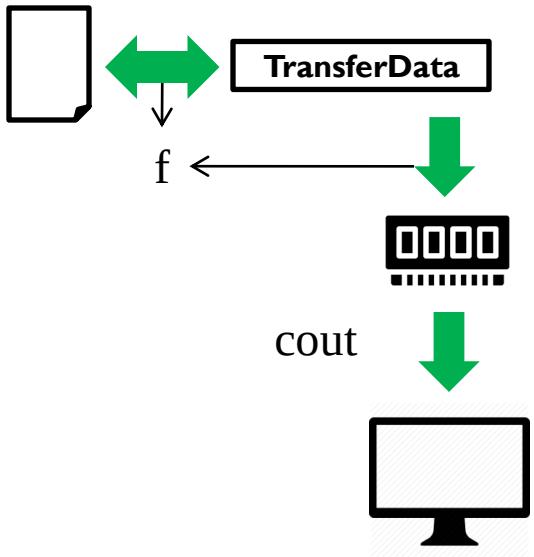
LẬP TRÌNH TRÊN TẬP TIN

- Kĩ thuật đọc tập tin: thực hiện các bước sau để đọc tập tin
  - Mở tập tin
  - Kiểm tra quá trình mở có thành công hay không
  - Chuyển nội dung từ tập tin sang thiết bị xuất
  - Khi chuyển nhớ kiểm tra tới cuối tập tin hay chưa.
  - Đóng tập tin

LẬP TRÌNH TRÊN TẬP TIN

- Ví dụ: đọc tập tin văn bản xuất ra màn hình

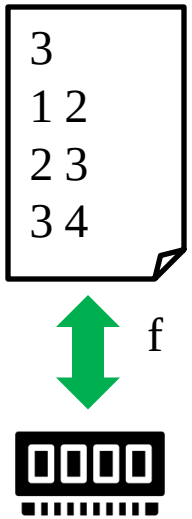
```
#include <fstream>
#include <iostream>
using namespace std;
void TransferData(istream& is, ostream& os){
  char c;
  while(is.get(c))
    os << c;
}
void main(){
  ifstream f("Data.txt");
  if(f){
    TransferData(f, cout);
    f.close();
  }
}
```



LẬP TRÌNH TRÊN TẬP TIN

- Ví dụ: đọc tập tin phân số

```
istream& operator>>(istream& is, vector<PhanSo> &arr){
  int n; arr.resize(0); is>>n;
  if(is.fail() || n <= 0) return is;
  for(int i = 0; i < n; i++){
    PhanSo p; is>>p;
    if(is.fail())break;
    arr.push_back(p);
  }
  return is;
}
void main(){
  ifstream f("Data.txt");
  if(f){
    vector<PhanSo> arr; f>>arr;
    //Bổ sung đoạn mã xuất ra màn hình
    f.close();
  }}
```



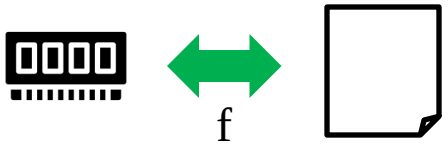
LẬP TRÌNH TRÊN TẬP TIN

- Kĩ thuật ghi tập tin: thực hiện các bước sau để ghi tập tin
  - Mở tập tin để ghi
  - Kiểm tra quá trình mở có thành công hay không
  - Chuyển nội dung từ thiết bị nhập ra thiết bị xuất
  - Đóng tập tin

LẬP TRÌNH TRÊN TẬP TIN

- Ví dụ: ghi một chuỗi trong bộ nhớ ra tập tin văn bản

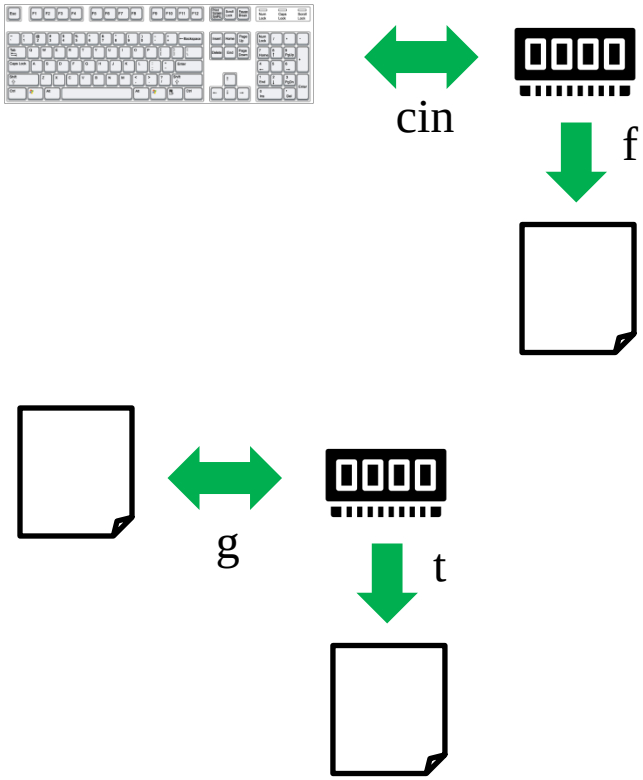
```
#include <fstream>
#include <iostream>
using namespace std;
void main(){
  ofstream f("Data.txt", ios::out);
  if(f){
    char* str = "Hello World!!!";
    while(*str){
      f.put(*str);
      str++;
    }
    f.close();
  }
}
```



LẬP TRÌNH TRÊN TẬP TIN

- Ví dụ: ghi một chuỗi trong bộ nhớ ra tập tin văn bản

```
#include <fstream>
#include <iostream>
using namespace std;
void main(){
  ofstream f("Data.txt", ios::out);
  if(f){
    TransferData(cin, f);
    f.close();
  }
  ifstream g("Source.cpp");
  ofstream t("Destination.cpp", ios::out);
  TransferData(f, t);
  g.close(); t.close();
}
```



NỘI DUNG

- Các mối liên hệ đối tượng
- Bàn luận về tính chất kế thừa
- Hệ thống nhập/xuất trong C++
- Lập trình trên tập tin
  - Đọc/ghi tập tin
  - Đọc/ghi trên cùng tập tin
- Lập trình tập tin văn bản dạng mở rộng

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Tập tin văn bản gồm các loại sau
  - Mỗi kí tự bên trong kích thước 8-bit (ASCII)
  - Mỗi kí tự bên trong kích thước 16-bit (UTF-16)
  - Mỗi kí tự bên trong kích thước 32-bit (UTF-32)
  - Mỗi kí tự bên trong kích thước từ 1 → 4 byte (UTF-8)
- Với ASCII và UTF-16, C++ hỗ trợ với hai kiểu dữ liệu char & wchar_t.
- Với UTF-8 và UTF-32 lập trình viên tự kiểm soát phạm vi các kí tự thông qua kiểu char* hay kiểu unsigned int.

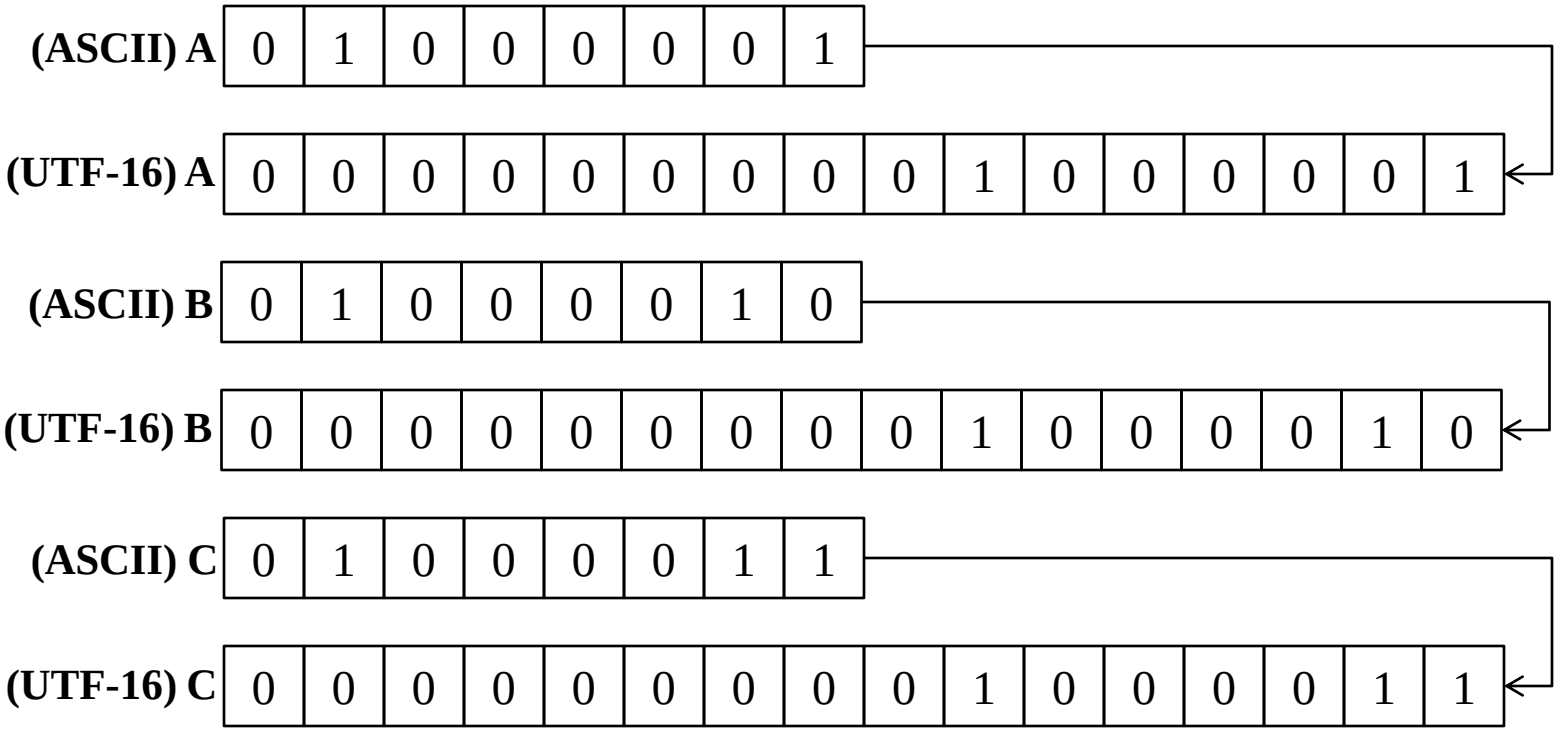
LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Một kí tự trong UTF-8 sẽ được ánh xạ ngược lại với một kí tự trong các bảng mã còn lại

Mã UTF-32 & UTF-16	Mã UTF-8
00000000 → 0000007F	0xxxxxxx (128 kí tự)
00000080 → 000007FF	110xxxxx 10xxxxxx (1920 kí tự)
00000800 → 0000FFFF	1110xxxx 10xxxxxx 10xxxxxx (63488)
00010000 → 0010FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx (1048576)

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển đổi từ chuỗi ASCII sang UTF-16



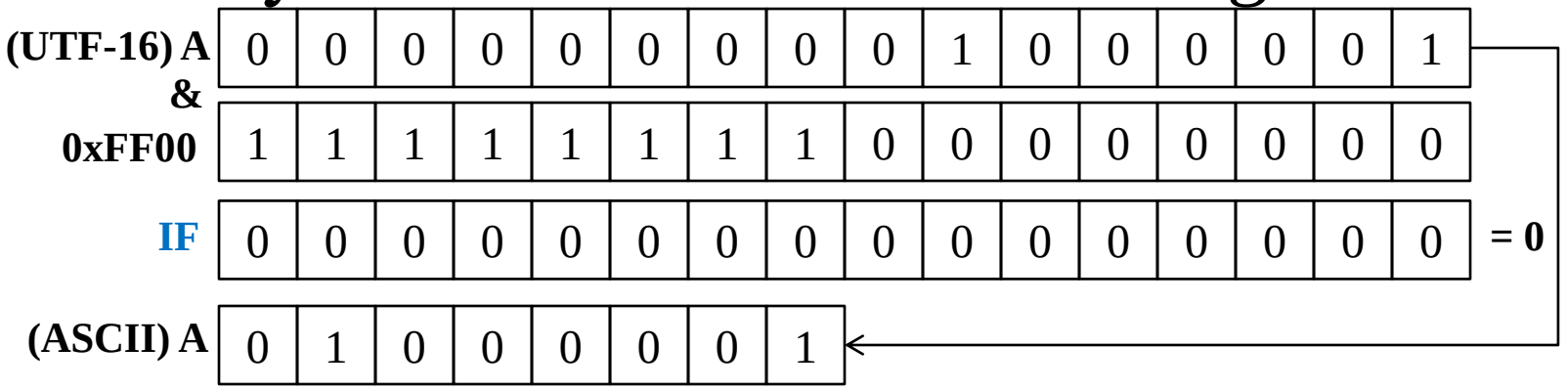

```
string ascii = "abc";
wstring uft16;
```

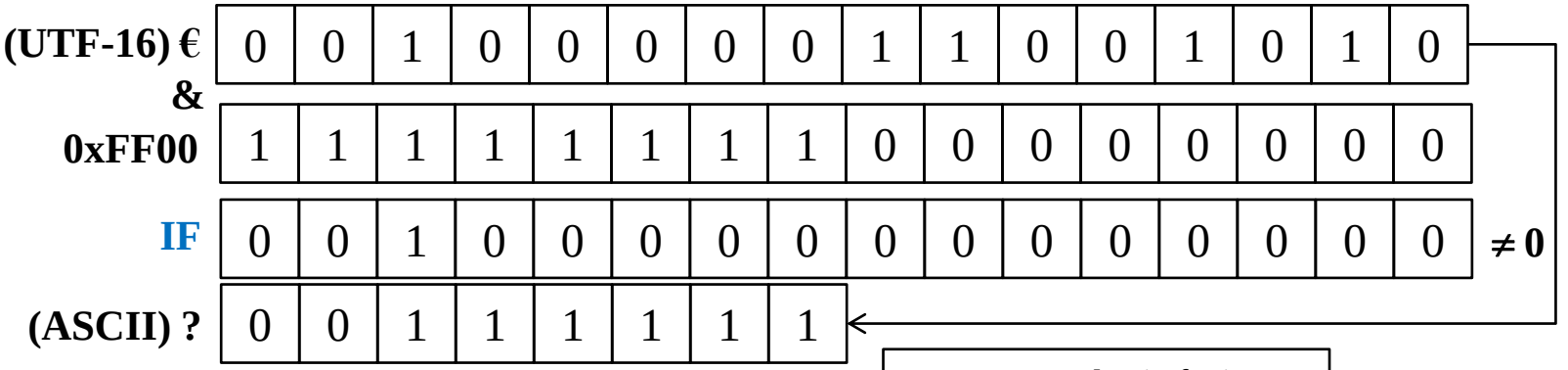
→

```
For: i = 0 → len(ascii)
  uft16+=ascii[i];
```

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển đổi từ chuỗi UTF-16 sang ASCII



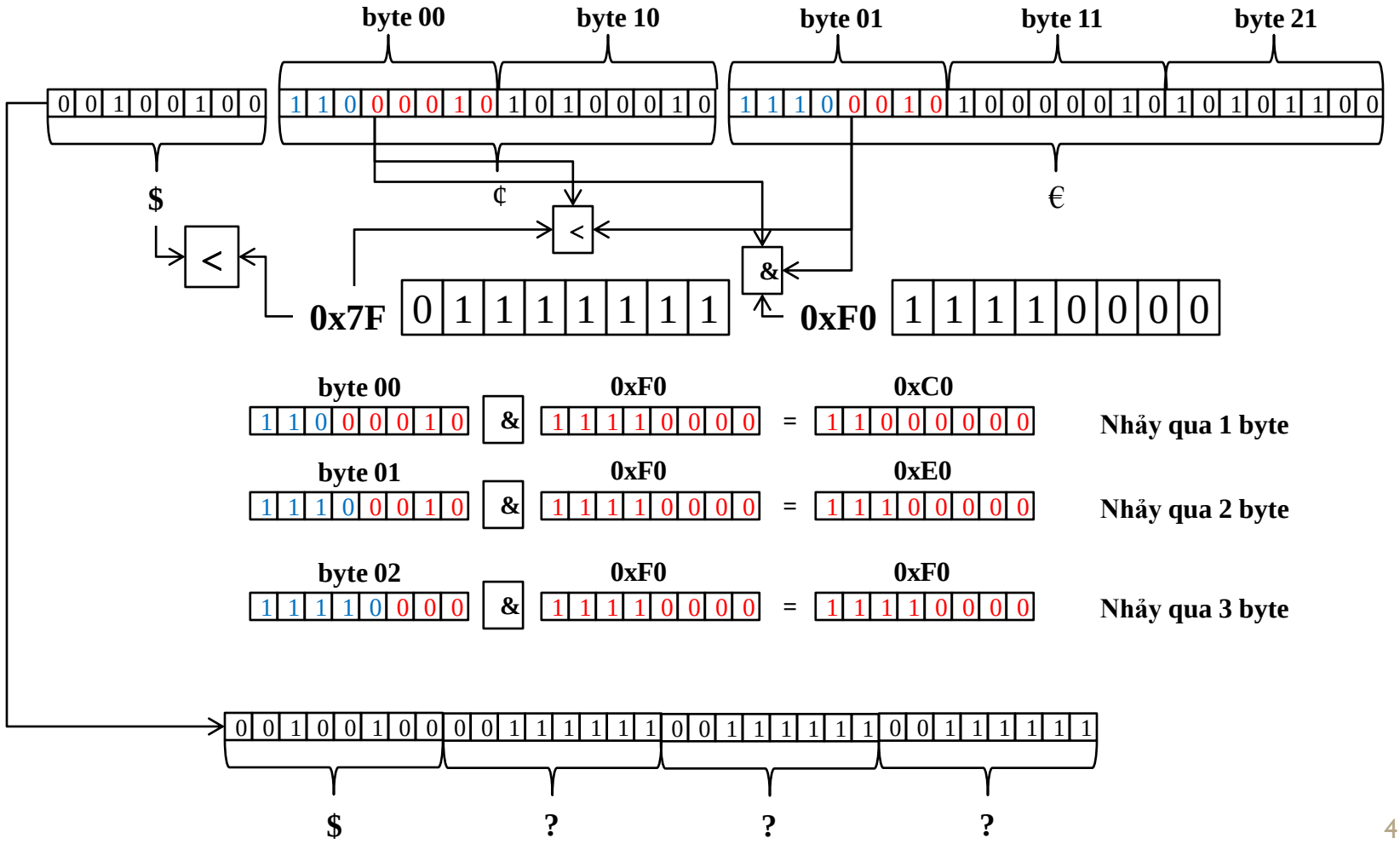



```
wstring utf16 = "A€";
string ascii;
```

```
For: i = 0 → len(utf16)
  IF: (utf16[i] & 0xff00) ≠ 0
    ascii += '?'
  ELSE:
    ascii+=utf16[i];
```

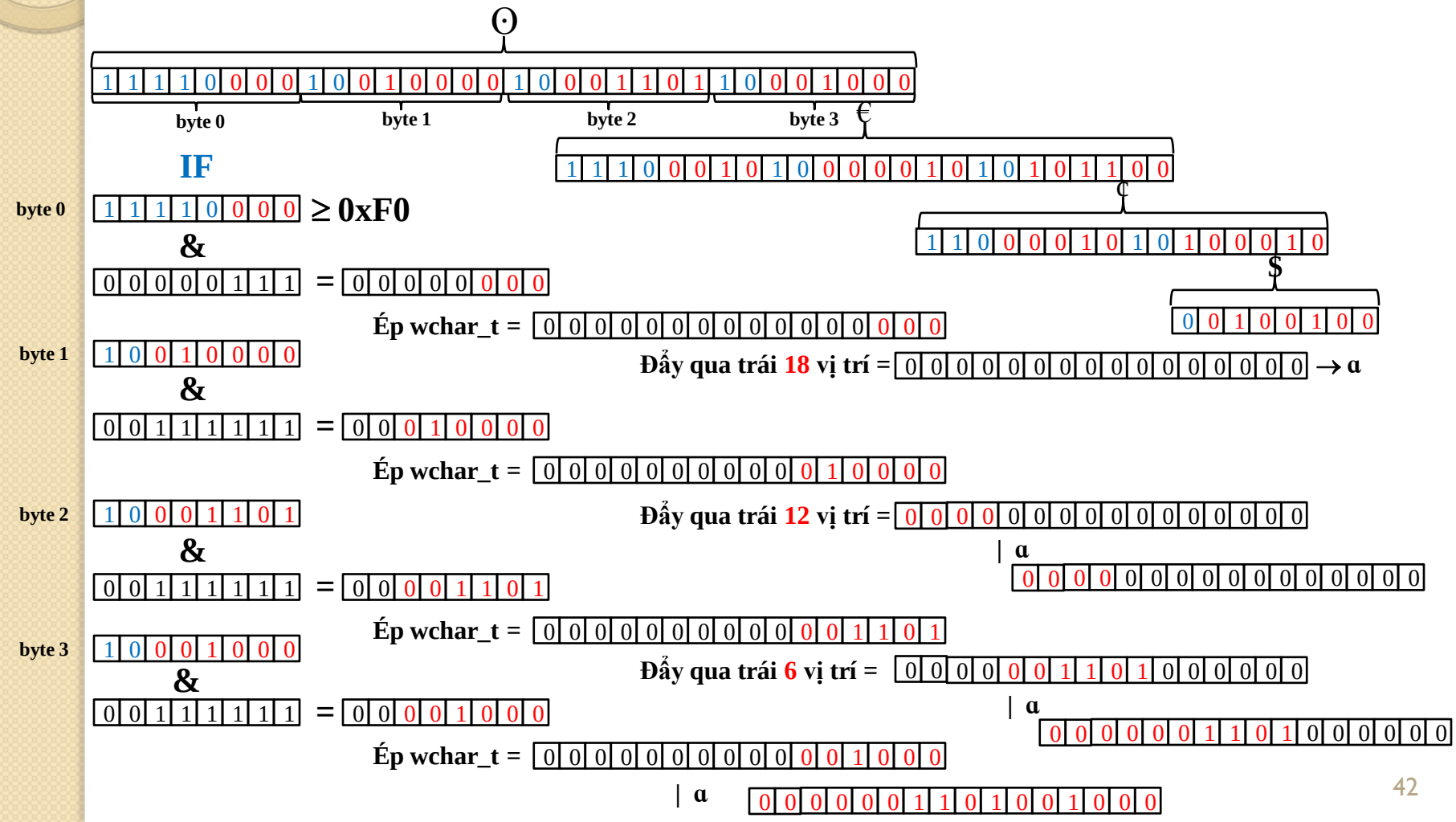
LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển chuỗi UTF-8 (“$¢€O”) → ASCII



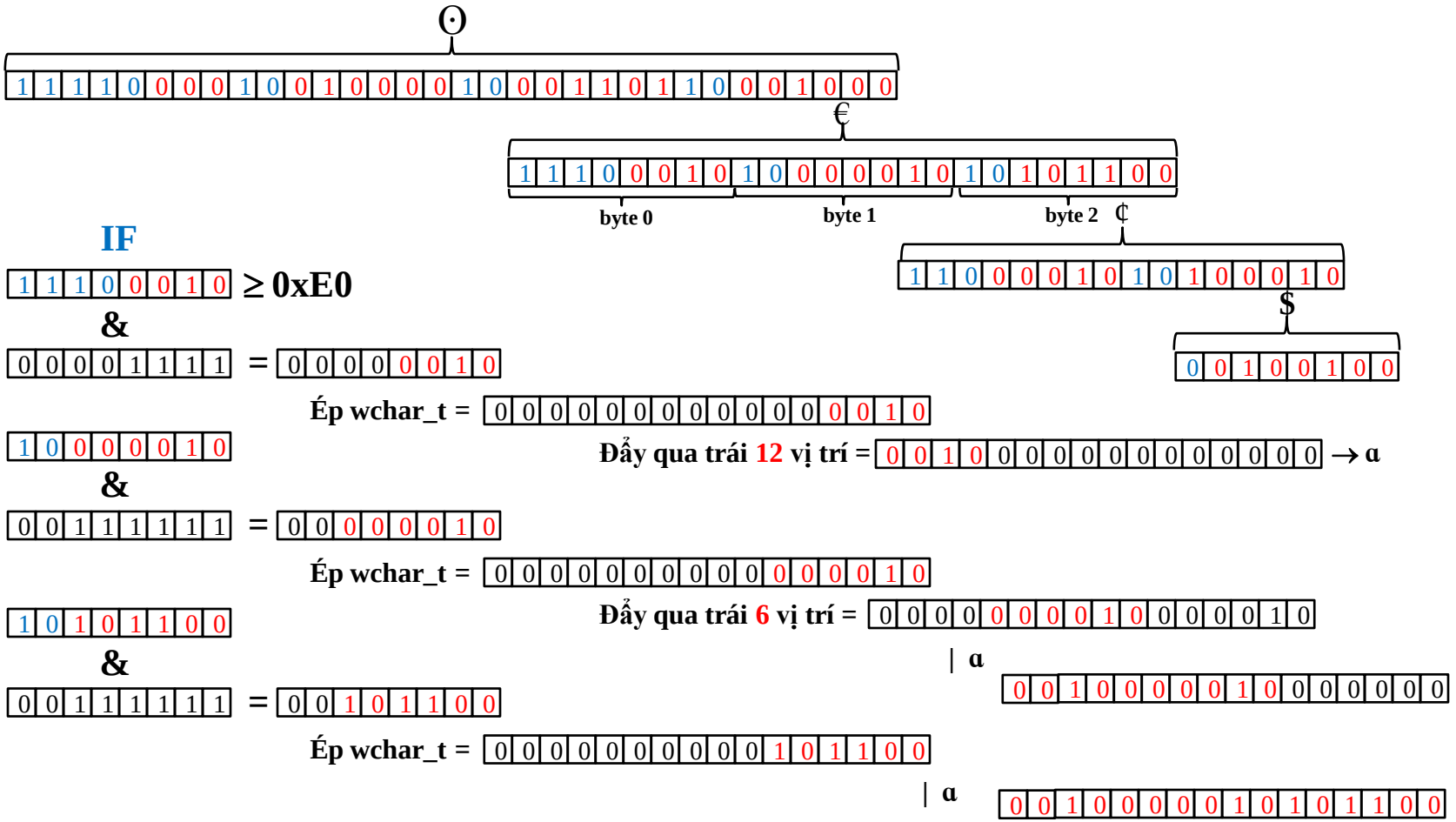
LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển chuỗi UTF-8 (“Ơ€¢$”)→UTF-16



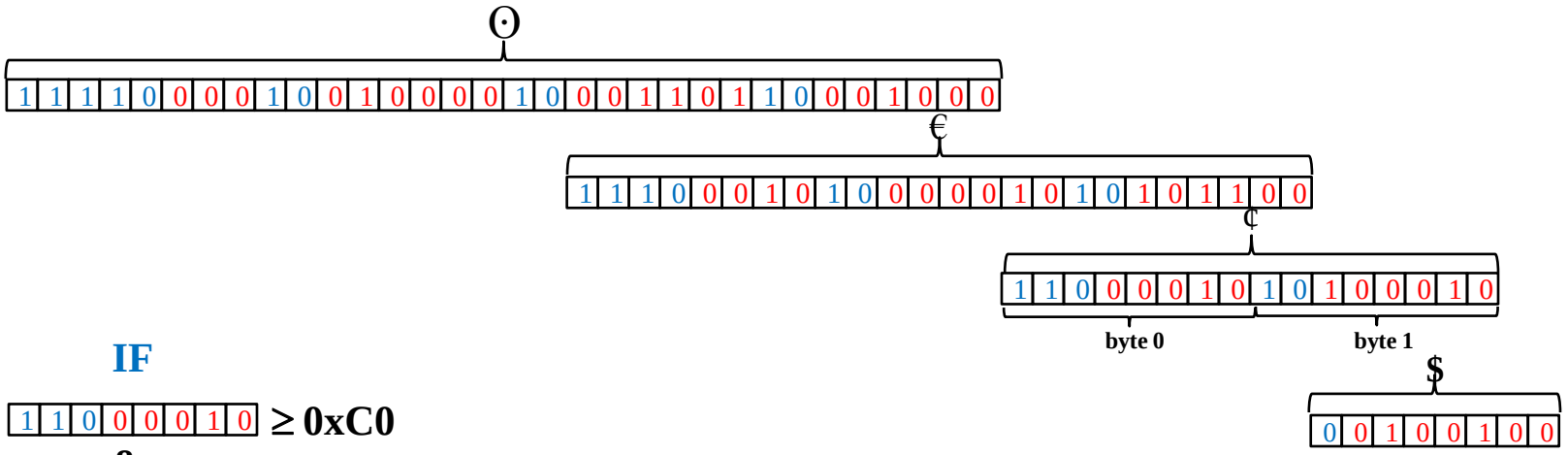
LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển chuỗi UTF-8 (“Ơ€¢$”)→UTF-16



LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển chuỗi UTF-8 (“Ơ€¢$”)→UTF-16



IF

byte 0 1 1 0 0 0 0 1 0 **≥ 0xC0**

&

0 0 0 1 1 1 1 1 **=** 0 0 0 0 0 0 1 0

Ép wchar_t = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

Đẩy qua trái 6 vị trí = 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 **→ α**

byte 1 1 0 1 0 0 0 1 0

&

0 0 1 1 1 1 1 1 **=** 0 0 1 0 0 0 1 0

Ép wchar_t = 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0

| α 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Đã thực hiện một số thao tác
  - Chuyển chuỗi ASCII ↔ UTF-16
  - Chuyển chuỗi ASCII ↔ UTF-8
  - Chuyển chuỗi UTF-8 ↔ UTF-16
- Vấn đề hàm 'len' với các chuỗi kí tự.
  - Chuỗi ASCII: kích thước chuỗi = **số lượng kí tự** + 1.
  - Chuỗi UTF-16: kích thước chuỗi = 2 × (**số lượng kí tự** + 1).
  - Chuỗi UTF-8: dựa vào cách thức lưu chuỗi để đếm chính xác **số lượng kí tự**.

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Nhận dạng chuỗi
  - Với chuỗi UTF-8: dễ dàng nhận biết thông qua cách lưu trữ
  - Với chuỗi ASCII, UTF-16 và UTF-32: chỉ có thể nhận biết được một vế.
    - Ví dụ: Có thể xác định chính xác **không phải** là chuỗi ASCII, UTF-16 hay UTF-32
- Cần nhận dạng cấu trúc tập tin để xác định nội dung thuộc định dạng gì
  - Ví dụ:
    - Tập tin UTF-8 có 3 byte {EF, BB, BF} ở đầu tập tin
    - Tập tin UTF-16 có 2 byte {FF, FE} ở đầu tập tin

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển tập tin ASCII → UTF-8
  - Mở tập tin ASCII ở chế độ đọc nhị phân
  - Tạo tập tin mới ở chế độ ghi nhị phân
  - Kiểm tra việc mở có thành công hay không
  - Ghi 3 byte {EF, BB, BF} vào tập tin mới
  - Chuyển lần lượt tất cả các byte trong tập tin ASCII sang tập tin mới
  - Đóng các tập tin

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển tập tin ASCII → UTF-8

```
static const unsigned char UTF8_FLAG[] = {0xEF, 0xBB, 0xBF};

void text_AnsiToUtf8(char* fAnsi, char* fUtf8){
  ifstream f1(fAnsi, ios::binary);
  ofstream f2(fUtf8, ios::binary);
  if(f1.fail() || f2.fail())
    return;
  f2.write((char*)UTF8_FLAG, sizeof(UTF8_FLAG));
  char c;
  while(f1.get(c))
    f2.put(c);
}
```

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển tập tin ASCII → UTF-16
  - Mở tập tin ASCII ở chế độ đọc nhị phân
  - Tạo tập tin mới ở chế độ ghi nhị phân
  - Kiểm tra việc mở có thành công hay không
  - Ghi 3 byte {FF, FE} vào tập tin mới
  - Chuyển lần lượt tất cả các byte trong tập tin ASCII sang tập tin mới bằng hàm chuyển chuỗi ASCII → UTF-16
  - Đóng các tập tin

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển tập tin ASCII → UTF-16

```
static const unsigned char UTF16_FLAG[] = {0xFF, 0xFE};

void text_AnsiToUtf16(char* fAnsi, char* fUtf16){
  ifstream f1(fAnsi, ios::binary);
  ofstream f2(fUtf16, ios::binary);
  if(f1.fail() || f2.fail())
    return;
  f2.write((char*)UTF16_FLAG, sizeof(UTF16_FLAG));
  char* data = //Lấy dữ liệu từ tập tin fAnsi
  wstring wstr = //Chuyển chuỗi ascii → utf16
  f2.write((char*)wstr.c_str(), sizeof(wstr[0]) * len(wstr));
}
```

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển tập tin UTF-8 → UTF-16
  - Lấy toàn bộ dữ liệu trong tập tin UTF-8 ra lưu vào một chuỗi
  - Chuyển chuỗi UTF-8 → UTF-16
  - Tạo tập tin để ghi dữ liệu dưới chế độ nhị phân
  - Ghi 2 byte {FF, FE} vào đầu tập tin mới
  - Ghi toàn bộ chuỗi vừa mới chuyển vào tập tin mới
  - Đóng các tập tin

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển tập tin UTF-8 → UTF-16

```
static const unsigned char UTF16_FLAG[] = {0xFF, 0xFE};

void text_Utf8ToUtf16(char* fUtf8, char* fUtf16){
  char* data = //Lấy dữ liệu trong  fUtf8
  wstring wstr = //Chuyển chuỗi data Utf-8 thành Utf-16
  ofstream f(fUtf16, ios::binary);
  f.write((char*)UTF16_FLAG, sizeof(UTF16_FLAG));
  f.write((char*)wstr.c_str(), sizeof(wstr[0]) * len(wstr));
}
```

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển tập tin UTF-16 → UTF-8
  - Lấy toàn bộ dữ liệu trong tập tin UTF-16 ra lưu vào một chuỗi
  - Chuyển chuỗi UTF-16 → UTF-8
  - Tạo tập tin để ghi dữ liệu dưới chế độ nhị phân
  - Ghi 3 byte {EF, BB, BF} vào đầu tập tin mới
  - Ghi toàn bộ chuỗi vừa mới chuyển vào tập tin mới
  - Đóng các tập tin

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển tập tin UTF-16 → UTF-8

```
static const unsigned char UTF8_FLAG[] = {0xEF, 0xBB, 0xBF};

void text_Utf16ToUtf8(char* fUtf16, char* fUtf8){
  wstring data = //Lấy dữ liệu trong  fUtf16
  string str = //Chuyển chuỗi data Utf-16 thành Utf-8
  ofstream f(fUtf8, ios::binary);
  f.write((char*)UTF8_FLAG, sizeof(UTF8_FLAG));
  f.write(str.c_str(), str.length());
}
```

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển tập tin UTF-8 → ASCII
  - Lấy toàn bộ dữ liệu trong tập tin UTF-8 ra lưu vào một chuỗi
  - Chuyển chuỗi UTF-8 → ASCII
  - Tạo tập tin để ghi dữ liệu dưới chế độ nhị phân
  - Ghi toàn bộ chuỗi vừa mới chuyển vào tập tin mới
  - Đóng các tập tin

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển tập tin UTF-8 → ASCII

```
void text_Utf8ToAnsi(char* fUtf8, char* fAnsi){
  char* data = //Lấy dữ liệu trong  fUtf8
  string str = //Chuyển chuỗi data Utf-8 thành ASCII
  ofstream f(fAnsi, ios::binary);
  f.write(str.c_str(), str.length());
}
```

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển tập tin UTF-16 → ASCII
  - Lấy toàn bộ dữ liệu trong tập tin UTF-16 ra lưu vào một chuỗi
  - Chuyển chuỗi UTF-16 → ASCII
  - Tạo tập tin để ghi dữ liệu dưới chế độ nhị phân
  - Ghi toàn bộ chuỗi vừa mới chuyển vào tập tin mới
  - Đóng các tập tin

LẬP TRÌNH TẬP TIN VĂN BẢN DẠNG MỞ RỘNG

- Chuyển tập tin UTF-16 → ASCII

```
void text_Utf16ToAnsi(char* fUtf16, char* fAnsi){
  wstring data = //Lấy dữ liệu trong  fUtf16
  string str = //Chuyển chuỗi data Utf-16 thành ASCII
  ofstream f(fAnsi, ios::binary);
  f.write(str.c_str(), str.length());
}
```

BÀI TẬP

- Bài tập 1: Xây dựng lớp CMessage gồm
  - Thuộc tính:
    - mNumber (số điện thoại nhắn tin, 9 kí số)
    - mVote (mã số bình chọn)
    - mDate (ngày bình chọn, dùng kiểu CDate)
    - mTime (giờ bình chọn, dùng kiểu CTime)
  - Phương thức:
    - fInput: đọc MỘT tập tin → đối tượng CMessage
    - fOutput: xuất MỘT đối tượng CMessage → tập tin

input

```
0907123456
7
10/03/2014
18:20:22
```

output

```
0907123456 7 10/03/2014 18:20:22
```

BÀI TẬP

- Bài tập 2: Xây dựng lớp CListMessage
  - Thuộc tính:
    - mMessage (mảng các CMessage)
    - mAmount (số lượng tin nhắn)
  - Phương thức:
    - fInput: đọc tập tin → mảng các đối tượng CMessage
    - fOutput: xuất mã số được bình chọn nhiều nhất ra tập tin

input

```
3
0907123456
7
10/03/2014
18:20:22
0917103856
2
10/03/2014
09:17:20
0987188411
7
11/03/2014
02:20:10
```

output

```
7
```