TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

GIÁO TRÌNH KỸ THUẬT LẬP TRÌNH

Huỳnh Minh Trí
Phan Tấn Quốc
Nguyễn Nhựt Đông

TP. Hồ Chí Minh, tháng 09 năm 2016

MỤC LỤC

MỤC LỤC i
Lời nói đầu 1
CHƯƠNG 1. THUẬT TOÁN 2
1.1. KHÁI NIỆM BÀI TOÁN 2
1.1.1. Bài toán 2
1.1.2. Các bước giải bài toán bằng máy tính 2
1.2. GIỚI THIỆU VỀ THUẬT TOÁN 4
1.2.1. Khái niệm thuật toán 4
1.2.2. Các đặc trưng của thuật toán 4
1.3. NGÔN NGỮ SƠ ĐỒ KHỐI 5
1.3.1. Tập ký hiệu 5
1.3.2. Cấu trúc rẽ nhánh 6
1.3.3. Cấu trúc lặp 8
1.4. NGÔN NGỮ MÃ GIẢ 14
1.4.1. Tập ký hiệu (mã giả theo ngôn ngữ C/C++) 14
1.4.2. Cấu trúc rẽ nhánh 14
1.4.3. Cấu trúc lặp 15
BÀI TẬP 18
CHƯƠNG 2. TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH C 20
2.1. CÁC KHÁI NIỆM CƠ BẢN 20
2.1.1. Kiểu dữ liệu số 21
2.1.2. Hằng 22
2.1.3. Biến 23
2.1.4. Biểu thức 24
2.2. LỆNH XUẤT DỮ LIỆU, LỆNH NHẬP DỮ LIỆU, LỆNH GÁN 24
2.2.1. Lệnh xuất dữ liệu với printf() 24
2.2.2. Lệnh nhập dữ liệu với scanf() 25
2.2.3. Lệnh gán 25

2.2.4. Lệnh xuất dữ liệu với cout 25
2.2.5. Lệnh nhập dữ liệu với cin 26
2.3. TOÁN TỬ 26
2.3.1. Toán tử so sánh 26
2.3.2. Toán tử logic 26
2.3.3. Toán tử tăng, giảm 27
2.3.4. Toán tử điều kiện 27
2.3.5. Thứ tự ưu tiên của các toán tử 27
2.3.6. Vấn đề chuyển đổi kiểu dữ liệu 28
2.4. CHÚ THÍCH TRONG CHƯƠNG TRÌNH 28
2.5. CẤU TRÚC CỦA MỘT CHƯƠNG TRÌNH C/C++ ĐƠN GIẢN 29
BÀI TẬP 33
CHƯƠNG 3. CÁC CẤU TRÚC ĐIỀU KHIỂN 35
3.1. CẤU TRÚC if else 35
3.2. CẤU TRÚC switch . . . case 38
3.3. CẤU TRÚC for 41
3.4. CẤU TRÚC while 44
3.5. CẤU TRÚC do while 47
3.6. MỘT SỐ CÂU LỆNH KHÁC 48
3.6.1. Câu lệnh break 48
3.6.2. Câu lệnh continue 49
BÀI TẬP 51
CHƯƠNG 4. CHƯƠNG TRÌNH CON 54
4.1. KHÁI NIỆM 54
4.1.1. Khái niệm chương trình con 54
4.1.2. Lợi ích của việc sử dụng chương trình con 54
4.2. CÁCH THIẾT KẾ VÀ SỬ DỤNG HÀM 54
4.2.1. Tham số hình thức biến, tham số hình thức trị 55
4.2.2. Biến toàn cục và biến địa phương 56
4.2.3. Hàm không có giá trị trả về, hàm có giá trị trả về 56
BÀI TẬP 68
CHƯƠNG 5. KIỂU DỮ LIỆU MẢNG 70
5.1. MẢNG MỘT CHIỀU 70
5.1.1. Khái niệm mảng 70

5.1.2. Khai báo mảng một chiều 70
5.2. THAO TÁC TRÊN MẢNG MỘT CHIỀU 71
5.2.1. Nhập các phần tử 71
5.2.2. Duyệt các phần tử 71
5.2.3. Tính tổng các phần tử 71
5.2.4. Đếm số phần tử thỏa điều kiện nào đó 72
5.2.5. Đặt cờ hiệu 73
5.2.6. Xóa một phần tử trong mảng 73
5.2.7. Chèn phần tử vào mảng 73
5.2.8. Đặt lính canh 74
5.2.9. Bài toán sắp xếp 74
5.2.10. Bài toán tìm kiếm 75
5.3. MẢNG HAI CHIỀU 84
5.3.1. Khai báo mảng hai chiều 84
5.3.2. Truy nhập đến thành phần của mảng 85
5.3.3. Khởi tạo giá trị cho mảng 85
5.4.CÁC THAO TÁC THƯỜNG GẶP TRÊN MẢNG HAI CHIỀU 86
5.4.1. Nhập mảng 86
5.4.2. Xuất mảng 86
5.4.3. Tính tổng 87
5.4.4. Đếm số phần tử thỏa điều kiện 87
5.4.5. Đặt lính canh 88
5.4.6. Đặt cờ hiệu 88
5.4.7. Sắp xếp 90
5.4.8. Xử lý dòng, cột 91
BÀI TẬP 93
CHƯƠNG 6. KIỂU DỮ LIỆU CÓ CẤU TRÚC 106
6.1. KHAI BÁO CẤU TRÚC- KHAI BÁO BIẾN CẤU TRÚC 106
6.2. TRUY XUẤT ĐẾN TỪNG THÀNH PHẦN CỦA CẤU TRÚC 108
BÀI TẬP 117
Chương 7. KỸ THUẬT LẬP TRÌNH ĐỆ QUY 119
7.1. HÀM ĐỆ QUI 119
7.2. PHÂN LOẠI ĐỆ QUY 119
7.2.1. Đệ quy tuyến tính 119

7.2.2. Đệ quy nhị phân 120
7.2.3. Đệ quy hỗ tương 122
7.2.4. Đệ quy phi tuyến 123
7.3. KHỬ ĐỆ QUY 124
7.4. THUẬT TOÁN ĐỆ QUY 125
BÀI TẬP 127
Chương 8. KỸ THUẬT LẬP TRÌNH CON TRỎ 129
8.1. ĐỊA CHỈ VÀ CON TRỎ 129
8.1.1. Địa chỉ ô nhớ 129
8.1.2. Con trỏ 129
8.1.3. Khai báo con trỏ 129
8.1.4. Phép lấy địa chỉ của một biến 129
8.1.5. Phép toán lấy giá trị tại một địa chỉ mà một con trỏ đang trỏ tới 130
8.2. QUY TẮC SỬ DỤNG CON TRỎ TRONG CÁC BIỂU THỨC 130
8.2.1. Sử dụng tên con trỏ 130
8.2.2. Sử dụng dạng khai báo của con trỏ 131
8.3. CÁC THAO TÁC TRÊN CON TRỎ 131
8.3.1. Phép toán dịch chuyển địa chỉ của con trỏ 131
8.3.2. Cấp phát động bộ nhớ 131
8.3.3. Giải phóng khối nhớ đã được cấp phát 133
8.4. CON TRỎ VỚI MẢNG MỘT CHIỀU 133
8.5. CON TRỎ VỚI MẢNG HAI CHIỀU 138
8.6. TỔ CHỨC DỮ LIỆU DẠNG DANH SÁCH 140
BÀI TẬP 143
Chương 9. KỸ THUẬT LẬP TRÌNH CHUỖI 145
9.1. KÝ TỰ VÀ CHUỖI KÝ TỰ 145
9.2. NHẬP/XUẤT KÝ TỰ 145
9.3. NHẬP/XUẤT CHUỖI KÝ TỰ 146
BÀI TẬP 159
Chương 10. KỸ THUẬT LẬP TRÌNH VỚI FILE 163
10.1. KHÁI NIỆM 163
10.2. KHAI BÁO CON TRỎ FILE 163
10.3. MỘT SỐ HÀM XỬ LÝ FILE VĂN BẢN THƯỜNG DÙNG 163
10.3.1. Mở tập tin 163

10.3.2. Đóng tập tin 164
10.3.3. Đưa dữ liệu vào tập tin 164
10.3.4. Đọc dữ liệu từ tập tin 164
10.4. MỘT SỐ HÀM XỬ LÝ FILE NHỊ PHÂN 174
10.4.1. File nhị phân chứa các số nguyên 174
10.4.2. File nhị phân với dữ liệu có cấu trúc 175
BÀI TẬP 191
Chương 11. KỸ THUẬT LẬP TRÌNH NÂNG CAO 200
11.1. THUẬT TOÁN CHIA ĐỂ TRỊ 200
11.2. THUẬT TOÁN QUAY LUI 203
11.2.1. Bài toán liệt kê 203
11.2.2. Lưu đồ thuật toán quay lui 203
11.3. THUẬT TOÁN QUY HOẠCH ĐỘNG 212
11.3.1. Phân rã 212
11.3.2. Ghi nhận lời giải 212
11.3.3. Tổng hợp lời giải 213
11.3.4. Một số bài toán quy hoạch động điển hình 213
11.4. THUẬT TOÁN THAM LAM 217
BÀI TẬP 223
ĐỀ THI THAM KHẢO 229
ĐỀ THI CƠ SỞ LẬP TRÌNH 01 229
ĐỀ THI CƠ SỞ LẬP TRÌNH 02 230
ĐỀ THI CƠ SỞ LẬP TRÌNH 03 231
ĐỀ THI CƠ SỞ LẬP TRÌNH 04 232
ĐỀ THI CƠ SỞ LẬP TRÌNH 05 233
ĐỀ THI CƠ SỞ LẬP TRÌNH 06 234
ĐỀ THI CƠ SỞ LẬP TRÌNH 07 235
ĐỀ THI CƠ SỞ LẬP TRÌNH 08 236
ĐỀ THI CƠ SỞ LẬP TRÌNH 09 237
ĐỀ THI CƠ SỞ LẬP TRÌNH 10 238
ĐỀ KIỂM TRA THỰC HÀNH CƠ SỞ LẬP TRÌNH 01 239
ĐỀ KIỂM TRA THỰC HÀNH CƠ SỞ LẬP TRÌNH 02 240
ĐỀ KIỂM TRA THỰC HÀNH CƠ SỞ LẬP TRÌNH 03 241
ĐỀ KIỂM TRA THỰC HÀNH CƠ SỞ LẬP TRÌNH 04 242

ĐỀ KIỂM TRA THỰC HÀNH CƠ SỞ LẬP TRÌNH 05 243
ĐỀ THI KỸ THUẬT LẬP TRÌNH 01 244
ĐỀ THI KỸ THUẬT LẬP TRÌNH 02 246
ĐỀ THI KỸ THUẬT LẬP TRÌNH 03 248
ĐỀ THI KỸ THUẬT LẬP TRÌNH 04 250
ĐỀ THI KỸ THUẬT LẬP TRÌNH 05 252
ĐỀ THI KỸ THUẬT LẬP TRÌNH 06 254
ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 01 256
ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 02 258
ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 03 259
ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 04 261
ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 05 263
ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 06 264
ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 07 265
ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 08 267
ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 09 269
ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 10 270
PHỤ LỤC. MỘT SỐ HÀM CHUẨN THƯỜNG SỬ DỤNG 272
TÀI LIỆU THAM KHẢO 277

Lời nói đầu

Kỹ thuật lập trình là học phần đóng vai trò quan trọng đối với các sinh viên thuộc các ngành thuộc lĩnh vực Công nghệ thông tin, Toán ứng dụng, Sư phạm tin học, Điện tử viễn thông,… Kỹ thuật lập trình nhằm cung cấp kiến thức nền tảng về lập trình cho sinh viên; Kỹ thuật lập trình nhằm hỗ trợ sinh viên học tốt các học phần về lập trình tiếp theo. Kỹ thuật lập trình là nội dung quan trọng trong khối kiến thức ở các kỳ thi hoàn chỉnh đại học và thi cao học các ngành thuộc lĩnh vực công nghệ thông tin.

Nội dung giáo trình Kỹ thuật lập trình này bám sát đề cương chi tiết của hai học phần Cơ sở lập trình và Kỹ thuật lập trình của trường Đại học Sài Gòn, giáo trình gồm 11 chương: Thuật toán, Mở đầu về ngôn ngữ lập trình C/C++, Các cấu trúc điều khiển, Chương trình con, Kiểu dữ liệu mảng, Kiểu dữ liệu cấu trúc, Lập trình đệ qui, Con trỏ, Chuỗi ký tự, File, Các kỹ thuật lập trình nâng cao. Mỗi chương gồm hai phần: Lý thuyết và Bài tập; cuối giáo trình có các đề thi lý thuyết và thực hành để sinh viên luyện tập.

Nội dung các chương 1,6,8,10 do tác giả Huỳnh Minh Trí biên soạn. Nội dung các chương 5,7,9,11 do tác giả Phan Tấn Quốc biện soạn. Nội dung các chương 2,3,4 và phần phụ lục do tác giả Nguyễn Nhựt Đông biên soạn. Nội dung của 31 đề thi lý thuyết và thực hành tham khảo do ba tác giả cùng biên soạn.

Chúng tôi trân trọng giới thiệu với bạn đọc quyển giáo trình Kỹ thuật lập trình này và hy vọng rằng nó sẽ giúp cho việc giảng dạy và học tập các môn Cơ sở lập trình và Kỹ thuật lập trình của Trường được thuận lợi hơn.

Cuối cùng, chúng tôi xin gởi lời cảm ơn các đồng nghiệp trong khoa công nghệ thông tin trường Đại Học Sài Gòn đã có nhiều đóng góp quý báu, đã cùng chúng tôi chia sẽ nội dung tập bài giảng các học phần Cơ sở lập trình và Kỹ thuật lập trình này trong suốt hàng chục năm qua.

Thành phố Hồ Chí Minh, ngày 05 tháng 09 năm 2016

HUỲNH MINH TRÍ-PHAN TẤN QUỐC-NGUYỄN NHỰT ĐÔNG

Bộ môn khoa học máy tính,
Khoa Công nghệ thông tin
Trường Đại học Sài Gòn

Giáo trình này được hoàn thành từ nguồn kinh phí của Trường Đại học Sài Gòn (CS2014-03, CS2015-01)

CHƯƠNG 1. THUẬT TOÁN

1.1. KHÁI NIỆM BÀI TOÁN

1.1.1. Bài toán

Trong phạm vi giáo trình này, bài toán được quan niệm là công việc nào đó ta muốn máy tính thực hiện, chẳng hạn viết một dòng chữ ra màn hình, giải phương trình bậc hai, quản lí điểm trong trường học,…

Khi dùng máy tính giải bài toán, ta cần quan tâm đến hai yếu tố: đưa vào máy thông tin gì (Input) và cần lấy ra thông tin gì (Output). Do đó để phát biểu một bài toán ta cần phải chỉ rõ Input và Output của bài toán đó.

Sau đây là ví dụ về một số bài toán tin học: Bài toán giải phương trình dạng $ax^2 + bx + c = 0$; bài toán tìm tất cả phương án để đặt n quân hậu trên bàn cờ $n \times n$; bài toán cho dãy n số, tìm dãy con tăng dài nhất (dãy con này có thể không gồm các phần tử liên tiếp nhau), bài toán tính 2^{1000},…

1.1.2. Các bước giải bài toán bằng máy tính

Khả năng khai thác máy tính phụ thuộc rất nhiều vào sự hiểu biết của người sử dụng. Việc giải bài toán trên máy tính được tiến hành qua các bước sau:

Bước 1: Xác định bài toán

Như đã trình bày, mỗi bài toán được đặc tả bởi hai thành phần: Input và Output. Việc xác định bài toán chính là xác định rõ hai thành phần này; các thông tin đó cần được nghiên cứu cẩn thận để có thể lựa chọn thuật toán, cách thể hiện các đại lượng đã cho và các đại lượng phát sinh trong quá trình giải bài toán và ngôn ngữ lập trình thích hợp.

Bước 2: Lựa chọn thuật toán giải

Mỗi bài toán có thể giải bằng nhiều thuật toán khác nhau. Việc chọn thuật toán nào cho bài toán cần giải quyết phụ thuộc người lập trình. Một thuật toán được đánh giá là tốt nếu thuật toán đó cho thời gian chạy nhanh (tiêu chí quan trọng nhất); sau đó mới đến các tiêu chí khác như thuật toán trong sáng, giản dị,…

Bước 3: Thiết kế thuật toán

Bước thiết kế thuật toán là bước quan trọng nhất để giải một bài toán. Mỗi thuật toán chỉ giải một bài toán nào đó, nhưng có thể có nhiều thuật toán khác nhau cùng giải một bài toán. Cần chọn một thuật toán phù hợp để giải bài toán đã cho.

Khi lựa chọn thuật toán người ta thường quan tâm đến các tài nguyên như thời gian thực thi trên máy tính, số lượng bộ nhớ cần dùng,... Hai tài nguyên về thời gian và bộ nhớ thường mâu thuẩn nhau, nói cách khác yêu cầu về thời gian và bộ nhớ là tỷ lệ nghịch nhau. Điều này đòi hỏi người lập trình phải cân nhắc giữa hai yêu cầu này. Trong các loại tài nguyên, người ta quan tâm nhiều nhất đến thời gian vì đó là dạng tài nguyên không tái tạo được.

Việc thiết kế và lựa chọn thuật toán để giải một bài toán cụ thể cần căn cứ vào lượng tài nguyên mà thuật toán đòi hỏi và lượng tài nguyên thực tế cho phép.

Bước 4: Viết chương trình

Việc viết chương trình là một tổng hợp hữu cơ giữa việc lựa chọn cấu trúc dữ liệu và ngôn ngữ lập trình để diễn đạt đúng thuật toán.

Khi viết chương trình ta cần lựa chọn một ngôn ngữ bậc cao, hoặc hợp ngữ, hoặc ngôn ngữ máy, hoặc một phần mềm chuyên dụng thích hợp cho thuật toán đã lựa chọn.

Bước 5: Hiệu chỉnh chương trình

Sau khi được viết xong, chương trình vẫn còn có thể có nhiều lỗi khác nhau. Các lỗi được phân làm hai loại là các lỗi về cú pháp và các lỗi về ngữ nghĩa.

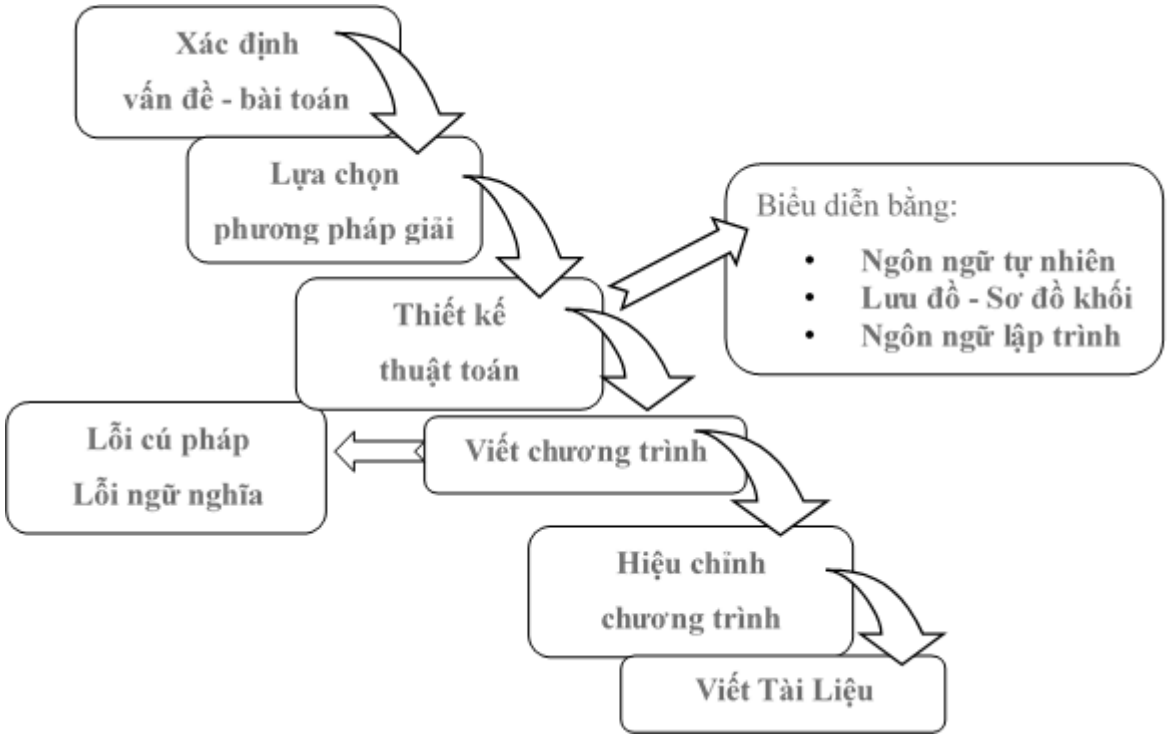
Viết chương trình trong ngôn ngữ nào ta cần phải tuân theo đúng quy định cú pháp của ngôn ngữ đó. Chương trình dịch có thể giúp ta phát hiện và thông báo đầy đủ các sai sót về mặt cú pháp.

Chương trình có thể không cho kết quả đúng mặc dù đã không còn lỗi về cú pháp. Các lỗi này là lỗi về ngữ nghĩa. Để phát hiện các lỗi về ngữ nghĩa, cần phải thử chương trình bằng cách thực hiện nó với một số bộ Input tiêu biểu phụ thuộc vào đặc thù của bài toán. Các bộ Input này gọi là các bộ Test. Nếu có sai sót, ta phải sửa chương trình rồi thử lại. Quá trình này được gọi là *hiệu chỉnh chương trình*.

Bước 6: Viết tài liệu

Tài liệu phải mô tả chi tiết bài toán, thuật toán, chương trình, kết quả thử nghiệm và hướng dẫn sử dụng. Tài liệu này rất có ích cho người sử dụng chương trình và cho việc đề xuất những khả năng hoàn thiện thêm.

Các bước trên có thể lặp đi lặp lại nhiều lần cho đến khi mà ta cho là chương trình đã làm việc đúng đắn.



Hình 1.1. *Các bước giải bài toán bằng máy tính*

1.2. GIỚI THIỆU VỀ THUẬT TOÁN

1.2.1. Khái niệm thuật toán

Thuật toán giải bài toán là một thủ tục xác định bao gồm một dãy hữu hạn các bước cần thực hiện để thu được đầu ra cho một đầu vào cho trước của bài toán.

1.2.2. Các đặc trưng của thuật toán

Đầu vào (Input): Mỗi thuật toán đều phải có dữ liệu đầu vào (tường mình hoặc không tường minh).

Đầu ra (Output): Mỗi thuật toán đều phải có dữ liệu đầu ra cụ thể.

Tính xác định: Các chỉ dẫn, qui tắc ở mỗi bước phải rõ ràng.

Tính kết thúc: Thuật toán phải dừng sau một số hữu hạn bước.

Tính đúng đắn: Thuật toán phải cho ra kết quả chính xác theo yêu cầu của bài toán.

Tính tổng quát: Thuật toán phải áp dụng được cho các bài toán cùng loại.

Tính khả thi: Mỗi thuật toán phải có thể thực hiện thành công trong khoảng thời gian hữu hạn.

Những thuật toán có các đặc trưng trên gọi là các thuật toán có lời giải đúng (các thuật toán cho lời giải gần đúng như: xấp xỉ, ngẫu nhiên, heuristic, metaheuristic,... không được đề cập trong giáo trình này).

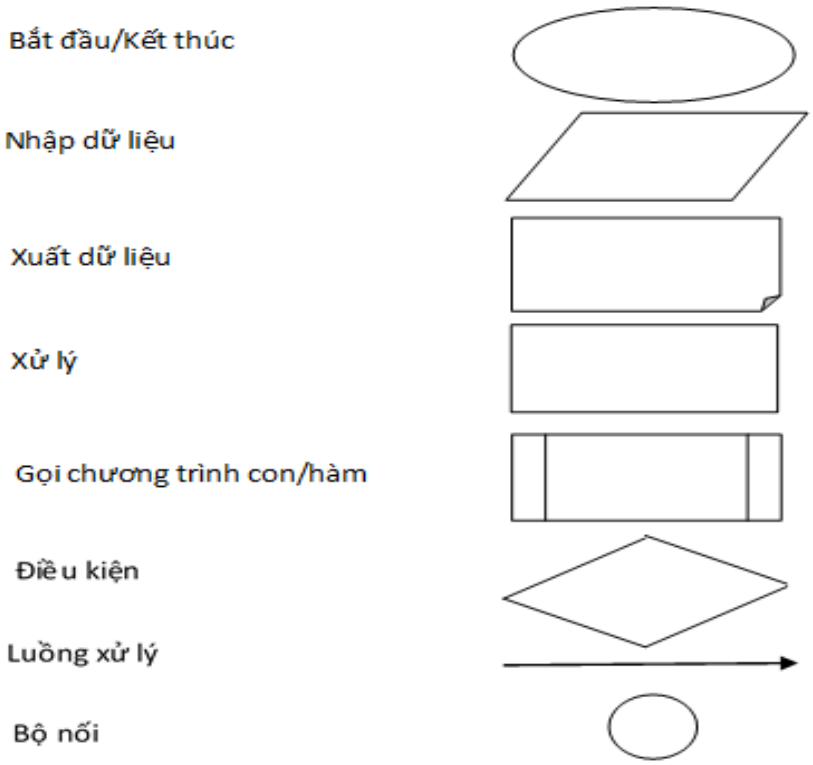
Thuật toán có thể được biểu diễn bằng ngôn ngữ tự nhiên, ngôn ngữ mã giả, ngôn ngữ lưu đồ-sơ đồ khối, hay bằng một ngôn ngữ lập trình cụ thể.

Giáo trình này sẽ trình bày cách biểu diễn thuật toán bằng ngôn ngữ sơ đồ khối và ngôn ngữ mã giả.

1.3. NGÔN NGỮ SƠ ĐỒ KHỐI

1.3.1. Tập ký hiệu

Ngôn ngữ sơ đồ khối thường sở dụng tập ký hiệu của ngôn ngữ sơ đồ khối sau:

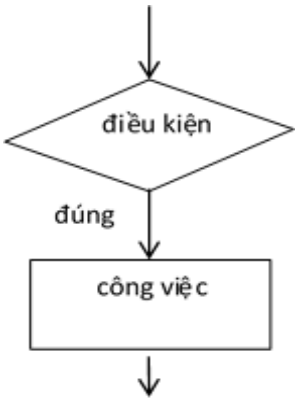


Hình 1.2. *Các sơ đồ khối thông dụng*

Sau đây là biểu diễn các cấu trúc bằng ngôn ngữ sơ đồ khối

1.3.2. Cấu trúc rẽ nhánh

Dạng 1 (dạng thiếu)

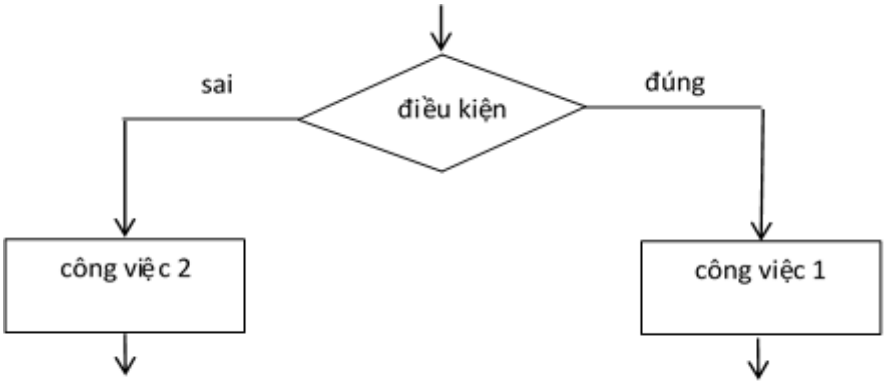


Hình 1.3. *Sơ đồ của cấu trúc rẽ nhánh-dạng 1*

Mô tả hoạt động

Nếu <điều kiện> là đúng thì sẽ thực hiện <công việc>, nếu <điều kiện> là sai thì sẽ không thực hiện <công việc>.

Dạng 2 (dạng đầy đủ)



Hình 1.4. *Sơ đồ của cấu trúc rẽ nhánh-dạng 2*

Mô tả hoạt động

Nếu <điều kiện> là đúng thì thực hiện <công việc 1>, ngược lại nếu <điều kiện> là sai thì sẽ thực hiện <công việc 2>.

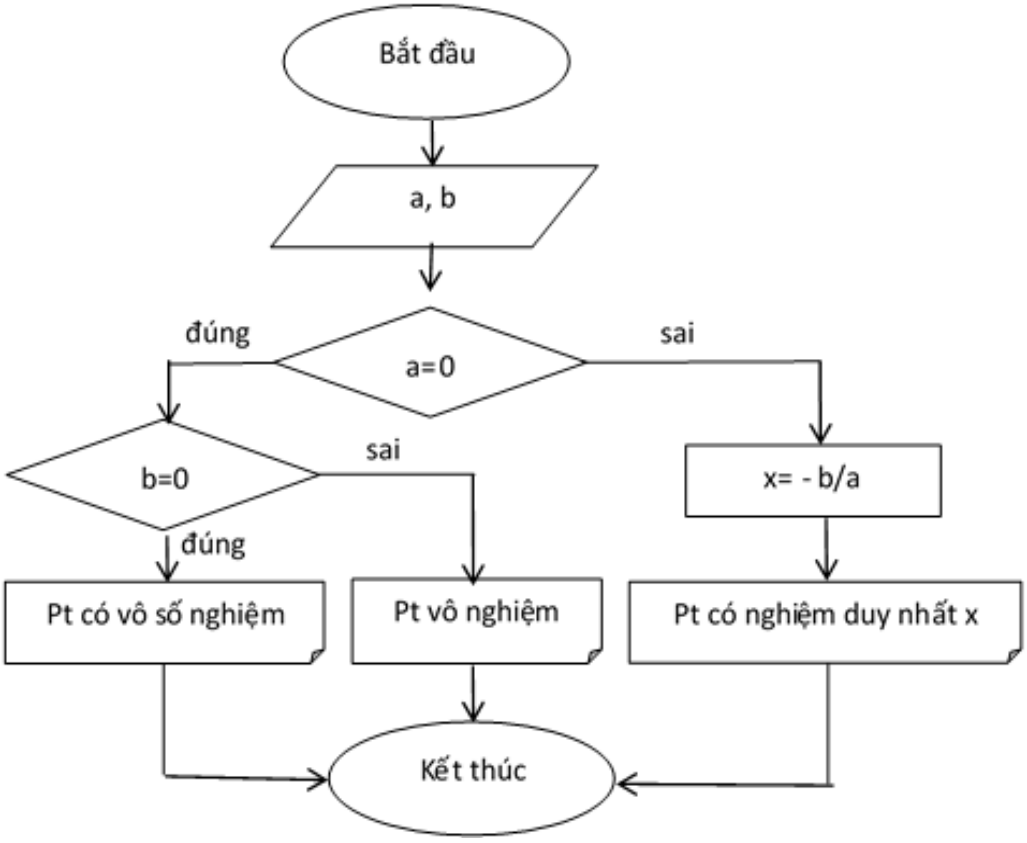
Lưu ý:

i. *<điều kiện> trong dạng 1 và 2 là một biểu thức logic, còn <công việc> là một công việc đơn giản hoặc nhiều công việc đơn giản gộp lại.*

ii. *Với cấu trúc dạng 2 này thì luôn luôn có một công việc sẽ được thực hiện bất kể biểu thức <điều kiện> là đúng hay sai.*

Ví dụ 1.1:

Hãy biểu diễn thuật toán giải phương trình bậc nhất dạng ax + b = 0 (sử dụng ngôn ngữ sơ đồ khối)



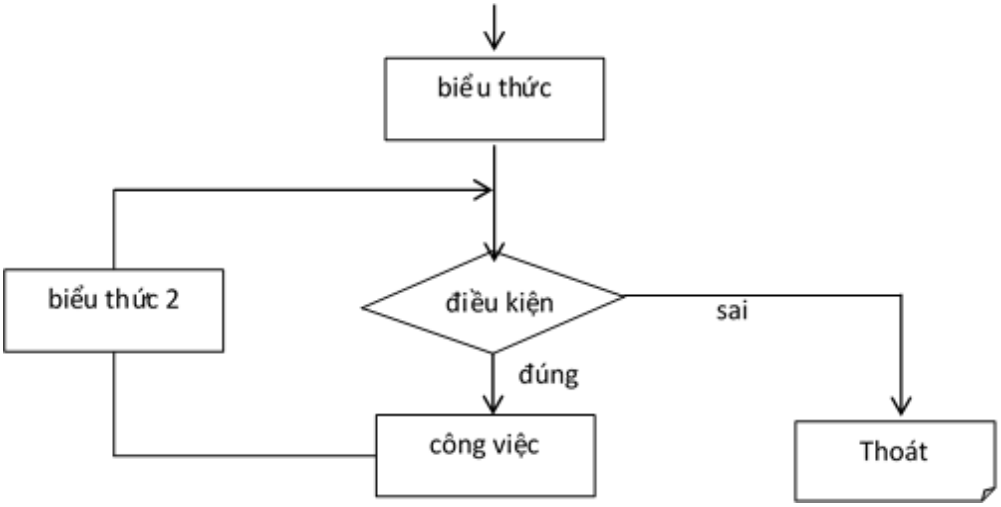
Hình 1.5. *Sơ đồ khối của ví dụ 1.1*

Một số bộ test:

Test	Hệ số a	Hệ số b	Kết quả
test 1	0	0	Phương trinh vố số nghiệm
test 2	0	2	Phương trinh vô nghiệm
test 3	1	3	Phương trinh có nghiệm duy nhất x=3
test 4	1	-4	Phương trinh có nghiệm duy nhất x=4

1.3.3. Cấu trúc lặp

***Dạng 1*:**



Hình 1.6. *Sơ đồ của cấu trúc lặp dạng 1*

Mô tả hoạt động

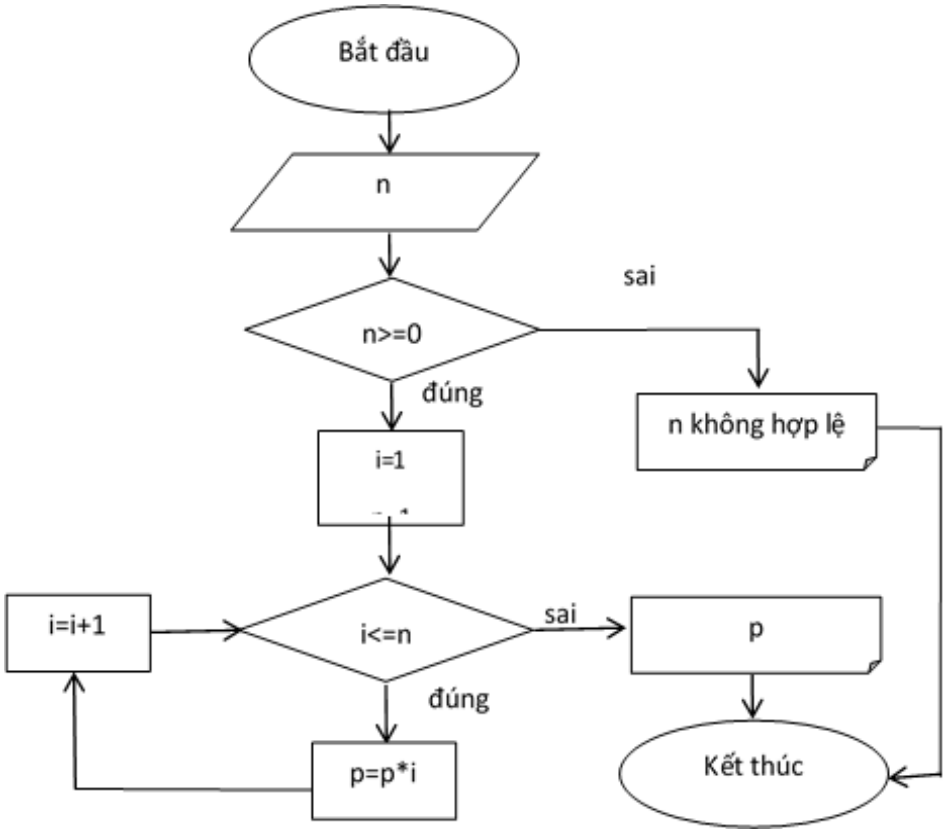
Đầu tiên sẽ thực hiện <biểu thức 1>, sau đó kiểm tra <điều kiện>. Nếu <điều kiện> là đúng thì sẽ thực hiện <công việc> rồi đến thực hiện <biểu thức 2>; sau khi thực hiện xong <biểu thức 2> thì cấu trúc này quay trở lại kiểm tra <điều kiện> và bắt đầu một vòng lặp mới,…cứ như thế đến khi <điều kiện> nhận giá sai thì cấu trúc lặp này sẽ kết thúc.

Lưu ý:

i. *<biểu thức 1> thường là phép gán giá trị khởi đầu của biến điều khiển vòng lặp,*
ii. *<biểu thức 2> thường là phép gán để thay đổi giá trị của biến điều khiển vòng lặp,*
iii. *<điều kiện> thường là một phép so sánh biến điều khiển vòng lặp với số lần lặp.*
iv. *Trong quá trình lặp lại này, <công việc> đến một lúc nào đó phải nhận giá trị sai để vòng lặp kết thúc.*

Ví dụ 1.2:

Biểu diễn thuật toán tính tích p = n!=1.2...n (sử dụng ngôn ngữ sơ đồ khối).

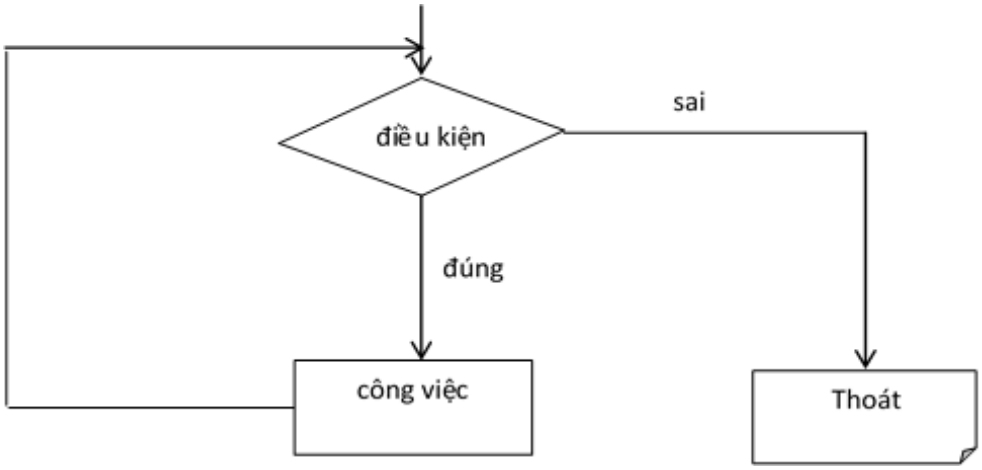


Hình 1.7. *Sơ đồ khối của ví dụ 1.2*

Một số bộ test:

Test	n	P
test 1	-1	n không hợp lệ
test 2	0	1
test 3	1	1
test 4	10	3628800

Dạng 2:



Hình 1.8. *Sơ đồ của cấu trúc lặp dạng 2*

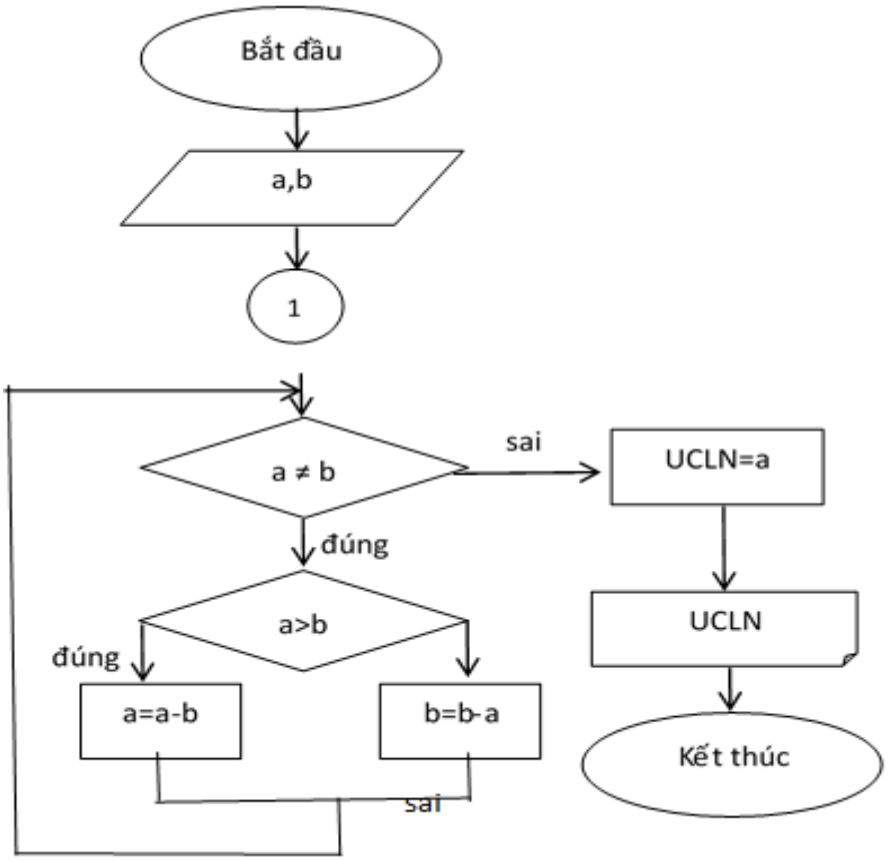
Mô tả hoạt động

Nếu <điều kiện> là đúng thì sẽ thực hiện <công việc>, sau khi thực hiện xong <công việc> thì sẽ quay trở lại kiểm tra <điều kiện> và bắt đầu một vòng lặp mới,…cứ như vậy vòng lặp tiếp tục đến khi nào <điều kiện> nhận giá trị sai thì kết thúc.

Rõ ràng với cấu trúc này thì trong <công việc> phải có một giai đoạn mà ở đó <điều kiện> phải mang giá trị sai để vòng lặp kết thúc như đã phân tích ở trên.

Ví dụ 1.3:

Biểu diễn thuật toán tìm ước số chung lớn nhất của 2 số nguyên dương a,b (sử dụng ngôn ngữ sơ đồ khối).

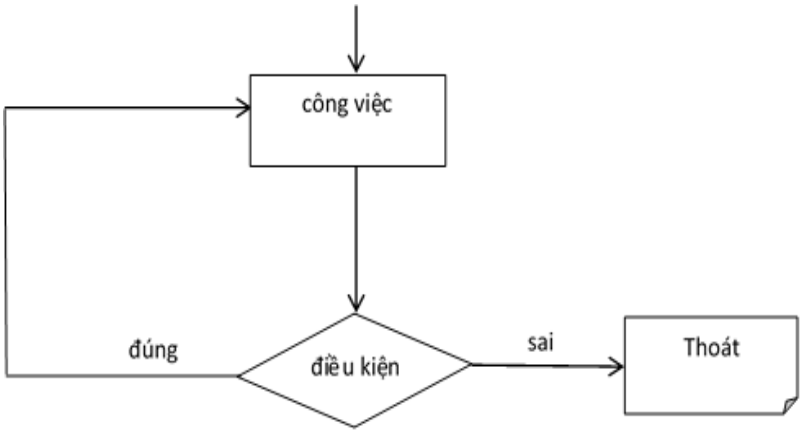


Hình 1.9. *Sơ đồ khối của ví dụ 1.3*

Một số bộ test:

	test 1	test 2	test 3	test 4
a	18	4	101	3
b	12	18	103	1000000
Kết quả	6	2	1	1

Dạng 3:



Hình 1.10. *Sơ đồ của cấu trúc lặp dạng 3*

Mô tả hoạt động

Trước hết <công việc> sẽ được thực hiện, sau đó thực hiện <điều kiện>. Nếu <điều kiện> là đúng thì lại bắt đầu một vòng lặp mới,…cứ như vậy đến khi <điều kiện> sai thì cấu trúc lặp này kết thúc.

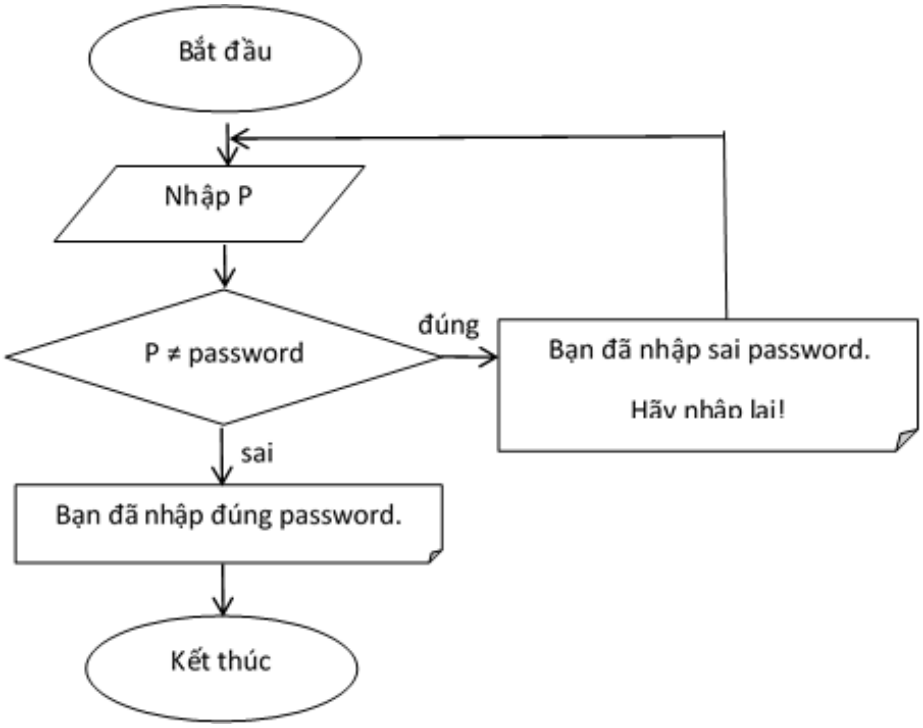
Tương tự như sơ đồ ở dạng 2, sau một số lần lặp <điều kiện> phải mang giá trị sai để vòng lặp kết thúc

Lưu ý:

Cấu trúc lặp dạng 3 thường dùng khi cần nhập dữ liệu và cần kiểm tra điều kiện hợp lệ của dữ liệu nhập.

Ví dụ 1.4:

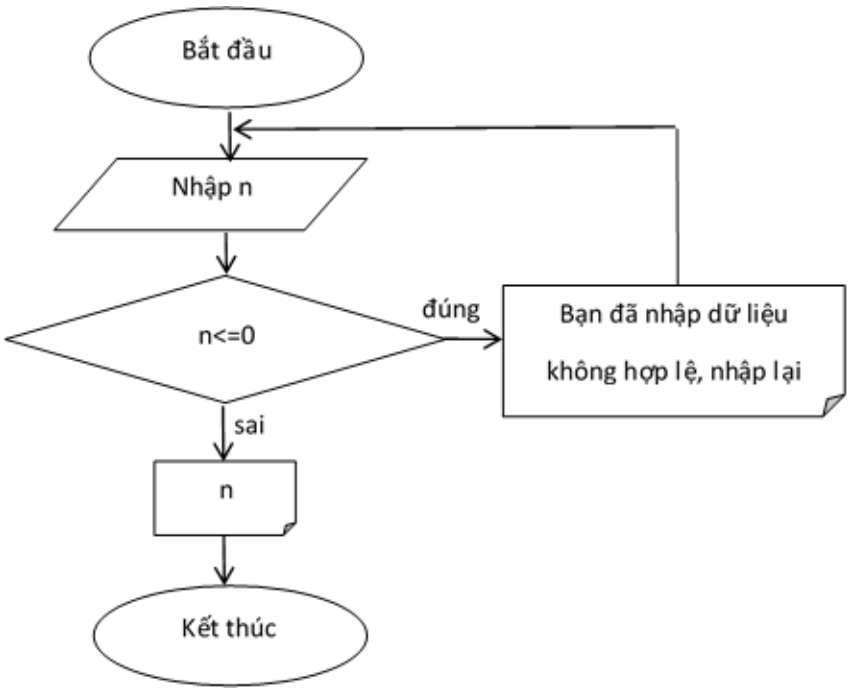
Viết chương trình kiểm tra password (sử dụng ngôn ngữ sơ đồ khối).



Hình 1.11. *Sơ đồ khối của ví dụ 1.4*

Ví dụ 1.5:

Viết chương trình nhập một số nguyên dương n (sử dụng ngôn ngữ sơ đồ khối).



Hình 1.12. *Sơ đồ khối của ví dụ 1.5*

1.4. NGÔN NGỮ MÃ GIẢ

1.4.1. Tập ký hiệu (mã giả theo ngôn ngữ C/C++)

Bắt đầu đoạn lệnh: {

Kết thúc đoạn lệnh: }

Lệnh nhập dữ liệu: cin>>

Lệnh xuất dữ liệu : cout<<

Lệnh xử lý dữ liệu: =

Biểu thức điều kiện: if (<điều kiện>)

Ghi chú: //

1.4.2. Cấu trúc rẽ nhánh

Dạng 1

```
if (<điều kiện>)
        <công việc>;
```

Mô tả hoạt động

Nếu <điều kiện> là đúng thì thực hiện <công việc>, nếu <điều kiện> là sai thì sẽ không thực hiện <công việc>.

Lưu ý: Nếu <công việc> có từ hai câu lệnh đơn trở lên thì cần đặt <công việc> giữa hai dấu { }.

Dạng 2

```
if (<điều kiện>)
        <công việc 1>;
else
        <công việc 2>;
```

Mô tả hoạt động

Nếu <điều kiện> là đúng thì sẽ thực hiện <công việc 1>, ngược lại nếu < điều kiện> là sai thì <công việc 2> sẽ được thực hiện.

Ví dụ 1.6:

Hãy biểu diễn thuật toán giải phương trình (Pt) bậc hai dạng $ax^2 + bx+c = 0$

(sử dụng ngôn ngữ mã giả)

```
cin>>a>>b>>c;
if (a=0)  //Pt bậc nhất
     if (b=0)
          if (c=0)
               cout<<"Pt có vô số nghiệm số";
          else   // b=0 và c≠0
               cout<<"Pt có vô nghiệm";
          else      // a=0 và b≠0
     {
          x=-c/b;
          cout << "Pt có nghiệm duy nhất: "<<x;
     }
else      //a≠0
     {
     Delta=b*b-4*a*c;
     if  (Delta<0)
          cout<<" Pt có vô nghiệm";
     else // Delta>=0
          if (Delta=0)
          {
               x=-b/(2*a);
               cout << "Pt có nghiệm kép: "<<x;
          }
          else
          {
               x1 = (-b-sqrt(Delta))/(2*a);      x2 = (-b+sqrt(Delta))/(2*a);
               cout << "Pt có hai nghiệm: "<< "x1="<<x1<<
"  x2="<<x2;
          }
}    // kết thúc else     //a≠0
```

1.4.3. Cấu trúc lặp

Dạng 1

for (<biểu thức 1>; <điều kiện>; <biểu thức2>)

<công việc>;

Mô tả hoạt động

Đầu tiên sẽ thực hiện <biểu thức 1>, sau đó kiểm tra <điều kiện>. Nếu <điều kiện> là đúng thì sẽ thực hiện <công việc> rồi đến <biểu thức 2>; sau khi thực hiện xong <biểu thức 2> thì cấu trúc này quay trở lại kiểm tra <điều kiện> và bắt đầu một vòng lặp mới,…cứ thế vậy đến khi <điều kiện> nhận giá trị sai thì cấu trúc lặp này sẽ kết thúc.

Ví dụ 1.7:

Biểu diễn thuật toán tính tổng $s= 1^2+2^2+...+n^2$ (sử dụng ngôn ngữ mã giả).

```
cin>> n;
if (n<0)
     cout <<"n không hợp lệ";
else
{
     s=0;
     for(i=1; i<=n; i++)
          s=s+i*i;
     cout<<s;
}
```

Dạng 2

while (<điều kiện>)

<công việc>;

Mô tả hoạt động

Nếu <điều kiện> là đúng thì sẽ thực hiện <công việc>, sau đó quay trở lại thực hiện <điều kiện>. Vòng lặp while này sẽ kết thúc khi <điều kiện> là sai.

Ví dụ 1.8:

Biểu diễn thuật toán tìm ước số chung lớn nhất của 2 số nguyên dương a,b (sử dụng ngôn ngữ mã giả).

```
cin>>a>>b;
r=a mod b; // r số dư của phép chia a cho b
while (r ≠ 0)
{
     a=b;
     b=r;
```

```
        r= a mod b;
}
cout<<b;
```

Dạng 3

```
do
{
        <công việc>;
}
while (<điều kiện>);
```

Mô tả hoạt động

Trước hết <công việc> sẽ được thực hiện, sau đó kiểm tra <điều kiện>. Nếu <điều kiện> là đúng thì quay trở lại thực hiện <công việc> và bắt đầu một vòng lặp mới. Vòng lặp do while này sẽ kết thúc khi <điều kiện> nhận giá trị sai.

Ví dụ 1.9:

Viết chương trình kiểm tra password (sử dụng ngôn ngữ mã giả).

```
do
{     cin>>p;
      if (p ≠ password)
            cout<< "Bạn đã nhập sai password. Hãy nhập lại!"
}
while (p ≠ password);
cout<<"Bạn đã nhập đúng password";
```

BÀI TẬP

Hãy trình bày thuật toán cho các bài toán sau bằng ngôn ngữ sơ đồ khối hay mã giả.

BT1-1.

Nhập vào bán kính của một hình tròn. Hãy tính chu vi và diện tích của đường tròn đó.

BT1-2.

Nhập vào một số xe (gồm 5 chữ số). Hãy cho biết số xe đó có bao nhiêu nút ?

BT1-3.

Nhập vào hai số nguyên. Tính min và max của hai số nguyên đó.

BT1-4.

Nhập vào 4 số nguyên *a,b,c,d*. Hãy sắp xếp 4 số trên theo thứ tự tăng dần.

BT1-5.

Nhập vào số tự nhiên *n*. Hãy kiểm tra xem *n* có phải là số hoàn chỉnh hay không ? (*n* là số hoàn chỉnh nếu nó bằng tổng các ước số thực sự của nó). Ví dụ 6 là số hoàn chỉnh vì 6=1+2+3.

BT1-6.

Nhập vào số tự nhiên *n*. Kiểm tra xem *n* có phải là số chính phương hay không ?

Ví dụ 1,4,9,16 là các số chính phương.

BT1-7.

Nhập vào số tự nhiên *n*. Kiểm tra xem *n* có phải là số nguyên tố hay không ?

Ví dụ 2,3,5,7 là các số nguyên tố.

BT1-8.

a. Nhập vào số tự nhiên *n*. Tìm các số nguyên tố nhỏ hơn hoặc bằng *n*.

Ví dụ: *n*=8 thì kết quả là: 2,3,5,7

b. Nhập vào số tự nhiên *n*. Tìm *n* số nguyên tố đầu tiên.

Ví dụ: *n*=8 thì kết quả là:2,3,5,7,11,13,17,19

BT1-9.

Trong mặt phẳng tọa độ OXY cho ba điểm $A(x_a,y_a)$,$B(x_b,y_b)$,$C(x_c,y_c)$ và một điểm $M(x_m,y_m)$. Xác định xem M nằm trong, nằm ngoài hay nằm trên các cạnh tam giác ABC ?

Ví dụ:

xa=0,ya=0; xb=0,yb=4; xc=4,yc=0,xm=0,ym=6 thì M nằm ngoài tam giác ABC

xa=0,ya=0; xb=0,yb=4; xc=4,yc=0,xm=1,ym=2 thì M nằm trong tam giác ABC

xa=0,ya=0; xb=0,yb=4; xc=4,yc=0,xm=2,ym=0 thì M nằm trên cạnh tam giác ABC

BT1-10.

Cho số tự nhiên *n* và một số thực *x*. Hãy tính tổng sau:

$S(n,x) = \sin(x) + \sin(\sin x) + ... + \sin(\sin(...(\sin(\sin x))...))$

n lần sin

CHƯƠNG 2. TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH C

2.1. CÁC KHÁI NIỆM CƠ BẢN

Trong mục này sẽ giới thiệu những thành phần cơ bản của ngôn ngữ lập trình C là : Tập ký tự, từ khoá và tên.

Tập ký tự dùng trong ngôn ngữ C : Chữ cái hoa : **A, B,…, Z**; chữ cái thường : **a, b, …z** ; chữ số : **0, 1,…9** ; các ký hiệu : **+ - * / = ()** ; ký tự gạch nối : **_** và các ký hiệu khác như : **. , ; : [ ] { } ? ! \ & | % # $ @** ,…Dấu cách là một khoảng trống dùng để tách các từ.

Khi viết chương trình ta không được sử dụng bất cứ ký hiệu nào khác ngoài các ký tự trên

Từ khoá : Từ khóa là từ có ý nghĩa xác định dùng để khai báo các kiểu dữ liệu, viết các toán tử và viết câu lệnh… Trong C có các từ khoá sau:

asm	const	else	for	interrupt	return	sizeof	void
break	continue	enum	goto	long	short	switch	volatile
case	default	extern	huge	near	static	typedef	while
cdecl	do	far	if	pascal	struct	union	
char	double	float	int	register	signed	unsigned	

Các từ khóa phải viết bằng ***chữ thường.*** Không được dùng từ khoá để đặt tên cho các hằng, biến, mảng, hàm, …

Tên : Khái niệm tên rất quan trọng trong quá trình lập trình, nó không những thể hiện rõ ý nghĩa trong chương trình mà còn dùng để xác định các đối tượng khác nhau khi thực hiện chương trình. Tên thường được đặt cho hằng, biến, mảng, hàm, con trỏ, nhãn, tên file, tên struct … Chiều dài tối đa của tên là 32 ký tự.

Tên biến hợp lệ là một chuỗi ký tự liên tục gồm: ***Ký tự chữ, số và dấu gạch dưới***. Ký tự đầu của tên phải là ***chữ hoặc dấu gạch dưới***. Khi đặt tên không được đặt trùng với các từ khóa.

Ví dụ 2-1.

Các tên hợp lệ : delta, a_1, x1, Num_ODD,

Các tên không hợp lệ :

3xyz _1	*(do ký tự đầu là số)*
num-odd	*(do sử dụng dấu gạch ngang)*
r#3	*(do sử dụng ký tự #)*
int	*(do đặt tên trùng với từ khóa)*
del ta	*(do có khoảng trắng)*
f(x)	*(do có dấu ngoặc tròn)*

Lưu ý :

Trong các tên, chữ hoa và chữ thường được xem là khác nhau như vậy ABC và abc là khác nhau. Trong ngôn ngữ C thường dùng chữ hoa để đặt tên cho các hằng và dùng chữ thường cho các đối tượng khác

2.1.1. Kiểu dữ liệu số

Ngôn ngữ C/C++ có nhiều kiểu dữ liệu; trong mục này chúng ta chỉ tìm hiểu về kiểu dữ liệu số nguyên và kiểu dữ liệu số thực.

Kich thước của các kiểu dữ liệu : Mỗi kiểu dữ liệu chỉ có khả năng lưu trữ các giá trị nằm trong một giới hạn nhất định. Giới hạn này tuỳ thuộc vào kích thước biểu diễn của kiểu dữ liệu đó.

Kiểu số nguyên

Các số nguyên thuộc tập ℤ được định nghĩa với các từ khoá, kích thước và miền giá trị như sau (các thông số trong mục chương này được lấy trên môi trường BC3.1).

Từ khóa	Kích thước	Miền giá trị
unsigned int	16 bits	0 .. 65535
int	16 bits	-32768 .. 32767
unsigned long	32 bits	0 .. 4294967295
long	32bits	-2147483648 .. 2147483647

Các phép toán số học đối với số nguyên (đã sắp theo độ ưu tiên)

Toán tử	Ý nghĩa	Ví dụ
+	cộng	
-	trừ	
*	nhân	
/	chia lấy thương nguyên	5 / 2 = 2
%	chia lấy số dư	6 % 4 = 2

Kiểu số thực

Từ khoá	**Kích thước**	**Miền giá trị của trị tuyệt đối**
float	32 bits	3.4 * (10^{-38}) đến 3.4 * (10^{38})
double	64 bits	1.7 * (10^{-308}) đến 1.7 * (10^{308})
long double	80 bits	3.4 * (10^{-4932}) đến 1.1 * (10^{4932})

Lưu ý: Trên miền số thực không có phép toán chia lấy dư.

2.1.2. Hằng

Hằng là những giá trị cố định đã biết trước, được định nghĩa ở đầu chương trình, và không thể thay đổi được trong chương trình. Trong C, mỗi hằng số đều có một kiểu dữ liệu của nó. Ta có thể có các loại hằng sau :

+ Hằng số : 10, -2
+ Hằng thực : 3.14 -2.3 1.2 E-3 .12e3
+ Hằng ký tự : ‘a’ ‘>’ ‘0’
+ Hằng chuỗi : “Đây là hằng chuỗi “ “a”
+ Biểu thức hằng : 2*3-0.1 ‘a’-‘A’

Khai báo hằng: Có 2 cách khai báo hằng phổ biến là:

```
#define MAX 10000
const   int MAX = 10000;
```

Lưu ý :

- Mỗi câu lệnh có thể viết trên một hay nhiều dòng nhưng phải kết thúc bằng dấu; Trong khi lập trình cần phải ghi chú để giải thích các biến, hằng, thao tác xử lý giúp cho chương trình rõ ràng dễ hiểu, dễ nhớ, dễ sửa chữa và để người khác đọc vào dễ hiểu. Trong C có các ghi chú sau: // hoặc /* nội dung ghi chú */

- Không nên nhầm lẫn giữa hằng ký tự và hằng chuỗi :

"a" được lưu trữ gồm 2 ký tự là a và \0

'a' được lưu trữ là a mà thôi

- Có một số ký tự đặc biệt không in được trên màn hình : '\n' (xuống dòng); '\0' (ký tự NULL); '\r' (Enter)…

Ví dụ 2-2.

```
int main()
{
        int a, b;                       //khai bao 2 bien a,b  kieu  int
        a = 1;                          //gan 1 cho a
        b =3;                           //gan 3 cho b
   /* thuat toan tim so lon nhat la neu a lon hon b thi a lon nhat nguoc lai b lon
      nhat */
        if (a > b)      printf("max: %d", a);
        else            printf("max: %d", b);
}
```

Khi biên dịch chương trình, C gặp cặp dấu ghi chú sẽ không dịch ra ngôn ngữ máy.

Tóm lại, đối với ghi chú

dạng // dùng để ghi chú một hàng và

dạng /* …. */ có thể ghi chú một hàng hoặc nhiều hàng.

2.1.3. Biến

Mỗi biến trong chương trình C đều phải được khai báo trước khi sử dụng chúng. Việc khai báo này giúp cho chương trình dịch biết được kích thước của biến đó, địa chỉ của chúng và ghi nhận sự có mặt của chúng trong chương trình

Khai báo biểu của biến có dạng :

Kiểu danh_sách_tên_biến ;

Ví dụ 2-3.

```
int     x, y, z;
char    dem;
```

2.1.4. Biểu thức

Biểu thức là một sự kết hợp hợp lệ, của những phép toán thực hiện trên các biến, các hằng hoặc giá trị của các hàm. Biểu thức được phân loại theo kiểu giá trị : nguyên và thực. Trong các mệnh đề logic, biểu thức được phân thành giá trị khác không (đúng) và giá trị bằng 0 (sai).

2.2. LỆNH XUẤT DỮ LIỆU, LỆNH NHẬP DỮ LIỆU, LỆNH GÁN

2.2.1. Lệnh xuất dữ liệu với printf()

Xuất ra màn hình giá trị của các biểu thức theo một khuôn dạng nào đó.

printf (<các hằng chuỗi định dạng>, <biểu thức 1>, <biểu thức 2>,. . .);

Hằng chuỗi định dạng chứa mã định dạng sau đây: **%[width].prec type**; trong đó:

width khoảng trống dành để chứa biểu thức cần xuất ra màn hình.

prec định kích thước của phần thập phân.

type định kiểu của biểu thức theo sau hằng chuỗi định dạng

- **d** số nguyên hệ 10
- **ld** số long
- **c** ký tự
- **s** chuỗi ký tự
- **f** số thực float
- **lf** số thực double

chẳng hạn để xuất hai giá trị a và b lên màn hình, trong đó a là kiểu số nguyên và b là kiểu số thực, ta có thể viết lệnh sau:

printf("%4d %6.2f", a, b);

Khi đó câu lệnh này sẽ dành 4 khoảng trắng để ghi giá trị a và 6 khoảng trắng để ghi giá trị b trong đó đã tính cả dấu chấm thập phân và 2 chữ số phần thập phân.

*Hằng **\n** thường được sử dụng để đưa con trỏ về đầu dòng tiếp theo để kết quả xuất ra dễ đọc hơn, hằng **\n** tương đương với hằng **endl**.*

2.2.2. Lệnh nhập dữ liệu với scanf()

Nhập các giá trị từ bàn phím và lưu giữ vào nội dung của các địa chỉ biến theo cú pháp sau:

scanf (<các hằng chuỗi định dạng>, <địa chỉ biến 1>, <địa chỉ biến 2>,. . .);

Hằng chuỗi dịnh dạng chứa mã định dạng sau đây:

d	số nguyên hệ 10
ld	số long
c	ký tự
s	chuỗi ký tự
f	số thực float
lf	số thực double

Để xuất, nhập dữ liệu chúng ta hoàn toàn có thể sử dụng hai lệnh trong thư viện ***stdio.h*** là ***printf, scanf*** như đã đề cập ở các ví dụ trên. Tuy nhiên chúng ta có thể sử dụng các lệnh ***cout, cin*** trong thư viện ***iostream.h*** để xuất nhập dữ liệu .

2.2.3. Lệnh gán

Lệnh gán dùng để gán giá trị cho các biến, cú pháp của lệnh gán như sau:

tên biến = <biểu thức>;

trong đó <**biểu thức**> có thể là một hằng, một biến hay một công thức. Trong một phép gán, biến cần được gán xuất hiện bên trái dấu = ; còn hằng, biến hoặc biểu thức để gán thì xuất hiện bên phải của dấu = . Thông thường hai vế trong câu lệnh gán phải có cùng kiểu. Tuy nhiên khi gán một giá trị nguyên cho một biến thực, thì giá trị nguyên sẽ được chuyển đổi thành giá trị thực tương ứng, còn khi thực hiện phép gán một số thực cho một biến nguyên, thì phần nguyên của số thực đó sẽ được gán cho biến (một số ngôn ngữ lập trình đòi hỏi hai vế của câu lệnh gán phải có cùng kiểu).

2.2.4. Lệnh xuất dữ liệu với cout

cout << <biểu thức1> [<<biểu thức2>] [...]

Lệnh này cho phép xuất giá trị của các biểu thức ra màn hình. Các ***<biểu thức i>*** cũng có thể là các hàm, các hằng, các biến,…

Một số lệnh sau đây (trong thư viện ***iomanip.h***) có thể được sử dụng chung với lệnh ***cout***:

int setw(int n); quy định độ rộng trên màn hình để xuất giá trị là n cột.

int setprecision(int n); quy định số lượng chữ số sau dấu chấm thập phân cần xuất ra màn hình là n.

2.2.5. Lệnh nhập dữ liệu với cin

cin >> <biến 1> [>> <biến 2>] [...];

Lệnh này cho phép nhập từ bàn phím giá trị cho các biến (riêng biến kiểu chuỗi không sử dụng lệnh ***cin*** để nhập), sau mỗi lần nhập giá trị phải nhấn enter hoặc phím khoảng cách.

2.3. TOÁN TỬ

2.3.1. Toán tử so sánh

toán tử	định nghĩa
<	nhỏ hơn
>	lớn hơn
==	bằng
<=	nhỏ hơn hoặc bằng
>=	lớn hơn hoặc bằng
!=	khác nhau

2.3.2. Toán tử logic

Phép phủ định !

Phép hội &&

Phép tuyển ||

Kết quả thực hiện các phép toán này được mô tả trong bảng sau:

a	b	!a	a && b	a ||b
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

2.3.3. Toán tử tăng, giảm

Ngôn ngữ C/C++ đưa ra hai phép toán một ngôi để tăng hoặc giảm các biến (nguyên và thực). Toán tử tăng ++ sẽ cộng 1 vào toán hạng của nó, toán tử giảm -- sẽ trừ đi 1. Chẳng hạn nếu n đang có giá trị bằng 5 thì sau phép tính ++ **n**, n có giá trị 6, còn sau phép tính **--n**, n có giá trị 4.

Dấu phép toán ++ và -- có thể đứng trước hoặc đứng sau toán hạng, như vậy có thể có tất cả 4 cách viết sau: **++n, n++, --n, n--**.

Sự khác nhau giữa ++ **a** và **a** ++ ở chỗ: trong khi phép **n** ++ thì **n** được tăng *sau khi* giá trị của nó đã được tính trong biểu thức, còn trong phép ++**n** thì **n** được tăng trước khi giá trị của nó được tính trong biểu thức. Sự khác nhau giữa **--n** và **n--** cũng tương ứng như vậy.

Chẳng hạn xét đoạn lệnh sau:

```
int a = 1; b = 3;
cout  << ++a + b++;
```

thì kết quả xuất là 5

2.3.4. Toán tử điều kiện

(<biểu thức điều kiện> ? <biểu thức 1> : <biểu thức 2>)

Ý nghĩa: Nếu <**biểu thức điều kiện**> là đúng thì trả về giá trị của <**biểu thức** 1> ngược lại trả về giá trị của <**biểu thức** 2>.

Chẳng hạn để tìm giá trị lớn nhất của 2 số a,b thì có thể viết đoạn lệnh như sau:

```
max = a > b ? a : b ;
```

Lưu ý:

*Các **<biểu thức i>** ở đây có thể là một đoạn lệnh, nhưng chúng ta không nên lạm dụng toán tử này cho những tính toán phức tạp để tránh làm cấu trúc chương trình rối rắm thêm. Tuy nhiên câu lệnh này lại thường được sử dụng khi cần trả về một điều kiện đúng / sai đơn giản.*

2.3.5. Thứ tự ưu tiên của các toán tử

Khi tính giá trị của một biểu thức, ngôn ngữ C/C++ quy ước thứ tự ưu tiên của các toán tử từ cao đến thấp, trên cùng một hàng thì cùng cấp như sau:

phép gọi hàm

!

/ , * , \ , %, &&

+, -, ||

< , >, ==, < =, > =, !=

Đoạn lệnh sau cho kết quả là 8.

```
int a =2, b = 3;
int m = a++ > b-- ? –a : b++;
cout << m+a+b;
```

Đoạn lệnh sau cho kết quả là 13.

```
int a = 5, b = 9;
int m = a>b ? a++ : b++;
int n = a<b ? --a : --b;
cout << --n + (++m);
```

2.3.6. Vấn đề chuyển đổi kiểu dữ liệu

Khi thực hiện các phép tính toán số học thường xảy ra nhu cầu chuyển đổi kiểu các toán hạng để tính toán theo cú pháp chuyển đổi kiểu như sau:

(kiểu dữ liệu) biểu thức

Chẳng hạn với a,b là hai số nguyên. Cần xuất giá trị của phân số a / b thì cần phải chuyển đổi dữ liệu của kết quả để có kết đúng. Có bốn cách sau đây để thực hiện: **(float)a / b, float(a) / b, a / float(b)**; riêng trường hợp nếu a hoặc b là hằng số thì việc chuyển a hoặc b thành số thực có thể thực hiện bằng cách ghi thêm **.0** phía sau hằng đó (ví dụ **1 / b** ghi thành **1.0 / b**).

2.4. CHÚ THÍCH TRONG CHƯƠNG TRÌNH

Khi viết chương trình nên đưa vào các dòng chú thích để chương trình dễ đọc, dễ hiểu . Điều này cũng giúp cho việc sửa đổi nâng cấp chương trình về sau được thuận lợi. Có hai cách sau để ghi lời chú thích:

Cách 1: đặt dấu // trước dòng muốn chú thích

Cách 2: bao đoạn lệnh cần chú thích giữa hai dấu /* … */

Lời chú thích có thể đặt ở bất kỳ vị trí nào trong chương trình. Nếu chương trình dài phức tạp nên đưa chú thích vào trước mỗi đoạn lệnh hoặc trước mỗi chương trình con.

2.5. CẤU TRÚC CỦA MỘT CHƯƠNG TRÌNH C/C++ ĐƠN GIẢN

Một chương trình trên C/C++ thường có cấu trúc như sau:

Khai báo các thư viện được sử dụng trong chương trình.

Định nghĩa các hằng số

Khai báo các biến toàn cục

Khai báo các tiêu đề hàm

Chương trình chính

Thiết kế chi tiết từng hàm tự tạo

Một chương trình đơn giản mẫu được mô tả sau đây.

Ví dụ 2-4.

Trong mặt phẳng tọa độ OXY cho 3 điểm A,B,C lần lượt có tọa độ là (xa,ya), (xb,yb), (xc,yc). Hãy tính chu vi, diện tích và tọa độ trọng tâm của tam giác này.

Hướng dẫn:

Gọi cv, dt lần lượt là chu vi,diện tích và a,b,c lần lượt là độ dài của các cạnh BC, AC, AB, ta có:

$$a = \sqrt{(xc - xb)^2 + (yc - yb)^2}$$

$$b = \sqrt{(xc - xa)^2 + (yc - ya)^2}$$

$$c = \sqrt{(xb - xa)^2 + (yb - ya)^2}$$

Chu vi của tam giác ABC:

cv = a + b + c

Áp dụng công thức Heron tìm diện tích của tam giác khi biết chiều dài 3 cạnh của tam giác, ta có:

$S = \sqrt{p(p-a)(p-b)(p-c)}$ với p là nửa chu vi của tam giác: p = cv / 2

Gọi (xg,yg) là tọa độ trọng tâm của tam giác ABC, ta có công thức:

xg = (xa+xb+xc)/3

yg = (ya+yb+yc)/3

```
/* tính chu vi, diện tích và tọa độ trọng tâm của tam giác */
1. #include <conio.h>
2. #include <iostream.h>
3. #include <math.h>
4. int main()
5. {
6. clrscr();// lệnh xóa màn hình
7. int    xa,ya,xb,yb,xc,yc;
8. double  a,b,c,p,cv,s,xg,yg;
9. cout << "Nhap toa do cua diem A(xa,yq):";cin >> xa >> ya;
10.  cout << "Nhap toa do cua diem B(xb,yb):";cin >> xb >> yb;
11.  cout << "Nhap toa do cua diem C(xc,yc):";cin >> xc >> yc;
12.  /*  hay ta dùng khi khai báo #include <stdio.h>
13.  printf ("Nhap toa do cua diem A(xa,ya):"); scanf ("%d,
   %d",&xa. &ya);
14.  printf ("Nhap toa do cua diem A(xb,yb):"); scanf ("%d,
   %d",&xb, &yb);
15.  printf ("Nhap toa do cua diem A(xc,yc) :"); scanf ("%d,
   %d",&xc . &yc);
16.  */
17.  a = sqrt((xb-xc)*(xb-xc)+(yb-yc)*(yb-yc));
18.  b = sqrt((xa-xc)*(xa-xc)+(ya-yc)*(ya-yc));
19.  c = sqrt((xa-xb)*(xa-xb)+(ya-yb)*(ya-yb));
20.  cv = a+b+c;
21.  p = (a+b+c)/2;
22.  s = sqrt(p*(p-a)*(p-b)*(p-c));
23.  xg = (xa+xb+xc)/3.0;              // chuyển mẫu số về số
   thực
24.  yg = (ya+yb+yc)/3.0;
25.  cout << "\nChu vi cua tam giac la   : " << cv;
26.  cout << "\nDien tich cua tam giac la: " << s;
27.  cout << "\nToa do trong tam cua tam giac la  : "<< xg <<
   "," << yg;
28.  /*  hay ta dùng khi khai báo #include <stdio.h>
```

```
29.  printf ("\n Chu vi cua tam giac la   : %d  " , cv) ;
30.  printf ( "\nDien tich cua tam giac la: %f  ", s);
31.  printf ( "\nToa do trong tam cua tam giac la  : (%f, %f)
   ", xg,  yg); */
32.  getch();       // lệnh dừng chương trình
33.  }
```

Một số bộ test:

Test	A	B	C	Chu vi	Diện tích	Trọng tâm
test 1	(0;3)	(0;-3)	(4;0)	16	12	(1.33;0)
test 2	(-3;0)	(3;0)	(0;4)	16	12	(0;1.33)
test 3	(0;4)	(4;0)	(0;0)	13.66	8	(1.33;1.33)

Ví dụ 2-5.

Nhập một số nguyên dương n có 3 chữ số. Hãy tính tổng các chữ số của số đó.

Hướng dẫn:

Tìm cách tách các chữ số của số n, sau đó tính tổng các chữ số đó:

chữ số đơn vị = **n %10**, chữ số hàng chục=**n % 100 /10**, chữ số hàng trăm = **n / 100**.

```
/* Tính tổng các chữ số của số nguyên có 3 chữ số */
1. #include <iostream.h>
2. int main()
3. {
4. int n;
5. cin >> n;
6. int hangdv = n%10;
7. int hangchuc = n%100/10;
8. int hangtram = n/100;
9. cout << hangdv + hangchuc + hangtram;
10.  }
```

Một số bộ test:

	test1	test2	test3	test4
n	197	135	111	100
Kết quả	17	9	3	1

Ví dụ 2-6.

Nhập vào 3 số thực a,b,c. Tìm giá trị lớn nhất của 3 số đó.

Hướng dẫn:

max=a;

Nếu (b > max) thì max=b;

Nếu (c > max) thì max=c;

```
     /* Tìm giá trị lớn nhất của 3 số  */
1. #include <iostream.h>
2. int main()
3. {
4. float a,b,c;
5. cin >> a >> b >> c;
6. float max=a;
7. max = b>max ? b : max;
8. max = c>max ? c : max;
9. cout << max;
10.  }
```

Một số bộ test:

Test	a	b	c	max
test 1	-2	-4	-6	-2
test 2	1.1	2.8	2.3	2.8
test 3	40000	30	52000	52000

Toán tử điều kiện có thể được sử dụng để giải các bài toán có cấu trúc rẽ nhánh đơn giản như giải phương trình, hệ phương trình,…; cấu trúc if else trong chương tiếp theo sẽ giải quyết vấn đề vừa nêu trên một cách hiệu quả hơn.

BÀI TẬP

Viết chương trình hoàn chỉnh cho các bài toán sau đây

(mỗi bài tập sinh viên nên ghi chú lại các bộ test mẫu để kiểm thử chương trình).

BT2-1.

Cho tam giác *ABC* có chiều dài 3 cạnh lần lượt là *a,b,c* (*a=BC*, *b=AC*, *c=AB*). Hãy tìm chu vi, diện tích của tam giác *ABC* và độ dài đường cao *AH*.

Ví dụ: *a*=4, *b*=5, *c*=3 thì chi vi là 12, diện tích là 6, đường cao *AH* là 3

BT2-2.

Nhập vào *t* giây, hãy đổi *t* giây ra dạng giờ – phút – giây.

Ví dụ: *t*=3700 thì kết quả là : 1*h* 1*m* 40*s*

BT2-3.

Một số nguyên dương có *k* chữ số được gọi là số Armstrong khi nó bằng tổng lũy thừa *k* của từng chữ số. Nhập vào một số nguyên dương *k* có 3 chữ số, hãy kiểm tra xem *k* có phải là số Armstrong hay không ?

Ví dụ: *k*=153 là số Armstrong vì $153=1^3+5^3+3^3$

BT2-4.

Cho số tự nhiên *n* có ba chữ số.

a. Tính tổng các chữ số của số *n* ?

b. Kiểm tra *n* có đối xứng hay không ?

BT2-5.

Cho một hình chữ nhật có cách cạnh song song với các trục tọa độ, tọa độ góc dưới trái là $A(x_1,y_1)$ và tọa độ góc trên phải là $B(x_2,y_2)$. Hãy kiểm tra xem điểm $M(x,y)$ có nằm trong hình chữ nhật trên hay không ? (nằm trên cạnh được xem như nằm trong). Giả sử tọa độ các điểm là các số nguyên.

Ví dụ: $(x_1,y_1) = (0,0)$, $(x_2,y_2) = (6,4)$, $M(x,y)=(-1,1)$ thì *M* nằm ngoài hình chữ nhật.

BT2-6.

Trong mặt phẳng tọa độ *OXY*, cho hai hình chữ nhật có các cạnh song song với các trục tọa độ: Hình chữ nhật thứ nhất có đỉnh dưới trái là *A* và đỉnh trên phải là *B*; hình chữ nhật thứ hai có đỉnh dưới trái là *C* và đỉnh trên phải là *D*. Giả sử hoành độ và tung độ của các đỉnh là các số nguyên.

Hãy viết chương trình tìm tọa độ góc dưới trái (*E*) và tọa độ góc trên phải (*F*) của hình chữ nhật bé nhất có các cạnh song song với các trục tọa độ và chứa cả hai hình chữ nhật đã cho.

Ví dụ: với dữ liệu nhập là *A*(1;0), *B*(5;3), *C*(3;-2), *D*(6;1) thì kết quả là *E*(1;-2), *F*(6;3).

BT2-7.

Trong mặt phẳng tọa độ *OXY*, cho hai hình chữ nhật có các cạnh song song với các trục tọa độ; mỗi hình chữ nhật biết tọa độ đỉnh dưới trái và đỉnh trên phải. Hãy tìm tâm và đường kính của đường tròn nhỏ nhất bao cả hai hình chữ nhật trên.

BT2-8.

Trong mặt phẳng tọa độ *OXY* cho 2 điểm *A, B*; điểm *A* có tọa độ là (*x*,*y*), điểm *B* có tọa độ là (*z*,0) với *x, y, z* là các số dương. Hãy tính diện tích tam giác *OAB*.

BT2-9.

Trong mặt phẳng tọa độ *OXY* cho 3 điểm *A,B,C* lần lượt có tọa độ là (x_a,y_a), (x_b,y_b), (x_c,y_c). Hãy tìm tọa độ trọng tâm, diện tích đường tròn nội tiếp, diện tích đường tròn ngoại tiếp của tam giác *ABC*.

BT2-10.

Tìm sin, cosin, tan, cotan của góc *alpha* độ, với *alpha* là dữ liệu nhập vào.

CHƯƠNG 3. CÁC CẤU TRÚC ĐIỀU KHIỂN

3.1. CẤU TRÚC if else

Đây là lệnh rẽ nhánh theo điều kiện. Cấu trúc **if** cho phép lựa chọn một trong hai nhánh rẽ tùy thuộc biểu thức điều kiện là bằng không hay khác không. Lệnh **if** có hai dạng.

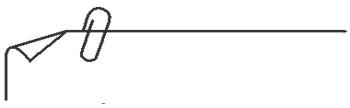
Dạng 1

if (<biểu thức >)

<khối lệnh>;

Mô tả hoạt động

Khi gặp lệnh if dạng 1, <biểu thức> được tính toán. Nếu kết quả của <biểu thức> có giá trị khác 0 thì chương trình sẽ thực hiện <khối lệnh>; nếu kết quả của <biểu thức> có giá trị bằng 0 thì chương trình sẽ bỏ qua <khối lệnh > để thực hiện câu lệnh tiếp theo sau if.



i. *<Biểu thức> có thể là kiểu nguyên hay kiểu số thực.*

ii. *<Biểu thức> sau if là biểu thức điều kiện và phải được đặt trong cặp dấu ngoặc ().*

iii. *Khối lệnh là một nhóm nhiều lệnh có thứ tự được đặt giữa một cặp ngoặc nhọn ({}) , mà về mặt ngữ pháp cũng được xem như là một lệnh. Khối lệnh thường được dùng để mô tả lệnh của một lệnh điều khiển khác, vì thông thường các lệnh điều khiển của ngôn ngữ C đòi hỏi chỉ lấy một lệnh thực hiện kèm theo mà thôi.*

iv. *Các <khối lệnh> nếu có từ hai câu lệnh đơn trở lên thì phải đặt trong cặp dấu ngoặc {}. Khi khối lệnh chỉ gồm một câu lệnh thì có thể bỏ dấu ngoặc nhọn đầu và cuối. Không được đặt dấu chấm phẩy sau dấu ngoặc nhọn kết thúc khối lệnh. Khối lệnh tương đương với câu lệnh đơn về mặt cú pháp. Nói cách khác, nơi nào trong chương trình đặt được một câu lệnh thì ở đó ta cũng có thể đặt được một khối lệnh.*

Dạng 2

if (<biểu thức >)

< khối lệnh 1>;

else

< khối lệnh 2>;

Mô tả hoạt động

Khi gặp lệnh if dạng 2, <biểu thức> được tính toán. Nếu kết quả của <biểu thức> là khác 0 thì chương trình sẽ thực hiện <khối lệnh 1>; nếu kết quả của <biểu thức> là bằng 0 thì chương trình sẽ thực hiện <khối lệnh 2>.

Ví dụ 3-1:

Giải hệ phương trình: $\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$

Hướng dẫn:

-Tính định thức d, dx, dy.

-Nếu d khác 0 thì hệ phương trình có nghiệm duy nhất là (dx/d; dy/d).

Ngược lại (khi d=0)

Nếu dx khác 0 hoặc dy khác 0 thì

hệ phương trình là vô nghiệm;

Ngược lại (cả d, dx, dy đều bằng 0).

hệ phương trình là vô số nghiệm.

```
1. #include <iostream.h>
2. int main()
3. {
4.    float a1, b1, c1, a2, b2, c2;
5.    cin >> a1 >> b1 >> c1;
6.    cin >> a2 >> b2 >> c2;
7.    float     d = a1*b2-a2*b1;
8.    float     dx = c1*b2-c2*b1;
9.    float     dy = a1*c2-a2*c1;
10.       if (d != 0)
11.            cout << dx/d << " : " << dy/d;
12.       if (d == 0 && (dx != 0 || dy !=0))
13.            cout <<"He pt vo nghiem";
14.       if (d == 0 && dx == 0 && dy == 0)
15.            cout << "He pt vo so nghiem";
16.  }
```

Chúng ta đã trình bày lời giải trên bằng cách sử dụng cấu trúc rẽ nhánh if else dạng 1, nếu sử dụng cấu trúc rẽ nhánh if else dạng 2, ta có thể giải như sau:

```
if ( d!=0)
        cout << dx/d << " : " << dy/d;
else
        if ( dx!=0  || dy != 0)
                cout << "He pt vo nghiem";
        else
                cout << "He pt vo so nghiem";
```

Một số bộ test:

test	a_1	b_1	c_1	a_2	b_2	c_2	**Kết quả**
test 1	1	1	36	2	4	100	22;14
test 2	1	3	5	6	18	4	Vô nghiệm
test 3	1	-1	0	1	-1	-4	Vô nghiệm
test 4	1	1	1	2	2	2	Vô số nghiệm

Ví dụ 3-2:

Viết chương trình nhập vào hai số nguyên a,b; sau đó thực hiện một trong bốn phép tính cộng, trừ, nhân, chia (nếu gõ ký tự + thì thực hiện phép cộng, nếu gõ ký tự - thì thực hiện phép trừ,...).

```
1. #include < iostream.h >
2. int main()
3. {
4. long a,b;
5. char pt;                  // pt là kiểu ký tự
6. cout << "\nChuong trinh thuc hien 4 phep toan so hoc\n";
7. cout << "Nhap vao so a = "; cin >> a;
8. cout << "Nhap vao so b = "; cin >> b;
9. cout << "\nGo dau + de thuc hien phep cong:";
10.  cout << "\nGo dau - de thuc hien phep tru :";
11.  cout << "\nGo dau * de thuc hien phep tich:";
12.  cout << "\nGo dau / de thuc hien phep chia:";
13.  cout << "\nHay nhap vao lua chon:";
```

```
14.  cin >> pt;
15.  if ( pt == '+' )
16.       cout << "Ket qua la : " << a+b;
17.  else
18.       if (pt == '-' )
19.            cout << "Ket qua la : " << a-b;
20.       else
21.            if (pt == '*')
22.                 cout << "Ket qua la : " << a*b;
23.            else
24.                 if ( pt == '/')
25.                 cout << "Ket qua la : " << (float)a / b;
26.                 else
27.                 cout << "Ban da chon khong dung phep toan!";
28.  }
```

Một số bộ test:

Test	a	b	+	-	*	/
1	1	-2	-1	3	-2	0
2	20000	30000	50000	-10000	600000000	0.666667

3.2. CẤU TRÚC switch . . . case

Đây là lệnh thể hiện quyết định có nhiều nhánh rẽ, dựa trên việc so sánh giá trị của biểu thức với một số giá trị hằng số và rẽ nhánh theo đó.

switch (<biểu thức>)
{ **case** hằng_1 : < khối lệnh 1>;
break;
case hằng_2 : < khối lệnh 2>;
break;
....
case hằng_n : < khối lệnh n>;
break;
[**default** :< khối lệnh 0>; break;]
}

Mô tả hoạt động

- Đầu tiên cấu trúc này tính giá trị của < biểu thức > sau đó so sánh giá trị của <biểu thức> với từng giá trị hằng_i nằm sau các từ khóa case. Nếu giá trị của <biểu thức> trùng với hằng_i nào thì < khối lệnh i > tương ứng sẽ được thực hiện, nếu sau < khối lệnh i > có lệnh break thì chương trình sẽ cho phép thoát ra khỏi cấu trúc switch case. Trong trường hợp < khối lệnh i > bằng rỗng thì < khối lệnh i+1> sẽ được xem xét tương tự.
- default là một thành phần không bắt buộc. Khi cấu trúc có sử dụng từ khóa default thì nghĩa là nếu giá trị của < biểu thức > không trùng với các giá trị từ hằng_1 đến hằng_n thì < khối lệnh 0 > sau default sẽ được thực hiện. Khi cấu trúc không có default thì sẽ thoát khỏi lệnh switch.
- Lệnh break có tác dụng ngắt chương trình, thoát khỏi < khối lệnh > chứa nó.

Ví dụ 3-3:

Nhập 2 giá trị tháng, năm của một ngày nào đó. Hãy cho biết tháng đó có bao nhiêu ngày ?

Lưu ý: các tháng 1,3,5,7,8,10,12 có 31 ngày; các tháng 4,6,9,11 có 30 ngày. Riêng tháng 2 thì có hai trường hợp sau: Trường hợp 1: Nếu là năm nhuận thì tháng đó có 29 ngày (năm nhuận là năm chia hết cho 4 nhưng không chia hết cho 100 hoặc là năm chia hết cho 400), trường hợp hai: Nếu năm không nhuận thì tháng đó có 28 ngày.

```
1. #include <iostream.h>
2. int main()
3. {
4. int thang, songay;     // so ngay cua thang
5. cin >> thang;;
6. switch (thang)
7. {
8. case 1:
9. case 3:
10.  case 5:
11.  case 7:
12.  case 8:
13.  case 10:
```

```
14.  case 12:  songay=31;     break;
15.  case 4:
16.  case 6:
17.  case 9:
18.  case 11:  songay=30;     break;
19.  case 2:   cin >> nam;
20.       if (( nam %4 == 0 && nam %100 !=0 ) || nam % 400 == 0)
21.            songay = 29;
22.       else
23.            songay = 28;
24.  break;
25.  }
26.  cout<<songay;
27.  }
```

Một số bộ test:

Test	Tháng/năm	Số ngày của tháng tương ứng
test 1	2/2008	29
test 2	2/2009	28
test 3	1/2009	31
test 4	11/2008	30
test 5	2/2000	28

Ví dụ 3-4:

Giải lại Ví dụ 3-2 trên bằng cách sử dụng cấu trúc switch case

```
1. #include < iostream.h >
2. int main()
3. {
4. long a,b;
5. char pt;
6. cout << "Chuong trinh thuc hien 4 phep toan so hoc\n";
7. cout << "Nhap vao so a = ";      cin >> a;
8. cout << "Nhap vao so b = ";      cin >> b;
9. cout << "\nGo dau + de thuc hien phep cong:";
10.  cout << "\nGo dau - de thuc hien phep tru :";
```

```
11.  cout << "\nGo dau * de thuc hien phep tich:";
12.  cout << "\nGo dau / de thuc hien phep chia:";
13.  cout << "\nHay nhap vao lua chon:";     cin >> pt;
14.  switch (pt)
15.  {
16.       case '+': cout<<"Ket qua la : "<< a+b;  break;
17.       case '-': cout<<"Ket qua la : "<< a-b;  break;
18.       case '*': cout<<"Ket qua la : "<< a*b;  break;
19.  case '/': cout<<"Ket qua la : "<< (float) a / b;  break;
20.       default :      cout<<"Ban  da  chon  khong  dung  phep
  toan!";
21.  }
22.  }
```

Lưu ý :

i. *Nếu khi một hoặc nhiều các < khối lệnh i > đã được thực hiện nhưng không gặp câu lệnh break thì < khối lệnh 0 > sau default vẫn được thực hiện.*

ii. *Cấu trúc if else tổng quát hơn so với cấu trúc switch theo nghĩa nếu đoạn lệnh giải được bằng cấu trúc switch thì cũng sẽ giải được bằng cấu trúc if else.*

iii. *Theo cú pháp switch (<biểu thức>) thì biểu thức phải là kiểu nguyên (số nguyên, ký tự, giá trị logic).*

iv. *<biểu thức> là một biểu thức có kết quả nguyên (char, int, long, unsigned,...). Hằng_i là hằng số, hằng kí tự hoặc biểu thức hằng có kết quả nguyên. Các hằng_i phải khác nhau ở các case. Thứ tự của các case và default là tùy ý.*

3.3. CẤU TRÚC for

Cấu trúc **for** được sử dụng để giải quyết vấn đề bài toán có số lần lặp lại xác định trước.

for (<biểu thức 1>; <biểu thức điều kiện>; <biểu thức 2>)

{

<khối lệnh> ;

}

Mô tả hoạt động

Trong cấu trúc của lệnh này, trước tiên sẽ thực hiện tính <**biểu thức 1**>, sau đó tính < **biểu thức điều kiện**>. Nếu giá trị của < **biểu thức điều kiện**> khác không (đúng) thì sẽ thực hiện <**khối lệnh**> rồi đến tính <**biểu thức 2**>. Sau khi thực hiện xong <**biểu thức 2**> thì quay trở lại tính giá trị của < **biểu thức điều kiện**> và bắt đầu một vòng lặp mới,…cứ thế vậy đến khi giá trị của < **biểu thức điều kiện**> nhận giá trị bằng không (sai) thì cấu trúc lặp này sẽ kết thúc.

Cũng như đã đề cập trong phần trước, < **biểu thức điều kiện**> là một biểu thức dạng logic và số lần lặp đã được biểu thị tường minh trong < **biểu thức điều kiện**> này.

Nhận xét:

- < biểu thức điều kiện > là điều kiện để tiếp tục lặp (chứ không phải là điều kiện để kết thúc lặp).
- Một trong các biểu thức : < biểu thức 1 >; < biểu thức điều kiện >; < biểu thức 2 > có thể không có (nhưng vẫn còn dấu ';' ngăn cách giữa chúng) thì câu lệnh vẫn hợp lệ.
- Nếu < biểu thức điều kiện > không có, hoặc là một hằng số khác không, thì vòng lặp for coi như lặp lại mãi mãi (chỉ có thể thoát ra bằng lệnh break hay return).

Ví dụ 3-5:

Nhập vào một số nguyên dương n <= 1000000000. Hãy cho biết số lượng các ước số dương của n.

Hướng dẫn:

Do các ước số thực sự của n chỉ có thể nhận giá trị từ 1 đến n/2, nên đặt biến d là số lượng các ước số của n tại thời điểm đang xét, ta có thuật toán như sau:

```
d=1; (hiểu phép gán đầu tiên d=1 là do n là ước số của n)
for (int i=1;i<=n/2;i++)
        if (n % i==0) d++;
```

```
1. #include<iostream.h>
2. int main()
3. {
4. long n;
5. int d=1;
6. cin>>n;
7. for (long i=1;i<=n/2;i++)
8. if (n % i==0)
9. d++;
10.  cout<<"So luong uoc so cua so n :"<<d;
11.  }
```

Một số bộ test:

	test 1	test 2	test 3	test 4
n	1	100	1001	10001
Kết quả	1	9	2	4

Ví dụ 3-6:

Giả sử n ≥1 và x là số thực. Hãy viết chương trình tính giá trị của biểu thức sau:

$$S(n,x) = \frac{x}{1} - \frac{x^2}{1+\frac{1}{2}} + \frac{x^3}{1+\frac{1}{2}+\frac{1}{3}} - \ldots + (-1)^{n-1}\frac{x^n}{1+\frac{1}{2}+\ldots+\frac{1}{n}}$$

Hướng dẫn:

Thoạt nhìn thì đây là một bài toán phức tạp; bài toán này có ba vấn đề cần giải quyết như sau: Thứ nhất là xử lý biểu thức chứa hàm mũ ở tử số, thứ hai là xử lý chuỗi số ở mẫu số, thứ ba là xử lý vấn đề đan dấu của chuỗi, thứ tư là tính tổng các số hạng. Việc tìm ra một thuật toán hay với độ phức tạp tuyến tính cho bài toán này là điều mà chúng ta cần nhắm tới.

Ta giải quyết từng vấn đề trên như sau: Thứ nhất, tử số của số hạng đứng sau bằng tử số của số hạng đứng liền trước nó nhân với x, do đó khi tính tử số của một số hạng ta không cần tính lại từ đầu; thứ hai, việc tính các tổng cho mẫu số cũng có tính chất tương tự; thứ ba, việc xử lý vấn đề đan dấu của chuỗi thì ta chỉ cần luân phiên đảo dấu của các số hạng bằng cách gán dau =-dau và dau được nhân với từng số hạng.

```
1. #include <iostream.h>
2. int main()
3. {
4. int n;  float x;
5. cin>>n>>x;
6. float s=0,tu=1,mau=0,dau=1;
7. for (int i=1;i<=n;i++)
8. {
9.    tu=tu*x;
10.       mau=mau+1.0/i;
11.       s=s+dau*tu/mau;
12.       dau=-dau;// có thể không sử dụng ý tưởng này nếu sử
   dụng tu=tu*-x
13.  }
14.  cout<<s;
15.  }
```

Một số bộ test:

Test	test 1	test 2	test 3	test 4
n	3	1	100	5
x	1	10	1	3
Kết quả	0.878788	10	0.530037	79.270622

3.4. CẤU TRÚC while

Cấu trúc **while** được sử dụng để giải quyết vấn đề bài toán có số lần lặp lại không xác định trước và <khối lệnh> chỉ được thực hiện khi <điều kiện> được thỏa.

while (< biểu thức >)

{

<khối lệnh>;

}

Mô tả hoạt động

Trong cấu trúc của lệnh này, trước tiên sẽ tính giá trị của < **biểu thức** >. Nếu giá trị của < **biểu thức** > khác không (đúng) thì sẽ thực hiện <**khối lệnh**>, sau đó quay trở lại tính

giá trị của < **biểu thức** >. Vòng lặp **while** này sẽ kết thúc việc lặp khi giá trị của < **biểu thức** > nhận giá trị không (sai).

Nhận xét:

- < biểu thức > trong vòng lặp while có thể là một biểu thức bất kỳ, ngôn ngữ lập trình C chỉ quan tâm rằng giá trị của < biểu thức > là bằng không hay khác không.
- Khi giá trị của < biểu thức > còn khác không thì còn làm. Khi giá trị của < biểu thức > bằng không thì thoát khỏi vòng lặp.
- <khối lệnh> trong thân vòng lặp while có thể không được thực hiện lần nào.

Ví dụ 3-7:

Viết chương trình nhập vào một số nguyên dương n (0 < n < 1000000000). Hãy tính tổng các chữ số của số đó.

Hướng dẫn:

Lần lượt tìm giá trị của từng chữ số của n và cộng dồn vào s.

(Ở đây n theo đề bài có thể lớn đến hàng tỉ, do đó cần khai báo biến n thuộc kiểu unsigned long).

```
1. #include <iostream.h>
2. int main()
3. {
4. long n;
5. cin>>n;
6. int s=0;
7. while (n>0)
8. {
9.    s=s+n%10;
10.       n=n/10;
11.   }
12.   cout<<s;
13.   }
```

Một số bộ test:

test	test 1	test 2	test 3	test 4
n	1	2010	999111	1010101
Kết quả	1	3	30	4

Ví dụ 3-8:

Nhập vào số tự nhiên n. Hãy tìm n số nguyên tố đầu tiên.

Hướng dẫn:

Thuật toán kiểm tra n có phải là số nguyên tố hay không đã được trình bày trong chương 1. Thuật toán ở đây là cho i bắt đầu từ số 2 trở đi, nếu i là số nguyên tố thì ghi nhận lại và tăng biến đếm số nguyên tố lên 1 đơn vị. Công việc này sẽ kết thúc khi đã tìm được n số nguyên tố (nếu sử dụng chương trình con thì chương trình giải bài toán này có thể viết gọn gàng hơn).

```
1. #include <iostream.h>
2. int main()
3. {
4. int n;
5. cout<<"Nhap vao so tu nhien n:";
6. cin>>n;
7. int demsont=0;
8. int i=2,d;
9. while (demsont<n)
10.  {    d=0;
11.       for (int j=1;j<=i;j++)
12.       if (i%j==0)
13.            d++;
14.       if (d==2)
15.            {
16.            cout<<i<<" ";
17.            demsont++;
18.       }
19.       i++;
20.  }
21.  }
```

Một số bộ test:

Test	n	dãy n số							
test 1	2	2	3						
test 2	5	2	3	5	7	11			
test 3	8	2	3	5	7	11	13	17	19

Nhận xét

Cú pháp của lệnh

for (<biểu thức 1>; <biểu thức điều kiện>; <biểu thức 2>)

{

<khối lệnh> ;

}

tương đương với nhóm lệnh sau :

<biểu thức 1>;

while (<biểu thức điều kiện >)

{

<khối lệnh >;

<biểu thức 2>;

}

3.5. CẤU TRÚC do while

Các vòng lặp **while** và **for** ở trên được gọi là các vòng lặp có điều kiện kiểm tra trước, khi đó < **khối lệnh** > có thể không được thực hiện lần nào; trong trường hợp mà trước khi vào vòng lặp, gia trị của điều kiện của vòng lặp bằng 0 (sai). Khi ta cần một quá trình lặp kiểm tra điều kiện sau (tức là lệnh lặp luôn luôn được thực hiện ít nhất một lần trước khi kiểm tra điều kiện lặp) ta phải dùng lệnh **do... while**

do

{

< khối lệnh >;

}

while < biểu thức >;

Mô tả hoạt động

Trước hết < **khối lệnh** > sẽ được thực hiện, sau đó tính giá trị của < **biểu thức** >. Nếu giá trị của < **biểu thức** > khác không (đúng) thì quay trở lại thực hiện < **khối lệnh** > sau đó lại tính giá trị của < **biểu thức** >. Vòng lặp này kết thúc khi giá trị của < **biểu thức** > nhận giá trị không (sai).

Lưu ý :

i. *Các lệnh lặp for, while, do…while có thể lồng vào chính nó, hoặc lồng vào lẫn nhau.*

ii. *Vòng lặp for thường sử dụng khi biết được số lần lặp xác định. Vòng lặp while, do…while thường sử dụng khi không biết rõ số lần lặp. Khi gọi vòng lặp while, do…while; nếu biểu thức sai; vòng lặp while sẽ không được thực hiện lần nào trong khi đó vòng lặp do…while thực hiện được 1 lần.*

3.6. MỘT SỐ CÂU LỆNH KHÁC

3.6.1. Câu lệnh break

Khi có nhiều vòng lặp lồng nhau, câu lệnh **break** sẽ thoát ra khỏi vòng lặp (hoặc lệnh **switch**) bên trong nhất chứa nó. Như vậy, **break** cho ta khả năng thoát sớm ra khỏi một vòng lặp (từ một điểm bất kỳ bên trong vòng lặp) mà không cần dùng đến điều kiện kết thúc vòng lặp.

Lệnh break nằm trong lệnh switch chỉ đưa chương trình ra khỏi lệnh switch. Lệnh break nằm ngoài lệnh switch mới đưa chương trình ra khỏi lệnh while.

Ví dụ 3-9: *(Giải lại ví dụ 3.8 bằng cách sử dụng câu lệnh break)*

Nhập vào số tự nhiên n. Hãy tìm n số nguyên tố đầu tiên.

(*n* là số nguyên tố nếu *n* có một ước số trong khoảng từ 2 đến *n*-1. Khi *n* đã có một ước số thì đã đủ cơ sở để dừng việc kiểm tra).

```
1. #include<iostream.h>
2. int main()
3. {
4.    int n;
5.    cin>>n;
6.    for (int i=1;i<=n;i++)
7.    {
8.          int j;
9.          for (j=2;j<i;j++)
10.                   if (i%j==0)    break;
11.              if (j==i)      cout<<i<<" ";
12.         }
13.   }
```

3.6.2. Câu lệnh continue

Trái với câu lệnh **break**, câu lệnh **continue,** chỉ được dùng trong vòng lặp, dùng để bắt đầu một lần lặp mới của vòng lặp bên trong nhất chứa nó. Nói một cách chính xác hơn: Khi gặp một câu lệnh **continue** bên trong thân của một cấu trúc **for**, chương trình sẽ chuyển tới bước khởi đầu lại.

Khi gặp một câu lệnh **continue** bên trong thân của một cấu trúc **while** hoặc **do while**, chương trình sẽ chuyển tới xác định giá trị biểu thức (viết sau từ khóa **while**) và sau đó tiến hành kiểm tra điều kiện kết thúc vòng lặp.

Ví dụ 3-10:

In các số nguyên từ 0 đến 21 lên màn hình, không in số 10.

```
1. #include<iostream.h>
2. int main()
3. {
4.    for( int i = 0; i < 21; i++ )
5.    {    if( i == 10 ) continue;
6.         cout << i << " ";
7.    }
8. }
```

Ví dụ 3-11:

Nhập vào số tự nhiên n. Hãy tìm n số nguyên tố đầu tiên.

```
1. #include<iostream.h>
2. int main()
3. {
4.    int n;
5.    cin>>n;
6.    for (int i=2;i<=n;i++)
7.    {     int j;
8.          for(j=2;j<i;j++)
9.          if (i%j==0) break;
10.               if (j<i)  continue;
11.               else cout<<i<<" ";
12.         }
13.   }
```

BÀI TẬP

Viết chương trình hoàn chỉnh cho các bài toán sau

BT3-1.

a. Giải phương trình dạng $ax^2 + bx + c = 0$ với *a,b,c* là các số thực.

b. Cho 4 số nguyên dương *a,b,c,d* khác nhau đôi một. Tìm giá trị lớn thứ nhì.

Ví dụ: Dữ liệu nhập vào là 7,8,2,1 thì kết quả là 7.

c. Cho 4 số nguyên dương *a,b,c,d* khác nhau đôi một. Xuất 4 số này ra màn hình theo chiều tăng dần.

Ví dụ: Dữ liệu nhập vào là 7,8,2,1 thì kết quả là 1,2,7,8.

d. Viết chương trình nhập vào 6 số nguyên *a,b,c,d,e,f.* Hãy đếm xem trong đó có bao nhiêu số âm? Bao nhiêu số bằng 0 ? Bao nhiêu số dương ?

e. Nhập vào 10 ký tự là chữ cái thường. Hỏi trong đó có bao nhiêu nguyên âm ? Bao nhiêu phụ âm ?

BT3-2.

Bảng đồng hồ điện tử gồm một dãy 3 số *h,p,s* thể hiện tương ứng giờ, phút, giây của thời điểm hiện tại. Cứ sau mỗi giây, giá trị của bộ 3 số *h,p,s* sẽ thay đổi thành h_1,p_1,s_1 tương ứng với thời điểm mới.

Nhập vào 3 số *h,p,s* tìm h_1,p_1,s_1 trong đó $0 \le h \le 23$ và $0 \le p,s \le 59$.

Ví dụ: *h,p,s* là 8 30 0 thì kết quả h_1,p_1,s_1 là 8 30 1

h,p,s là 23 59 59 thì kết quả h_1,p_1,s_1 là 24 0 0

BT3-3.

a. Nhập 3 giá trị ngày, tháng, năm. Hãy tìm ngày kề sau của ngày vừa nhập.

b. Nhập 3 giá trị ngày, tháng, năm. Hãy tìm ngày kề trước của ngày vừa nhập.

BT3-4.

Theo quy định, phân biệt 3 loại điện trong thanh toán: Điện tiêu dùng (loại 1), điện sản xuất (loại 2) và điện kinh doanh (loại 3). Mỗi loại có một cách thanh toán riêng.

Đối với loại điện tiêu dùng 50 KWh đầu tiên mỗi KWh tính với giá A_1 đồng, từ KWh thứ 51 tới KWh thứ 150 được tính với giá B_1, còn từ KWh thứ 151 trở đi mỗi KWh tính với giá C_1 đồng.

Đối với loại điện sản xuất 200 KWh đầu tiên mỗi KWh tính với giá A_2 đồng, từ KWh thứ 201 tới KWh thứ 1000 được tính với giá B_2 còn từ KWh thứ 1001 trở đi mỗi KWh tính với giá C_2 đồng.

Đối với loại điện kinh doanh 100 KWh đầu tiên mỗi KWh tính với giá A_3 đồng, từ KWh thứ 101 tới KWh thứ 200 được tính với giá B_3, còn từ KWh thứ 201 trở đi mỗi KWh tính với giá C_3 đồng.

Công ty Alpha có 3 đồng hồ điện tương ứng với 3 loại. Chỉ số của đồng hồ đầu tháng cuối tháng đối với loại 1 lần lượt là là X_1 và Y_1, loại 2 là X_2 và Y_2, loại 3 là X_3 và Y_3.

Yêu cầu: Tính tổng số tiền T mà công ty phải trả trong tháng.

Dữ liệu vào từ bàn phím

6 số nguyên X_1, Y_1, X_2, Y_2, X_3 và Y_3 ($0 \le X_2 \le Y_2 \le 10^7$, $0 \le X_3 \le Y_3 \le 10^7$, $0 \le X_1 \le Y_1 \le 10^7$).

9 số nguyên A_1, B_1, C_1, A_2, B_2, C_2, A_3, B_3 và C_3 ($0 \le A_1, B_1, C_1, A_2, B_2, C_2, A_3, B_3, C_3 \le 1000$).

Kết quả xuất ra màn hình:

Số nguyên T.

Ví dụ: Dữ liệu nhập vào là:

0 100 0 700 0 500

1 2 3 2 3 4 3 4 5

Thì kết quả là: 4250

BT3-5.

a.Tính tổng $S=1+\frac{1}{3}+\frac{1}{5}+...+\frac{1}{(2n+1)}$ với n là số tự nhiên ≥ 0.

b.Tính tổng $S=\frac{1}{2}+\frac{1}{4}+...+\frac{1}{2n}$ với n là số tự nhiên ≥ 1.

c.Tính tổng $S(x,n)=x+\frac{x^2}{1+2}+\frac{x^3}{1+2+3}+...+\frac{x^n}{1+2+3+...+n}$

BT3-6.

a.Tính tổng $S(x,n)=-x+\frac{x^2}{2!}-\frac{x^3}{3!}+...+(-1)^n\frac{x^n}{n!}$

b.Tính tổng S(x,n) = $1-\frac{x}{1+2}+\frac{x^2}{2+3}-...+(-1)^n\frac{x^n}{n+(n+1)}$

c.Tính tổng S = 1.2 + 2.3.4 +3.4.5.6+...+ n(n+1)...(2n)

d.Cho S sẽ được tính theo công thức S=1+3+5+...+(2n+1) với n≥0 và số nguyên M. Hãy tìm giá trị n nhỏ nhất sao cho S >M với M được cho trước.

Ví dụ: M=20 thì n tìm được là 4.

BT3-7.

a. Viết chương trình hiển thị tất cả các số có 3 chữ số sao cho tổng các chữ số của số đó bằng tích của chúng. Ví dụ: Số 123 là thỏa mãn điều kiện đề bài.

b. Tìm các số có 5 chữ số khác nhau đôi một. Có bao nhiêu số như vậy ?

   Ví dụ: Số 42876 là số thỏa điều kiện đề bài.

c. Cần trả 200000 đồng từ 3 loại giấy bạc 1000 đồng, 2000 đồng, 5000 đồng. Hỏi có bao nhiêu phương án có thể để chi trả ?

d. Tìm các bộ ba số tự nhiên x,y,z thỏa $1 \leq x,y,z \leq 10000$ và thỏa mãn $x^2 + y^2 = z^2$. Ví dụ {6,8,10} là một bộ nghiệm thỏa mãn.

BT3-8.

a. Trong mặt phẳng tọa độ *OXY* cho 4 điểm *A, B, C, D* lần lượt có tọa độ là (x_a,y_a), (x_b,y_b), (x_c,y_c), (x_d,y_d). Hãy kiểm tra xem tứ giác *ABCD* có phải là một hình bình hành không ?

b. Trong mặt phẳng tọa độ *OXY* cho *n* điểm (X_i,Y_i) có tọa độ nguyên. Hãy đếm xem có bao nhiêu điểm nằm hẳn ở mỗi phần tư I, II, III, IV của mặt phẳng và bao nhiêu điểm nằm trên các trục tọa độ ?

BT3-9.

Trong mặt phẳng tọa độ Oxy cho hai hình chữ nhật có cạnh song song với các trục tọa độ, hình chữ nhật thứ nhất có tọa độ gốc dưới trái là $A(x_1,y_1)$ và tọa độ gốc trên phải là $B(x_2,y_2)$. Hình chữ nhật thứ hai có tọa độ gốc dưới trái là $C(x_3,y_3)$ và tọa độ gốc trên phải là $D(x_4,y_4)$. Các toạ độ là các số nguyên. Hãy tìm phần diện tích giao nhau của hai hình chữ nhật trên?

BT3-10.

Trong mặt phẳng tọa độ *OXY* cho hai đường thẳng có phương trình lần lượt là: $a_1x + b_1y + c_1 = 0$ và $a_2x + b_2y + c_2 = 0$ (trong đó a_i,b_i,c_i là các số thực). Hãy kiểm tra xem hai đường thẳng này là trùng nhau? song song nhau? cắt nhau ? hay vừa cắt nhau vừa vuông góc nhau?

CHƯƠNG 4. CHƯƠNG TRÌNH CON

4.1. KHÁI NIỆM

4.1.1. Khái niệm chương trình con

Khi lập trình giải quyết một vấn đề bài toán, để thuận tiện cho việc viết chương trình; người ta thường tách chương trình đó thành các đoạn chương trình nhỏ hơn - mỗi đoạn chương trình nhỏ này giải quyết *trọn vẹn* một công đoạn cụ thể của bài toán; mỗi đoạn chương trình nhỏ đó được gọi là một **chương trình con**.

Chương trình con trong *C*/*C++* được gọi chung là **hàm**. Hàm trong *C*/*C++* có hai dạng là hàm có giá trị trả về và hàm không có giá trị trả về.

Từ khái niệm chương trình con, cần lưu ý rằng mỗi hàm chỉ nên thiết kế để giải quyết một chức năng cụ thể của vấn đề bài toán.

4.1.2. Lợi ích của việc sử dụng chương trình con

Việc phân nhỏ chương trình thành các chương trình con trước hết sẽ làm cho việc phân tích thiết kế chương trình được thuận lợi. Tiếp theo là nó sẽ giúp thuận lợi trong việc kiểm thử, nâng cấp, viết tài liệu cho chương trình, rút ngắn thời gian lập trình, tránh lặp lại các đoạn lệnh tương tự nhau,…

Bản thân ngôn ngữ lập trình *C*/*C++* đã được hỗ trợ một số hàm; các hàm này được gọi là các hàm chuẩn. Tùy theo nhu cầu thực tế mà người lập trình có thể thiết kế thêm các hàm khác; các hàm này được gọi là các hàm tự tạo (phần phụ lục của giáo trình này có trình bày một số hàm chuẩn thường được sử dụng của một số thư viện chuẩn).

Trong chương này; giáo trình tập trung bàn về cách thức xây dựng các hàm tự tạo phục vụ cho từng vấn đề bài toán cụ thể.

4.2. CÁCH THIẾT KẾ VÀ SỬ DỤNG HÀM

Để thuận tiện trong việc diễn đạt nội dung phần lý thuyết liên quan, chúng ta xét một số chương trình sau.

4.2.1. Tham số hình thức biến, tham số hình thức trị

Ví dụ 4-1:

Minh họa cách sử dụng tham số hình thức biến và tham số hình thức trị.

```
1. #include<iostream.h> // khai báo thư viện
2. void test1(int a, int b, int &c);// dòng tiêu đề hàm, kết thúc
   bằng dấu ;
3. int main()
4. {
5.    int a=1,b=2,c=3;
6.    test1(a,b,c);// lời gọi hàm
7.    cout<<a<<"     "<<b<<"   "<<c;
8. }
9. void test1(int a, int b, int &c) //dòng tiêu đề hàm, kết thúc
   không có dấu ;
10.   {
11.         a=a+1;
12.         b=b+1;
13.         c=c+a+b;
14.   }
```

Các biến liệt kê trong các dòng *tiêu đề hàm* thường được gọi là các **tham số**. Hàm có thể không có tham số hoặc có nhiều tham số. Các tham số mà ngay trước nó không có chỉ dẫn & thì được hiểu là tham số đó được truyền theo kiểu **tham trị**, các biến mà ngay trước nó có chỉ dẫn & thì được hiểu là tham số đó được truyền theo kiểu **tham biến**. Danh sách các tham số ở *lời gọi hàm* được gọi là **tham số hình thức**.

Các tham số được truyền theo kiểu tham trị thì những thay đổi trong thân hàm chứa nó sẽ không được giữ lại khi ra khỏi hàm chứa nó; ngược lại, nếu tham số được truyền theo kiểu tham biến thì mọi sự thay đổi trong thân hàm đó sẽ được giữ lại khi ra khỏi hàm đó.

Chẳng hạn xét dòng tiêu đề hàm

void test1(int *a*, int *b*, int &*c*);

hàm này có ba tham số là *a, b, c*; trong đó *a,b* là các tham trị, *c* là tham biến. Khi thực hiện chương trình ở ví dụ 4-1 thì sẽ cho kết quả là : 1 2 8

4.2.2. Biến toàn cục và biến địa phương

Các biến được khai báo trong các hàm - kể cả hàm int main () được gọi là các **biến cục bộ** (biến địa phương) và nó chỉ có ảnh hưởng trong phạm vi thân hàm chứa nó. Khi hàm chứa nó kết thúc thì các biến này cũng mất tác dụng theo. Ngược lại, các biến được khai báo ngoài được gọi là các **biến toàn cục**; biến toàn cục ảnh hưởng đến cả chương trình. Khi lập trình chúng ta cần tránh tối đa việc sử dụng các biến toàn cục.

Ví dụ 4-2:

Minh họa cách sử dụng biến toàn cục và biến địa phương.

```
1.  #include<iostream.h>
2.  void test2(int a, int b);
3.  int c;// biến toàn cục
4.  int main()
5.  {
6.     int a=1,b=7,d=5; // các biến cục bộ
7.     c=3;
8.     test2(a,b);
9.     cout<<a<<"     "<<b<<"   "<<c<<"   "<<d;
10. }
11. void test2(int a, int b)
12. {
13.    c=a+b;
14.    int d=a+b;
15.    a=a+1;
16.    b=b+1;
17. }
```

Chương trình trên có sử dụng bốn biến *a,b,c,d*; trong đó *c* là biến toàn cục; còn *a,b,d* là các biến cục bộ.

Khi thực hiện chương trình ở ví dụ 4-2 thì sẽ cho kết quả là: 1 7 8 5

4.2.3. Hàm không có giá trị trả về, hàm có giá trị trả về

Hàm không có giá trị trả về (trả về kiểu *void*), hàm có giá trị trả về (trả về kiểu khác *void*).

Nếu hàm cần thiết kế chỉ có một giá trị trả về thì có thể thiết kế hàm theo dạng hàm không có giá trị trả hoặc có giá trị trả về; khi đó ta cần thêm thông tin sau để quyết định việc lựa chọn một trong hai cách trên: Nếu tên hàm không được sử dụng trong các hàm hoặc biểu thức khác nữa thì nên thiết kế theo dạng **hàm không có giá trị trả về**, ngược lại thì nên thiết kế theo dạng **hàm có giá trị trả về**. Hàm được thiết kế theo dạng có giá trị trả về thì trong thân hàm đó phải có lệnh return để trả về giá trị là kết quả cần trả về của hàm đó.

Khi cần thiết kế một hàm có nhiều hơn một giá trị thì có thể giải quyết theo ba cách sau: *thứ nhất*, phân rã chức năng của hàm thành nhiều chức năng nhỏ hơn nữa, để mỗi hàm có đúng một chức năng riêng biệt và nó sẽ trả về đúng một giá trị; *thứ hai*, trả về nhiều giá trị thông qua kỹ thuật truyền tham biến – cần trả về bao nhiêu giá trị thì sử dụng thêm bấy nhiêu tham biến; *thứ ba*, sử dụng biến toàn cục – dùng biến toàn cục để ghi nhận giá trị cần trả về.

Ví dụ 4-3:

Minh họa cách sử dụng hàm có giá trị trả về và hàm không có giá trị trả về.
*Ví dụ 4-3 có 3 hàm: hàm **test3** là hàm có giá trị trả về, hàm **test4** là hàm không có giá trị trả về, hàm **test5** là hàm không có giá trị trả về nhưng có trả về 3 giá trị thông qua 3 tham biến a,b,c.*

```
1.  #include<iostream.h>
2.  int test3 (int a, int b, int c);
3.  void test4 (int a, int b, int c);
4.  void test5 (int &a, int &b, int &c);
5.  int main()
6.  {
7.     int a=10,b=20,c=30;
8.     cout<< test3(a,b,c)<<endl; // trả về giá trị 63
9.     test4(a,b,c); // trả về giá trị 63
10.    test5(a,b,c);
11.    cout<<a+b+c<<endl;// trả về giá trị 63; a,b,c là các tham
   biến của test 5
12. }
13. int test3 (int a, int b, int c)
```

```
{
    a=a+1;
    b=b+1;
    c=c+1;
    return a+b+c;
}
void test4 (int a, int b, int c)
{
    a=a+1;
    b=b+1;
    c=c+1;
    cout<<a+b+c<<endl;
}
void test5 (int &a, int &b, int &c)
{
    a=a+1;
    b=b+1;
    c=c+1;
}
```

Ví dụ 4-4:

Viết chương trình tính diện tích của hình vuông khi biết độ dài cạnh, hình chữ nhật khi biết độ dài 2 cạnh, hình tam giác khi biết độ dài 3 cạnh, hình tròn khi biết độ dài bán kính.

Hướng dẫn:

Trong ví dụ 4-4, chúng ta sẽ xem xét ba kiểu thiết kế chương trình con: Chương trình con mà các tiêu đề hàm không có tham số nào (Ví dụ 4-4a), chương trình con không có giá trị trả về (Ví dụ 4-4b), chương trình con có giá trị trả về (Ví dụ 4-4c).

Ví dụ 4-4a:

```
1. #include <iostream.h>
2. void dthinhvuong();
3. void dthinhchunhat();
4. void dthinhtamgiac();
5. void dthinhtron();
6. int main()
7. {
8.    dthinhvuong();
9.    dthinhchunhat();
10.       dthinhtamgiac();
11.       dthinhtron();
12.   }
13.   void dthinhvuong()
14.   {
15.       int a;
16.       cout<<"\nChuong trinh tinh dien tich hinh vuong";
17.       cout<<"\nNhap chieu dai canh hinh vuong : ";
18.       cin>>a;
19.       int s=a*a;
20.       cout<<"dien tich la : "<<s<<endl;
21.   }
22.   void dthinhchunhat()
23.   {
24.       //bạn đọc dễ dàng hoàn chỉnh đoạn chương trình này
25.   }
26.   void dthinhtamgiac()
27.   {
28.       //bạn đọc dễ dàng hoàn chỉnh đoạn chương trình này
29.   }
30.   void dthinhtron()
31.   {
32.       //bạn đọc dễ dàng hoàn chỉnh đoạn chương trình này
33.   }
```

Ví dụ 4-4b:

```
1. #include <iostream.h>
2. void dthinhvuong(int a);
3. void dthinhchunhat(int a, int b);
4. void dthinhtamgiac(int a, int b, int c);
5. void dthinhtron(int r);
6. int main()
7. {
8.    int a,b,c,r;
9.    dthinhvuong(a);
10.       dthinhchunhat(a,b);
11.       dthinhtamgiac(a,b,c);
12.       dthinhtron(r);
13.   }
14.   void dthinhvuong(int a)
15.   {
16.       cout<<"\nChuong trinh tinh dien tich hinh vuong";
17.       cout<<"\nNhap chieu dai canh hinh vuong : ";
18.       cin>>a;
19.       int s=a*a;
20.       cout<<"dien tich la : "<<s<<endl;
21.   }
22.   void dthinhchunhat(int a, int b)
23.   {
24.       //bạn đọc dễ dàng hoàn chỉnh đoạn chương trình này
25.   }
26.   void dthinhtamgiac(int a, int b, int c)
27.   {
28.       //bạn đọc dễ dàng hoàn chỉnh đoạn chương trình này
29.   }
30.   void dthinhtron(int r)
31.   {
32.       //bạn đọc dễ dàng hoàn chỉnh đoạn chương trình này
33.   }
```

Ví dụ 4-4c:

```
1. #include <iostream.h>
2. int dthinhvuong(int a);
3. int dthinhchunhat(int a, int b);
4. float dthinhtamgiac(int a, int b, int c);
5. float dthinhtron(int r);
6. int main()
7. {
8.    int a,b,c,r;
9.    cout<<"\nChuong trinh tinh dien tich hinh vuong";
10.       cout<<"\nNhap chieu dai canh hinh vuong : ";
11.       cin>>a;
12.       cout<<dthinhvuong(a);
13.       cout<<dthinhchunhat(a,b);
14.       cout<<dthinhtamgiac(a,b,c);
15.       cout<<dthinhtron(r);
16.   }
17.   int dthinhvuong(int a)
18.   {
19.       return a*a;
20.   }
21.   int dthinhchunhat(int a, int b)
22.   {
23.       //bạn đọc dễ dàng hoàn chỉnh đoạn chương trình này
24.   }
25.   float dthinhtamgiac(int a, int b, int c)
26.   {
27.       //bạn đọc dễ dàng hoàn chỉnh đoạn chương trình này
28.   }
29.   float dthinhtron(int r)
30.   {
31.       //bạn đọc dễ dàng hoàn chỉnh đoạn chương trình này
32.   }
```

Ví dụ 4-5:

Viết chương trình in các số nguyên tố nhỏ hơn hoặc bằng n. Với n là số nguyên dương cho trước.

Hướng dẫn:

Chương trình này có thể thiết kế các hàm sau:

void nhap(int &*n*) //Hàm này cho phép nhập giá trị *n*

int nguyento(int *n*) //Hàm này kiểm tra xem *n* có phải là số nguyên tố hay không ? Nếu *n* là nguyên tố thì trả về giá trị 1, ngược lại thì trả về giá trị 0.

void insonguyento(int *n*) //Hàm này cho phép in các số nguyên tố nhỏ hơn hoặc bằng *n*.

```
1.  #include<iostream.h>
2.  #include<math.h> // thư viện chứa các hàm toán học
3.  void nhap(int &n);
4.  void insonguyento(int n);
5.  int main()
6.  {
7.     int n;
8.     nhap(n);
9.     insonguyento(n);
10. }
11. void nhap(int &n) // n cần phải được truyền theo kiểu tham
    biến
12. {
13.    cout<<"Nhap vao so n = "; cin>>n;
14. }
15. int nguyento(int n)
16. {
17.    if (n<2) return 0;
18.    int k=sqrt(n);// hàm lấy giá trị căn bậc 2. Hàm này trả về
    giá trị là số thực
19.    for (int i=2;i<=k;i++)   //Các cận của vòng for ở đây phải
    là 2 và k
20.         if (n%i==0)
21.              return 0;
```

```
22.    return 1;
23. }
24. void insonguyento(int n)
25. {
26.    for (int i=1;i<=n;i++)
27.    if (nguyento(i)) // lời gọi hàm có giá trị trả về
28.          cout<<i<<" ";
29. }
```

Ví dụ 4-6:

Nhập 3 giá trị ngày, tháng, năm. Hãy viết chương trình tìm ngày kề sau của ngày vừa nhập.

Hướng dẫn:

Ta thiết kế hai hàm sau:

int songay(int *thang*, int *nam*)

void ngaymai(int &*ngay*, int &*thang*, int &*nam*)

Hàm **songay** cho biết số ngày của một tháng nào đó, hàm **ngaymai** cho biết ngày kề sau của ngày hiện tại. Ở đây cần trả về 3 giá trị là *ngay, thang, nam* nên cả 3 giá trị này cần được trả về thông qua tham biến và hàm này được thiết kế theo kiểu hàm không có trị trả về như trên.

```
1.  #include <iostream.h>
2.  void ngaymai(int &ngay, int &thang, int &nam);
3.  int main()
4.  {
5.     int ngay,thang,nam;
6.     cin>>ngay>>thang>>nam;
7.     ngaymai(ngay,thang,nam);
8.     cout<<ngay<<" "<<thang<<" "<<nam;
9.  }
10. int songay(int thang, int nam)
11. {
12.    switch (thang)
13.    {
14.          case 1:
```

```
15.         case 3:
16.         case 5:
17.         case 7:
18.         case 8:
19.         case 10:
20.         case 12:return 31;
21.         case 4:
22.         case 6:
23.         case 9:
24.         case 11:return 30;
25.    case 2:if ((nam %4==0 && nam %100!=0) || (nam % 400 ==0))
26.                    return 29;
27.               return 28;
28.    }
29. }
30. void ngaymai(int &ngay, int &thang, int &nam)
31. {
32.    if (ngay<songay(thang,nam))
33.         ngay++;
34.    else
35.    {
36.         ngay=1;
37.         thang++;
38.         if (thang>12)
39.         {
40.              thang=1;
41.              nam++;
42.         }
43.    }
44. }
```

Ví dụ 4-7:

Viết chương trình nhập vào 2 phân số a/b và c/d. Hãy tính tổng của 2 phân số. Yêu cầu phân số kết quả phải ở dạng tối giản. Ví dụ: 1/2 + 1/6 có kết quả là 2/3.

Hướng dẫn:

Với ví dụ này ta thiết kế các hàm sau:

void nhapps(int &*tu*, int &*mau*);

void xuatps(int *tu*, int *mau*);

void congps(int *tu1*, int *mau1*,int *tu2*, int *mau2*, int &*tu*, int &*mau*);

int uscln(int *a*, int *b*);

Trong đó, hàm nhapps thì các tham số phải là tham biến do các phân số đầu vào còn được sử dụng ở hàm congps, hàm xuatps thì các tham số là tham trị. Do cả hai hàm nhập phân số và xuất phân số này không được gọi ở một hàm hoặc một biểu thức khác nên tốt nhất là thiết kế theo kiểu hàm không có trị trả về. Còn hàm congps thì do kết quả trả về là một phân số - nghĩa là có tử số và mẫu số, nên không thể trả về thông qua tên hàm do đó ta cần viết hàm không có trị trả về và khi đó giá trị trả về được trả về thông qua hai tham biến của hàm.

(nếu coi phân số như là một biến của kiểu struct thì có thể viết hàm có trị trả về cho trường hợp này. Vấn đề này sẽ được xử lý trong chương 7).

```
1.  #include <iostream.h>
2.  void    nhapps(int &tu, int &mau);
3.  void    xuatps(int tu, int mau);
4.  void    congps(int tu1, int mau1,int tu2, int mau2, int &tu,
    int &mau);
5.  int     uscln(int a, int b);
6.  int  main()
7.  {
8.     int tu1,mau1,tu2,mau2,tu,mau;
9.     nhapps(tu1,mau1);
10.    nhapps(tu2,mau2);
11.    congps(tu1,mau1,tu2,mau2,tu,mau);
12.    xuatps(tu,mau);
13. }
14. void    nhapps(int &tu, int &mau)
15. {
16.    cout<<"Nhap vao tu so  : ";   cin>>tu;
17.    cout<<"Nhap vao mau so : ";cin>>mau;
18. }
```

```
19. void    xuatps(int tu, int mau)
20. {
21.    int uc=uscln(tu,mau);
22.    tu=tu/uc;
23.    mau=mau/uc;
24.    cout<<"Ket qua: "<<tu<<"/"<<mau<<endl;
25. }
26. void    congps(int tu1, int mau1,int tu2, int mau2, int &tu,
    int &mau)
27. {
28.    tu=tu1*mau2+tu2*mau1;
29.    mau=mau1*mau2;
30. }
31. int uscln(int a, int b)
32. {
33.    int r=a%b;
34.    while (r!=0)
35.    {
36.          a=b;
37.          b=r;
38.    }
40.    return b;
41. }
```

Lưu ý:

i. *Khi thiết kế một hàm nên đặt ra các câu hỏi: Hàm này nên đặt tên là gì cho có tính gợi nhớ ? Không nên viết tắt tên hàm, hạn chế không nên viết hoa tên hàm. Hàm này cần có bao nhiêu tham số ? Trong đó có bao nhiêu tham số được truyền theo kiểu tham trị và bao nhiêu tham số được truyền theo kiểu tham biến ? Hàm này có giá trị trả về hay không ?*

ii. *Các dòng tiêu đề hàm đặt phía trước hàm int main () phải có dấu chấm phẩy, các dòng tiêu đề hàm đặt phía sau hàm int main () - nghĩa là khi viết chi tiết cho các hàm thì các dòng tiêu đề hàm không có dấu chấm phẩy. Các hàm hỗ trợ cho các hàm khác không nhất thiết phải được khai báo dòng tiêu đề hàm phía trước hàm int man().*

iii. *Hàm có trị trả về thì trong thân hàm đó phải có lệnh return.*

iv. *Nên thiết kế chương trình chi tiết đến từng chức năng, rõ ràng. Tiêu chí quan trọng nhất để đánh giá một chương trình là thời gian của chương trình đó; điều này phụ thuộc vào thuật toán cài đặt chương trình đó (một chương trình có dài hơn vài chục dòng lệnh cũng không quan trọng).*

v. *Hàm có giá trị trả về thì lời gọi hàm phải được sử dụng như một tên biến, chứ không được gọi độc lập như loại hàm không có giá trị trả về.*

vi. *Khi một chương trình có nhiều chương trình con, nên cho kiểm thử từng hàm một; hàm này đúng rồi mới kiểm thử hàm tiếp theo và cuối cùng là kiểm thử toàn bộ chương trình.*

BÀI TẬP

Viết chương trình hoàn chỉnh cho các bài toán sau

BT4-1.

a. Đếm xem từ 1 đến 1tỉ có bao nhiêu số nguyên tố ? Bao nhiêu số chính phương ? Bao nhiêu số hoàn chỉnh ? (số hoàn chỉnh là số bằng tổng các ước thực sự của nó, Ví dụ 6, 28 là các số hoàn chỉnh).

b. Tìm tất cả các số nhỏ hơn 1 tỷ sao cho nó vừa là số nguyên tố và là số đối xứng. Ví dụ 7, 131, 10001 là các số đối xứng.

c. Đếm xem trong đoạn [*a*,*b*] có bao nhiêu số nguyên tố mà tất cả các chữ số của số nguyên tố đó cũng là số nguyên tố ? (*a*,*b* là các số nguyên dương, *a*<=*b*).

d. Một số nguyên dương có k chữ số được gọi là số Armstrong khi nó bằng tổng lũy thừa *k* của từng chữ số. Hãy in ra các số Armstrong trong đoạn [1..100000].

e. Liệt kê tất cả các số tự nhiên *k* thỏa mãn đồng thời ba điều kiện: k là số có 5 chữ số; *k* là số nguyên tố, *k* là số đối xứng.

f. Phân tích số tự nhiên *n* thành tích các số nguyên tố. Ví dụ 90=2 * 3 * 3 * 5.

BT4-2.

Nhập thông tin ngày (ngày/tháng/năm) - là các số nguyên dương). Hãy cho biết ngày đó là ngày thứ bao nhiêu trong năm ? (ngày 1 tháng 1 là ngày thứ nhất của năm).

Ví dụ: Ngày 3/2/2014 là ngày thứ 34 trong năm.

BT4-3.

a.Tính tổng sau với x là số thực và n là số nguyên dương.

$$1-\frac{x}{1+2}+\frac{x^2}{2+3}-...+(-1)^n\frac{x^n}{n+(n+1)}$$

b.Cho $f_0=1$; $f_1=1$; $f_n=f_{n-1}+f_{n-2}$ với mọi n>1.

$$S=\frac{1}{1+f_o}+\frac{2}{1+f_1}+...+\frac{n}{1+f_{n-1}}$$ (n là số nguyên dương)

c.S=1 + 1.2 + 1.2.3 +... + 1.2.3.4....n (n là số nguyên dương).

BT4-4.

a. Tìm ước số chung lớn nhất của 2 số tự nhiên *a*,*b*.

b. Chuyển đổi một số *n* trong hệ đếm thập phân thành số trong hệ đếm cơ số *b*.

c. Cho *x* là số thực và *n* là số nguyên dương, tính tính x^n.

d. Tính $s(n)=\sqrt{2+\sqrt{2+...+\sqrt{2+\sqrt{2}}}}$ có n dấu căn. Hãy viết hàm tính $s(n)$

BT4-5.

a. Giả sử A và B là 2 số nguyên cho trước và x,y,z là 3 số nguyên thỏa mãn đồng thời hệ bất phương trình sau:

$x^2 + 2y^2 + 3z^2 \le A^2$

$y^2 + z^2 \le B^2$

Viết hàm tìm giá trị lớn nhất của biểu thức $F(x,y,z)=x^2+y^4+z^8$

b. Tìm các bộ nghiệm nguyên dương (x,y,z)thỏa mãn hệ phương trình sau:

$x + y + z = n$

$x^k + y^k + z^k = m$

(trong đó m,n,k là các số nguyên dương)

Tìm một thuật toán hiệu quả cho bài toán.

c. Tìm các bộ ba số tự nhiên x,y,z thỏa $1\le x,y,z \le 10000$ và thỏa mãn $x^2 + y^2 = z^2$.

BT4-6.

a. Trong mặt phẳng tọa độ cho 3 điểm O,A,B. trong đó O,A,B lần lượt có toạ độ là (0,0), (X,Y) $(Z,0)$ với X,Y và Z là các số nguyên dương. Viết chương trình đếm số điểm có tọa độ nguyên nằm bên trong tam giác OAB – không đếm các điểm trên cạnh.

b. Trong mặt phẳng toạ độ OXY cho n hình chữ nhật có các cạnh song song với các trục toạ độ. Hãy tìm diện tích phần mặt phẳng được phủ bởi n hình chữ nhật trên.

BT4-7.

Trong biểu thức: $((((a_1?a_2)?a_3)?a_4)?a_5) = a_6$. Hãy điền vào các vị trí dấu '?' bởi một trong bốn phép tóan +, - , * , / sao cho giá trị của biểu thức đã cho hai vế bằng nhau. Hãy tìm tất cả các lời giải.

BT4-8.

Viết chương trình nhập số tự nhiên n. Hãy in ra chữ số thứ n của dãy vô hạn các số chính phương 149162536496481100121….

BT4-9.

Hãy tìm các số p,q $(0<p<q<n)$ sao cho tổng các ước số thực sự của p bằng q và tổng các ước số thực sự của q bằng p.

BT4-10.

a. Trên mặt phẳng cho một tam giác và một đoạn thẳng. Tính độ dài của đoạn thẳng nằm trong tam giác.

b. Trên mặt phẳng cho hai tam giác. Tính diện tích phần chung của hai tam giác

CHƯƠNG 5. KIỂU DỮ LIỆU MẢNG

5.1. MẢNG MỘT CHIỀU

5.1.1. Khái niệm mảng

Để biểu diễn một dãy các biến ta có thể dùng nhiều biến rời rạc, tuy nhiên cách làm này là không thuận lợi (thậm chí không khả thi) khi số lượng các biến là nhiều. Việc sử dụng mảng là cách tốt nhất và là bắt buộc trong nhiều trường hợp xử lý các vấn đề bài toán thực tế.

Mảng là một số hữu hạn các phần tử có *cùng một kiểu giá trị* và có *chung một tên gọi*. Mỗi phần tử biểu diễn được một giá trị. Số phần tử của mảng phải được xác định ngay từ khi định nghĩa mảng (đối với mảng tĩnh). Mỗi phần tử của mảng được truy cập trực tiếp thông qua tên mảng cùng với chỉ số của mảng. Ngôn ngữ C/C++ cho phép làm việc với mảng một chiều, mảng hai chiều hoặc mảng nhiều chiều.

Trong giáo trình này, chúng tôi chỉ đề cập đến mảng một chiều, kiến thức về mảng hai và mảng nhiều chiều sẽ được đề cập đến trong giáo trình kỹ thuật lập trình tiếp theo.

5.1.2. Khai báo mảng một chiều

< kiểu dữ liệu> tênmảng[const n];

Ví dụ để khai báo mảng một chiều có tên là a chứa 10 phần tử thuộc kiểu số nguyên ta thực hiện câu lệnh như sau:

```
int  a[10];
```

Phần tử của mảng một chiều sẽ được truy xuất thông qua *tên mảng*[chỉ số], chẳng hạn để truy xuất phần tử thứ i của mảng một chiều có tên là a ta viết a[i].

Chỉ số của phần tử đầu tiên của mảng trong C/C++ mặc nhiên là 0, chẳng hạn trong khai báo trên mảng a gồm các phần tử a[0], a[1], ..., a[9].

Lưu ý: Trong một số tình huống, để dễ diễn đạt, chúng ta có thể sử dụng chỉ số đầu tiên của mảng là 1 hoặc là một giá trị khác mà chấp nhận lãng phí một số ô nhớ dành cho mảng.

5.2. THAO TÁC TRÊN MẢNG MỘT CHIỀU

Cho mảng một chiều gồm *n* phần tử; mỗi phần tử là một số nguyên. Sau đây là một số thao tác thường gặp trên mảng một chiều.

5.2.1. Nhập các phần tử

Kỹ thuật nhập từng phần tử của mảng một chiều có thể được thực hiện theo sơ đồ sau:

```
void nhapmang(int a[], int &n)
{
        cin>>n;
        for (int i=0;i<n;i++)
                cin>>a[i];
}
```

Tham số *n* ở đây cần phải được truyền theo kiểu tham biến vì *n* được nhập bên trong hàm này; nếu truyền theo kiểu tham trị thì các hàm khác sẽ không nhận được giá trị *n* nhập vào từ hàm ***nhapmang***.

5.2.2. Duyệt các phần tử

Kỹ thuật duyệt từng phần tử của mảng một chiều có thể được thực hiện theo sơ đồ sau:

```
void duyetmang(int a[], int n)
{
        for (int i=0;i<n;i++)
                <Thực hiện thao tác trên phần tử ai>
}
```

Thao tác thường gặp nhất là thao tác xuất các phần tử của mảng, khi đó câu lệnh tổng quát *<Thực hiện thao tác trên phần tử a_i>* được thay thế bởi câu lệnh cout<<a[i]. Trong các ví dụ và bài tập trong giáo trình này, chúng ta sẽ thường xuyên sử dụng thao tác duyệt mảng.

5.2.3. Tính tổng các phần tử

Kỹ thuật tính tổng các phần tử của mảng một chiều thỏa mãn điều kiện nào đó có thể được thực hiện như sơ đồ sau.

Thiết kế theo kiểu hàm có giá trị trả về (kiểu int)

```
int tinhtong(int a[], int n)
{
    int tong=0;
    for (int i=0;i<n;i++)
    if (a[i] thỏa mãn điều kiện bài toán)
        tong =tong+a[i];
    return tong;
}
```

Hoặc cũng có thể thiết kế theo kiểu hàm không có giá trị trả về (trả về void)

```
void tinhtong(int a[], int n)
{
    int tong=0;
    for (int i=0;i<n;i++)
        if (a[i] thỏa mãn điều kiện bài toán)
        tong =tong+a[i];
    cout<<tong;
}
```

5.2.4. Đếm số phần tử thỏa điều kiện nào đó

Kỹ thuật đếm các phần tử của mảng một chiều thỏa mãn điều kiện nào đó có thể được thực hiện như sơ đồ sau.

```
int demso(int a[], int n)
{
    int dem=0;
    for (int i=0;i<n;i++)
        if (a[i] thỏa mãn điều kiện bài toàn)
        dem++;
    return dem;
}
```

5.2.5. Đặt cờ hiệu

Duyệt lần lượt từng phần tử của mảng, nếu có phần tử thỏa điều kiện thì dừng thuật toán và trả về một giá trị *k* nào đó; nếu không có phần tử nào thỏa điều kiện thì trả trả về giá trị *k*' nào đó; giá trị *k*' mang ý đảo nghĩa với giá trị *k*; thường thì kỹ thuật cờ hiệu này nhằm trả lời cho câu hỏi: *có hay không ? đúng hay sai ? thuộc hay không thuộc ?....*

Kỹ thuật đặt cờ hiệu có thể được thực hiện theo sơ đồ sau:

```
int datcohieu(int a[], int n)
{
        for (int i=0;i<n;i++)
                if (a[i] thỏa mãn điều kiện bài toán)
                        return 0/1;
        return 1/0;
}
```

5.2.6. Xóa một phần tử trong mảng

Để xóa một phần tử tại vị trí *k* của mảng, ta thực hiện như sau: Dịch chuyển các phần tử từ vị trí thứ *k* + 1 đến cuối mảng qua trái một vị trí. Sau khi dịch chuyển xong thì giảm số phần tử của mảng đi một đơn vị; nghĩa là cho *n* = *n*-1. Do *n* thay đổi sau khi thực hiện hàm nên *n* cần được truyền theo kiểu tham biến.

Kỹ thuật xóa một phần tử tại vị trí *k* của mảng có thể được thực hiện theo sơ đồ sau:

```
void xoaphantu(int a[], int &n, int k)
{
        for (int i=k;i<n;i++)
                a[i]=a[i+1];
        n--;
}
```

5.2.7. Chèn phần tử vào mảng

Chèn phần tử *x* vào tại vị trí thứ *k* của mảng ta thực hiện như sau: Dịch chuyển các phần tử từ vị trí thứ cuối mảng đến vị trí *k* + 1 qua phải một vị trí; tiếp theo chèn phần tử *x* vào vị trí *k* và cuối cùng là tăng *n* lên một đơn vị; nghĩa là *n* = *n*+1. Cũng như hàm xóa ở trên; biến *n* phải được truyền theo kiểu tham biến.

Kỹ thuật chèn phần tử *x* vào tại vị trí thứ *k* của mảng có thể được thực hiện theo sơ đồ sau:

```
void chenphantu(int a[], int &n, int k, int x)
{
        for (int i=n;i>k;i--)
        a[i]=a[i-1];
        a[k]=x;
        n++;
}
```

5.2.8. Đặt lính canh

Phần tử lính canh có thể hiểu là một ví trị nào đó của mảng, hoặc là một giá trị nào đó của mảng hoặc cũng có thể là một giá trị tùy theo từng yêu cầu cụ thể của từng bài toán,… Sau đó tiến hành duyệt các phần tử của mảng, nếu có phần tử nào thỏa điều kiện của bài toán thì tiến hành thay đổi vị trí lính canh (hoặc giá trị canh). Vị trí lính canh này chính là thông tin cần tìm.

Tuy nhiên tùy thông tin đầu vào của vấn đề bài toán mà xác định thông tin về phần tử lính canh cho phù hợp.

5.2.9. Bài toán sắp xếp

Trong phần này chúng tôi chỉ xét đến một cách sắp xếp đơn giản nhất đó là phương pháp sắp xếp đổi chỗ trực tiếp.

Xét một mảng *n* số $a_0, a_1,.., a_{N-1}$. Nếu có $i < j$ và $a_i > a_j$, thì ta gọi đó là một nghịch thế. Mảng chưa sắp xếp sẽ có nghịch thế, và ngược lại mảng đã có thứ tự sẽ không còn chứa nghịch thế. Để sắp xếp một mảng, ta có thể tìm cách làm giảm số các nghịch thế trong mảng này bằng cách hoán vị các phần tử a_i,a_j nếu có $i < j$ và $a_i > a_j$ theo một quy luật nào nào đó.

Như đã đề cập ở trên, để sắp xếp một dãy số, ta có thể xét các nghịch thế có trong dãy và làm triệt tiêu dần chúng đi. Ý tưởng chính của thuật toán là xuất phát từ đầu dãy, tìm tất cả nghịch thế chứa phần tử này, triệt tiêu chúng bằng cách đổi chỗ phần tử này với phần tử tương ứng trong cặp nghịch thế. Lặp lại xử lý trên với các phần tử tiếp theo trong dãy.

Đoạn mã thể hiện ý tưởng sắp xếp đổi chỗ trực tiếp như hàm sau:

```
void sapxep(int a[], int n)
{       for (int i=0; i<n-1;i++)
        for (int j=i+1;j<n ;j++)
        if (a[i]>a[j]) hoanvi(a[i],a[j])
}
```

trong đó hàm hoán vị được viết như sau:

```
void hoanvi(int &a, int &b)
{
        int temp=a;
                a=b;
                b=temp;
}
```

Trong hàm hoanvi này các biến a và b phải được truyền theo kiểu tham biến. Hàm sắp xếp trên sẽ sắp xếp các phần tử theo chiều tăng dần, nếu cần sắp giảm thì phép so sánh trong câu lệnh if (a[i]>a[j]) cần được đổi lại là if (a[i]<a[j]).

5.2.10. Bài toán tìm kiếm

Cho dãy n số nguyên $a_0,a_2,...,a_{n-1}$ và một số nguyên x. Hãy tìm xem x có thuộc vào dãy số trên hay không ? Nếu tìm được ở vị trí thứ i thì xuất kết quả là i, ngược lại nếu không tìm thấy thì xuất kết quả là –1 (chú ý dãy bắt đầu từ chỉ số 0, nếu dãy có nhiều số bằng x thì xuất vị trí i nhỏ nhất).

Sau đây là thuật toán tìm kiếm thường được sử dụng nhất có tên gọi là *tìm kiếm tuyến tính*. Ý tưởng chính của thuật toán tìm kiếm này như sau: Bắt đầu từ phần tử thứ nhất a[0], ta lần lượt so sánh x với các giá trị a[i]. Nếu có a[i] bằng x thì i chính là kết quả cần tìm và kết thúc thuật toán. Nếu trong dãy không có số a[i] nào bằng x thì xuất kết quả là –1 và cũng kết thúc thuật toán.

Đoạn mã thể hiện ý tưởng tìm kiếm tuyến tính như hàm sau:

```
int timkiem (int a[], int n, int x)
{
  int i = 0;
  while ( i<n  && a[i]!=x)
```

```
        i++;
        if( i==n)
        return -1; //    tìm hết nhưng không có x
        return i; //     tìm thấy a[i] là phần tử có khóa x
    }
```

Ví dụ 5-1:

Cho mảng một chiều a chứa n số nguyên. Hãy viết các hàm thực hiện các công việc sau:

a. *Đếm xem trong mảng có bao nhiêu số nguyên tố ?*
b. *Kiểm tra trong mảng có được sắp tăng dần hay không ? Nếu có trả về giá trị 1, nếu không có trả về giá trị 0.*
c. *Sắp xếp các số nguyên tố trong mảng tăng dần, còn các số khác khác thì giữ nguyên giá trị và vị trí.*
d. *Tìm tổng các chữ số của tất cả các phần tử của mảng.*

Ví dụ: n=8

19 11 8 1 15 3 2 22

Thì kết quả là :

a. 4

b. 0

c. 2 3 8 1 15 11 19 22

d. 36

Hướng dẫn:

Câu a.

Áp dụng kỹ thuật đếm: Duyệt từng phần tử của mảng, nếu phần tử nào là số nguyên tố thì tăng biến đếm lên 1 đơn vị. Hàm này có áp dụng hàm *kiểm tra n có phải là số nguyên tố* hay không ở chương 4.

```
1. int demsonguyento(int a[], int n)
2. {
3.      int dem=0;
4.      for (int i=0;i<n;i++)
5.      if (nguyento(a[i]))
6.              dem++;
```

```
7.      return dem;
8.  }
```

Câu b.

Áp dụng kỹ thuật cờ hiệu: duyệt từng cặp phần tử a[i],a[i+1]; nếu tồn tại a[i]>a[i+1] nghĩa là mảng không được sắp tăng dần và dừng thuật toán; nếu tất cả phép so sánh a[i]>a[i+1] đều sai; đồng nghĩa với mảng đã được sắp tăng dần thì hàm trả về giá trị 1. Lưu ý do có sử dụng phần tử a[i+1] nên biến i ở đây chạy từ 0 đến nhỏ hơn n-1.

```
1.  int kiemtramangtang(int a[], int n)
2.  {
3.      for (int i=0;i<n-1;i++)
4.      if (a[i]>a[i+1])
5.              return 0;
6.      return 1;
7.  }
```

Câu c.

Áp dụng kỹ thuật sắp xếp: Chỉ sắp xếp các phần tử là số nguyên tố.

```
1.  void sapxepnguyento(int a[], int n)
2.  {
3.      for (int i=0;i<n-1;i++)
4.      for (int j=i+1;j<n;j++)
5.              if (nguyento(a[i]) && nguyento(a[j]) && a[i]>a[j])
6.                      hoanvi(a[i],a[j]);
7.  }
```

Câu d.

Áp dụng kỹ thuật tính tổng: Viết hàm tính tổng các chữ số của một số n, sau đó áp dụng hàm này để tính tổng tất cả các chữ số của tất cả các phần tử của mảng.

```
1.  int tongchusocuaso(int n)
2.  {
3.      int s=0;
4.      while(n>0)
5.      {
6.              s=s+n%10;
7.              n=n/10;
```

```
8.      }
9.      return s;
10. }
11. int tongchusocuaday(int a[], int n)
12. {
13.     int s=0;
14.     for (int i=0;i<n;i++)
15.             s=s+ tongchusocuaso(a[i]);
16.     return s;
17. }
```

Ví dụ 5-2:

Cho mảng một chiều a chứa n số nguyên. Hãy viết các hàm thực hiện các công việc sau:

a. *Tìm giá trị lớn nhất của mảng*
b. *Hãy viết hàm tìm vị trí của số dương lẻ lớn nhất của mảng; nếu không tìm thấy thì trả về giá trị -1.*
c. *Tìm số nguyên tố lớn nhất; nếu không có trả về giá trị 0.*

Ví dụ: n=8

19 11 8 1 15 3 2 22

Thì kết quả là :

a. 22
b. 0
c. 19

Hướng dẫn:

Câu a.

Áp dụng kỹ thuật lính canh: đặt biến lính canh max bằng phần tử đầu tiên của mảng, sau đó duyệt các phần tử tiếp theo của mảng, nếu phần tử nào có giá trị lớn hơn max thì gán max bằng phần tử đó. Khi duyệt xong mảng thì max chính là giá trị lớn nhất cần tìm.

```
1.  int  giatrilonnhat(int a[], int n)
2.  {
3.      int max=a[0];
4.      for (int i=1;i<n;i++)
5.      if (a[i]>max)
```

```
6.              max=a[i];
7.      return max;
8.  }
```

Lưu ý: Nếu cần tìm giá trị nhỏ nhất của mảng thì thay phép so sánh > ở hàm trên bằng phép so sánh <. Đây là bài toán áp dụng lính canh không có điều kiện. Với các bài toán như tìm số nguyên tố nhỏ nhất/lớn nhất trong mảng và nếu không tìm thấy thì trả về giá trị k chẳng hạn; thì đây là dạng bài toán cần sử dụng kỹ thuật lính canh có điều kiện một cách hợp lý.

Câu b.

Áp dụng kỹ thuật lính canh có điều kiện:

Bài toán tìm số dương lẻ lớn nhất, nên phần tử lính canh ở đây nên được chọn là 0; max=0 và cũng do bài toán yêu cầu nếu không tìm thấy thì trả về -1 nên ta sử dụng thêm một lính canh nữa cho vị trí là vitri = -1. Áp dụng kỹ thuật lính canh duyệt tất cả các phần tử của mảng, nếu phần tử nào là số lẻ và lớn hơn lính canh max thì ta cập nhật lại giá trị của hai lính canh max và vitri. Khi duyệt xong mảng thì lính canh vitri chính là kết quả cần tìm.

```
1.  int soduonglelonnhat(int a[],int n)
2.  {
3.      int vitri=-1,max=0;
4.      for (int i=0;i<n;i++)
5.      if (a[i]%2!=0 && a[i]>max)
6.      {
7.              max=a[i];       vitri=i;
8.      }
9.      return vitri;
10. }
```

Câu c.

Áp dụng kỹ thuật lính canh có điều kiện như câu b.

```
1.  int  nguyentolonnhat(int a[], int n)
2.  {
3.      int max=0;
4.      for (int i=1;i<n;i++)
5.      if (nguyento(a[i]) && a[i]>max)
```

```
6.              max=a[i];
7.      return max;
8.  }
```

Ví dụ 5-3:

Cho dãy n số nguyên dương. Hãy viết các hàm thực hiện các công việc sau:
Viết hàm xóa tất cả các số nguyên tố trong mảng.

Ví dụ: n=8

5 121 5 5 11 22 2 22

Thì kết quả là :

121 22 22

Hướng dẫn:

Áp dụng kỹ thuật xóa phần tử trong mảng: Trước hết viết một hàm bổ trợ xoaphantu để xóa một phần tử tại vị trí k như đã mô tả ở phần trên, sau đó duyệt từng phần tử của mảng; nếu vị trí i nào mà a[i] là số nguyên tố thì xóa phần tử tại vị trí đó; lưu ý khi xóa xong một phần tử thì cần lùi i về bên trái 1 vị trí. Tham số *n* trong hàm xoasonguyento phải được truyền theo kiểu tham biến.

Sau đây là hai hàm minh họa cho ý tưởng trên.

```
1.  void xoaphantu(int a[], int &n, int k)
2.  {
3.      for (int i=k;i<n;i++)
4.              a[i]=a[i+1];
5.      n--;
6.  }
7.  void xoasonguyento(int a[], int &n)
8.  {
9.      for (int i=0; i<n;i++)
10.     if (nguyento(a[i]))
11.             {
12.             xoaphantu(a,n,i);
13.             i--;
14.             }
15. }
```

Ví dụ 5-4:

Cho dãy n số nguyên dương. Hãy viết các hàm thực hiện các công việc sau:

a. *Viết hàm in các số trong mảng thỏa tính chất: Số đó là số nguyên tố và khi đảo ngược các chữ số của số đó thì cũng thu được một số nguyên tố (ví dụ các số 2,113,149 thỏa cả tính chất trên)*
b. *Tìm chiều dài của đoạn con liên tiếp dài nhất chứa toàn số nguyên tố. Nếu không có thì trả về giá trị 0.*
c. *Viết hàm đếm tần số xuất hiện của các giá trị.*

Ví dụ: n=8

5 121 5 5 11 22 2 22

Thì kết quả là :

a. 5 5 5 11 2

b. 3

c. 5 3
121 1
11 1
22 2
2 1

Hướng dẫn:

Câu a.

Để giải quyết yêu cầu này cần viết 2 hàm sau: thứ nhất, hàm kiểm tra xem *n* có phải là số nguyên tố hay không; thứ hai, là hàm tìm số đảo ngược của một số. Hàm tìm số đảo ngược của một số là một bài tập nhỏ được viết như sau.

```
1. int  sodaonguoc(int n)
2. {
3.      int s=0;
4.      while (n>0)
5.      {       s=s*10+n%10;
6.              n=n/10;
7.      }
8.      return s;
9. }
```

```
10. void songuyentokep(int a[], int n)
11. {
12.     for (int i=0;i<n;i++)
13.     if (nguyento(a[i]) && nguyento(sodaonguoc (a[i])))
14.             cout<<a[i]<<" ";
15. }
```

Câu b.

Sử dụng 2 biến d và max; d là biến đếm số lượng số nguyên tố của mỗi đoạn; khi bắt đầu lại mỗi đoạn mới thì d lại được gán bằng 0; dấu hiệu để nhận biết để khởi đầu mỗi đoạn là a[i] không phải là số nguyên tố; max là giá trị lớn nhất của các giá trị d. Lưu ý là khi duyệt xong mảng thì phải kiểm tra giá trị d cuối cùng với giá trị max. giá trị lớn nhất của d cuối cùng và max chính là kết quả cần tìm.

```
1.  int doannguyentodainhat(int a[], int n)
2.  {
3.      int d=0,max=0;
4.      for (int i=0;i<n;i++)
5.              if (nguyento(a[i]))
6.                      d++;
7.              else
8.              {
9.                      if (d>max)      max=d;
10.                     d=0;
11.             }
12.     return (d>max?d:max);
13. }
```

Câu c.

Sử dụng thêm mảng phụ d có cùng kích thước với mảng a; mảng d sẽ lưu trữ tần số của các giá trị của mảng a. Đầu tiên mảng d được gán toàn giá trị 1; tiếp theo duyệt từng cặp phần tử a[i],a[j]; nếu a[i] bằng a[j] thì tăng tần số d[i] của phần tử a[i] lên 1 đơn vị và giảm tần số d[j] của phần tử a[j] về bằng 0. Việc gán d[j] bằng 0 có ý nghĩa là giá trị a[j] đã được xét ở các lần lặp trước đó; do đó khi xét điều kiện a[i] bằng a[j] cần phải thêm điều kiện d[i] > 0. Cuối cùng, d[i] chính là tần số của giá trị a[i], việc thêm điều kiện

d[i]>0 có ý nghĩa mỗi giá trị của mảng chỉ được đếm tần số một lần cho phù hợp với yêu cầu của bài toán,

```
1.  void tansoxuathien(int a[], int n)
2.  {
3.      int d[1000];
4.      for (int i=0;i<n;i++)
5.          d[i]=1;
6.      for (i=0;i<n-1;i++)
7.      for (int j=i+1;j<n;j++)
8.          if (a[i]==a[j] && d[i]>0)
9.          {
10.             d[i]=d[i]+1;
11.             d[j]=0;
12.         }
13.     for (i=0;i<n;i++)
14.         if (d[i]>0)
15.             cout<<"\n"<<a[i]<<" xuat hien "<<d[i]<<" lan ";
16. }
```

Lưu ý:

i. *Mỗi bài toán đặt ra thường có một kỹ thuật lập trình tối ưu tương ứng cho nó, nếu xác định đúng kỹ thuật lập trình để giải quyết thì sẽ có thuật toán tốt, ngược lại, sẽ là thuật toán tồi.*

ii. *Khi tìm phần tử lớn nhất trong mảng thì giá trị lính canh được đặt là -∞ và khi tìm phần tử nhỏ nhất trong mảng thì giá trị lính canh được đặt là +∞ (các giá trị -∞, +∞ thường các ngôn ngữ lập trình đã có hàm xác định; chính là giá trị nhỏ nhất/lớn nhất trong miền giá trị của kiểu dữ liệu đó). Trong trường hợp mảng chỉ chứa các số nguyên dương thì lính canh cho việc tìm giá trị lớn nhất thường được gán bằng 0.*

iii. *Khi cần nhập giá trị cho mảng, nên tạo sẵn một số bộ test mẫu và copy bộ test mẫu này vào màn hình nhập liệu chứ không nhất thiết phải liên tục nhập lại dữ liệu khi kiểm thử chương trình: Vừa mất thời gian vừa có thể thiếu chính xác.*

5.3. MẢNG HAI CHIỀU

Để biểu diễn một dãy các biến ta có thể dùng nhiều biến rời rạc có tên khác nhau, tuy nhiên cách làm này là không thuận lợi (thậm chí không khả thi) khi số lượng các biến là đủ nhiều. Việc sử dụng mảng là cách tốt nhất và là bắt buộc trong nhiều trường hợp xử lý các vấn đề bài toán thực tế.

Mảng là một số hữu hạn các phần tử có cùng một kiểu giá trị và có chung một tên gọi. Mỗi phần tử biểu diễn được một giá trị. Số phần tử của mảng phải được xác định ngay từ khi định nghĩa mảng. Mỗi phần tử của mảng được truy cập trực tiếp thông qua tên mảng cùng với chỉ số của mảng. Ngôn ngữ C/C++ cho phép làm việc với mảng một chiều, mảng hai chiều hoặc mảng nhiều chiều.

(Trong giáo trình này, chúng tôi chỉ đề cập chủ yếu đến mảng hai chiều; kiến thức về mảng một chiều đã được đề cập đến trong giáo trình Cơ sở lập trình trước đây).

5.3.1. Khai báo mảng hai chiều

Cú pháp khai báo mảng hai chiều

< kiểu dữ liệu> tên mảng[const m][const n];

Ví dụ sau đây khai báo một mảng hai chiều có tên là *a* gồm 20 dòng, 50 cột, các phần tử là các số nguyên.

int *c*[20][50];

Cấp phát bộ nhớ cho mảng

Các phần tử trong mảng được cấp phát các ô nhớ kế tiếp nhau trong bộ nhớ. Kích thước của mảng bằng kích thước một phần tử nhân với số phần tử của mảng.

Mảng hai chiều *m* dòng *n* cột được xem như là một bảng hình chữ nhật chứa *m*n* phần tử cùng kiểu dữ liệu.

Sau đây là ví dụ về một mảng hai chiều 3 dòng, 4 cột chứa các số nguyên.

	cột 0	cột 1	cột 2	cột 3
dòng 0	2	3	4	6
dòng 1	3	8	4	7
dòng 2	3	1	2	5

5.3.2. Truy nhập đến thành phần của mảng

Biến mảng lưu trữ địa chỉ ô nhớ đầu tiên trong vùng nhớ được cấp phát. Ngôn ngữ C/C++ đánh chỉ số các phần tử trong mảng bắt đầu từ 0. Các phần tử của mảng được truy nhập thông qua Tên mảng và Chỉ số của phần tử của phần tử trong mảng.

Mỗi phần tử của mảng được truy xuất thông qua tên mảng[chỉ số dòng][chỉ số cột], chẳng hạn để truy xuất phần tử tại dòng thứ *i* cột thứ *j* của mảng *a*, ta ghi là *a*[*i*][*j*].

5.3.3. Khởi tạo giá trị cho mảng

Các phần tử của mảng có thể được khởi tạo giá trị ngay khi khai báo

Chẳng hạn: int a[4] = {1,4,6,2}; //các phần tử 0,1,2,3.

int b[2][3]={ {1,2,3}, {4,5,6} }; //dòng 0 và 1

Số lượng giá trị khởi tạo không được lớn hơn số lượng phần tử trong mảng. Nếu số lượng này nhỏ hơn, các phần tử còn lại được khởi tạo giá trị 0.

int a[3][4] = { {1}, {4,5} }; //a[0][0]=1, a[1][0]=4, a[1][2]=5

//các giá trị khác của mảng a đều bằng 0.

int a[3][4] = { }; ←Tất cả đều mang giá trị 0

Có thể xác định kích thước mảng thông qua số giá trị khởi tạo nếu để trống kích thước mảng.

int a1 [8] = {2, 4, 6, 8, 10, 12, 14, 16}; //các phần tử từ a1[8] trở đi được gán //số nguyên ngẫu nhiên

int a2 [] = {2, 4, 6, 8, 10, 12, 14, 16};

5.4.CÁC THAO TÁC THƯỜNG GẶP TRÊN MẢNG HAI CHIỀU

5.4.1. Nhập mảng

Ví dụ 5.5.

Nhập một mảng hai chiều a có m dòng n cột; các phần tử là các số nguyên.

```
1. void nhapmang(int a[maxm][maxn], int &m, int &n)
2. {
3.  cout<<"Nhap vao so dong: ";cin>>m;
4.  cout<<"Nhap vao so cot : ";cin>>n;
5.  for (int i=0;i<m;i++)
6.  for (int j=0;j<n;j++)
7.   cin>>a[i][j];
8. }
```

5.4.2. Xuất mảng

Ví dụ 5.6.

Xuất một mảng hai chiều b có m dòng n cột lên màn hình.

```
1. void xuatmang(int a[maxm][maxn], int  m, int  n)
2. {
3.  for (int i=0;i<m;i++)
4.  {
5.  for (int j=0;j<n;j++)
6.   cout<<a[i][j]<<" ";
7.  cout<<endl;
8.  }
9. }
```

5.4.3. Tính tổng

Ví dụ 5.7.

Cho mảng hai chiều a có m dòng, n cột; các phần tử là các số nguyên. Hãy viết hàm tính tổng các phần tử của mảng thỏa điều kiện nào đó.

```
1. void tinhtong(int a[maxm][maxn], int  m, int  n)
2. {
3.  int tong=0;
4.  for (int i=0;i<m;i++)
5.  for (int j=0;j<n;j++)
6.   if (a[i] thỏa mãn điều kiện bài toán)
7.        tong = tong +a[i][j];
8.  cout<<"Tong cac phan tu cua mang la : "<< tong;
9. }
```

5.4.4. Đếm số phần tử thỏa điều kiện

Ví dụ 5.8.

Cho mảng hai chiều a có m dòng, n cột; các phần tử là các số nguyên. Hãy đếm xem có bao nhiêu phần tử thỏa các điều kiện cho trước.

```
1. int demso (int a[maxm][maxn], int  m, int  n)
2. {
3.  int dem=0;
4.  for (int i=0;i<m;i++)
5.  for (int j=0;j<n;j++)
6.   if (a[i] thỏa mãn điều kiện bài toán)
7.        dem++;
8.  return dem;
9. }
```

5.4.5. Đặt lính canh

Phần tử lính canh có thể hiểu là một ví trị nào đó của mảng, hoặc là một giá trị nào đó của mảng hoặc cũng có thể là một giá trị tùy theo từng yêu cầu cụ thể của bài toán,… Sau đó tiến hành duyệt các phần tử của mảng, nếu có phần tử nào thỏa điều kiện của bài toán thì tiến hành thay đổi vị trí lính canh (hoặc giá trị canh). Tùy thông tin đầu vào của vấn đề bài toán mà xác định thông tin về phần tử lính canh cho phù hợp.

Ví dụ 5.9.

Cho mảng hai chiều a có m dòng, n cột; các phần tử là các số nguyên. Hãy tìm số chính phương lớn nhất, nếu không tìm thấy trả về giá trị 0.

```
1. int chinhphuongln(int a[maxm][maxn], int  m, int  n)
2. {
3.  int cpmax=0;    // lính canh
4.  for (int i=0;i<m;i++)
5.  for (int j=0;j<n;j++)
6.  if ((sqrt(a[i][j])==int(sqrt(a[i][j]))) && (a[i][j]>cpmax))
7.   cpmax=a[i][j];
8.  return cpmax;
9. }
```

5.4.6. Đặt cờ hiệu

Duyệt lần lượt từng phần tử của mảng, nếu có phần tử thỏa điều kiện thì dừng thuật toán và trả về một giá trị *k* nào đó; nếu không có phần tử nào thỏa điều kiện thì trả trả về giá trị *k*’ nào đó; giá trị *k*’ mang ý đảo nghĩa với giá trị *k*; thường thì kỹ thuật cờ hiệu này nhằm trả lời cho câu hỏi: *có hay không ? đúng hay sai ? thuộc hay không thuộc ?....*

Khác với kỹ thuật lính canh là luôn phải duyệt hết các phần tử của mảng; kỹ thuật cờ hiệu chỉ cần tìm một phần tử thỏa điều kiện.

Ví dụ 5.10.

Cho mảng hai chiều a có m dòng, n cột; các phần tử là các số nguyên. Hãy kiểm tra xem mảng có chứa dòng nào tăng dần hay không ? Nếu có trả về giá trị 1; nếu không có trả về 0.

```
1. int kiemtradongtang(int a[maxm][maxn], int  m, int  n)
2. {
3.    int flag;
4.    for (int i=0;i<m;i++)
5.    {
6.          flag=1;   // đặt cờ
7.          for (int j=0;j<n-1;j++)
8.          if (a[i][j]>a[i][j+1])
9.                flag=0;
10.               if (flag==1) return 1;
11.         }
12.         return 0;
13.   }
```

Lưu ý:

Có thể giải quyết yêu cầu này bằng cách không sử dụng biến; cụ thể như sau: trong mỗi dòng i xét từng cặp phần tử kề nhau (a[i][j], a[i][j+1]), nếu có a[i][j] > a[i][j+1] thì bỏ qua dòng i; nghĩa là j sẽ chưa đi đến giá trị n, nếu tất cả các cặp (a[i][j], a[i][j+1]) đều thỏa a[i][j] ≤ a[i][j+1] thì dòng i là dòng tăng dần cần tìm.

5.4.7. Sắp xếp

Ví dụ 5.11.

Cho mảng hai chiều a có m dòng, n cột; các phần tử là các số nguyên. Hãy thực hiện các công việc sau đây:

a. *Sắp xếp các phần tử tăng dần trên mỗi dòng (từ trái qua phải).*

b. *Sắp các phần tử tăng dần trên mỗi dòng (từ trái qua phải) và tăng dần trên mỗi cột (từ trên xuống dưới), khi đó mỗi phần tử của dòng dưới phải lớn hơn tất cả các phần tử của dòng trên.*

Thuật toán

Câu a.

Xem mỗi dòng như là một mảng một chiều, áp dụng thuật toán sắp xếp mảng một chiều ta sẽ được mảng tăng dần trên từng dòng như đoạn chương trình sau:

```
1. void sapdong(int a[maxm][maxn], int  m, int  n)
2. {
3.  for (int i=0;i<m;i++)
4.  for (int j=0;j<n-1;j++)
5.  for (int k=j+1;k<n;k++)
6.   if (a[i][j]>a[i][k])
7.         hoanvi(a[i][j],a[i][k]);
8. }
```

Câu b.

Cách 1: Dùng mảng phụ

Chuyển dữ liệu từ mảng hai chiều *a* sang mảng một chiều *b*, sau đó sắp xếp các phần tử tăng dần trên mảng một chiều *b* và cuối cùng chuyển dữ liệu từ mảng một chiều *b* qua lại mảng hai chiều *a* (cách này bạn đọc tự giải xem như bài tập).

Cách 2: Không dùng mảng phụ

```
1. void sapxep(int a[maxm][maxn], int  m, int  n)
2. {
3.  for (int i=0;i<m*n-1;i++)
4.  for (int j=i+1;j<m*n;j++)
5.   if (a[i/n][i%n]>a[j/n][j%n])
6.        hoanvi(a[i/n][i%n],a[j/n][j%n]);
7. }
```

5.4.8. Xử lý dòng, cột

Ví dụ 5.12.

Cho mảng hai chiều a có m dòng, n cột; các phần tử là các số nguyên.

a. *Hoán đổi nội dung hai dòng k,l của mảng a.*

b. *Dịch xuống xoay vòng các dòng một vị trí b (dòng 1 cuối cùng sẽ đưa lên vị trí dòng đầu tiên).*

Giải

a.

```
1. void hoanvidong(int a[maxm][maxn],int m, int n,int k,int l)
2. {
3.  for (int i=0;i<n;i++)
4.   hoanvi(a[k][i],a[l][i]);
5. }
```

b.

```
6. void xoayvong(int a[maxm][maxn], int  m, int  n)
7. {  for (int i=m-1;i>0;i--)
8.   hoanvidong(a,m,n,i,i-1);
9. }
```

Các thuật toán quay hoạch động và tham lam được trình bày các trong phần tiếp theo có một vai trò đặc biệt trong lĩnh vực thiết kế thuật toán; cả hai thuật toán này **thông thường** sử dụng cấu trúc dữ liệu mảng để lưu trữ.

BÀI TẬP

Viết chương trình hoàn chỉnh cho các bài toán sau đây

BT5-1.

Cho mảng một chiều *a* chứa *n* số nguyên dương.

a.Kiểm tra mảng có tồn tại giá trị chẵn nhỏ hơn 2014 hay không ? Nếu có thì trả về giá trị 1 nếu không có thì trả về giá trị 0.

b.Kiểm tra các phần tử trong mảng có lập nên một cấp số cộng không ? Nếu có hãy chỉ ra công sai *d*; nếu không trả về giá trị 0.

c.Tìm số nguyên tố nhỏ nhất; nếu không có trả về giá trị 0.

d.Tìm vị trí của số chẵn nhỏ nhất của mảng. Nếu không tìm thấy thì trả về giá trị -1.

e.Hãy đếm xem trong dãy có bao nhiêu số hoàn chỉnh ? (số hoàn chỉnh là số bằng tổng các ước số thực sự của nó, chẳng hạn 6 là số hoàn chỉnh vì 6 = 1 + 2 + 3).

BT5-2.

Cho mảng một chiều *a* chứa *n* số nguyên dương.

a.Hãy tìm tất cả các cặp số nguyên tố cùng nhau (hai số a,b là nguyên tố cùng nhau nếu ước số chung lớn nhất của chúng bằng 1).

b.Tìm ước số chung lớn nhất, bội số chung nhỏ nhất của n số trên.

c.Tìm chiều dài của đoạn con liên tiếp dài nhất chứa toàn số chẵn. Nếu không có thì trả về giá trị 0.

d.Tìm chiều dài của dãy con tăng dài nhất.

BT5-3.

Cho dãy *a* gồm *n* số tự nhiên $a_o, a_1, a_2, ..., a_{n-1}$. Hãy thực hiện các công việc sau:

a.Hãy đưa các số chẵn (ngoại trừ số 0) về đầu mảng, các số lẻ về cuối mảng, còn các số 0 ở giữa.

b.Tìm giá trị lớn thứ *k* của mảng (giả sử mảng có giá trị lớn thứ *k*).

c.Tìm độ dài của dãy con liên tiếp dài nhất của dãy *a* chứa toàn số nguyên tố.

d.Đếm số lượng số đối xứng có từ hai chữ số trở lên trong dãy *a*. Ví dụ 11, 121, 64446,… là các số đối xứng.

e.Tìm dãy *b*, biết số b_i là viết đảo ngược của số ai; ví dụ a_i =5334 thì b_i tương ứng là 4335.

f.Đếm số lượng số của dãy *a* thỏa điều kiện: 2 số liền kề trái/phải với nó là số nguyên tố.

g.Tìm dãy *c*, biết c_i là số lượng các số nguyên tố nhỏ hơn a_i; ví dụ a_i =18 thì c_i tương ứng là 7.

BT5-4.

Cho mảng một chiều *a* chứa *n* số nguyên dương (kiểu unsigned long). Hãy thực hiện các công việc sau:

a.Tìm tổng các chữ số của tất cả các số của mảng *a*.

b.Tìm số nguyên tố lớn nhất của mảng *a*; nếu mảng *a* không có số nguyên tố nào thì xuất giá trị 0.

c.Đếm xem trong mảng *a* có bao nhiêu số nguyên tố.

d.Tính tổng các ước số thực sự của mỗi số của mảng *a*. Ví dụ 6 có các ước số thực sự là 1,2,3

e.Tìm giá trị trung bình cộng các số của mảng *a*.

f.Tìm số nguyên tố nhỏ nhất của mảng *a*; nếu mảng *a* không có số nguyên tố nào thì xuất giá trị 0.

g.Cho biết số lượng các ước số của từng số của mảng *a*. Ví dụ 6 có 4 ước số là 1,2,3,6.

h.Biến đổi mỗi số của mảng *a* về số nguyên tố nhỏ nhất lớn hơn hoặc bằng nó.

BT5-5.

Cho dãy *n*+1 số nguyên a_0, a_1,…,a_{n-1}, a_n và một số thực *x*. Hãy tính giá trị của biểu thức $P(a,n,x) = a_nx^n+a_{n-1}x^{n-1}$ +…+a_1x+a_0.

BT5-6.

a. Cho một dãy *n* số nguyên $a_0,a_2,...a_{n-1}$, mỗi số có giá trị tuyệt đối không vượt quá 32000. Hãy tìm ra 3 số trong dãy trên có tích lớn nhất.

b. Cho hai tập *X* và *Y*; mỗi tập gồm *n* số nguyên và số nguyên *k*. Hỏi có tìm được một số *x* thuộc *X* và một số *y* thuộc *Y* sao cho *x + y = k* hay không ?

c. Cho dãy *n* số nguyên không âm và số nguyên *k*. Hỏi dãy đã cho có chứa hai số với tổng là bằng *k* hay không ? Nếu có hãy đưa ra hai số đó ($n <= 10^6$, $k <= 10^9$).

d. Cho dãy *n* số nguyên $a_1,a_2,…,a_n$. Số a_p ($1<=p<=n$) được gọi là một số trung bình cộng trong dãy nếu tồn tại 3 chỉ số i,j,k ($1<=i,j,k<=n$) đôi một khác nhau, sao cho $a_p=(a_i+a_j+a_k)/3$. Cho *n* và dãy $a_1,a_2,…,a_n$. Hãy tìm số lượng các số trung bình cộng trong dãy.

BT5-7.

Cho hai dãy số nguyên {*a*} và {*b*}, trong đó dãy {*a*} có *n* phần tử và dãy {*b*} có m phần tử. Các số trong các dãy này đã được sắp tăng dần. Hãy trộn các phần tử của hai dãy này thành một dãy {*c*} sao cho dãy {*c*} cũng được sắp tăng dần.

BT5-8.

Cho mảng chứa các ký tự chữ hoa hoặc chữ thường và ký tự khoảng trắng.

a. Đếm xem trong mảng có bao nhiêu ký tự chữ hoa ? Bao nhiêu ký tự là chữ thường.

b. Tìm tần số xuất hiện của các ký tự.

c. Tìm mã ASCII của mỗi ký tự.

d. Sắp xếp các ký tự theo chiều tăng (theo mã ASCII), các ký tự khoảng trắng giữ nguyên vị trí.

e. Các ký tự liền nhau gọi là một từ; hãy tìm một từ đầu tiên bên trái, một từ đầu tiên bên phải.

f. Tìm một từ dài nhất, từ dài nhất này có bao nhiêu ký tự ?

g. Hãy chuyển ký tự đầu của mỗi từ thành chữ hoa, các ký tự khác thành chữ thường.

h. Loại bỏ các ký tự khoảng trắng trước từ bên trái (nếu có), Loại bỏ các ký tự khoảng trắng sau từ bên phải (nếu có) và giữa các từ chỉ giữ lại đúng một ký tự khoảng trắng.

BT5-9.

Cho dãy *n* số nguyên {a}, dãy con liên tiếp là dãy mà thành phần của nó là các thành phần liên tiếp nhau trong {a}. Ta gọi tổng của dãy con là tổng tất cả các thành phần của nó. Tìm tổng lớn nhất trong tất cả các tổng của các dãy con của {a}.

Ví dụ nếu *n* = 7 và dãy số như sau:

4 -5 6 -4 2 3 -7

thì kết quả tổng là 7.

BT5-10.

Cho hai tập *A* và *B* chứa các phần tử là các số nguyên (trong mỗi tập hợp các giá trị phải khác nhau).

a. Tính: $A \cap B$,

b. Tính: $A \cup B$,

c. Tính: $A - B$

Ví dụ : A={1,2,4,6}, B={2,1,7,3,8} thì kết quả là :

$A \cap B$={1,2}, $A \cup B$ ={1,2,4,6,7,3,8}, $A - B$={4,6}

BT5-11.

Cho ma trận m dòng n cột chứa các số nguyên dương. Vị trí dòng dòng, cột của mảng bắt đầu từ 0.

a. Hãy đếm xem có bao nhiêu số hoàn chỉnh ?

b. Tìm giá trị số nguyên tố nhỏ nhất ?

c. Tính tổng các chữ số của tất cả các số.

d. Đếm xem có bao nhiêu số thỏa điều kiện "là số nguyên tố và khi viết đảo ngược của nó cũng là số nguyên tố" ?

e. Tìm vị trí (i,j) của phần tử thỏa điều kiện:là phần tử lớn nhất ở dòng i và là phần tử nhỏ nhất ở cột j hoặc là phần tử nhỏ nhất ở dòng i và là phần tử lớn nhất ở cột j. Dòng, cột của mảng bắt đầu từ 0.

f. Hoán đổi nội dung của 2 dòng d_1, d_2 $(0<=d_1,d_2<\text{m}, d_1 \text{ khác } d_2)$

g. Hoán đổi nội dung của 2 cột c_1, c_2 $(0<=c_1,c_2<\text{n}, c_1 \text{ khác } c_2)$

BT5-12.

Cho mảng hai chiều a có m dòng n cột; các phần tử của a là các số nguyên. Vị trí dòng dòng, cột của mảng bắt đầu từ 0. Hãy viết các hàm thực hiện các công việc sau:

a. *Tính tổng các phần tử có chỉ số dòng là k (0≤ k<n).*

b. *Tính tổng các phần tử nằm trên các đường biên của mảng.*

c. *Liệt kê chỉ số các dòng có chứa ít nhất một số nguyên tố.*

d. *Liệt kê chỉ số các dòng mà các phần tử trên dòng đó đều là các số nguyên tố.*

Ví dụ : m=3,n=4

1 2 3 4

2 7 3 11

5 2 7 8

Khi k =1 thì kết quả câu a là : 23

Thì kết quả câu b là : 45

kết quả câu c là : 0,1,2

kết quả câu d là : 1

BT5-13.

Cho mảng hai chiều a có n dòng n cột ; các phần tử của a là các số nguyên. Vị trí dòng dòng, cột của mảng bắt đầu từ 0. Hãy viết các hàm thực hiện các công việc sau đây:

a. *Tìm giá trị lớn nhất của từng dòng.*

b. *Kiểm tra ma trận a có đối xứng qua đường chéo chính hay không ?*

c. *Kiểm tra xem đường chéo chính có được sắp tăng dần từ trên xuống dưới hay không ?*

d. *Tìm giá trị lớn nhất của các phần tử ở ma trận tam giác trên (các phần tử nằm trên đường chéo chính).*

e. *Dịch xuống k vị trí cho tất cả các dòng của mảng.*

f. *Tìm tần số xuất hiện của các giá trị.*

g. *Chèn thêm một dòng (mảng một chiều b có n phần tử) vào dòng thứ k của mảng a.*

h. *Xóa một dòng thứ k của a.*

BT5-14.

Cho hai ma trận a,b cùng có n dòng n cột ; các phần tử của a là các số nguyên. Hãy viết các hàm thực hiện các công việc sau:

a. *Tính tổng của hai ma trận a,b.*

b. *Tính tích của hai ma trận a,b.*

c. *Tạo ma trận mới bằng cách nhân các phần tử của ma trận a với giá trị lớn nhất của dòng tương ứng của ma trận b.*

Ví dụ :

Ma trận a Ma trận b

1 2 3 2 1 3

4 3 2 3 2 1

5 3 7 4 5 2

Kết quả câu a.	*Kết quả câu b.*	*Kết quả câu c.*
3 3 6	*20 20 11*	*3 6 9*
7 5 3	*25 20 19*	*12 9 6*
9 8 9	*47 46 32*	*25 15 35*

BT5-15.

Cho mảng hai chiều a có m dòng và n cột; các phần tử là các số nguyên dương.

a. *Sắp xếp các phần tử tăng dần theo từng cột.*

b. *Tạo mảng b từ mảng a với b_{ij} bằng tổng các chữ số tương ứng của phần tử a_{ij} ở mảng a.*

c. *Tạo mảng c từ mảng a với c_{ij} là số nguyên tố gần với a_{ij} nhất.*

d. *Tạo mảng d từ mảng a với d_{ij} là số lượng số chẵn bao quanh a_{ij} (8 hướng, không kể chính nó)*

e. *Tìm vị trí (dòng,cột) của các phần tử mà các phần tử bao quanh nó đều là số chẵn. Dòng, cột của mảng bắt đầu từ 0.*

f. *Tìm mảng b, biết $b_{ij}=a_{ij}*k$ với k là giá trị nhỏ nhất của dòng i nếu i là số lẻ và $b_{ij}=a_{ij}*k$ với k là giá trị lớn nhất của dòng i nếu i là số chẵn.*

g. *Tìm mảng c, biết $c_{ij}=l$, với l là tổng số lượng số nguyên tố trên dòng i và trên cột j (nếu a_{ij} là số nguyên tố thì chỉ được đếm một lần trên dòng hoặc trên cột).*

Ví dụ:

				Kết quả câu a				Kết quả câu b				Kết quả câu c				Kếtquả câu d			
3	4																		
3	5	2	3	3	5	2	2	3	5	2	3	3	5	2	3	1	3	3	3
11	22	44	2	11	20	44	3	2	4	8	2	11	23	43	2	2	4	6	4
19	20	100	54	19	22	100	54	10	2	1	9	19	19	101	53	2	3	5	3

BT5-16.

Cho bảng số gồm M dòng và N cột (3 ≤ M, N ≤ 100). Xác định bảng con gồm 3 dòng và 3 cột của bảng đã cho sao cho tổng các số trong bảng con là lớn nhất.

Dữ liệu: *Vào từ tập tin văn bản BANGSO.INP. Dòng đầu của tập tin là hai số nguyên M, N lần lượt là số dòng và số cột của bảng. Trên các dòng từ thứ 2 đến thứ M+1, dòng thứ i+1 chứa N số của dòng thứ i trong bảng.*

Kết quả: *Ghi ra tập tin văn bản BANGSO.OUT. Dòng đầu là tổng số lớn nhất tìm được. Dòng tiếp theo là hai số nguyên, chỉ dòng và cột chứa số trên cùng bên trái của bảng con 3 × 3 có tổng lớn nhất. Nếu có nhiều bảng con 3 × 3 có cùng tổng, chọn bảng con có số trên cùng bên trái có chỉ số dòng nhỏ nhất. Nếu có nhiều bảng con có cùng tổng và số trên cùng bên trái có cùng chỉ số dòng, chọn bảng con có số trên cùng bên trái có chỉ số cột nhỏ nhất.*

Ví dụ:

BANGSO.INP	***BANGSO.OUT***
6 5 *5 6 7 4 6* *7 7 8 6 5* *9 9 8 3 5* *8 8 7 6 4* *4 5 2 4 5* *3 4 2 3 4*	*71* *2 1*

BT5-17.

Cho mảng chứa các ký tự chữ hoa hoặc chữ thường và ký tự khoảng trắng.

a. *Đếm xem trong mảng có bao nhiêu ký tự chữ hoa ? Bao nhiêu ký tự là chữ thường.*

b. *Tìm tần số xuất hiện của các ký tự.*

c. *Tìm mã ASCII của mỗi ký tự.*

d. *Sắp xếp các ký tự theo chiều tăng (theo mã ASCII), các ký tự khoảng trắng giữ nguyên vị trí.*

e. *Các ký tự liền nhau gọi là một từ; hãy tìm một từ đầu tiên bên trái, một từ đầu tiên bên phải.*

f. *Tìm một từ dài nhất, từ dài nhất này có bao nhiêu ký tự ?*

g. *Hãy chuyển ký tự đầu của mỗi từ thành chữ hoa, các ký tự khác thành chữ thường.*

h. *Loại bỏ các ký tự khoảng trắng trước từ bên trái (nếu có), Loại bỏ các ký tự khoảng trắng sau từ bên phải (nếu có) và giữa các từ chỉ giữ lại đúng một ký tự khoảng trắng.*

BT5-18.

Trong toán học thống kê, số trung vị của một dãy số (gồm một số lẻ phần tử) là số đứng giữa của dãy số sau khi dãy số được sắp theo thứ tự tăng dần.

Bài toán: *Cho bảng vuông N × N (3 ≤ N ≤ 99, N là số nguyên lẻ) mà ở dòng i cột j của bảng chứa số nguyên R_{ij} với (1 ≤ R_{ij} ≤ 1000).*

Yêu cầu: *Tìm số trung vị của dãy các số tạo bởi dãy các số trung vị của các dòng của bảng.*

Dữ liệu: *Vào từ tập tin văn bản TRUNGVI.INP. Dòng đầu là của tập tin là số nguyên lẻ N, trên các dòng từ 2 đến N+1, dòng i+1 chứa N số của dòng i của bảng vuông.*

Kết quả: *Ghi ra tập tin văn bản TRUNGVI.OUT một số nguyên duy nhất là số trung vị tìm được.*

Ví dụ:

TRUNGVI.INP	**TRUNGVI.OUT**
5 1 5 3 9 5 2 5 3 8 1 6 3 5 9 2 8 8 3 3 2 5 4 4 4 4	4

Giải thích:

Số trung vị của các dòng lần lượt là 5, 3, 5, 3, 4. Số trung vị của dãy 5 3 5 3 4 là 4.

BT5-19.

a. *Hãy in tam giác Pascal với chiều cao h.*

Ví dụ với h=5 thì kết quả là:

```
    1
    1   1
    1   2   1
    1   3   3   1
    1   4   6   4   1
    1   5  10  10   5   1
```

b. *Hãy điền các số từ 1 đến n*n vào bảng sau theo quy luật được mô tả với những trường hợp cụ thể (n = 5,6) như sau:*

n=5

```
    11    10    4     3     1
    19    12    9     5     2
    20    18    13    8      6
    24    21    17    14    7
    25    23    22    16    15
```

n=6

```
    1     2     6     7     15    16
    3     5      8    14    17     26
    4     9     13    18    25     27
    10    12    19    24    28    33
    11    22    23    29    32    34
    21    22    30    31    35    36
```

Hãy nhập n từ bàn phím (1 < n <20) và in bảng lên màn hình theo quy luật trên.

c. *Hãy lập trình để xuất mảng hai chiều có kích thước là n với 7<n<25 theo quy luật được nhận biết từ cách biểu diễn với các trường hợp cụ thể sau:*

Với n=6

11 9 7 5 3 1

9 7 5 3 1 2

7 5 3 1 2 4

5 3 1 2 4 6

3 1 2 4 6 8

1 2 4 6 8 10

Với n=7

13 11 9 7 5 3 1

11 9 7 5 3 1 2

9 7 5 3 1 2 4

7 5 3 1 2 4 6

5 3 1 2 4 6 8

3 1 2 4 6 8 10

1 2 4 6 8 10 12

BT5-20.

a.Tìm ma phương lẻ cấp n (n>=3, n là số lẻ)

Ma phương là một bảng vuông cấp n, trong mỗi ô nhận một giá trị sao cho mỗi hàng, mỗi cột và mỗi đường chéo đều thoã màn một tính P nào đó cho trước.

Bài toán ma phương ở đây như sau:

*"Hãy điền các số từ 1 đến n*n vào ma trận cấp n (n>=3, chỉ xét trường hợp n là số lẻ) với tính chất P là tổng các số bằng nhau".*

(bài này có nhiều thuật toán, sau đây là một thuật toán điển hình)

Đầu tiên đặt số 1 vào ô chính giữa của dòng cuối.

i=n; j=(n/2) + 1;a[i,j]=1;

khi đã điền số k ta tại ô i,j tiến hành điền số tiếp theo(k+1) theo hướng Đông-Nam

(tăng i=i+1,j+j+1) và có thể gặp những sự cố sau đây:

Nếu (j>n) và (i<=n) nghĩa là ô đang xét có cột đã vượt ra ngoài bảng thì j=1 - đẩy phần tử k+1 qua cực trái

Nếu (i>n) và (j<=n) nghĩa là ô đang xét có dòng đã vượt ra ngoài bảng thì i=1 - đẩy phần tử k+1 lên dòng đầu tiên

Nếu (i>n) và (j>n) nghĩa là ô đang xét có cả dòng và cột đã vượt ra ngoài bảng thì i=i-2; j=j-1;

Nếu a[i,j]<>0 nghĩa là tại ô đang xét đã điền số thì i=i-2; j=j-1;

Ví dụ với N =5 thì OUTPUT là:

11	18	25	2	9
10	12	19	21	3
4	6	13	20	22
23	5	7	14	16
17	24	1	8	15

b.*Tìm ma phương chẵn cấp n(n>=4, n là số chẵn)*

1	14	15	4
12	7	6	9
8	11	10	5
13	2	3	16

BT5-21.

Siêu mã là một loại mã có nhiều ứng dụng quan trọng trong lĩnh vực mã hóa và truyền tin. Trong bài này, ta xét bài toán đơn giản sau đây về siêu mã. Cho u và v là hai xâu kí tự khác rỗng có độ dài hữu hạn. Xâu u được gọi là xâu con của xâu v nếu u có thể nhận được từ v bằng cách xóa bớt ít nhất một kí tự trong v. Một tập X các xâu khác rỗng có độ dài hữu hạn được gọi là ***siêu mã*** *nếu mọi cặp u, v bất kỳ thuộc X, u không là xâu con của v và v không là xâu con của u.*

Cho trước một tập X = {x_1, x_2, ..., x_N} gồm N xâu khác rỗng, mỗi kí tự trong xâu là 0 hoặc 1. Hãy kiểm tra xem X có là một siêu mã hay không?

Dữ liệu: *vào từ file văn bản* ***HCODE.INP*** *có định dạng như sau:*

- *Dòng đầu tiên chứa số nguyên dương N (N ≤ 500);*
- *Dòng thứ i trong N dòng tiếp theo ghi xâu x_i của tập X, độ dài của xâu x_i không quá 15, với i = 1, 2, ..., N.*

Kết quả: *ghi ra file văn bản* ***HCODE.OUT*** *có định dạng như sau:*

- *Nếu X là siêu mã thì ghi số 1;*
- *Nếu X không là siêu mã thì dòng đầu tiên ghi số 0, dòng thứ hai ghi chỉ số i nhỏ nhất mà hoặc x_i là xâu con của x_j hoặc x_j là xâu con của x_i, với x_i, x_j thuộc X, 1 ≤ i < j ≤ N.*

Ví dụ:

HCODE.INP	***HCODE.OUT***
5 *1111* *100101* *01011* *000* *0001000*	*0* *2*

HCODE.INP	***HCODE.OUT***
3 *010* *1000* *11*	*1*

BT5-22.

Một nhóm nghiên cứu xử lý ảnh đang giải quyết bài toán nhận dạng mặt người trong ảnh. Ảnh chụp mặt người sau khi đã xử lý là một bảng vuông A kích thước N x N (10 ≤ N ≤ 800) với mỗi ô (I,J) (1 ≤ I, J ≤ N) có giá trị từ 0 đến 255 là mức xám của ảnh tại ô này (trong đó 0 là màu nền). Để xác định vị trí có thể là mặt người, nhóm cần thống kê các đặc trưng có dạng hình vuông kích thước K x K (1 ≤ K ≤ 40) trong đó tất các các giá trị trong hình vuông đều phải khác 0.

Yêu cầu : Từ một ảnh chụp mặt người, hãy giúp nhóm nghiên cứu đếm tất cả các đặc trưng có trong ảnh đó.

Dữ liệu *: Vào từ file văn bản FEATURE.INP trong đó :*

- *Dòng đầu chứa hai sô N và K*
- *Dòng thứ I trong N dòng tiếp theo chứa tương ứng dòng thứ I của bảng A.*

Các số ghi trên một dòng được ghi cách nhau bởi ít nhất một khoảng trắng. Mỗi dòng có N số nguyên. Dòng thứ I là các giá trị của N phần tử trong dòng thứ I-1 trong bảng vuông A

Kết quả : G*hi ra file văn bản FEATURE.OUT số lượng đặc trưng tìm được.*

Ví dụ:

FEATURE.INP
6 2 0 12 15 0 33 30 17 19 23 15 16 0 11 12 0 14 14 0 0 10 11 8 10 0 0 8 7 12 0 0 0 0 11 13 0 0

FEATURE.OUT
7

BT5-23.

*Cho hình vuông n * n (n>1). các dòng/cột được đánh số từ 1 đến n. Mỗi số trong bảng có giá trị tuyệt đối không quá 10000. Đường chép chính của hình vuông là đoạn thẳng nối ô (1,1) với ô(n,n) và hình vuông có tất cả là 2n-1 đường chéo song song với đường chéo chính, các đường chéo này được đánh số từ 1 đến 2n-1 được tính từ đường chéo có chỉ số dòng từ nhỏ đến lớn.*

a.Hãy tìm đường chéo chỉ chứa toàn số nguyên tố

b.Hãy tìm đường chéo có tổng các phần tử trên đường chéo đó là lớn nhất.

c.Hãy tìm đường chéo có chứa số nguyên tố lớn nhất

(nếu mỗi câu có nhiều kết quả thì in các kết quả đó trên cùng dòng)

HINHVUONG.INP

2

2 2

2 2

HINHVUONG.OUT

1 2 3

2

1 2 3

BT5-24.

Cho lưới ô vuông kích thước M x N. các ô của lưới được sơn bởi một trong hai màu đen hăng trắng

Cần tìm hình vuông gồm toàn ô đen có kích thước lớn nhất

Dữ liệu vào:

File văn bản: LUOI.INP có cấu trúc như sau:

Dòng đầu ghi hai số M,N

Trong m dòng tiếp theo mỗi dòng ghi N số , trong đó a[I,j]=1 nếu ô (i.j) có màu đen và a[I,j]=0 nếu ô(I,j) có màu trắng.

Kết quả:

Ghi ra file văn bản LUOI.OUT có cấu trúc như sau:

Dòng thứ nhất ghi số k là kích thước của hình vuông tìm được

Dòng thứ hai ghi 4 số p,q,r,s. trong đó (p,q) là tọa độ của ô ở góc trên trái còn (r,s) là tọa độ của ô ở góc dưới bên phải của hình vuông tim được.

Ví dụ:

LUOI.INP

```
8 10
0 1 0 0 0 0 0 0 0 0
1 1 1 1 0 0 1 1 1 1
1 1 1 1 0 0 1 1 1 1
0 0 1 1 1 0 0 0 0 0
0 0 1 1 1 1 1 1 1 0
0 0 1 1 1 1 1 1 1 1
0 0 0 0 0 1 1 1 1 1
0 0 0 0 0 1 1 1 1 1
```

LUOI.OUT

```
4
5 6 8 9
```

CHƯƠNG 6. KIỂU DỮ LIỆU CÓ CẤU TRÚC

Trong chương 5 chúng ta đã nói về cấu trúc mảng để lưu trữ các phần tử dữ liệu, tuy nhiên mảng chỉ lưu trữ các phần tử có cùng kiểu dữ liệu nào đó (như mảng số nguyên, mảng các số thực hoặc mảng các chuỗi,…). Vấn đề là chúng ta cần lưu trữ các phần tử phức tạp hơn, trong đó mỗi phần tử có thể có nhiều thành phần con (thường còn được gọi là thuộc tính); mỗi thành phần này thuộc về một kiểu dữ liệu nào đó. Kiểu dữ liệu của từng thành phần này có thể là một kiểu dữ liệu chuẩn sẵn có của ngôn ngữ lập trình hoặc cũng có thể là một kiểu dữ liệu do người lập trình định nghĩa trước đó.

Ví dụ về một bảng tính lương đơn giản

TT	Họ và tên	Ngày sinh	Lương cơ bản	thưởng	Thực lãnh
1	Nguyễn Văn Thành	12/10/1972	960000	1200000	2160000
2	Bùi Phương Thảo	01/01/1981	780000	1200000	1980000
3	Lê Tuấn	11/12/1983	750000	1200000	1950000

Qua bảng này, ta thấy rằng dữ liệu cần lưu trữ gồm nhiều phần tử (nhiều dòng), mà mỗi dòng là tập hợp nhiều thành phần con như họ và tên, ngày sinh, lương cơ bản, thưởng, thực lãnh và mỗi thành phần này lại thuộc về một kiểu dữ liệu cụ thể nào đó.

Tóm lại có thể nói rằng: *kiểu dữ liệu cấu trúc là một sự mở rộng của kiểu dữ liệu mảng mà mỗi phần tử của mảng lúc này có nhiều thành phần.*

6.1. KHAI BÁO CẤU TRÚC- KHAI BÁO BIẾN CẤU TRÚC

Đinh nghĩa một kiểu dữ liệu có cấu trúc mới

```
struct tên_kiểu_cấu_trúc
{       //khai báo các thành phần của cấu trúc
        <kiểu dữ liệu> <tên thuộc tính 1>
        <kiểu dữ liệu> <tên thuộc tính 2>
        <kiểu dữ liệu> <tên thuộc tính 3>
        …
};
```

Trong đó struct là từ khóa, tên_kiểu_cấu_trúc là một tên bất kỳ do người lập trình tự đặt, thành phần của cấu trúc có thể là : biến, mảng,. . . hoặc một cấu trúc khác mà kiểu của nó đã được định nghĩa trước đó.

chẳng hạn cấu trúc của bảng lương có thể được mô tả như sau:

```
struct hoso
{
        char hoten [30];
        struct ngaysinh
        long luongcoban
        long thuong;
        long thuclanh;
};
```

thành phần của cấu trúc có thể có kiểu dữ liệu là một cấu trúc đã được định nghĩa trước đó chẳng hạn như ví dụ sau:

```
struct ngaythang
{
        int ngay;
        int thang;
        int nam;
};
struct hoso
{
        char hoten [30];
        struct ngaythang ns;// kiểu ngaythang đã được định nghĩa trước
        long luongcoban;
        long thuong;
        long thuclanh;
};
```

Khai báo biến cấu trúc

<kiểu cấu trúc> <tên biến cấu trúc>

Chẳng hạn:

hoso nv[1000];

hoso *nv;

Lưu ý: Sử dụng lệnh gets để nhập chuỗi; trước mỗi lệnh nhập chuỗi cần sử dụng lệnh xóa vùng đệm bàn phím bằng lệnh *flushall() hoặc fflush(stdin).*

6.2. TRUY XUẤT ĐẾN TỪNG THÀNH PHẦN CỦA CẤU TRÚC

Biến cấu trúc được khai báo tĩnh

Với cấu trúc đơn thì theo cú pháp sau:

tên biến cấu trúc**.**tên thành phần

Ví dụ:

nv[i].hoten; // chỉ *họ tên* của nhân viên thứ i

nv[i].thuong; // chỉ *thưởng* của nhân viên thứ i

Với cấu trúc phức thì theo cú pháp sau:

Tên biến cấu trúc**.**tên biến cấu trúc**.**tên thành phần

Ví dụ:

nv[i]. ns.ngay;

nv[i]. ns.thang;

nv[i]. ns.nam;

Biến cấu trúc được khai báo động

tên biến cấu trúc ->tên thành phần.

Ví dụ 6-1.

Viết chương trình cộng hai phân số a/b và c/d trong đó a,b,c,d là các số nguyên. Yêu cầu phân số kết quả phải ở dạng tối giản.

```
1. #include <iostream.h>
2. struct phanso
3. {
4.    int tu;
5.    int mau;
6. };
7. void    nhapps(phanso &ps);
8. void    xuatps(phanso ps);
```

```
9. phanso congps( phanso ps1, phanso ps2);
10.  int  uscln(int a, int b);
11.  int main()
12.  {
13.       phanso ps1,ps2;
14.       nhapps(ps1);
15.       nhapps(ps2);
16.       xuatps(congps(ps1,ps2));
17.  }
18.  void nhapps(phanso &ps)
19.  {
20.       cout<<"Nhap vao tu so  : ";   cin>>ps.tu;
21.       cout<<"Nhap vao mau so : ";cin>>ps.mau;
22.  }
23.  void xuatps(phanso ps)
24.  {
25.       int uc=uscln(ps.tu,ps.mau);
26.       ps.tu=ps.tu/uc;
27.       ps.mau=ps.mau/uc;
28.       cout<<"Ket qua: "<<ps.tu<<"/"<<ps.mau<<endl;
29.  }
30.  phanso congps( phanso ps1, phanso ps2)
31.  {
32.       phanso ps;
33.       ps.tu=ps1.tu*ps2.mau+ps2.tu*ps1.mau;
34.       ps.mau=ps1.mau*ps2.mau;
35.       return ps;
36.  }
37.  int uscln(int a, int b)
38.  {
39.       int r=a%b;
40.       while (r!=0)
41.       {
42.            a=b;
43.            b=r;
```

```
44.             r=a%b;
45.         }
46.         return b;
47.   }
```

Ví dụ 6-2.

Cho mảng một chiều chứa n phân số (tử số và mẫu số là các số nguyên).

a.Hãy tìm phân số có giá trị lớn nhất.

b.Sắp xếp các phân số theo chiều tăng dần

```
1. #include <math.h>
2. #include <iostream.h>
3. #include <iostream.h>
4. struct phanso
5. {
6.    int tu;
7.    int mau;
8. };
9. void nhapmangps(phanso ps[], int &n);
10.   void xuatmangps(phanso ps[], int &n);
11.   void timpslonnhat(phanso ps[], int n);
12.   void sapxepmangps(phanso ps[], int n);
13.   int main()
14.   {
15.         phanso ps[1000];int n;
16.         nhapmangps(ps,n);
17.         timpslonnhat(ps,n);
18.         sapxepmangps(ps,n);
19.         xuatmangps(ps,n);
20.   }
21.   void nhapmangps(phanso ps[], int &n)
22.   {
23.         cout<<"Nhap so luong phan so n = ";
24.         cin>>n;
25.         for (int i=0; i<n;i++)
```

```
26.       {
27.            cout<<"Nhap phan so thu "<<i<<":"<<endl;
28.            cin>>ps[i].tu;
29.            cin>>ps[i].mau;
30.       }
31.  }
32.  void xuatmangps(phanso ps[], int &n)
33.  {
34.       cout<<endl;
35.       for (int i=0; i<n;i++)
36.            cout<<ps[i].tu<<"/"<<ps[i].mau<<";";
37.  }
38.  void timpslonnhat(phanso ps[], int n)
39.  {
40.       phanso psmax;
41.       psmax =ps[0];
42.       for (int i=1; i<n;i++)
43.            if (ps[i].tu*psmax.mau>psmax.tu* ps[i].mau)
44.                 psmax=ps[i];
45.       cout<<"\nPhan so lon nhat la : "<<psmax.tu<<"/"<<psmax.mau;
46.  }
47.  void hoanvi(phanso &ps1, phanso &ps2)
48.  {    phanso temp=ps1;
49.       ps1=ps2;
50.       ps2=temp;
51.  }
52.  void sapxepmangps(phanso ps[], int n)
53.  {
54.       for (int i=0;i<n-1;i++)
55.       for (int j=i+1;j<n;j++)
56.            if (ps[i].tu* ps[j].mau>ps[j].tu* ps[i].mau)
57.                 hoanvi(ps[i],ps[j]);
58.  }
```

Ví dụ 6-3.

Cho một danh sách lưu thông tin của các thí sinh khi thi tuyển vào lớp 10 với ba môn thi là Toán, Văn và Tiếng Anh, thông tin gồm :

- Số báo danh (chuỗi, tối đa 8 ký tự)

- Họ tên (chuỗi, tối đa 32 ký tự)

- Điểm môn Toán (số nguyên)

- Điểm môn Văn (số nguyên)

- Điểm môn Tiếng Anh (số nguyên)

Hãy in lên màn hình những thí sinh có điểm tổng ≥ 18 và không có môn nào dưới điểm 5; danh sách này phải được sắp theo thứ tự điểm tổng giảm dần.

```
1. #include <iostream.h>
2. #include <string.h>
3. #include < iostream.h>
4. struct hoso
5. {
6.    char      sobd[8];
7.    char      hoten[20];
8.    int  toan;
9.    int  van;
10.       int       anh;
11.       int  tong;
12.   };
13.   void nhap(hoso hs[], int &n);
14.   void xuat(hoso hs[], int n);
15.   void sapxep(hoso hs[], int n);
16.   void xuatkq(hoso hs[], int n);
17.   int main()
18.   {
19.       hoso hs[1000];
20.       int n;
21.       nhap(hs,n);
22.       xuat(hs,n);
23.       sapxep(hs,n);
```

```
24.       xuatkq(hs,n);
25.  }
26.  void nhap(hoso hs[], int &n)
27.  {
28.       cin>>n;
29.       for (int i=0;i<n;i++)
30.       {
31.            cout<<"So bao danh: ";
32.            flushall();gets(hs[i].sobd);
33.            cout<<"Ho va ten: ";  flushall();gets(hs[i].hoten);
34.            cout<<"Diem Toan:";cin>>hs[i].toan;
35.            cout<<"Diem Van:";cin>>hs[i].van;
36.            cout<<"Diem Tieng Anh:";cin>>hs[i].anh;
37.            hs[i].tong=hs[i].toan+hs[i].van+hs[i].anh;
38.            cout<<endl;
39.       }
40.  }
41.  void xuat(hoso hs[], int n)
42.  {
43.       for (int i=0;i<n;i++)
44.       cout<<hs[i].sobd<<hs[i].hoten<<hs[i].toan<<hs[i].van
  <<hs[i].anh<<endl;
45.  }
46.  void sapxep(hoso hs[], int n)
47.  {
48.       for (int i=0;i<n-1;i++)
49.       for (int j=i+1;j<n;j++)
50.            if (hs[i].tong<hs[j].tong)
51.            {
52.                 hoso temp=hs[i];
53.                 hs[i]=hs[j];
54.                 hs[j]=temp;
55.            }
56.  }
57.  void xuatkq(hoso hs[], int n)
```

```
58.  {
59.  for (int i=0;i<n;i++)
60.  if  (hs[i].tong>=18  &&  hs[i].toan>=5  &&  hs[i].ly>=5  &&
   hs[i].anhvan>=5)
61.       cout<<hs[i].sobd<<hs[i].hoten<<hs[i].toan<<hs[i].ly<
   <hs[i].anhvan<<endl;
62.  }
```

Ví dụ 6-4.

Viết chương trình thực hiện các công việc sau:

a.*Nhập vào hồ sơ của n nhân viên với các thông tin: Họ và tên, ngày sinh, lương cơ bản, thưởng, thực lãnh; trong đó thực lãnh = lương cơ bản + thưởng.*

b*.In danh sách nhân viên theo bậc lương giảm dần.*

```
1. #include<iostream.h>
2. #include<stdlib.h>
3. #include<iomanip.h>
4. struct ngaysinh
5. {
6.   int ngay;
7.   int thang;
8.   int nam;
9. };
10.  struct hoso
11.  {
12.       char hoten [30];
13.       struct ngaysinh ns;
14.       long luongcoban;
15.       long thuong;
16.       long thuclanh;
17.  };
18.  void nhap(hoso nv[], int &n);
19.  void xuat(hoso nv[], int n);
20.  void sapxep(hoso nv[], int n);
21.  int main()
```

```
22.  {
23.       int n;
24.       hoso nv[1000];
25.       nhap(nv,n);
26.       sapxep(nv,n);
27.       xuat(nv,n);
28.  }
29.  void nhap(hoso nv[], int &n)
30.  {
31.       cout<<"Ho so co bao nhieu nhan vien:";cin>>n;
32.       for(int i=0;i<n;i++)     //bắt đầu từ 1 để dễ theo dõi
33.       {
34.            cout<<"\Nhap nhan vien thu: "<<i;
35.            cout<<"\nHo ten:"; flushall();gets(nv[i].hoten);
36.            cout<<"Ngay:";cin>>nv[i]. ns.ngay;
37.            cout<<"Thang:";cin>>nv[i]. ns.thang;
38.            cout<<"Nam:";cin>>nv[i]. ns.nam;
39.            cout<<"Luong co ban:";cin>>   nv[i].luongcoban;
40.            cout<<"Thuong:");cin>>nv[i].thuong;
41.            nv[i].thuclanh = nv[i].luongcoban+ nv[i].thuong;
42.       }
43.  }
44.  void xuat(hoso nv[], int n)
45.  {
46.  cout<<"  Ho ten   Ngay sinh  LCB  Thuong  Thuc lanh"<<endl;
47.       for(int i=0;i<n;i++)
48.       cout<<nv[i].hoten<<setw(5)<<nv[i].
  ns.ngay<<setw(5)<<nv[i].   ns.thang<<setw(5)<<nv[i].  ns.nam
  <<setw(5)<<nv[i].luongcoban<<setw(5)<<nv[i].thuong<<setw(5)<
  <nv[i].thuclanh;
49.  }
50.  void sapxep(hoso nv[], int n)
51.  {
52.       for(int i=0;i<n-1;i++)
53.       for(int j=i+1;j<n;j++)
```

```
54.         if(nv[i].thuclanh > nv[j].thuclanh)
55.         {
56.               hoso temp = nv[i];
57.               nv[i] = nv[j];
58.               nv[j] = temp;
59.         }
60.   }
```

BÀI TẬP

Viết chương trình cho các bài toán sau đây

BT6-1.

Cho mảng một chiều các phân số (tử số và mẫu số là số nguyên), hãy thực hiện các công việc sau đây:

a. Viết hàm sắp xếp các phân số có giá trị tăng dần.

b. Tính tổng giá trị các phân số.

c. In các phân số của mảng ở dạng tối giản

d. Đếm xem trong mảng có bao nhiêu phân số có giá trị lớn hơn 0 và nhỏ hơn 1

e. Đếm xem trong mảng có bao nhiêu phân số (ở dạng tối giản) mà tử số và mẫu số của phân số đó đều là các số nguyên tố

BT6-2.

Trong mặt phẳng tọa độ *OXY* cho *n* điểm có tọa độ x_i,y_i (i=1..*n*).

a. Đếm số lượng điểm nằm ở phần tư thứ I (x>0 và y>0)

b. Tìm cặp điểm gần nhau nhất

c. Tìm hình chữ nhật nhỏ nhất chứa hết n điểm trên (hình chữ nhật này có các cạnh song song với các trục tọa độ; điểm nằm trên cạnh hcn xem như thuộc hcn). (nghĩa là tìm đỉnh dưới trái, trên phái)

d. Tìm 3 điểm trong số n điểm trên sao cho diện tích của tam giác đó là lớn nhất

e. Đếm xem trong n điểm trên có tạo nên bao nhiêu đoạn thẳng song song với trục tung hoặc trục hoành (một cặp điểm tạo nên một đoạn thẳng)

BT6-3.

Cho dãy *n* khối lập phương. Trên mỗi mặt của mỗi khối lập phương ghi một giá trị nguyên từ 1 đến 9.

a. Tổng giá trị của mỗi khối lập phương bằng tổng các giá trị trên các mặt của khối lập phương đó. Hãy tìm tổng lớn nhất.

b. Hãy cho biết có bao nhiêu khối lập phương có giá trị của các mặt là các số khác nhau ?

c. Hãy cho biết giá trị nào xuất hiện nhiều lần nhất ? Bao nhiêu lần ?

BT6-4.

Trong kỳ thi tuyển sinh cao học có n thí sinh tham gia; mỗi thí sinh cần quản lý các thông tin sau:

- Số báo danh (kiểu chuỗi, 8 ký tự)
- Họ và tên (kiểu chuỗi, tối đa 32 ký tự)
- Điểm môn ngoại ngữ (số nguyên)

- Điểm môn cơ sở (số nguyên)
- Điểm môn chuyên ngành (số nguyên)

a. Tìm những thí sinh có điểm môn chuyên ngành là cao nhất.
b. Tìm những thí sinh có điểm của cả ba môn thi đều lớn hơn hoặc bằng 5.
c. Tìm những thí sinh có ít nhất một môn thi nhỏ hơn 5.
d. Tìm những thí sinh có điểm thi môn ngoại ngữ ≥ 5 và có tổng điểm môn cơ sở và môn chuyên ngành ≥ 12 ; trong đó tổng điểm môn cơ sở và môn chuyên ngành được sắp giảm dần.

BT6-5.

Cho n hình chữ nhật có các cạnh song song với các trục tọa độ, mỗi hình chữ nhật biết toạ độ dưới trái, trên phải là x_1[i],y_1[i],x_2[i],y_2[i] (i=1..n). Hãy tính diện tích của hình được phủ bởi N hình trên.

Chương 7. KỸ THUẬT LẬP TRÌNH ĐỆ QUY

7.1. HÀM ĐỆ QUI

Trong thực tế ta thường gặp những đối tượng bao gồm chính nó hoặc được định nghĩa dưới dạng của chính nó. Ta nói các đối tượng đó được xác định một cách đệ qui.

Một hàm được gọi là có tính đệ quy nếu trong bản thân hàm đó có lệnh gọi lại chính nó một cách trực tiếp hay gián tiếp. Một chương trình được gọi là có đệ qui nếu nó có chứa ít nhất một hàm đệ qui (gọi tắt là chương trình đệ qui).

Một hàm đệ qui (recursive function) gồm bước cơ sở và bước đệ qui.

Bước cơ sở (hay còn gọi là bước dừng): Mô tả cấp độ giải được của bài toán.

Bước đệ quy: Bước gọi lại chính nó nhưng với cấp độ thấp hơn.

Khi một hàm gọi đệ qui đến chính nó, thì ở mỗi lần gọi tới, chương trình sẽ tạo ra một tập các biến cục bộ hoàn toàn độc lập với các tập biến cục bộ đã được tạo ra trong các lần gọi trước đó, có bao nhiêu lần gọi tới hàm đệ quy thì cũng có bấy nhiêu lần thoát ra khỏi hàm, cứ mỗi lần thoát ra khỏi hàm thì một tập các biến cục bộ sẽ được giải phóng, sự tương ứng giữa các lần gọi tới hàm và thoát ra khỏi hàm được thực hiện theo thứ tự ngược, nghĩa là lần ra đầu tiên ứng với lần vào cuối cùng và lần ra khỏi hàm cuối cùng ứng với lần đầu tiên gọi tới hàm (*cơ chế vào sau ra trước*).

7.2. PHÂN LOẠI ĐỆ QUY

Các chương trình đệ quy thường gặp có thể thuộc vào bốn loại sau: *Đệ quy tuyến tính, đệ quy nhị phân, đệ quy hỗ tương và đệ quy phi tuyến* (việc phân loại này chỉ mang tính hình thức).

Trong mục này, chúng ta sẽ lần lượt xem xét các loại đệ quy kể trên, ở mỗi loại chúng ta minh họa bằng một số ví dụ đơn giản điển hình.

7.2.1. Đệ quy tuyến tính

Một hàm được gọi là đệ quy tuyến tính (đệ qui đơn) nếu một lần gọi hàm nó chỉ phát sinh tối đa một lời gọi đệ quy.

Ví dụ 7.1.

Giai thừa được định nghĩa theo kiểu quy nạp như sau: n! = n(n-1)!*

Viết chương trình tính n!, với n là một số nguyên không âm.

```
1. #include <iostream.h>
2. long giaithua(int n);
3. int main()
4. {
5.  cout<<giaithua(5);
6. }
7. long giaithua(int n)
8. {
9.  if (n==0|| n==1)
10.       return 1;
11.  else
12.       return n*giaithua(n-1);
13. }
```

7.2.2. Đệ quy nhị phân

Một hàm được gọi là đệ quy nhị phân nếu mỗi lần gọi hàm nó phát sinh tối đa một số ít (thường là nhỏ hơn 4) lời gọi đệ quy.

Ví dụ 7.2.

Viết chương trình tính số hạng thứ n của dãy fibonacci f_n xác định theo công thức đệ quy sau:

$f_1=1$,

$f_2=1$,

$f_n= f_{n-1} + f_{n-2}$, $n \geq 3$.

```
1. #include <iostream.h>
2. int  f(int n);
3. int main()
4. {
5.  cout<<f(7);
6. }
7. int  f(int n)
8. {
9.  if (n==1 || n==2)
10.       return 1;
11.  else
12.       return f(n-1)+f(n-2);
13. }
```

Ví dụ 7.3.

Cho một dãy số được định nghĩa theo công thức quy nạp sau (với n là số nguyên ≥1)

$f(1) = 1; f(2) = 2; f(3) = 3.$

$f(n+3) = 2f(n+2) + f(n+1) - 3f(n)$

Viết chương trình tính f(n).

```
1. #include <iostream.h>
2. long  f(int n);
3. int main()
4. {
5.  int n;
6.  cin>>n ;
7.  cout<<f(n);
8. }
```

```
9. long  f(int n)
10. {
11.  if (n==1||n==2||n==3) return n;
12.  return 2*f(n-1) + f(n-2)-3*f(n-3);
13. }
```

Một số bộ test:

Test	Test 1	Test 2	Test 3
n	8	20	30
f(*n*)	13	-7546	1338706

7.2.3. Đệ quy hỗ tương

Hai hàm P,Q được gọi là đệ quy hỗ tương nếu hàm P có lời gọi đến hàm Q và ngược lại một cách trực tiếp hay gián tiếp.

Ví dụ 7-4.

Viết chương trình tính số hạng thứ n của hai dãy sau:

$x_0 = 1, y_0 = 0,$

$x_n = x_{n-1} + y_{n-1}$ *với mọi* $n > 0,$

$y_n = 3*x_{n-1} + 2*y_{n-1}$ *với mọi* $n > 0.$

```
1. #include<iostream.h>
2. int tinhxn(int n);
3. int tinhyn(int n);
4. int main()
5. {
6.  cout<<tinhxn(3)<<endl;
7.  cout<<tinhyn(3);
8. }
```

```
9. int tinhxn(int n)
10. {
11.  if (n==0) return 1;
12.  return tinhxn(n-1)+tinhyn(n-1);
13. }
14. int tinhyn(int n)
15. {
16.  if (n==0) return 0;
17.  return 3*tinhxn(n-1)+2*tinhyn(n-1);
18. }
```

7.2.4. Đệ quy phi tuyến

Một hàm được gọi là đệ quy phi tuyến (đệ qui phức) nếu mỗi lần gọi hàm thì nó phát sinh ra khoảng n lần gọi đệ quy. Thông thường với loại này; lời gọi đệ quy được đặt trong một hoặc nhiều vòng lặp.

Ví dụ 7-5.

Dãy A_n được cho như sau :

$A_1=1$

$A_n=n(A_1+A_2+...+A_{n-1})$

Viết hàm tính A_n có sử dụng đệ quy

```
1. int An(int n)
2. {
3.  if (n==1) return 1 ;
4.  int s=0 ;
5.  for (int i=1;i<n ;i++)
6.   s=s+An(i) ;
7.  return s*n ;
8. }
```

7.3. KHỬ ĐỆ QUY

Mặc dù nhiều bài toán có thể giải bằng kỹ thuật đệ quy, thế nhưng chúng ta không nên lạm dụng đệ quy và nếu một bài toán không quá khó để tìm được một lời giải không đệ quy thì nên chọn cách giải không đệ quy. Lời giải không đệ quy được gọi là khử đệ quy.

Trong mục này giáo trình trình bày cách viết lại các hàm đệ qui trên bằng cách không sử dụng hàm đệ qui; việc khử đệ qui bằng ngăn xếp (stack) sẽ nói đến trong các môn học tiếp theo.

Ví dụ 7-6.

Viết chương trình tính n! bằng cách không dùng đệ quy.

```
1. #include <iostream.h>
2. long giaithua(int n);
3. int main()
4. {
5.  cout<<giaithua(10);
6. }
7. long giaithua(int n)
8. {
9.  long temp=1;
10.  for (int i=1;i<=n;i++)
11.        temp *=i;
12.  return temp;
13. }
```

Ví dụ 7.7.

Viết chương trình tính số hạng thứ n của dãy fibonacci f_n được xác định theo công thức đệ quy:

$f_1=1$,

$f_2=1$,

$f_n= f_{n-1} + f_{n-2}$, $n \geq 3$.

```
1. #include <iostream.h>
2. int f(int n);
3. int main()
4. {
5.  int n;
6.  cout<<f(n);
7. }
8. int f(int n)
9. {
10.  int f1=1,f2=1,f=1;
11.  for (int i=3;i<=n;i++)
12.  {
13.        f=f2+f1;
14.        f1=f2;
15.        f2=f;
16. }
17. return f;
18. }
```

7.4. THUẬT TOÁN ĐỆ QUY

Thuật toán đệ qui là thuật toán gọi đến chính nó với đầu vào kích thước nhỏ hơn.

Việc phát triển thuật toán đệ qui là thuận tiện khi cần xử lý với các đối tượng được định nghĩa đệ qui (cây có gốc, tập hợp, biểu thức quy nạp,…).

Các thuật toán chia để trị và quay lui được trình bày các trong phần tiếp theo có một vai trò đặc biệt trong lĩnh vực thiết kế thuật toán; cả hai thuật toán này **thông thường** được mô tả dưới dạng các thuật toán đệ qui.

BÀI TẬP

Viết chương trình hoàn chỉnh cho các bài toán sau đây

BT7-1.

a. *Tìm ước số chung lớn nhất của 2 số tự nhiên a,b bằng hai cách sử dụng đệ quy và không sử dụng đệ quy.*

b. *Chuyển đổi một số n trong hệ đếm thập phân thành số trong hệ đếm cơ số b bằng hai cách sử dụng đệ quy và không sử dụng đệ quy.*

c. *Cho x là số thực và n là số nguyên dương, tính tính x^n bằng 2 cách sử dụng sử dụng đệ quy và không sử dụng đệ quy.*

d. *Cho* $s(n) = \sqrt{2 + \sqrt{2 + \ldots + \sqrt{2 + \sqrt{2}}}}$ *; biểu thức này có n dấu căn. Hãy viết hàm tính s(n) bằng hai cách đệ quy và không đệ quy.*

BT7-2.

a. *Với n ≥ 1, n là số nguyên dương, biết rằng f(n) được tính theo công thức đệ quy sau đây:*

f(1) =1,

f(2n) = 2f(n),

f(2n+1)= 2f(n) + 3f(n+1)

Viết hàm tính f(n) bằng hai cách đệ qui và không đệ qui.

b. *Với mỗi n ≥ 1, số Y_n được tính như sau :*

Y_1=1, Y_2=2, Y_3=3,

$Y_n=Y_{n-1} + 2Y_{n-2} + 3Y_{n-3}$ nếu n ≥ 4

Viết hàm tính Y_n bằng hai cách đệ qui và không đệ qui.

c. *Với mỗi số nguyên n ≥ 1, số X_n được tính như sau :*

X_1=1,

X_2=1,

$X_n=X_{n-1} + (n-1)X_{n-2}$ với n ≥3

Viết hàm tính X_n bằng hai cách đệ qui và không đệ qui.

BT7-3.

a. *Cho dãy số x_n được định nghĩa như sau:*

$x_0 = 1$,

$x_1 = 2$,

$x_n = nx_0 + (n - 1)x_1 + \ldots + x_{n-1}$

Viết hàm đệ quy tính x_n (với $n \geq 0$) bằng hai cách đệ qui và không đệ qui.

b. *Dãy A_n được cho như sau :*

$A_1=1$,

$A_{2n}=n + A_n + 2$,

$A_{2n+1}=n^2 + A_n.A_{n+1} +1$

Viết hàm tính A_n bằng đệ quy và không sử dụng đệ quy.

BT7-4.

Sử dụng thuật toán chia để trị giải các bài toán sau.

a. *Tìm kiếm nhị phân,*

b. *Quick sort,*

c. *Tính các hệ số của nhị thức $(a+b)^n$,*

d. *Cho dãy n số nguyên, hãy chuyển k phần tử đầu dãy về cuối dãy.*

BT7-5.

Bài toán tháp Hà Nội.

Giả sử có 3 cọc A, B, C. Ban đầu tại A đặt một số đĩa với thứ tự đĩa trên nhỏ đĩa dưới to.

Yêu cầu của bài toán là chuyển toàn bộ số đĩa trên sang cọc C, trong quá trình chuyển được phép sử dụng cọc C, mỗi lần chuyển đúng 01 đĩa và luôn bảo đảm nguyên tắc đĩa nhỏ nằm trên đĩa lớn trong suốt quá trình chuyển, đồng thời số lần di chuyển đĩa là ít nhất.

Chương 8. KỸ THUẬT LẬP TRÌNH CON TRỎ

8.1. ĐỊA CHỈ VÀ CON TRỎ

8.1.1. Địa chỉ ô nhớ

Liên quan đến một biến ta có các khái niệm: tên biến, kiểu của biến, giá trị của biến. Ví dụ với khai báo:

double *x*; // *x* là biến kiểu double

x=12.5; //*x* có giá trị là 12.5

và lúc này, biến *x* chiếm một số ô nhớ liên tiếp trong bộ nhớ, mỗi ô nhớ có kích thước 1 byte, địa chỉ của ô nhớ được biểu thị ở hệ đếm 16.

Lưu ý:-Một biến kiểu int chiếm 4 byte, một biến kiểu double chiếm 8 byte, một biến kiểu char chiếm 1 byte,...

-Địa chỉ của hai biến kiểu int liên tiếp cách nhau 4 byte, địa chỉ của hai biến kiểu double liên tiếp cách nhau 8 byte,… (tham số này là theo CFREE5-2010).

8.1.2. Con trỏ

Con trỏ là một kiểu dữ liệu đặc biệt dùng để quản lý địa chỉ của các ô nhớ. Một con trỏ quản lý các địa chỉ mà dữ liệu tại các địa chỉ này có kiểu *T* thì con trỏ đó được gọi là con trỏ kiểu *T*. Con trỏ kiểu *T* chỉ được dùng để chứa địa chỉ của biến kiểu *T* (nghĩa là con trỏ kiểu int chỉ được dùng để chứa địa chỉ của biến kiểu int, con trỏ kiểu double chỉ được dùng để chứa địa chỉ của các biến kiểu double).

8.1.3. Khai báo con trỏ

[type] *< pointer_var_name >

Chẳng hạn với khai báo

int x, *px; // *x* là biến int, còn *px* là con trỏ kiểu int.

8.1.4. Phép lấy địa chỉ của một biến

Cú pháp: &< var_name >

Ví dụ &*a* là phép lấy địa chỉ của các biến a, chẳng hạn với khai báo

int *a*;

Muốn lấy địa chỉ của biến *a* ta viết là:

cout<<&*a*;

Lưu ý:-Khi một con trỏ chưa trỏ tới bất kỳ một địa chỉ nào, thì con trỏ đó được gọi là con trỏ NULL.

-Con trỏ NULL có địa chỉ là 0000.

8.1.5. Phép toán lấy giá trị tại một địa chỉ mà một con trỏ đang trỏ tới

Cú pháp: *< pointer_var_name >

chẳng hạn với khai báo

int a=10; // biến a có giá trị là 10

int px=&a; // px là con trỏ trỏ đến địa chỉ của biến a

thì lúc này nội dung tại địa chỉ mà con trỏ *px* trỏ đến (tức là giá trị của biến *a*) là 10.

Cần chú ý rằng việc thay đổi giá trị của **px* sẽ làm cho giá trị của *a* thay đổi theo và ngược lại. Chẳng hạn với khai báo:

int a=10;

int *px=&a;

*px=*px+2

sẽ làm cho biến *a* cũng có giá trị là 12.

8.2. QUY TẮC SỬ DỤNG CON TRỎ TRONG CÁC BIỂU THỨC

8.2.1. Sử dụng tên con trỏ

Do con trỏ cũng là một biến, nên khi tên của nó xuất hiện trong một biểu thức thì giá trị của nó sẽ được sử dụng trong biểu thức này.

Lưu ý: -Giá trị của một con trỏ là địa chỉ của một biến nào đó. Trong phép gán mà con trỏ đứng bên trái thì giá trị bên phải phải là một địa chỉ.

-Nội dung của một con trỏ cũng có thể được thay đổi.

8.2.2. Sử dụng dạng khai báo của con trỏ

Nếu px là con trỏ trỏ tới biến *x* thì các cách viết *x* và **px* là tương đương nhau.

Khi biết được địa chỉ của một biến thì chẳng những chúng ta có thể sử dụng giá trị của nó mà còn có thể gán cho nó một giá trị mới (làm thay đổi nội dung của nó); điều này sẽ được áp dụng khi cần để nhận biết kết quả của hàm thông qua các tham biến.

8.3. CÁC THAO TÁC TRÊN CON TRỎ

8.3.1. Phép toán dịch chuyển địa chỉ của con trỏ

Cú pháp : < pointer_var_name > + số kiểu int //phép cộng

< pointer_var_name > - số kiểu int //phép trừ

```
int x,*px;
px=&x;
cout<<px<<endl;     //giả sử địa chỉ của px là ff34
px=px+2;
cout<<px<<endl;     //địa chỉ của px sẽ là ff3c
px=px-1;
cout<<px<<endl;     //địa chỉ của px sẽ là ff38
```

8.3.2. Cấp phát động bộ nhớ

+Toán tử new

Dạng 1: <pointer_var_name>=new <type>;

Cấp phát sizeof (<type>) bytes nhớ trong khối nhớ heap: Nếu cấp phát thành công, hàm new trả về địa chỉ đầu của khối nhớ cấp phát được cho con trỏ pointer_var_name; nếu khối nhớ heap không đủ để cấp phát thì hàm trả sẽ về con trỏ NULL.

Dạng 2: <pointer_var_name>=new <type>[N_blocks];

Tương tự như trên, câu lệnh dạng này cấp phát N_blocks*sizeof (<type>) bytes nhớ. Chẳng hạn với khai báo:

char *pc;

pc=new char[10];//cấp phát 10 khối nhớ 1 byte và cho con trỏ pc trỏ tới địa chỉ đầu tiên của khối nhớ này.

Lưu ý: Cần kiểm tra xem việc cấp phát có thành công hay không. Nếu cấp phát thành công, con trỏ mới bắt đầu trỏ tới địa chỉ đầu của khối nhớ được cấp phát và lúc này chúng ta mới có thể thực hiện các thao tác trên con trỏ này. Việc kiểm tra này có thể được thực hiện như sau:

```
int *px;
px= new int;
if (px==NULL)
{
        cout<<"khong du bo nho";
        exit(0);
}
```

+Từ khóa sizeof

Cú pháp: sizeof(<type>) hoặc

sizeof(<expression>)

Cho biết kích thước lưu trữ các giá trị có kiểu dữ liệu <type> hoặc của kiểu dữ liệu của giá trị biểu thức expression. Chẳng hạn với đoạn lệnh sau:

```
cout<<sizeof(char);
cout<<sizeof(int);
cout<<sizeof(float);
cout<<sizeof(double);
cout<<sizeof(long double);
        thì kết quả sẽ là 1 4 4 8 12                    // Tham số theo CFREE 5.0
```

hoặc với đoạn lệnh

```
cout<<sizeof('c');
cout<<sizeof(2003);
cout<<sizeof(54321);
cout<<sizeof(200.3);
```

thì kết quả sẽ là 1 4 4 8 // Tham số theo CFREE 5.0

8.3.3. Giải phóng khối nhớ đã được cấp phát

Hàm delete

void delete <pointer_var_name>

Giải phóng khối nhớ được cấp phát bằng toán tử new mà khối nhớ này đang được quản lý bởi con trỏ <pointer_var_name >.

8.4. CON TRỎ VỚI MẢNG MỘT CHIỀU

Trong ngôn ngữ *C*/*C*++, tên của một mảng là một hằng con trỏ trỏ tới địa chỉ của phần tử đầu tiên của mảng. Chúng ta có thể truy cập tới các phần tử của mảng thông qua địa chỉ của nó bằng cách xuất phát từ địa chỉ của phần tử đầu tiên. Việc sử dụng con trỏ trong mảng làm việc quản lý bộ nhớ được linh động và hiệu quả hơn.

Nếu a là mảng một chiều thì trong mọi ngữ cảnh, các cách viết sau đây là tương đương:

```
a, &a[0];
a+i, &a[i]
*(a+i), a[i];
```

Ví dụ 8-1.

Cho dãy n số nguyên. Hãy thực hiện các công việc sau:

a. ***T****ìm giá trị lớn nhất của dãy.*

b. *Sắp xếp các phần tử của dãy theo thứ tự tăng dần.*

```
1. #include <alloc.h>
2. #include <iostream.h>
3. void nhap(int *&a, int &n);
4. void xuat(int *a, int n);
5. int giatrilonnhat(int *a, int n);
6. void swap(int *a, int *b);
7. void sapxep(int *a, int n);
8. int main()
9. {
10.  int *a,n;
11.  nhap(a,n);
12.  cout<<giatrilonnhat(a,n)<<endl;
13.  sapxep(a,n);
14.  xuat(a,n);
15.  delete a;
16. }
17. void nhap(int *&a, int &n)
18. {
19.  cout<<"Nhap so phan tu cua mang :";cin>>n;
20.  a=new int [n];
21.  for (int i=0;i<n;i++)
22.       cin>>*(a+i);
23. }
24. void xuat(int *a, int n)
25. {
26.  for (int i=0;i<n;i++)
27.       cout<<*(a+i)<<" ";
```

```
28. }
29. int giatrilonnhat(int *a, int n)
30. {
31.  int max=*a;
32.  for (int i=1;i<n;i++)
33.       if (*(a+i)>max)
34.            max=*(a+i);
35.  return max;
36. }
37. void swap(int *a, int *b)
38. {
39.  int t=*a;
40.  *a=*b;
41.  *b=t;
42. }
43. void sapxep(int *a, int n)
44. {
45.  for (int i=0;i<n-1;i++)
46.  for (int j=i+1;j<n;j++)
47.       if (*(a+i)>*(a+j))
48.            swap(a+i,a+j);
49. }
```

Ví dụ 8-2.

Cho hai dãy số nguyên a và b; trong đó dãy a có n số, dãy b có m số. Nối dãy b vào sau dãy a tạo thành dãy c; xuất dãy c lên màn hình.

```
1. #include <alloc.h>
2. #include <iostream.h>
3. void nhap(int *&a, int &n);
4. void noimang1(int *a, int *b, int *&c,int n, int m);
5. int *noimang2(int *a, int *b, int *c,int n, int m);
6. void xuat(int *a, int n);
7. int main()
8. {
9.  int *a,*b,*c,n,m;
10.  nhap(a,n);
11.  nhap(b,m);
12.  noimang1(a,b,c,n,m);
13.  int *d=noimang2(a,b,c,n,m);
14.  xuat(c,n+m);
15.  xuat(d,n+m);
16.  delete a;
17.  delete b;
18.  delete c;
19.  delete d;
20. }
21. void nhap(int *&a, int &n)
22. {
23.  cout<<"Nhap so phan tu cua mang :";cin>>n;
24.  a=new int [n];
```

```
25.  for (int i=0;i<n;i++)
26.        cin>>*(a+i);
27. }
28. void xuat(int *a, int n)
29. {
30.  for (int i=0;i<n;i++)
31.        cout<<*(a+i)<<" ";
32.  cout<<endl;
33. }
//Cách 1: Trả về biến con trỏ c
34. void noimang1(int *a, int *b, int *&c,int n, int m)
35. {c=new int [n+m];
36.  int t=0;
37.  for (int i=0;i<n;i++)          c[t++]=a[i];
38.  for (int i=0;i<m;i++)          c[t++]=b[i];
39. }
//Cách 2: Trả về hàm kiểu con trỏ
40. int *noimang2(int *a, int *b, int *c,int n, int m)
41. {
42.  c=new int [n+m];
43.  int t=0;
44.  for (int i=0;i<n;i++)
45.        c[t++]=a[i];
46.  for (int i=0;i<m;i++)
47.        c[t++]=b[i];
48.  return c;
49. }
```

8.5. CON TRỎ VỚI MẢNG HAI CHIỀU

Như chúng ta đã nói trong phần mảng một chiều với con trỏ. Tên của mảng là một hằng con trỏ trỏ đến phần tử đầu tiên của mảng, dựa vào phần tử đầu tiên này mà có thể truy cập đến các phần tử khác trong mảng.

Tuy nhiên cần chú ý với mảng một chiều

*a=1; *(a+1)=2; *(a+2)=3; *(a+3)=4;

thì phép truy xuất *a*[2] là hợp lý, tuy nhiên ta không thể truy xuất a[i][j], mà khi đó phần tử ở dòng *i* cột *j* của mảng *a* phải được truy xuất bởi *(*a*+*n***i*+*j*).

	cột 1	cột 1	cột 2	cột 3
dòng 0	2 a0	3 a1	4 a2	6 a3
dòng 1	3 a4	8 a5	4 a6	7 a7
dòng 2	3 a8	1 a9	2 a10	5 a11

Nhắc lại rằng *việc cấp phát vùng nhớ động để lưu trữ dữ liệu mảng làm tránh lãng phí bộ nhớ*. Chẳng hạn nếu khai báo tĩnh int a[20][50] thì máy đã cấp phát đúng 20*50 = 1000 khối nhớ 2 byte cho mảng a và nếu thực tế *m*, *n* là một số rất bé thì mảng *a* vẫn chiếm hết 1000 khối nhớ 2 byte này. Vấn đề này đã được khắc phục bằng cách sử dụng cấp phát động sử dụng biến con trỏ.

Ví dụ 8-3.

Viết chương trình nhập mảng hai chiều a có m dòng n cột, các phần tử là các số nguyên và một số nguyên x.

a. *Hãy đếm xem trong mảng có bao nhiêu số bằng x ?*

b. *Cho biết vị trí của các phần tử bằng x ?*

```
1. #include<iostream.h>
2. #include<alloc.h>
3. void    nhap(int *&a,int &m, int &n, int &x);
4. void    demsobangx(int *a, int m, int n, int x);
5. int     main()
6. {
7.  int *a,m,n,x;
8.  nhap(a,m,n);
9.  demsobangx(a,m,n,x);
10.  delete a;
11. }
12. void  nhap(int *&a,int &m, int &n, int &x);
13. {
14.  cout<<"Nhap vao so dong:";    cin>>m;
15.  cout<<"Nhap vao so cot:";     cin>>n;
16.  a=new int [m*n];
17.  for (int i=0;i<m;i++ )
18.  for (int j=0;j<n;j++ )
19.       cin>>*(a+n*i+j);
20.  cout<<"Nhap vao gia tri x:";  cin>>x;
21. }
22. void demsobangx(int *a, int m, int n, int x)
23. {
24.  int d=0;
25.  for (int i=0;i<m;i++ )
26.  for (int j=0;j<n;j++ )
27.       if (x==*(a+n*i+j))
```

```
28.       {
29.            d++;
30.            cout<<i<<" "<<j<<endl;
31.       }
32.  cout<<"So lan xuat hien cua x la : "<<d;
33. }
```

(Nội dung con trỏ với chuỗi sẽ được trình bày trong chương tiếp theo của giáo trình).

8.6. TỔ CHỨC DỮ LIỆU DẠNG DANH SÁCH

Trong các mục trên, các khối nhớ được cấp phát liên tiếp nhau trong bộ nhớ; và với cách tổ chức dữ liệu như mảng 1 chiều hoặc mảng 2 chiều thì mối liên kết giữa các phần tử là một đặc trưng quan trọng của cấu trúc dữ liệu mảng; nó thuận tiện cho nhiều bài toán trong thực tế; nhưng cũng bất lợi trong việc xử lý nhiều bài toán khác; chẳng hạn như các bài toán chèn/xóa phần tử trong dãy,…

Mục này trình bày cách tổ chức dữ liệu linh hoạt hơn, đó là tổ chức dữ liệu dạng danh sách; mỗi phần tử của danh sách được trỏ bởi một con trỏ; các khối nhớ được cấp cho một danh sách không nhất thiết phải liên tục nhau. Một phần tử của danh sách có hai thành phần: Phần thứ nhất lưu trữ các thông tin về bản thân phần tử, phần thứ hai lưu trữ địa chỉ của phần tử kế tiếp trong danh sách, hoặc lưu trữ giá trị NULL nếu là phần tử cuối danh sách.

Ví dụ 8-4.

Nhập vào một danh sách các số nguyên, xuất các số nguyên đó lên màn hình.

```
1. #include     <iostream.h>
2. struct  node
3.  {
4.   int  info;
5.   struct    node *next;
6.  };
```

```
7. struct  list
8.  {
9.   node *head,*tail;
10.  };
11. void  themnut(list &l,node *p);
12. void  nhapdanhsach(list &l);
13. void  xuatdanhsach(list l);
14. list l;
15. int main()
16. {
17.  nhapdanhsach(l);
18.  xuatdanhsach(l);
19. }
20. void  themnut(list &l,node *p)
21. {
22.  if (l.head==NULL)
23.       {
24.       l.tail=p;
25.       l.head=p;
26.       }
27.  else
28.  {
29.       l.tail->next=p;
30.       l.tail=p;
31.  }
32. }
33. void nhapdanhsach(list &l)
```

```
34. {
35.  int x,n;
36.  cout<<"Danh sach co bao nhieu phan tu : ";cin>>n;
37.  for (int i=1;i<=n;i++)
38.  {    cin>>x;
39.       node *p=new node;p->info=x;   p->next=NULL;
40.       themnut(l,p);
41.  }
42. }
43. void xuatdanhsach(list l)
44. {
45.  cout<<endl;
46.  node *p = l.head;
47.  while (p!=NULL)
48.  {    cout<<p->info<<" ";
49.       p=p->next;
50.  }
51. }
```

BÀI TẬP

Giải các bài toán sau bằng cách sử dụng kiến thức về con trỏ

BT8-1.

Nhập vào n số nguyên dương. Đếm số lượng số chẵn, số lượng số lẻ.

BT8-2.

*Nhập vào hai ma trận A và B đều có n dòng n cột, các phần tử là các số nguyên. Tính A+B, A-B, A*B.*

BT8-3.

Cho dãy n phần tử các số nguyên. Hãy xóa phần tử ở vị trí thứ k của dãy.

BT8-4.

Cho dãy n phần tử các số nguyên. Hãy chèn phần tử x vào vị trí thứ k của dãy.

BT8-5.

Cho bảng số m dòng, n cột chứa các số nguyên. Hãy xóa dòng thứ k của bảng.

BT8-6.

Cho bảng số m dòng, n cột chứa các số nguyên. Hãy chèn n phần tử vào dòng thứ k của bảng (các dòng từ dòng thứ k của bảng bị đẩy xuống dưới một vị trí).

BT8-7.

Cho hai dãy a,b chứa các số nguyên đã được sắp tăng; dãy a có n phần tử, dãy b có m phần tử. Hãy sắp xếp hai dãy này thành một dãy tăng.

BT8-8.

Cho dãy n phần tử các số nguyên dương. Hãy xóa các số nguyên tố khỏi dãy.

BT8-9.

Cho hai dãy A,B chứa các số nguyên đôi một khác nhau (mỗi dãy xem là một tập hợp). Trong đó dãy A có n phần tử, dãy B có m phần tử.

a. *Tìm A ∩ B*

b. *Tìm A ∪ B*

c. *Tìm A – B.*

BT8-10.

Cho một bảng có m dòng, n cột chứa các số nguyên dương. Hãy xóa các số nguyên tố trong bảng. Giả sử khi xóa một số tại (dòng i, cột j) thì dịch chuyển số (dòng i, cột j+1) vào vị trí (dòng i, cột j); nếu j là cột cuối của bảng thì dịch chuyển số đầu tiên ở dòng tiếp theo vào vị trí (dòng i, cột j). Xuất bảng số còn lại lên màn hình (khi đó một số dòng cuối của bảng có thể không đủ n số).

BT8-11.

Dùng danh sách liên kết đơn để biểu diễn một đa thức $P(a_i,n,x)$*. Viết chương trình thực hiện các công việc sau:*

a. *Nhập một đa thức,*

b. *Xuất một đa thức,*

c. *Tính giá trị của một đa thức tại điểm* x_o*,*

d. *Cộng hai đa thức, xuất đa thức kết quả lên màn hình,*

e. *Nhân hai đa thức, xuất đa thức kết quả lên màn hình.*

Chương 9. KỸ THUẬT LẬP TRÌNH CHUỖI

9.1. KÝ TỰ VÀ CHUỖI KÝ TỰ

Trong ngôn ngữ C/C++, ký tự được đặt giữa hai dấu nháy đơn. Chuỗi ký tự là một dãy các ký tự liên tiếp kết thúc bằng ký tự NULL('\0'), trên màn hình sẽ không nhìn thấy ký tự kết thúc này còn trong bộ nhớ thì nó sẽ chiếm một byte. Chuỗi ký tự được đặt trong dấu nháy kép.

Chẳng hạn để khai báo biến ký tự *ch* và gán ký tự *A* cho *ch* ta làm như sau:

```
char    ch='A';
```

Để khai báo biến chuỗi ký tự *st* ta có thể làm như sau:

```
char*  st="Thanh Pho Ho Chi Minh";
```

hoặc

```
char    st[]="Thanh Pho Ho Chi Minh";
```

Lưu ý:-Với chuỗi *s*, cú pháp *s* + *k* sẽ trả về chuỗi tính từ ký tự thứ *k* trở về cuối của chuỗi *s*.

-Với chuỗi *s*, cú pháp *s*[*k*]='\0' trả về chuỗi *s* trong đó *s* được kết thúc ở vị trí *k*.

Trong phần phụ lục cuối giáo trình này, chúng tôi trình bày cú pháp và ý nghĩa của một số hàm quan trọng của ngôn ngữ *C*/*C*++ để bạn đọc thuận tiện theo dõi.

9.2. NHẬP/XUẤT KÝ TỰ

Hàm getchar()

Dùng để đọc một ký tự từ bàn phím và trả về ký tự đó. Hàm này luôn chờ cho đến khi gặp ký tự xuống dòng mới trả về, vì vậy khi sử dụng luôn phải gõ thêm phím ENTER và sẽ làm phát sinh thêm một ký tự nữa vào bộ nhớ đệm bàn phím. Khi gọi getchar() lần nữa thì sẽ lấy ký tự trong vùng đệm này.

Hàm getch()

Có cùng chức năng như hàm getchar(), nhưng getch() nhập ký tự mà không hiện ra màn hình, trong khi getchar() nhập ký tự và ký tự đó có hiện ra màn hình.

Hàm putchar()

Dùng để xuất một ký tự ra màn hình.

9.3. NHẬP/XUẤT CHUỖI KÝ TỰ

Hàm gets(*s*)

Cho nhập từ bàn phím một chuỗi ký tự cho đến khi gặp ký tự xuống dòng thì đưa chuỗi này vào biến *s* và thay ký tự xuống dòng bằng ký tự NULL (‘\0’). Lưu ý lệnh cin>> không được sử dụng để nhập chuỗi.

Chuỗi *s* có chiều dài *n* thì *n* ký tự này được truy xuất như sau: *s*[0], *s*[1],…, *s*[*n*-1].

Hàm puts(*s*)

Xuất chuỗi ký tự *s* ra màn hình.

Ví dụ 9.1.

Nhập vào hai chuỗi và sau đó xuất hai chuỗi vừa nhập lên màn hình.

```
1. #include<iostream.h>
2. #include<string.h>
3. #include<alloc.h>
4. int main()
5. {
6.  char *s1;
7.  s1=new char[1000];
8.  gets(s1);
9.  char *s2;
10.  s2=new char[1000];
```

```
11.  gets(s2);
12.  cout<<s1<<endl; // có thể sử dụng puts(s1);
13.  cout<<s2<<endl; // có thể sử dụng puts(s2);
14.  delete s1;
15.  delete s2;
16. }
```

Ví dụ 9.2.

Nhập từ bàn phím một chuỗi s; giả thiết thêm rằng chuỗi không có khoảng trắng dư thừa ở đầu và cuối chuỗi và giữa các từ có duy nhất một khoảng trắng. Hãy viết các hàm thực hiện các công việc sau (các công việc là độc lập với nhau).

a. *Tách một từ bên phải (bên trái) của chuỗi s ?*

b. *Đưa các ký tự đầu của mỗi từ thành chữ hoa, còn các ký tự khác thành chữ thường.*

c. *Đếm xem chuỗi s có bao nhiêu từ ?*

d. *Đếm xem chuỗi s có bao nhiêu từ bắt đầu bằng ký tự nguyên âm ? (các ký tự a, u, i, o, e là các nguyên âm).*

e. *Đếm số ký tự của mỗi từ của chuỗi s.*

f. *Sắp xếp các ký tự của chuỗi s theo chiều tăng (theo mã ASCII); trong đó các ký tự khoảng trắng giữ nguyên vị trí.*

g. *Cho biết tần số xuất hiện của các chữ cái trong chuỗi s (không kể ký tự trống).*

h. *Đếm số lần xuất hiện chuỗi y trong chuỗi s.*

```
1. #include <iostream.h>
2. #include <string.h>
3. #include <ctype.h>
4. char *tachtutrai(char *s); // câu a
5. char *tachtuphai(char *s); // câu a
6. char *chuanhoatu(char *s); // câu b
7. int demsotu(char *s); // câu c
```

```
8.int demtubatdaunguyenam(char s[])// câu d
9. void demkytucuamoitu(char *s); // câu e
10.  void sapxepkytutang(char *s); // câu f
11.  void tansocackytu( char *s)// câu g
12.  int main()
13.  {
14.       char *s;
15.       s="thanH pHO Ho Chi Minh";
16.       cout<<chuanhoatu(s);
17.  }
18.  char *tachtutrai(char *s)
19.  {
20.       return strrev(strrchr(strrev(s),' ')+1);
21.  }
22.  char *tachtuphai(char *s)
23.  {
24.       return strrchr(s,' ')+1;
25.  }
26.  char *chuanhoatu(char *s)
27.  {
28.       strlwr(s);
29.       for (int i=0;i<strlen(s);i++)
30.            if (s[i]==' ' && s[i+1]!=' ' )
31.                 s[i+1]=toupper(s[i+1]);
32.       if (s[0]!=' ')
33.            s[0]=toupper(s[0]);
34.       return s;
```

```
35.  }
36.  int demsotu(char *s)
37.  {
38.       int l=strlen(s),d=1;
39.       for (int i=0;i<l;i++)
40.            if (s[i]==' ' && s[i+1]!=' ')
41.                 d++;
42.       return d;
43.  }
44.  int demtubatdaunguyenam(char s[])
45.  {
46.       int l=strlen(s),d=0;
47.       for (int i=0;i<l;i++)
48.            if (s[i]==' ' && (s[i+1]=='a' ||s[i+1]=='u' ||
   s[i+1]=='i' ||
49.                 s[i+1]=='o' || s[i+1]=='e'))
50.                      d++;
// Kiểm tra ký tự đầu tiên
51.       if (s[0]=='a' ||s[0]=='u' || s[0]=='i' || s[0]=='o' ||
   s[0]=='e')
52.            d++;
53.       return d;
54.  }
55.  void demkytucuamoitu(char *s)
56.  {
57.       int sokytumoitu=0;
58.       for (int i=0;i<strlen(s);i++)
```

```
59.       if (s[i]!=' ') sokytumoitu++;
60.       else
61.       {
62.            cout<<sokytumoitu<<" ";
63.            sokytumoitu=0;
64.       }
65.            cout<<sokytumoitu;
66.  }
67.  void sapxepkytutang(char *s)
68.  {
69.       for(int i=0;i<strlen(s)-1;i++)
70.       for(int j=i+1;j<strlen(s);j++)
71.            if(*(s+i)>*(s+j) && *(s+i)!=' ' &&*(s+j)!=' ' )
72.            {
73.                 char temp=*(s+i);
74.                 *(s+i)=*(s+j);
75.                 *(s+j)=temp;
76.            }
77.  }
78.  void tansocackytu( char *s)
79.  {
80.       int l=strlen(s);
81.       int d[255];
82.       for (int i=0;i<l;i++)
83.            d[s[i]]=0;
84.       for (i=0;i<l;i++)
85.       d[s[i]]++;
```

```
86.  for (i=0;i<l;i++)
87.       if (d[s[i]]!=0)
88.       {
89.            cout<<s[i]<<"  xuat  hien  "<<d[s[i]]<<"  lan"
   <<endl;
90.            d[s[i]]=0;
91.       }
92.  }
93.  int demchuoicon(char *s, char *y)
94.  {
95.       int d=0;
96.       while (strstr(s,y)!=NULL)
97.            {
98.            d=d+1;
99.            strcpy(s,strstr(s,y)+1);
100.      }
101. return d;
102. }
```

Lưu ý: Vấn đề tách một từ bên trái chuỗi *s* sẽ được giải quyết nếu xác định được vị trí *cuối* của từ bên trái *cộng thêm 1*; để xác định được vị trí đó ta có thể sử dụng vòng lặp để tìm khoảng trắng đầu tiên tính từ bên trái; ta cũng có thể sử dụng cú pháp strchr(*s*,' ') – *s* để tìm vị trí *k* một từ bên trái.

Ví dụ 9.3.

Nhập từ bàn phím một chuỗi; giả thiết chuỗi không có khoảng trắng dư thừa ở đầu và cuối chuỗi và giữa các từ có duy nhất một khoảng trắng. Hãy viết các hàm thực hiện các công việc sau (các công việc là độc lập với nhau).

a. *Tìm từ có k ký tự, với k=1..7.*

b. *Tạo các chuỗi con được ghép từ từ đầu tiên và từ cuối cùng của mỗi chuỗi, giữa hai từ này có một khoảng trắng.*

c. *Đào ngược chuỗi. Ví dụ chuỗi nhập vào là "truong dai hoc sai gon" thì kết quả sẽ là "gon sai hoc dai truong".*

d. *Đảo ngược các ký tự trong mỗi từ. Ví dụ chuỗi nhập vào là "truong dai hoc sai gon" thì kết quả sẽ là "gnourt iad coh ias nog".*

e. *Sắp xếp các từ theo chiều tăng. Ví dụ chuỗi nhập vào là "truong dai hoc sai gon" thì kết quả sẽ là "dai gon hoc sai truong".*

f. *Tìm một từ dài nhất của chuỗi. Ví dụ chuỗi nhập vào là "truong dai hoc sai gon" thì kết quả sẽ là "truong".*

```
1. #include<iostream.h>
2. #include<string.h>
3. int demsotukkytu(char *s, int k);
4. void demtuloaik(char *s);
5. char *daonguocchuoi(char *s);
6. char *daonguoctu(char *s);
7. char *sapxep(char *s);
8. char *tudainhat(char *s);
9. int main()
10.  {
11.       char s[1000]="truong dai hoc sai gon";
12.       //demtuloaik(s);
13.       //cout<<daonguocchuoi(s);
14.       //cout<<daonguoctu(s);
15.       //cout<<sapxep(s);
16.       cout<<tudainhat(s);
17.  }
```

```
18.  int demsotukkytu(char *s, int k)
19.  {
20.       int demkytu=0,demtukkytu=0;
21.       for (int i=0;i<strlen(s);i++)
22.       if (s[i]!=' ') demkytu++;
23.       else
24.            {
25.            if (demkytu==k)
26.                 demtukkytu++;
27.            demkytu=0;
28.            }
29.       return demkytu==k?demtukkytu+1:demtukkytu;//??
30.  }
31.  void demtuloaik(char *s)
32.  {
33.       for (int i=1;i<=7;i++)
34.       cout<<demsotukkytu(s,i)<<" ";
35.  }
36.  char *gheptu(char *s)
37.  {
38.       char stemp[256];
39.       strcpy(stemp,s);
40.       int k=strchr(s,' ')-s;
41.       s[k]='\0';
42.       return strcat(s,strrchr(stemp,' '));
43.  }
44.  char *daonguocchuoi(char *s)
```

```
45.  {
46.       strcat(strrev(s)," ");
47.       int i=0;
48.       while (i<strlen(s))
49.       {
50.            int j=i;
51.            while (s[j]!=' ')j++;
52.            {
53.                 int u=i,v=j-1;
54.                 while (u<v) // dao doan u,v cua chuoi
55.                 {
56.                      char ch=s[u];
57.                      s[u]=s[v];
58.                      s[v]=ch;
59.                      u++;
60.                      v--;
61.                 }
62.            }
63.            i=j+1;
64.       }
65.       s[strlen(s)-1]='\0';
66.       return s;
67.  }
68.  char *daonguoctu(char *s)
69.  {
70.       strcat(s," ");
71.       int i=0;
```

```
72.       while (i<strlen(s))
73.       {
74.            int j=i;
75.            while (s[j]!=' ')j++;
76.            {
77.                 int u=i,v=j-1;
78.                 while (u<v) // dao doan u,v cua chuoi
79.                 {
80.                      char ch=s[u];
81.                      s[u]=s[v];
82.                      s[v]=ch;
83.                      u++;
84.                      v--;
85.                 }
86.            }
87.            i=j+1;
88.       }
89.       s[strlen(s)-1]='\0';
90.       return s;
91.  }
92.  char *tutrai(char *s)
93.  {
94.       int k=strchr(s,' ') - s;
95.       s[k]='\0';
96.       return s;
97.       //if (s[k-1]==10)   s[k-1]='\0';
98.  }
```

```
99.  int demtu(char *s)
100. {
101.      s=strcat(s," ");
102.      int d=0;
103.      for (int i=0;i<strlen(s);i++)
104.      if (s[i]==' ') d++;
105.      return d;
106. }
107. char *sapxep(char *s)
108. {
109.      char *stemp;
110.      stemp=new char[256];
111.      int d=0;
112.      int sotu=demtu(s);
113.      char *strsub[100];
114.      while (d<sotu)
115.      {
116.           strcpy(stemp,s);
117.           strsub[d]=tutrai(s);
118.           strcpy(s,stemp);
119.           s=s+strlen(tutrai(s))+1;
120.           d++;
121.      }
122.      for (int i=0;i<sotu-1;i++)
123.      for (int j=i+1;j<sotu;j++)
124.      if (strcmp(strsub[i],strsub[j])>0)
125.           {
```

```
126.                stemp=strsub[i];
127.                strsub[i]=strsub[j];
128.                strsub[j]=stemp;
129.           }
130.      strcpy(s,"");
131.      for (int i=0;i<sotu;i++)
132.      {
133.           strcat(s,strsub[i]);
134.           strcat(s," ");
135.      }
136.      s[strlen(s)-1]='\0';
137.      return s;// cung co the viet tra ve void neu ... char
   *&s..
138. }
139. char *tudainhat(char *s)
140. {
141.      strcat(s," ");
142.      char *skq;
143.      skq=new char [256];
144.      int i=0,max=0;
145.      while (i<strlen(s))
146.      {
147.           int j=i;
148.           while (s[j]!=' ')j++;
149.           if (j-i+1>max)
150.                {
151.                     max=j-i+1;
```

```
152.                        int u=0;
153.                        for (int k=i;k<=j;k++)
154.                        skq[u++]=s[k];
155.                    }
156.              i=j+1;
157.        }
158.        skq[max-1]='\0';
159.        return skq;
160. }
```

BÀI TẬP

Viết chương trình hoàn chỉnh cho các bài toán sau đây

BT9-1.

Cho chuỗi s gồm các từ; mỗi từ cách nhau ít nhất một khoảng trắng. Chuỗi chỉ gồm các chữ cái thường từ a đến z và ký tự khoảng trắng. Hãy viết các hàm thực hiện các công việc sau:

a. *Đếm xem chuỗi s có bao nhiêu ký tự là ký tự nguyên âm ? Bao nhiêu phụ âm ? (các ký tự a i o u e là nguyên âm).*

b. *Xuất bảng mã ASCII của từng ký tự (khác ký tự khoảng trắng) trong chuỗi. Ví dụ ký tự a có mã ASCII là 97.*

c. *Sắp xếp các ký tự tăng dần (theo mã ASCII), còn các ký tự khoảng trắng đứng ở cuối chuỗi.*

d. *Cắt bỏ các khoảng trắng dư thừa ở đầu, ở cuối và ở giữa chuỗi (giữa các từ chỉ giữ lại một khoảng trắng).*

BT9-2.

Cho chuỗi s gồm các từ, mỗi từ gồm các ký tự chữ cái (giả sử chuỗi s không có khoảng trắng ở đầu chuỗi, cuối chuỗi và giữa các từ có một khoảng trắng).

a. *Từ dài nhất có bao nhiêu ký tự ?*

b. *Tìm các từ có độ dài dài nhất.*

c. *Đếm xem chuỗi có bao nhiêu từ có đúng k ký tự ?*

d. *Loại một từ bên trái và một từ bên phải của chuỗi, hãy tìm chuỗi còn lại.*

BT9-3.

Cho chuỗi s gồm các từ; mỗi từ cách nhau đúng một khoảng trắng (đầu và cuối chuỗi không có khoảng trắng). Hãy xuất các từ theo chiều đảo ngược các ký tự của nó (Ví dụ s="truong dai hoc" thì kết quả là "gnourt iad coh").

BT9-4.

a. *Cho chuỗi s gồm các từ, mỗi từ gồm các ký tự chữ cái (giả sử chuỗi s không có khoảng trắng ở đầu chuỗi, cuối chuỗi và giữa các từ có một khoảng trắng).*

Chèn một chuỗi y vào vị trí thứ k của chuỗi s.

Ví dụ: s=”abc abc bcad cba cdba”

y=”xyz”, k=0 thì kết quả là: ”xyzabc abc bcad cba cdba”

b. *Cho chuỗi s gồm các từ, mỗi từ gồm các ký tự chữ cái (giả sử chuỗi s không có khoảng trắng ở đầu chuỗi, cuối chuỗi và giữa các từ có một khoảng trắng).*

Xóa n ký tự trong chuỗi s bắt đầu từ vị trí thứ k của chuỗi s

Ví dụ: s=”abc abc bcad cba cdba”

n=4, k=4 thì kết quả là: ”abc acad cba cdba”

BT9-5.

Cho chuỗi s gồm các từ, mỗi từ gồm các ký tự chữ cái (giả sử chuỗi s không có khoảng trắng ở đầu chuỗi, cuối chuỗi và giữa các từ chỉ có đúng một khoảng trắng). Xem một từ trong chuỗi là một chuỗi con; hãy sắp xếp các chuỗi con này theo thứ tự tăng.

Ví dụ: s=”abc abc bcad cba cdba”

Kết quả là: “abc abc bcad cba cdba”

BT9-6.

Cho chuỗi S. Hãy tạo một chuỗi S_1 có cấu trúc như sau: Từ đầu tiên của chuỗi S_1 là một từ bên trái cùng của chuỗi S, từ tiếp theo của chuỗi S_1 là một từ bên phải cùng của chuỗi S, tiếp theo là các từ còn lại của chuỗi S. Chuỗi S_1 cũng không có khoảng trắng đầu chuỗi, không có khoảng trắng cuối chuỗi, và giữa các từ chỉ có duy nhất một khoảng trắng.

Ví dụ:

Dữ liệu nhập	*Kết quả*
Que huong la chum khe ngot	*Que ngot huong la chum khe*

BT9-7.

Cho chuỗi S gồm các từ, mỗi từ cách nhau đúng một ký tự trắng, đầu chuỗi và cuối chuỗi không có khoảng trắng. Mã hóa chuỗi: Các ký tự a,b,c,….,x,y,z theo thứ tự đó được xếp thành vòng tròn. Hãy thay mỗi ký tự khác khoảng trắng của S bằng ký tự đứng sau nó đúng k vị trí theo thứ tự trên (Ví dụ k=5, thì ký tự a được thay bằng ký tự f, nghĩa là ký tự b được thay bằng ký tự g,…, ký tự z được thay bằng ký tự e). Giả thiết chỉ xét ký tự chữ thường.

BT9-8.

Cho một chuỗi s chứa tối đa 256 ký tự. Hãy tìm các chuỗi con có chiều dài là k lặp lại nhiều lần nhất.

Ví dụ :s=ABCXYABCABC12345XYHKHABC và k=3

Thì kết quả là ABC (ghi các chuỗi con tìm được, mỗi chuỗi trên một dòng)

BT9-9.

Một chuỗi được gọi là đối xứng nếu nó không ít hơn một ký tự và nếu ta đọc từ trái sang phải hay từ phải sang trái đều được kết quả giống nhau.

Ví dụ:"A" ; "TET" ; "CAOOAC" là các chuỗi đối xứng.

Viết chương trình nhập vào một chuỗi ký tự S, có chiều dài n ($1 \le n \le 1000$). Hãy cho biết chiều dài chuỗi con đối xứng dài nhất. Chuỗi con của S là chuỗi gồm một số ký tự liên tiếp nhau trong S có độ dài nhỏ hơn hoặc bằng n.

Dữ liệu vào được cho trong tập tin văn bản CHUOI.INP gồm 2 dòng.

-dòng đầu ghi số n

-dòng sau ghi n ký tự liên tiếp gồm các chữ cái in hoa (A → Z)

Dữ liệu ra cần được ghi vào tập tin văn bản CHUOI.OUT gồm một số duy nhất là độ dài của chuỗi con đối xứng dài nhất.

Ví dụ

CHUOI.INP	*CHUOI.INP*
18	*19*
IKACOBEGIGEBOCAHTM	*IKACOBEGIGEMHBEGIGE*
CHUOI.OUT	*CHUOI.OUT*
13	*5*

BT9-10.

Hãy đọc tiếng việt không dấu một số nguyên ≥ 0. Ví dụ:2016 sẽ được đọc là 'Hai nghin khong tram muoi sau'.

BT9-11.

Cho n từ (mỗi nhóm ký tự liên tiếp nhau không chứa ký tự khoảng trắng gọi là một từ).

a. *Đếm số lượng từ có 3 ký tự.*

b. *Đếm số từ có chứa ký tự n.*

c. *Sắp xếp các từ theo thứ tự tăng.*

Kết quả xuất lên màn hình theo mẫu bên.

Chương 10. KỸ THUẬT LẬP TRÌNH VỚI FILE

10.1. KHÁI NIỆM

Chương này sẽ trình bày các thao tác cần thiết nhất đối với các loại file văn bản và file nhị phân để giải quyết vấn đề lưu trữ dữ liệu bền vững trên đĩa từ. Chương này rất hữu ích cho các vấn đề bài toán cần xử lý dữ liệu vào/ra từ file như các thuật toán trong các học phần cấu trúc dữ liệu, hệ điều hành, trí tuệ nhân tạo,...

10.2. KHAI BÁO CON TRỎ FILE

Khi thao tác với các file dữ liệu, chúng ta thường không thao tác trực tiếp lên các file dữ liệu đó mà thông qua một vùng đệm của file là một phần của bộ nhớ. Để quản lý các vùng đệm của các file dữ liệu này thì C sử dụng các con trỏ kiểu FILE.

Cách khai báo các con trỏ quản lý file như sau:

FILE * <filepointer>;

Trong đó filepointer là tên biến con trỏ kiểu FILE ; từ lúc này, mọi thao tác tác động lên con trỏ file này thực chất là sẽ tác động lên file dữ liệu tương ứng.

10.3. MỘT SỐ HÀM XỬ LÝ FILE VĂN BẢN THƯỜNG DÙNG

10.3.1. Mở tập tin

FILE *fopen (const char * filename, const char *mode);

Lệnh này có tác dụng mở một file có tên là filename theo cách thức quy định ở trong mode. Trong đó:

-filename là một chuỗi gồm đầy đủ cả đường dẫn, tên tập tin và phần mở rộng (nếu không có đường dẫn thì đường dẫn mặc nhiên là thư mục hiện hành. Khi đánh đường dẫn cần dùng hai dấu “\\”).

-mode nhận các giá trị là wt khi cần mở file mới để ghi theo kiểu văn bản (nếu file đã tồn tại nó sẽ bị xoá) còn rt cho biết mở một file để đọc theo kiểu văn bản.

10.3.2. Đóng tập tin

void fclose (FILE * filepointer);

Lệnh này có tác dụng đóng file văn bản tương ứng với filepointer. Lệnh này thường được sử dụng sau khi đã kết thúc việc đọc hoặc ghi file.

10.3.3. Đưa dữ liệu vào tập tin

void fprintf (FILE * filepointer,const char formattext, varname);

Lệnh này có tác dụng đưa dữ liệu varname - ứng với định dạng formattext vào file đang mở được quản lý bởi con trỏ file có tên là filepointer.

Chẳng hạn nếu muốn ghi số nguyên x vào file có filepointer là f ta thực hiện câu lệnh như sau:

fprintf(f,"%d",x);

Nếu muốn ghi một dòng trống vào file có filepointer là f thì ta thực hiện câu lệnh như sau:

fprintf(f,"\n");

10.3.4. Đọc dữ liệu từ tập tin

void fscanf (FILE *filepointer, &varname);

Lệnh này đọc dữ liệu tại vị trí con trỏ file và đưa giá trị vào varname. Nếu có nhiều varname, thì chúng cách nhau bởi một dấu phẩy.

int feof (filepointer);

Lệnh này nhận giá trị là khác 0 khi con trỏ file filepointer chưa được đặt ở cuối file.

Ví dụ 10-1. File văn bản với mảng một chiều các số nguyên

Viết chương trình đọc file có tên là songuyen.inp có cấu trúc như sau:

-Dòng đầu tiên ghi số n

-Trong các dòng tiếp theo ghi n số nguyên, các số cách nhau ít nhất một khoảng trắng.

Hãy sắp xếp các phần tử theo chiều tăng dần. Ghi kết quả sắp xếp vào file songuyen.out

```
1. #include<stdio.h>
2. #define fi "songuyen.inp"
3. #define fo "songuyen.out"
4. void docfile(int a[], int &n);
5. void sapxep(int a[], int n);
6. void ghifile(int a[], int n);
7. int  main()
8. {
9.  int a[10000],n;
10.  docfile(a,n);
11.  sapxep(a,n);
12.  ghifile(a,n);
13. }
14. void docfile(int a[], int &n)
15. {
16.  FILE *f=fopen(fi,"rt");
17.  fscanf(f,"%d",&n);
18.  for (int i=0;i<n;i++)
19.       fscanf(f,"%d",&a[i]);
20.  fclose(f);
21. }
22. void swap(int &a, int &b)
23. {
24.  int temp=a;
25.  a=b;
26.  b=temp;
27. }
28. void sapxep(int a[], int n)
29. {
30.  for (int i=0;i<n-1;i++)
31.  for (int j=i+1;j<n;j++)
32.  if (a[i]>a[j])
33.       swap(a[i],a[j]);
34. }
```

```
35. void ghifile(int a[], int n)
36. {
37.  FILE *f=fopen(fo,"wt");
38.  fprintf(f,"%d\n",n);
39.  for (int i=0;i<n;i++)
40.  fprintf(f,"%6d",a[i]);
41.  fclose(f);
42. }
```

Ví dụ 10-2. File văn bản với mảng hai chiều các số nguyên

Viết chương trình tạo file văn bản có tên là "matran.inp" có cấu trúc như sau:

-Dòng đầu ghi hai số m,n.

-Trong m dòng tiếp theo mỗi dòng ghi n số các số cách nhau ít nhất một khoảng trắng.

- ***a.*** *Hãy tìm giá trị lớn nhất, giá trị lớn nhì của ma trận*
- ***b.*** *Hãy đếm số lượng số nguyên tố trên từng cột của ma trận*
- ***c.*** *Hãy sắp xếp các dòng của ma trận theo chiều tăng từ trái qua phải*
- ***d.*** *Hãy sắp xếp các dòng của ma trận theo chiều tăng từ trái qua phải và từ trên xuống dưới.*

Kết quả của các câu trên được ghi trong file matran.out (ghi chú rõ đó là kết quả của câu nào)

```
1. #include<stdio.h>
2. #include<iostream.h>
3. #include<math.h>
4. #define fi "matran.inp"
5. #define fo "matran.out"
6. #define max 100
7. void docfile(int a[max][max], int &m, int &n);
8. void timgiatri(int a[max][max], int  m, int  n);
9. void demnguyento(int a[max][max], int  m, int  n);
10.  void sapdong(int a[max][max], int m, int n);
11.  void sapbang(int a[max][max], int m, int n);
12.  FILE *f=fopen(fo,"wt");
13.  int main()
14.  {
15.       int a[max][max],m,n;
16.       docfile(a,m,n);
17.       timgiatri(a,m,n);
18.       demnguyento(a,m,n);
19.       sapdong(a,m,n);
20.       sapbang(a,m,n);
21.       fclose(f);
22.  }
23.  void docfile(int a[max][max], int &m, int &n)
24.  {
25.       FILE *f=fopen(fi,"rt");
26.       fscanf(f,"%d%d",&m,&n);
27.       for (int i=0;i<m;i++)
```

```
28.       for (int j=0;j<n;j++)
29.       fscanf(f,"%d",&a[i][j]);
30.       fclose(f);
31.  }
32.  void timgiatri(int a[max][max], int  m, int  n)
33.  {
34.       int gtlonnhat=a[0][0],gtlonnhi=-INT_MAX;
35.       for (int i=0;i<m;i++)
36.       for (int j=0;j<n;j++)
37.            if (a[i][j]>gtlonnhat)
38.            {
39.            gtlonnhi=gtlonnhat;
40.            gtlonnhat=a[i][j];
41.            }
42.            else
43.            if (a[i][j]>gtlonnhi && a[i][j]!=gtlonnhat)
44.                 gtlonnhi=a[i][j];
45.       fprintf(f,"Cau a:%d\t%d\n",gtlonnhat,gtlonnhi);
46.  }
47.  int  nguyento(int n)
48.  {
49.       if (n<2) return 0;
50.       long k=sqrt(n);
51.       for (int i=2;i<=k;i++)
52.            if (n%i==0)
53.                 return 0;
54.       return 1;
```

```
55.  }
56.  void demnguyento(int a[max][max], int  m, int  n)
57.  {
58.       fprintf(f,"Cau b:");
59.       for (int i=0;i<n;i++)
60.       {
61.       int demnt=0;
62.       for (int j=0;j<m;j++)
63.       if (nguyento(a[j][i]))
64.            demnt++;
65.       fprintf(f,"%d\t",demnt);
66.       }
67.       fprintf(f,"\n");
68.  }
69.  void swap(int &a, int &b)
70.  {
71.       int temp=a;
72.       a=b;
73.       b=temp;
74.  }
75.  void sapdong(int a[max][max], int  m, int  n)
76.  {
77.       for (int i=0;i<m;i++)
78.       for (int j=0;j<n-1;j++)
79.       for (int k=j+1;k<n;k++)
80.            if (a[i][j]>a[i][k])
81.                 swap(a[i][j],a[i][k]);
```

```
82.       fprintf(f,"Cau c:\n");
83.       for (i=0;i<m;i++)
84.       {
85.            for (int j=0;j<n;j++)
86.                 fprintf(f,"%6d",a[i][j]);
87.            fprintf(f,"\n");
88.       }
89.  }
90.  void sapbang(int a[max][max], int  m, int  n)
91.  {
92.       for (int i=0;i<m*n-1;i++)
93.       for (int j=i+1;j<m*n;j++)
94.            if (a[i/n][i%n]>a[j/n][j%n])
95.                 swap(a[i/n][i%n],a[j/n][j%n]);
96.       fprintf(f,"Cau d:\n");
97.       for (i=0;i<m;i++)
98.       {
99.       for (int j=0;j<n;j++)
100.      fprintf(f,"%6d",a[i][j]);
101. fprintf(f,"\n");
102. }
103. }
```

File văn bản với các ký tự

Hàm fputc(int ch, FILE *filepointer) ;

ch là một giá trị nguyên, filepointer là con trỏ file.

Hàm này ghi lên file filepointer một ký tự có mã bằng m=ch%255.

Trong đó ch được xem là một số nguyên không dấu.

Ví dụ 10-3.

Chương trình tạo một file có tên là filechar.inp chứa 255 ký tự chữ cái ngẫu nhiên (ký tự chữ thường) mỗi dòng 10 ký tự.

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. int main()
4. {
5.    FILE *f;
6.    char ch;
7.    randomize();
8.    f=fopen("filechar.inp","wt");
9.    for (int i=1;i<=255;i++)
10.       {
11.            fputc(random(25)+97,f);
12.            fputc(' ',f);
13.            if (i%10==0)
14.                 fprintf(f,"\n") ;
15.       }
16.       fclose(f);
17.  }
```

Lưu ý: Khi f được mở theo kiểu văn bản thì thì ký tự xuống dòng "\n" được ghi thành mã 10. Do đó với ví dụ trên, khi chỉ xử các ký tự khác ký tự khoảng trắng và tất nhiên cả ký tự cuối mỗi dòng thì điều kiện là: ch!=32 && ch!=10

Hàm fgetc(FILE *filepointer)

Hàm đọc một ký tự từ file filepointer. Nếu thành công hàm cho mã đọc được (có giá trị từ 0 đến 255).

Ví dụ 10-4.

Đọc từng ký tự trong file có tên là filechar.inp và xuất ra màn hình.

```
1. #include <stdio.h>
2. int main()
3. {
4.    FILE *f;
5.    char ch;
6.    f=fopen("filechar.inp","rt");
7.    while (!feof(f))
8.    {
9.          ch=fgetc(f);
10.               putchar(ch);
11.         }
12.         fclose(f);
13.   }
```

File văn bản với chuỗi ký tự

int fputs(const char*s, FILE * filepointer);

Trong đó:

-s là con trỏ trỏ tới địa chỉ đầu của một chuỗi ký tự kết thúc bằng '\0'.

-filepointer là con trỏ file.

Lệnh này có tác dụng ghi chuỗi s lên file filepointer (dấu '\0' không ghi lên file). Khi thành công, hàm này trả về ký tự cuối cùng được ghi lên file. Thường khi ghi nhiều dòng, mỗi phần tử trên mỗi dòng thì việc ghi một dấu kết thúc dòng có thể sử dụng lệnh fputc (10,f) hoặc cú pháp fprintf(f,"\n") như đã trình bày trong những phần trước.

Ví dụ 10-5.

Tạo một file chứa n chuỗi, mỗi chuỗi được ghi trên một dòng.

```
1. #include <iostream.h>
2. #include <stdio.h>
3. int main()
4. {
5.    char *s;
6.    FILE *f;
7.    s=new char[256];
8.    int n;
9.    f=fopen("chuoi.inp","wt");
10.       cin>>n;
11.       for (int i=1;i<=n;i++)
12.       {
13.            gets(s);
14.            fputs(s,f);
15.            fprintf(f,"\n");
16.       }
17.       fclose(f);
18.  }
```

char *fgets(char *s, int n, FILE *filepointe)

Trong đó:

-s là con trỏ kiểu char trỏ tới một vùng nhớ đủ lớn để chứa chuỗi ký tự đọc từ file.

-n là số nguyên xác định độ dài cực đại của dãy cần đọc.

Lệnh này có tác dụng đọc một dãy ký tự từ file filepointe chứa vào vùng nhớ s. Việc đọc sẽ kết thúc khi: hoặc đã đọc n-1 ký tự hoặc gặp dấu xuống dòng (mã 10). Khi đó mã 10 được đưa vào chuỗi kết quả hoặc là kết thúc file.

Chuỗi kết quả sẽ được bổ sung thêm dấu hiệu kết thúc chuỗi '\0'.

Khi thành công, hàm trả về địa chỉ vùng nhận kết quả.

Ví dụ 10-6.

Đọc một file chuoi.inp chứa mỗi chuỗi trên một dòng.

```
1. #include <stdio.h>
2. int main()
3. {  char *s;
4.    FILE *f;
5.    s=new char [80];
6.    f=fopen("chuoi.inp","rt");
7.    while (!feof(f))
8.    {
9.          fgets(s,80,f);
10.               puts(s);
11.         }
12.         fclose(f);
13.   }
```

10.4. MỘT SỐ HÀM XỬ LÝ FILE NHỊ PHÂN

10.4.1. File nhị phân chứa các số nguyên

Hàm putw: ghi một số nguyên

int putw(int n, FILE *filepointe);

Trong đó: n là giá trị nguyên

Công dụng: Ghi giá trị n lên file filepointe, nếu thành công hàm trả về số nguyên được ghi.

Hàm getw: đọc một số nguyên

int getw(FILE *filepointe);

Công dụng: đọc một số nguyên từ file filepointe, nếu thành công hàm trả về số nguyên đọc được.

Ví dụ 10-7.

Sử dụng hàm putw để ghi 100 số nguyên từ 1 đến 100 lên file songuyen.inp và sau đó sử dụng hàm getw để đọc file songuyen.inp và in các số nguyên đó lên màn hình.

```
1. #include <stdio.h>
2. int main()
3. {
4.    FILE *f;
5.    f=fopen("songuyen.inp","wb");
6.    for (int i=1;i<=100;i++)
7.         putw(i,f);
8.    fclose(f);
9.    f=fopen("songuyen.inp","rb");
10.        while ((i=getw(f))!=EOF)
11.             printf("%4d",i) ;
12.   }
```

10.4.2. File nhị phân với dữ liệu có cấu trúc

Hàm fwrite ghi các mẫu tin lên file

int fwrite(void *ptr, int size, int n, FILE *filepointer);

Trong đó:

ptr là con trỏ trỏ tới vùng nhớ chứa dữ liệu cần ghi,

size là kích thước của mẫu tin theo byte,

n là số mẫu tin cần ghi,

filepointer là con trỏ file/

Công dụng :

Ghi n mẫu tin kích thước size byte từ vùng nhớ ptr lên file filepointer. Hàm trả về một giá trị bằng số mẫu tin thực sự ghi được.

Hàm fread : đọc các mẫu tin từ file

int fwrite(void *ptr, int size, int n, FILE *filepointer);

Trong đó:

ptr là con trỏ trỏ tới vùng nhớ chứa dữ liệu đọc được,

size là kích thước của mẫu tin theo byte,

n là số mẫu tin cần đọc,

filepointer là con trỏ file

Công dụng :

Đọc n mẫu tin kích thước size byte từ file filepointer vào vùng nhớ ptr. Hàm trả về một giá trị bằng số mẫu tin thực sự đọc được.

Các hàm fwrite, fread thường được sử dụng để đọc và ghi các đối tượng dữ liệu có cùng độ lớn như cấu trúc, số thực.

Ví dụ 10-8.

Tạo một file chứa n phân số (có tử số và mẫu số là các số nguyên), sau đó đọc các phân số và xuất lên màn hình.

```
1. #include <stdio.h>
2. #include <iostream.h>
3. #include <stdlib.h>
```

```
4. struct phanso
5. {  int tuso;
6.    int mauso;
7. };
8. phanso ps;
9. int main()
10.   {     FILE *f;
11.         f=fopen("phanso.inp","wb");
12.         int n;
13.         cin>>n;
14.         for (int i=1;i<=n;i++)
15.         {     cin>>ps.tuso>>ps.mauso;
16.               fwrite(&ps,sizeof(ps),1,f);
17.         }
18.         fclose(f);
19.         f=fopen("phanso.inp","rb");
20.         while (fread(&ps,sizeof(ps),1,f))
21.         printf("\n%d/%d",ps.tuso,ps.mauso);
22.         fclose(f);
23.   }
```

Ví dụ 10-9.

Cho file THI.INP có nhiều chuỗi, mỗi chuỗi trên một dòng, mỗi chuỗi chứa các ký tự chữ cái thường tiếng Anh và khoảng trắng, giữa các từ có đúng một khoảng trắng, đầu và cuối mỗi chuỗi không có khoảng trắng.

Hãy lập trình thực hiện các công việc sau:

CÂU 1.Tìm một chuỗi dài nhất trong số các chuỗi của file, cho biết chiều dài của chuỗi tìm được.

CÂU 2.Tìm số lượng ký tự của mỗi từ trong mỗi chuỗi.

CÂU 3.Tạo các chuỗi con được ghép từ *từ đầu tiên* và *từ cuối cùng* của mỗi chuỗi, giữa hai từ này có một khoảng trắng.

CÂU 4.Xóa *n* ký tự trong mỗi chuỗi bắt đầu từ vị trí *đầu tiên của từ thứ 2* của chuối đó.

CÂU 5.Xem mỗi từ trong mỗi chuỗi là một chuỗi con. Hãy sắp xếp các chuỗi con (trong mỗi chuỗi) tăng dần.

BỘ TEST THAM KHẢO

THI.INP

what good is money if it can not buy happiness

do not waste your time on a man who is not willing to waste their time on you

it is what is in yourself that makes you happy or unhappy

a true friend is someone who reaches for your hand and touches your heart

THI.OUT

CAU 1:

do not waste your time on a man who is not willing to waste their time on you

77

CAU 2:

4 4 2 5 2 2 3 3 3 9

2 3 5 4 4 2 1 3 3 2 3 7 2 5 5 4 2 3

2 2 4 2 2 8 4 5 3 5 2 7

1 4 6 2 7 3 7 3 4 4 3 7 4 5

CAU 3:

what happiness

do you

it unhappy

a heart

CAU 4: nếu n là 14 thì có kết quả:

what if it can not buy happiness

do time on a man who is not willing to waste their time on you

it yourself that makes you happy or unhappy

a someone who reaches for your hand and touches your heart

CAU 5:

buy can good happiness if is it money not what

a do is man not not on on their time time to waste waste who willing you your

happy in is is it makes or that unhappy what you yourself

a and for friend hand heart is reaches someone touches true who your your

```
1. #include<iostream.h>
2. char *xoankytu(char s[],int n, int p)
3. {
4.    char *s1,*s2;
5.    s1=new char [256];
6.    s2=new char [256];
7.    strcpy(s2,s);
8.    s2=s+p+n;
9.    strcpy(s1,s);
10.       s1[p]='\0';
11.       return strcat(s1,s2);
12.   }
13.   char *gheptu(char *s)
14.   {
15.       char stemp[256];
16.       strcpy(stemp,s);
17.       int k=strchr(s,' ')-s;
18.       s[k]='\0';
19.       return strcat(s,strrchr(stemp,' '));
20.   }
21.   char *strsort(char *s)
22.   {
23.       strcat(s," ");
```

```
24.       char stemp[256], strsub[100][256];
25.       int sotu=0;    for (int i=0;i<strlen(s);i++) if
   (s[i]==' ') sotu++;
26.       int i=0;
27.       while (i<sotu)
28.       {
29.            strcpy(stemp,s);
30.            int k=strchr(stemp,' ')-stemp;
31.            stemp[k]='\0';
32.            if (stemp[k-1]==10) stemp[k-1]='\0';
33.            strcpy(strsub[i],stemp);
34.            int l=strlen(strsub[i]);
35.            strcpy(s,s+l+1);
36.            i++;
37.       }
38.       //sap xep cac chuoi con
39.       for (int i=0;i<sotu-1;i++)
40.       for (int j=i+1;j<sotu;j++)
41.       if (strcmp(strsub[i],strsub[j])>0)
42.            {
43.                 strcpy(stemp,strsub[i]);
44.                 strcpy(strsub[i],strsub[j]);
45.                 strcpy(strsub[j],stemp);
46.            }
47.       // noi chuoi con
48.       char skq[256];
49.       strcpy(skq,"");
```

```
50.       for (int i=0;i<sotu;i++)
51.       {
52.            strcat(skq,strsub[i]);
53.            strcat(skq," ");
54.       }
55.       skq[strlen(skq)-1]='\0';// xoa di 1 khoang trang
  cuoi do truoc do da them vao
56.       return skq;
57.  }
58.  int main()
59.  {
60.  //CAU 1
61.  FILE *f,*g;
62.  char *s,*slenmax;
63.  s=new char[256];
64.  g=fopen("THI.OUT","wt");
65.  slenmax=new char[256];
66.  f=fopen("THI.INP","rt");
67.  int strmax=0;
68.  while(!feof(f))
69.  {
70.       fgets(s,256,f);
71.       if (strlen(s)>strmax)
72.            {
73.            strmax=strlen(s);
74.            strcpy(slenmax,s);
75.            if (feof(f)) strmax++;
```

```
76.            }
77.   }
78.   fclose(f);
79.   fprintf(g,"CAU1:\n");
80.   fprintf(g,"%s",slenmax);
81.   fprintf(g,"%d",strmax-1);
82.   //CAU 2
83.   f=fopen("THI.INP","rt");
84.   fprintf(g,"\nCAU2:\n");
85.   while(!feof(f))
86.   {
87.         fgets(s,256,f);
88.         int d=0;
89.         for (int i=0;i<strlen(s);i++)
90.               if  (s[i]!=' ')     d++;
91.               else
92.                     {
93.                           fprintf(g,"%d ",d);
94.                           d=0;
95.                     }
96.         if (!feof(f)) d--;
97.         fprintf(g,"%d\n",d);
98.   }
99.   fclose(f);
100.  //CAU 3
101.  f=fopen("THI.INP","rt");
102.  fprintf(g,"CAU3:\n");
```

```
103. while(!feof(f))
104. {
105.       fgets(s,256,f);
106.       fprintf(g,"%s",gheptu(s));
107. }
108. fclose(f);
109. //CAU 4
110. f=fopen("THI.INP","rt");
111. fprintf(g,"\nCAU4:\n");
112. int n=14;
113. while(!feof(f))
114. {
115.       fgets(s,256,f);
116.       fprintf(g,"%s",xoankytu(s,n,strchr(s,' ')-s+1));
117. }
118. fclose(f);
119. //CAU 5
120. f=fopen("THI.INP","rt");
121. fprintf(g,"\nCAU5:\n");
122. while(!feof(f))
123. {     fgets(s,256,f);
124.       fprintf(g,"%s\n",strsort(s));
125. }
126. fclose(f);
127. fclose(g);
128. }
```

Ví dụ 10.10.

Cho file THI.INP có cấu trúc như sau:

-Dòng đầu ghi hai số nguyên dương m,n.

-Trong m dòng tiếp theo, mỗi dòng ghi n số nguyên trong phạm vi 0 đến 99999.

Các số cách nhau ít nhất một khoảng trắng. Đặt mảng a ứng với dữ liệu của m dòng n cột này. Hãy lập trình thực hiện các công việc sau:

CÂU 1.*Tìm số lớn nhất của mảng a, mảng a có bao nhiêu số bằng số lớn nhất này ?*

CÂU 2.*Tạo mảng b từ mảng a, sao cho b_{ij} là số nguyên tố gần với a_{ij} nhất – có thể bằng chính a_{ij}. Trong trường hợp x có số nguyên tố kề trên là q và có số nguyên tố kề dưới là p mà x – p = q - x thì sẽ ưu tiên lấy số kề trên làm kết quả. Đếm xem trong mảng nguyên tố b có bao nhiêu số mà khi viết các chữ số của nó theo chiều từ trái qua phải hay từ phải qua trái thì ta cùng được một kết quả ?*

CÂU 3.*Tạo mảng c từ mảng a, sao cho $c_{ij} = a_{ij} \times k_i$ với k_i là số nhỏ nhất trên dòng i. Tính tổng các chữ số của tất cả các số của mảng c.*

CÂU 4.*Tạo mảng d từ mảng a,. Gọi k_{ij} là số lượng các số nguyên tố có trên dòng i và có trên cột j. Mỗi số nguyên tố chỉ được tính là một lần trên dòng hoặc trên cột chứa nó. Tính tổng các số của mảng d.*

CÂU 5.*Tạo mảng e từ mảng a với e_{ij} là số lượng số chẵn bao quanh a_{ij} (8 hướng, không kể chính nó). Có bao nhiêu số trong mảng e lớn hơn hoặc bằng 5 ?*

Kết quả ghi vào file THI.OUT có cấu trúc như sau:

-Dòng đầu ghi hai số nguyên là kết quả của CÂU 1.

-Các dòng tiếp theo, mỗi dòng ghi một số nguyên là kết quả của các CÂU 2,3,4,5.

BỘ TEST THAM KHẢO

THI.INP

4 5

101 12 21 42 49

5 101 13 22 10

101 2 3 101 4

22 101 33 44 10

THI.OUT

CAU1:101 5

CAU2:11

CAU3:136

CAU4:72

CAU5:1

(Lưu ý: kiến thức về đọc/ghi file text dạng số nguyên sinh viên đã học trong học phần Cơ sở lập trình).

```
1. #include<iostream.h>
2. #include<math.h>
3. #define maxm 121
4. #define maxn 151
5. void readfile(int a[maxm][maxn], int &m, int &n);
6. void cau1(int a[maxm][maxn], int m, int n);
7. void cau2(int a[maxm][maxn], int m, int n);
8. void cau3(int a[maxm][maxn], int m, int n);
9. void cau4(int a[maxm][maxn], int m, int n);
10.  void cau5(int a[maxm][maxn], int m, int n);
11.  FILE *f;
12.  int main()
13.  {
14.       int a[maxm][maxn],m,n;
15.       readfile(a,m,n);
16.       f=fopen(".OUT","wt");
17.       cau1(a,m,n);
18.       cau2(a,m,n);
19.       cau3(a,m,n);
20.       cau4(a,m,n);
21.       cau5(a,m,n);
```

```
22.       fclose(f);
23.  }
24.  int prime(int n)
25.  {
26.       if (n<2) return 0;
27.       int k=sqrt(n);
28.       for (int i=2;i<=k;i++)
29.       if (n%i==0) return 0;
30.       return 1;
31.  }
32.  int symmetric(int n)
33.  {
34.       int t=n,s=0;
35.       while (t>0)
36.       {
37.            s=s*10+t%10;
38.            t=t/10;
39.       }
40.       return s==n;
41.  }
42.  int tongchuso(int n)
43.  {
44.       int s=0;
45.       while (n>0)
46.       {
47.            s=s+n%10;
48.            n=n/10;
```

```
49.       }
50.       return s;
51.  }
52.  void readfile(int a[maxm][maxn], int &m, int &n)
53.  {
54.       FILE *f;
55.       f=fopen("THI.inp","rt");
56.       fscanf(f,"%d%d\n",&m,&n);
57.       for (int i=0;i<m;i++)
58.       for (int j=0;j<n;j++)
59.       fscanf(f,"%d",&a[i][j]);
60.       fclose(f);
61.  }
62.  void cau1(int a[maxm][maxn], int m, int n)
63.  {
64.       int max=0,d=0;
65.       for (int i=0;i<m;i++)
66.       for (int j=0;j<n;j++)
67.       if (a[i][j]>max) max=a[i][j];
68.       for (int i=0;i<m;i++)
69.       for (int j=0;j<n;j++)
70.       if (a[i][j]==max) d++;
71.       fprintf(f,"CAU1:%d %d",max,d);
72.  }
73.  int primenear(int n)
74.  {
75.       int u=n;
```

```
76.       while (!prime(u)) u++;
77.       int v=n;
78.       while (!prime(v)&& v>0) v--;
79.       return (u-n<=n-v)?u:v;
80.  }
81.  void cau2(int a[maxm][maxn], int m, int n)
82.  {
83.       int d=0;
84.       for (int i=0;i<m;i++)
85.       for (int j=0;j<n;j++)
86.       if (symmetric(primenear(a[i][j])))
87.            d++;
88.       fprintf(f,"\nCAU2:%d",d);
89.  }
90.  int sonhonhatdong(int a[maxm][maxn], int m, int n, int row)
91.  {
92.       int min=INT_MAX;
93.       for (int i=0;i<n;i++)
94.            if (a[row][i]<min)
95.                 min=a[row][i];
96.       return min;
97.  }
98.  void cau3(int a[maxm][maxn], int m, int n)
99.  {
100.      int s=0,ki;
101.      for (int i=0;i<m;i++)
102.      {
```

```
103.            ki=sonhonhatdong(a,m,n,i);
104.            for (int j=0;j<n;j++)
105.                 s=s+tongchuso(a[i][j]*ki);
106.       }
107.       fprintf(f,"\nCAU3:%d",s);
108. }
109. int tongnguyentodongcot(int a[maxm][maxn], int m, int n,
   int row, int column)
110. {
111.       int d=0;
112.       for (int i=0;i<n;i++) // dong row
113.            if (prime(a[row][i])) d++;
114.       for (int j=0;j<m;j++) //cot column
115.            if (prime(a[j][column])) d++;
116.       return d;
117. }
118. void cau4(int a[maxm][maxn], int m, int n)
119. {
120.       int s=0;
121.       for (int i=0;i<m;i++)
122.       for (int j=0;j<n;j++)
123.       if (prime(a[i][j]))
124.       s=s+tongnguyentodongcot(a,m,n,i,j)-1;
125.       else
126.       s=s+tongnguyentodongcot(a,m,n,i,j);
127.       fprintf(f,"\nCAU4:%d",s);
128. }
```

```
129. int demsochan8huong(int a[maxm][maxn], int m, int n, int d,
   int c)
130. {
131.      int dem=0;
132.      if (d-1>=0 && a[d-1][c]%2==0) dem++;
133.      if (d-1>=0 && c+1<n && a[d-1][c+1]%2==0) dem++;
134.      if (c+1<n && a[d][c+1]%2==0) dem++;
135.      if (d+1<m && c+1<n && a[d+1][c+1]%2==0) dem++;
136.      if (d+1<m && a[d+1][c]%2==0) dem++;
137.      if (d+1<m && c-1>=0 && a[d+1][c-1]%2==0) dem++;
138.      if (c-1>=0 && a[d][c-1]%2==0) dem++;
139.      if (d-1>=0 && c-1>=0 && a[d-1][c-1]%2==0) dem++;
140.      return dem;
141. }
142. void cau5(int a[maxm][maxn], int m, int n)
143. {
144.      int d=0;
145.      for (int i=0;i<m;i++)
146.      for (int j=0;j<n;j++)
147.           if (demsochan8huong(a,m,n,i,j)>=5)
148.                d++;
149.      fprintf(f,"\nCAU5:%d",d);
150. }
```

BÀI TẬP

Viết chương trình hoàn chỉnh cho các bài toán sau đây

BT10-1.

a. *Cho file chứa n ký tự; trong đó mỗi ký tự cách nhau đúng một khoảng trắng. Hãy tạo ra file chứa mã ASCII của từng ký tự đó, mỗi dòng ghi đúng 80 số (có thể trừ dòng cuối cùng).*

b. *Cho file chứa các chuỗi, mỗi chuỗi chỉ trên một dòng. Hãy tạo file chứa các dòng (chuỗi) có chứa 2 từ "VIET NAM", trong file kết quả mỗi chuỗi cũng được ghi trên một dòng.*

c. *Cho file chứa n số thực dương và nhỏ hơn 1000, mỗi số thực có đúng 2 chữ số ở phần thập phân. Hãy tính tổng của n số. Hãy tính trung bình cộng của n số. Hãy tìm một số a trong dãy, sao cho trong dãy có d1 số lớn hơn a và d2 số nhỏ hơn a mà d1 và d2 chênh lệch nhau là ít nhất.*

BT10-2.

Viết chương trình tạo một file văn bản có tên là "SONGUYEN.INP" chứa n số nguyên ngẫu nhiên trong khoảng từ 1 đến 65535, mỗi dòng ghi 10 số.

Hãy đếm xem trong file SONGUYEN.INP có bao nhiêu số nguyên tố, bao nhiêu số chính phương? bao nhiêu số hoàn chỉnh. Kết quả ghi vào file văn bản "SONGUYEN.OUT".

BT10-3.

Hãy tạo file chứa các số nguyên tố nhỏ hơn 1 triệu Cho một file chứa các ký tự chữ cái (có thể là chữ hoa hoặc chữ thường), mỗi ký tự cách nhau ít. nhất một khoảng trắng. Hãy cho biết tần số xuất hiện của mỗi ký tự.

BT10-4.

Cho hai file văn bản SOLIEU1.INP, SOLIEU2.INP chứa các số nguyên đã được sắp tăng dần. Hãy trộn hai file này thành một file văn bản SOLIEU.OUT mà các phần tử cũng được sắp tăng dần (không dùng biến mảng).

BT10-5.Tìm vị trí

Cho file bangso.inp có cấu trúc như sau:
-Dòng đầu ghi 2 số m,n.
-Trong m dòng tiếp theo mỗi dòng ghi n số nguyên dương; các số cách nhau ít nhất một khoảng trắng.
Vị trí của mỗi phần tử là (chỉ số dòng, chỉ số cột) của phần tử đó; vị trí dòng, cột bắt đầu từ 1.
Viết một chương trình hoàn chỉnh thực hiện các công việc sau (độc lập với nhau):

a. *Tìm vị trí (i,j) của các phần tử lớn nhất trên từng dòng.*
b. *Tìm vị trí (i,j) của phần tử có tổng các chữ số của nó là lớn nhất (chỉ cần tìm một vị trí thỏa).*
c. *Tìm vị trí (i,j) của phần tử vừa là lớn nhất trên dòng i vừa là nhỏ nhất trên cột j; nếu không có phần tử thỏa thì trả về vị trí (0;0) (chỉ cần tìm một vị trí thỏa).*
d. *Tìm vị trí (i,j) của các phần tử là số nguyên tố lớn nhất của bảng (nếu không có trả về giá trị 0,0).*
e. *Tìm vị trí (i,j) của phần tử có (tổng các phần tử trên dòng i + tổng các phần tử trên cột j) là lớn nhất (chỉ cần tìm một vị trí thỏa).*
f. *Tìm vị trí (i,j) của các phần tử bao quanh nó (8 hướng; không kể chính nó) đều là số chẵn. (nếu không có trả về giá trị 0,0).*

Ví dụ:

Bangso.inp

```
3    4
16   101  8    2
4    424  101  4
2    800  4    6
```

Bangso.out

```
Cau a: (1,2); (2,2); (3;2);
Cau b: (2;2);
Cau c: (1;2);
Cau d: (1,2); (2,3);
Cau e: (3,2);
```

Cau f: (3;1);

BT10-6.

Cho file Bangso.inp có cấu trúc như sau:

-Dòng đầu ghi 2 số m,n.

*-Trong m dòng tiếp theo mỗi dòng ghi n số nguyên dương; các số cách nhau ít nhất một khoảng trắng (bảng có tất cả là m*n phần tử). Vị trí dòng, cột bắt đầu từ 1.*

Viết một chương trình thực hiện các công việc sau (các câu là độc lập với nhau):

a. *Đếm số lượng số đối xứng lớn hơn hoặc bằng 10 có trong bảng (3 điểm).*

b. *Tìm tích của 3 giá trị lớn nhất của bảng (3 giá trị này không nhất thiết khác nhau) (3 điểm).*

c. *Tìm tổng các phần tử của bảng b, biết $b_{ij}=a_{ij}*k$ với k là giá trị nhỏ nhất của dòng i nếu i là số lẻ và $b_{ij}=a_{ij}*k$ với k là giá trị lớn nhất của dòng i nếu i là số chẵn. (2 điểm).*

d. *Tìm tổng các phần tử của mảng c, biết $c_{ij}=l$, với l là tổng số lượng số nguyên tố trên dòng i và trên cột j (nếu a_{ij} là số nguyên tố thì chỉ được đếm một lần trên dòng hoặc trên cột). (2 điểm).*

Ví dụ:

Bangso.inp

```
4    5
16   171  5    7    29
28   2    2    11   31
121  19   5    7    37
6    11   3    23   25
```

Bangso.out

Cau a: 4

Cau b: 765567

Cau c: 6079

Cau d: 112

BT10-7.

Cho file chứa n chuỗi, mỗi chuỗi trên mỗi dòng, mỗi chuỗi gồm các từ, mỗi từ cách nhau đúng một khoảng trắng (giả sử đầu và cuối chuỗi không có khoảng trắng nào dư thừa). Từ có k ký tự gọi là từ loại k; k có thể là 1,2,3,4,5,6,7. Hãy tìm từ loại xuất hiện nhiều lần nhất trong cả file.

Ví dụ chuỗi s là "Thanh ***Pho*** *Ho* ***Chi*** *Minh* ***Gia*** *Dinh" thì kết quả là từ loại 3.*

Cho file text TULOAI.INP có cấu trúc như sau:

Dòng đầu ghi số chuỗi

Trong n dòng tiếp theo mỗi dòng ghi một chuỗi.

Kết quả ghi ra file TULOAI.OUT ghi 1 số nguyên duy nhất là từ loại xuất hiện nhiều lần nhất ứng với chuỗi tương ứng ở file TULOAI.INP

BT10-8.Chuỗi

Cho file STR.INP chứa các chuỗi; mỗi chuỗi trên một dòng; đầu và cuối mỗi chuỗi không có khoảng trắng, giữa các từ có duy nhất một khoảng trắng. Mỗi chuỗi có tối đa 256 ký tự là chữ cái thường hoặc khoảng trắng.

Hãy lập trình thực hiện các công việc sau:

a. *Đếm số lượng ký tự nguyên âm, ký tự phụ âm có trong tất cả các chuỗi của file.*

b. *Chuyển các ký tự đầu của mỗi từ thành chữ hoa (tất cả các chuỗi)*

c. *Tìm số ký tự của mỗi từ trong mỗi chuỗi*

d. *Tìm số từ của mỗi chuỗi*

e. *Cho biết mỗi ký tự (khác ký tự khoảng trắng) xuất hiện bao nhiêu lần ?*

f. *Tìm một chuỗi dài nhất; cho biết chiều dài của chuỗi này ?*

g. *Với mỗi chuỗi tạo thành một chuỗi con bằng cách ghép từ đầu và từ cuối của chuỗi đó.*

h. *Cho biết từ dài nhất của file, từ dài nhất này có bao nhiêu ký tự ?*

i. *Hãy cho biết mỗi loại từ xuất hiện bao nhiêu lần ? (loại từ k gồm các từ có k ký tự: k=1,2,3,….)*

j. *Mỗi chuỗi gồm nhiều từ; mỗi từ xem là mỗi chuỗi con, hãy sắp xếp các chuỗi con trong mỗi chuỗi tăng dần.*

k. *Xóa các từ thứ 2,3,4 của mỗi chuỗi.*

l. *Chèn chuỗi "sai gon university" vào sau khoảng trắng đầu tiên của mỗi chuỗi.*

m. *Đảo ngược các từ trong mỗi chuỗi, trong mỗi từ thì thứ tự của các ký tự là giữ nguyên.*

n. *Đảo ngược các ký tự trong mỗi từ, thứ tự của các từ là giữ nguyên.*

o. *Mã hóa chuỗi: Thay mỗi ký tự khác khoảng trắng bằng ký tự đứng sau nó k vị trí (Vi du khi k=5: ký tự a được thay bằng ký tự f,…, ký tự z được thay bằng ký tự e,…)*

p. *Xóa tất cả khoảng trắng giữa các từ trong các chuỗi.*

q. *Tìm một từ xuất hiện xuất hiện nhiều lần nhất trong file.*

r. *Tìm các ký tự có xuất hiện ở tất cả các chuỗi*

s. **Xóa ít nhất một số ký tự trong mỗi chuỗi sao cho chuỗi còn lại là chuỗi đối xứng.*

t. **Chèn ít nhất một số ký tự vào trong mỗi chuỗi sao cho tạo chuỗi đối xứng.*

u. **Tìm chuỗi con chung dài nhất trong hai chuỗi đầu tiên của file (các chuỗi con này không nhất thiết phải liên tiếp).*

BT10-9.

Cho file THI.INP chứa các chuỗi; mỗi chuỗi trên một dòng; đầu và cuối mỗi chuỗi không có khoảng trắng, giữa các từ có duy nhất một khoảng trắng. Mỗi chuỗi có tối đa 256 ký tự là chữ cái thường hoặc khoảng trắng.

Hãy lập trình thực hiện các công việc sau:

a. *Có bao nhiêu chuỗi chứa chuỗi "que huong" ?*

b. *Mỗi chuỗi chứa bao nhiêu từ có 5 ký tự ?*

c. *Cho biết số lượng nguyên âm của mỗi từ trong mỗi chuỗi ?*

d. *Tìm một chuỗi có nhiều từ nhất; và cho biết chuỗi tìm được này có bao nhiêu từ ?*

e. *Sắp xếp các chuỗi tăng theo thứ tự chiều dài của các chuỗi.*

f. *Mã hóa bậc k: Mỗi ký tự nằm trong từ có k ký tự thì sẽ được thay thế bằng ký tự đứng sau nó k vị trí trong bảng chữ cái tiếng Anh (giả thiết bảng các chữ cái này được sắp theo vòng tròn; nghĩa là ký tự z sẽ được thay bằng ký tự e nếu k=5; rõ ràng cùng một ký tự có thể được mã hóa thành các ký tự khác nhau).*

Kết quả ghi vào file THI.OUT

BỘ TEST THAM KHẢO

THI.INP

que huong la gi ho me ma co giao day phai yeu que huong la gi ho me ai di xa cung nho nhieu

que huong la chum khe ngot cho con treo hai moi ngay

que huong la duong di hoc con ve rop buom vang bay

que huong la con dieu biec tuoi tho con tha tren dong que huong la con do nho em dem khua nuoc ven song

que huong la vang hoa bi la hong tim giau mong toi

la do doi bo dam but mau hoa sen trang tinh khoi

que huong moi nguoi chi mot nhu la chi mot me thoi que huong neu ai khong nho se khong lon noi thanh nguoi

THI.OUT

CAU a: 6

CAU b: 3 1 2 2 1 1 7

CAU c: 2 2 1 1 1 1 1 1 3 1 2 2 2 2 1 1 1 1 2 1 1 1 1 3

(câu này gồm 7 dòng dạng như trên)

CAU d: que huong la gi ho me ma co giao day phai yeu que huong la gi ho me ai di xa cung nho nhieu

24

CAU e:la do doi bo dam but mau hoa sen trang tinh khoi

que huong la duong di hoc con ve rop buom vang bay

que huong la vang hoa bi la hong tim giau mong toi

que huong la chum khe ngot cho con treo hai moi ngay

que huong la gi ho me ma co giao day phai yeu que huong la gi ho me ai di xa cung nho nhieu

que huong la con dieu biec tuoi tho con tha tren dong que huong la con do nho em dem khua nuoc ven song

que huong moi nguoi chi mot nhu la chi mot me thoi que huong neu ai khong nho se khong lon noi thanh nguoi

CAU f: (gồm 7 dòng)

txh mztsl… // đây là kết quả mã hóa của hai từ que huong

BT10-10.

Cho file THI.INP chứa các chuỗi; mỗi chuỗi trên một dòng; đầu và cuối mỗi chuỗi không có khoảng trắng, giữa các từ có duy nhất một khoảng trắng. Mỗi chuỗi có tối đa 256 ký tự là chữ cái thường hoặc khoảng trắng.

Hãy lập trình thực hiện các công việc sau:

a. *Đếm số từ bắt đầu bằng ký tự nguyên âm a,o,u,i,e của mỗi chuỗi ?*

b. *Tách 3 từ đầu bên trái và 3 từ cuối bên phải của mỗi chuỗi để tạo thành một chuỗi con tương ứng. Cac chuoi con nay khong co khoang trang dau va cuoi, giua cac tu của nó có một khoang trang.*

c. *Tìm một chuỗi chứa nhiều nguyên âm nhất ? Và cho biết chuỗi tìm được có bao nhiêu nguyên âm ?*

d. *Mỗi từ loại xuất hiện bao nhiêu lần ? (giải thích: một từ được gọi là từ loại k nếu từ đó chua k ky tu. Giả sử chỉ xét các từ loại 1,2,3,4,5,6,7,8,9,10,11,12).*

e. *Sắp xếp các chuỗi tăng theo thứ tự số lượng từ của các chuỗi.*

f. *Mã hóa bậc k: Mỗi ký tự nằm trong từ có k ký tự sẽ được thay thế bằng ký tự đứng trước nó k vị trí trong bảng chữ cái tiếng Anh (giả thiết bảng các chữ cái này được sắp theo vòng tròn; nghĩa là ký tự a sẽ được thay bằng ký tự v nếu k=5; rõ ràng cùng một ký tự có thể được mã hóa thành các ký tự khác nhau).*

BỘ TEST THAM KHẢO

THI.INP

que huong la gi ho me ma co giao day phai yeu que huong la gi ho me ai di xa cung nho nhieu

que huong la chum khe ngot cho con treo hai moi ngay

que huong la duong di hoc con ve rop buom vang bay

que huong la con dieu biec tuoi tho con tha tren dong que huong la con do nho em dem khua nuoc ven song

que huong la vang hoa bi la hong tim giau mong toi

la do doi bo dam but mau hoa sen trang tinh khoi

que huong moi nguoi chi mot nhu la chi mot me thoi que huong neu ai khong nho se khong lon noi thanh nguoi

THI.OUT

CAU a: 1 0 0 1 0 0 1

CAU b: que huong la cung nho nhieu

que huong la hai moi ngay

….

CAU c: chuỗi tìm được

số lượng nguyên âm

CAU d:1 số lần xuất hiện từ có 1 ký tự

2 số lần xuất hiện từ có 2 ký tự

3

…..

12 số lần xuất hiện từ có 12 ký tự

CAU e:que huong la chum khe ngot cho con treo hai moi ngay

que huong la duong di hoc con ve rop buom vang bay

que huong la vang hoa bi la hong tim giau mong toi

la do doi bo dam but mau hoa sen trang tinh khoi

que huong la gi ho me ma co giao day phai yeu que huong la gi ho me ai di xa cung nho nhieu

que huong la con dieu biec tuoi tho con tha tren dong que huong la con do nho em dem khua nuoc ven song

que huong moi nguoi chi mot nhu la chi mot me thoi que huong neu ai khong nho se khong lon noi thanh nguoi

CAU f: (gồm 7 dòng)

nrb cpjib… // đây là kết quả mã hóa của hai từ que huong

Chương 11. KỸ THUẬT LẬP TRÌNH NÂNG CAO

11.1. THUẬT TOÁN CHIA ĐỂ TRỊ

Chia để trị (divide and conquer) là một trong những phương pháp thiết kế thuật toán cơ bản bao gồm các thao tác:

Chia (divide) bài toán cần giải ra thành một số bài toán con.

-Các bài toán con có kích thước nhỏ hơn và cùng dạng với bài toán cần giải.

Trị (conquer) các bài toán con.

-Giải các bài toán con một cách đệ qui, và

-Bài toán con có kích thước đủ nhỏ sẽ được giải trực tiếp.

Tổ hợp (combine**)** lời giải của các bài toán con.

-Thu đươc lời giải của bài toán xuất phát.

Ý tưởng của chiến lược chia để trị như sau: Người ta phân bài toán cần giải thành các bài toán con. Các bài toán con lại được tiếp tục phân thành các bài toán con nhỏ hơn, cứ thế tiếp tục cho tới khi ta nhận được các bài toán con hoặc đã có thuật toán hoặc là có thể dễ ràng đưa ra thuật toán. Sau đó ta tìm cách kết hợp các nghiệm của các bài toán con để nhận được nghiệm của bài toán con lớn hơn, để cuối cùng nhận được nghiệm của bài toán cần giải. Thông thường các bài toán con nhận được trong quá trình phân chia là cùng dạng với bài toán ban đầu, chỉ có cỡ của chúng là nhỏ hơn.

Thuật toán chia để trị có thể biểu diễn bằng hàm đệ quy như sau:

```
void DivideConquer(A,x); // tìm nghiệm x của bài toán A
{
if (A đủ nhỏ)
    Solve(A);
else
```

{

Phân A thành các bài toán con $A_1, A_2, ..., A_m$;

for (i=1;i<=m;i++)

DivideConquer(A_i, x_i);

Kết hợp các nghiệm x_i (i=1,2,..,m) của các bài toán con A_i để nhận được nghiệm của bài toán A;

}

}

Trong hàm trên, Solve(*A*) là thuật toán giải bài toán *A* trong trường hợp *A* có cỡ đủ nhỏ.

Ví dụ 11-1.

Sử dụng thuật toán chia để trị cho phương pháp sắp xếp trộn (Merge sort).

Chia:

Chia dãy gồm *n* phần tử cần sắp xếp ra thành 2 dãy, mỗi dãy có *n*/2 phần tử.

Trị:

Sắp xếp mỗi dãy con một cách đệ qui sử dụng ***thuật toán trộn***. Khi dãy con chỉ còn một phần tử thì trả lại phần tử này.

Tổ hợp:

Trộn hai dãy con được sắp xếp để thu được dãy được sắp xếp gồm tất cả các phần tử của cả hai dãy con.

Trong đó ý tưởng của thuật toán trộn như sau:

Có hai dãy con đã được sắp xếp. Chọn phần tử nhỏ hơn trong số hai phần tử ở đầu của hai dãy, và đưa nó vào dãy kết quả. Phần tử của dãy con nào được đưa vào dãy kết quả thì phần tử tiếp theo từ dãy đó sẽ được tiếp tục xem xét.

Lặp lại công việc trên cho đến khi tất cả các phần tử của hai dãy con được đưa vào dãy kết quả.

Hai hàm trên được thiết kế chi tiết như sau:

```
1. void mergesort(int A[], int n, int l, int r)
2. {
3.  if(l < r)  {
4.   int m = (l + r) / 2;
5.   mergesort(A, n, l, m);
6.   mergesort(A, n, m + 1, r );
7.   merge(A, n, l, m, r);
8.  }
9. }
10. void merge(int A[], int n, int l, int m, int r)
11. {
12.  int L[maxn], R[maxn], n1, n2;
13.  n1 = m - l + 1;
14.  n2 = r - m;
15.  for (int k = 1; k <= n1; k++)
16.       L[k] = A[l + k -1];
17.  for (int k = 1; k <= n2; k++)
18.       R[k] = A[m + k];
19.  L[n1 + 1] = INT_MAX;
20.  R[n2 + 1] = INT_MAX;
21.  int i = 1, j = 1;
22.  for(int k = l; k <= r; k++)
23.  if(L[i] <= R[j])
24.       A[k] = L[i++];
25.  else
26.       A[k] = R[j++];
27. }
```

11.2. THUẬT TOÁN QUAY LUI

Quay lui (backtracking algorithm) là một trong những thuật toán cơ bản và quan trọng được áp dụng để giải quyết nhiều vấn đề bài toán; đặc biệt là bài toán liệt kê.

11.2.1. Bài toán liệt kê

Cho A_1, A_2,..., A_n là các tập hữu hạn, ký hiệu $X = A_1 \times A_2 \times ... \times A_n = \{(x_1, x_2, ..., x_n): x_i \in A_i$, i=1, 2,..., $n\}$. Giả sử P là tính chất cho trên tập X. Bài toán đặt ra là cần liệt kê tất cả các phần tử của X thoả mãn tính chất P. Gọi $D = \{ x = (x_1, x_2,...,x_n) \in X$: x thoả mãn tính chất $P\}$; khi đó các phần tử của tập D được gọi là các lời giải chấp nhận được.

Tất cả các bài toán liệt kê tổ hợp cơ bản đều có thể phát biểu dưới dạng bài toán liệt kê (Q).

Ví dụ một số bài toán liệt kê

-Bài toán liệt kê xâu nhị phân độ dài n dẫn về việc liệt kê các phần tử của tập $B_n = \{(x_1, ..., x_n): x_i \in \{0, 1\}$, i=1, 2, ..., $n\}$.

-Bài toán liệt kê các tập con m phần tử của tập $N = \{1, 2, ..., n\}$ dẫn về việc liệt kê các phần tử của tập $S(m,n) = \{(x_1,..., x_m) \in N^m: 1 \le x_1 < ... < x_m \le n \}$.

- Bài toán liệt kê các hoán vị của các số tự nhiên 1, 2, ..., n dẫn về việc liệt kê các phần tử của tập $\Pi_n = \{(x_1,..., x_n) \in N^n: x_i \ne x_j ; i \ne j \}$.

11.2.2. Lưu đồ thuật toán quay lui

Nội dung chính của thuật toán này là việc xây dựng dần các thành phần của cấu hình bằng cách thử tất cả các khả năng. Giả thiết cấu hình cần tìm được mô tả bằng một bộ gồm n thành phần $x_1,x_2,\ldots,x_n$. Giả sử đã xác định được i-1 thành phần $x_1,x_2,\ldots,x_{i-1}$ (mà ta sẽ gọi là lời giải bộ phận cấp i-1), bây giờ ta xác định thành phần x_i bằng cách duyệt tất cả các khả năng có thể đề cử cho nó. Với mỗi khả năng j, kiểm tra xem j có chấp nhận được không. Xảy ra hai trường hợp sau:

-Nếu chấp nhận j thì xác định x_i theo j, sau đó nếu $i = n$ thì ta được một cấu hình, còn ngược lại thì ta tiến hành việc xác định x_{i+1}.

-Nếu thử tất cả các khả năng mà không có khả năng nào được chấp nhận thì quay lại bước trước để xác định lại bước i-1.

Điểm quan trọng của thuật toán là phải ghi nhớ tại mỗi bước đã đi qua, những khả năng nào đã thử để tránh trùng lặp. Rõ ràng những thông tin này cần được lưu trữ theo cơ chế ngăn xếp (stack). Vì thế thuật toán này rất phù hợp với việc lập trình trên một ngôn ngữ cho phép goi đệ qui. Bước xác định xi có thể diễn tả qua thủ tục được tổ chức đệ qui dưới đây:

```
void try(int i)
{
for (int j=1;j<=n;j++)
if (<chấp nhận j>)
        {
                <Xác định xi theo j>;
                if (i==n)       <Ghi nhận một cấu hình>
                else            try(i+1);
        }
}
```

Phần quan trọng nhất trong thủ tục trên là việc đưa ra được một danh sách các khả năng đề cử và việc xác định giá trị của biểu thức logic <chấp nhận j>. Thông thường giá trị này, ngoài việc phụ thuộc *j*, còn phụ thuộc vào việc đã chọn các khả năng tại *i*-1 bước trước. Trong những trường hợp như vậy, cần ghi nhớ trạng thái mới của quá trình tìm kiếm sau khi <xác định x_i theo *j*> và trả lại trạng thái cũ sau lời gọi try(i+1). Các trạng thái này được ghi nhận nhờ một biến tổng thể, gọi là các biến trạng thái.

Sau khi xây dựng thủ tục đệ qui *try*, đoạn chương trình chính giải bài toán liệt kê có dạng:

```
int main()
{
        init();
        try(1);
```

}

Trong đó init là thủ tục khởi tạo các giá trị ban đầu (nhập các giá trị tham số của bài toán, khởi gán các biến trạng thái, biến đếm,…). Tiếp theo chúng ta sẽ xem xét việc áp dụng thuật toán quay lui giải một số bài toán liệt kê tổ hợp cơ bản.

Ví dụ 11-2.

Liệt kê các dãy nhị phân độ dài n.

Biểu diễn giải nhị phân dưới dạng $b_1,b_2,\ldots,b_n$, trong đó $b_i \in \{0,1\}$. Thủ tục đệ quy try(i) xác định b_i, trong đó các giá trị đề cử là 0 và 1. Các giá trị này mặc nhiên được chấp nhận mà không phải thỏa mãn điều kiện gì (vì thế bài toán không cần biến trạng thái). Thủ tục init nhập giá trị *n* và khởi gán biến đếm *count*, thủ tục *result* đưa ra dãy tìm được.

```
1. #include<iostream.h>
2. int x[1000];
3. int n,count;
4. void binary_browser(int i);
5. int main()
6. {
7.  cout<<"n = ";cin>>n;
8.  count=0;
9.  binary_browser(1);
10. }
11. void result()
12. {
13.  cout<<++count<<": ";
14.  for (int i=1;i<=n;i++)
15.       cout<<x[i]<<" ";
16.  cout<<endl;
```

```
17. }
18. void binary_browser(int i)
19. {
20.  for (int j=0;j<=1;j++)
21.  {    x[i]=j;
22.       if (i==n) result();
23.       else        binary_browser(i+1);
24.  }
25. }
```

Ví dụ 11-3.

Liệt kê các dãy hoán vị của {1,2,...,n}.

Biểu diễn hoán vị dưới dạng $p_1,p_2,\ldots,p_n$, trong đó p_i nhận giá trị từ 1 đến n và $p_i<>p_j$ với mọi $i<>j$. Các giá trị từ 1 đến n được lần lượt đề cử cho p_i, trong đó giá trị j được chấp nhận nếu nó chưa được dùng. Vì thế cần phải ghi nhớ đối với mỗi giá trị j xem nó đã được dùng hay chưa. Điều này được thực hiện nhờ một dãy biến logic b_j, trong đó b_j bằng *true* nếu j chưa được dùng. Các biến này cần phải được gán giá trị *true* trong thủ tục *init*. Sau khi gán j cho p_i cần ghi nhận *false* cho b_j và phải gán lại *true* khi thực hiện xong *result* hay *try*(i+1).

```
1. #include <iostream.h>
2. int x[1000],b[1000],n,count;
3. void permutation(int i);
4. int main()
5. {
6.  cout<<"n = ";cin>>n;
7.  count=0;
8.  for (int i=1;i<=n;i++)    b[i]=1;
9.  permutation(1);
```

```
10. }
11. void  result()
12. {
13.  cout<<++count<<":";
14.  for (int i=1;i<=n;i++)
15.  cout<<x[i]<<" ";
16.  cout<<endl;
17. }
18. void permutation(int i)
19. {
20.  for (int j=1;j<=n;j++)
21.  if (b[j]==1)
22.  {
23.       x[i]=j;
24.       b[j]=0;
25.       if (i==n) result();
26.       else      permutation(i+1);
27.       b[j]=1;
28.  }
29. }
```

Ví dụ 11-4.

Liệt kê các tổ hợp chặp m của {1,2,...,n}

Biểu diễn tổ hợp dưới dạng $c_1,c_2,\ldots,c_m$, trong đó

$1 \le c_1 < c_2 < \ldots < c_m \le n$

Từ đó suy ra các giá trị đề cử cho c_i là từ $c_{i-1}+1$ đến $n-m+i$. Để điều này đúng cho cả trường hợp $i=1$, cần thêm vào c_0 với $c_0 = 0$. Các giá trị đề cử này mặc nhiên được chấp nhận.

Trong nhiều bài toán, việc xác định điều kiện <***chấp nhận j***> không phải là đơn giản, nó đòi hỏi một kinh nghiệm tổ chức nhất định. Rõ ràng độ phức tạp của bài toán phụ thuộc rất nhiều vào độ phức tạp của điều kiện này. Nói chung, người ta cố gắng thu hẹp diện đề cử các khả năng *j* và tinh giản tối đa việc tính điều kiện <*chấp nhận j*>. Đây cũng là cách chủ yếu để nâng cao tính khả thi của các bài toán liệt kê.

```
1. #include<iostream.h>
2. void combination(int i);
3. int x[1000],n,k,count;
4. int main()
5. {
6. cout<<"n = ";cin>>n;
7. cout<<"k = ";cin>>k;
8. count=0;
9. combination(1);
10. }
11. void result()
12. {
13.  cout<<++count<<":";
14.  for (int i=1;i<=k;i++)
15.  cout<<x[i]<<" ";
16.  cout<<endl;
17. }
18. void combination(int i)
19. {
20. for (int j=x[i-1]+1;j<=n-k+i;j++)
21.  {     x[i]=j;
22.        if (i==k)     result();
```

```
23.        else           combination(i+1);
24.  }
25. }
```

Ví dụ 11-5.

Liệt kê các chỉnh hợp lặp chặp k của {1,2,…,n}

```
//arrangement with repetition
1. #include<iostream.h>
2. void arrangement(int i);
3. int x[1000];
4. int n,k,count;
5. int main()
6. {
7.  count=0;
8.  cout<<"n = "; cin>>n;
9.  cout<<"k = "; cin>>k;
10.  arrangement(1);
11. }
12. void result()
13. {
14.  cout<<++count<<":";
15.  for (int i=1;i<=k;i++)
16.       cout<<x[i]<<" ";
17.  cout<<endl;
18. }
19. void arrangement(int i)
20. {
21.  for (int j=1;j<=n;j++)
```

```
22.  {
23.        x[i]=j;
24.        if (i==k)        result();
25.        else          arrangement(i+1);
26.  }
27. }
```

Ví dụ 11-6.

Bài toán xếp Hậu: Liệt kê tất cả các cách xếp n quân Hậu trên bàn cờ n x n sao cho chúng không khống chế lẫn nhau.

Đánh số cột và dòng của bàn cờ từ 1 đến n. Mỗi dòng được xếp đúng 1 quân Hậu. Vấn đề còn lại là xem mỗi quân Hậu được xếp vào cột nào. Từ đó dẫn đến việc biểu diễn một cách xếp bộ n thành phần $x_1,x_2,\ldots,x_n$, trong đó $x_i = j$ nghĩa là quận Hậu dòng i được xếp vào cột j. Các giá trị đề cử cho x_i là từ 1 đến n. Giá trị j là được chấp nhận nếu ô (i,j) chưa bị các quân Hậu trước khống chế. Để kiểm soát được điều này, ta cần phải ghi nhận trạng thái của bàn cờ trước cũng như sau khi xếp được một quân Hậu. Để ý rằng, theo luật cờ Vua, quân Hậu khống chế các quân cờ khác trên cùng hàng, cùng cột và hai đường chéo chứa nó.

Việc kiểm soát theo chiều ngang là không cần thiết vì mỗi dòng được xếp đúng một quân Hậu. Việc kiểm soát theo chiều dọc được ghi nhận nhờ dãy biến logic a_j với quy ước a_j=*true* nếu cột j còn trống. Đối với hai đường chéo ta nhận xét rằng một đường có phương trình $i + j = const$, còn đường kia $i – j = const$ ($2 \le i + j \le 2n$, $1 – n \le i – j \le n - 1$), từ đó đường chéo thứ nhất được ghi nhận nhờ dãy biến logic b_j ($2 \le j \le 2n$) và đường chéo thứ hai, nhờ dãy biến logic c_j ($1 - n \le j \le n-1$) với quy ước các đường này còn trống nếu biến tương ứng có giá trị *true*. Các biến trạng thái a_j,b_j,c_j cần được khởi gán giá trị *true* trong thủ tục *init*. Như vậy giá trị j được chấp nhận khi và chỉ khi cả 3 biến a_j,b_{i+j},c_{i-j} cùng có giá trị *true*. Các biến này cần gán lại *false* khi xếp xong quân Hậu thứ i và trả lại *true* sau khi gọi *result* hay *try*(i+1).

Lưu ý rằng bài toán xếp hậu không phải là luôn có lời giải, chẳng hạn bài toán không có lời giải khi n=2,3. Do đó điều này cần được thông báo khi kết thúc thuật toán. Thuật toán trình bày ở trên là chưa hiệu quả.

```
1. #include <iostream.h>
2. int const n=8;
3. int a[n],b[2*n-1],c[2*n-1],x[n],count=0;
4. void result()
5. {
6.  cout<<++count<<":";
7.  for (int j=1;j<=n;j++)    cout<<x[j]<<" ";
8.  cout<<endl;
9. }
10. void  Queen(int i)
11. {
12.  if (i>n) result();
13.  else
14.  for (int j=1;j<=n;j++)
15.            if (a[j]==0 && b[i+j]==0 && c[i-j+n]==0)
16.            {
17.       x[i]=j;a[j]=1;b[i+j]=1;c[i-j+n]=1;
18.       Queen(i+1);
19.       a[j]=0;b[i+j]=0;c[i-j+n]=0;
20.            }
21. }
22. int main()
23. {Queen(1);
24. }
```

Các bài toán duyệt tổ hợp đã được đề cập ở trên là thuật toán cơ sở để giải quyết nhiều bài toán khác. Các thuật toán duyệt tổ hợp này dễ cài đặt và tìm được lời giải chính xác. Tuy nhiên, các thuật toán này có độ phức tạp không phải thời gian đa thức, do đó chỉ giải được bài toán với kích thước nhỏ.

Thuật toán nhánh cận là một cải tiến được biết đến rộng rãi của thuật toán quay lui. Thuật toán nhánh cận đưa ra các điều kiện cho phép cắt bỏ một số nhánh của mà chắc chắn nhánh nó không dẫn đến lời giải tối ưu. Thuật toán nhánh cận cho phép giải được bài toán có kích thước lớn hơn so với việc sử dụng giải bằng thuật toán quay lui; tuy nhiên việc cải thiện này đa số là không đáng kể.

11.3. THUẬT TOÁN QUY HOẠCH ĐỘNG

Quy hoạch động (Dynamic programming – viết tắt là DP) là một kỹ thuật thiết kế khá vạn năng để phát triển thuật toán cho nhiều lớp bài toán khác nhau. Kỹ thuật này là một trong những công cụ mạnh nhất để thiết kế thuật toán giải các bài toán tối ưu hóa.

Việc phát triển thuật toán dựa trên DP bao gồm 3 giai đoạn: Phân rã, ghi nhận lời giải và tổng hợp lời giải.

11.3.1. Phân rã

Chia bài toán cần giải thành những bài toán con nhỏ hơn có cùng dạng với bài toán ban đầu sao cho bài toán con kích thước nhỏ nhất có thể giải một cách trực tiếp.

Bài thân bài toán xuất phát có thể coi là bài toán con có kích thước lớn nhất trong họ các bài toán con này.

11.3.2. Ghi nhận lời giải

Lưu trữ lời giải của các bài toán con vào một bảng.

Việc làm này là cần thiết vì lời giải của những bài toán con thường được sử dụng lại nhiều lần, và điều đó nâng cao hiệu quả của thuật toán do không phải giải lặp lại cùng một bài toán con nhiều lần.

11.3.3. Tổng hợp lời giải

Lần lượt từ lời giải của các bài toán con kích thước nhỏ hơn tìm cách xây dựng lời giải của bài toán con kích thước lớn hơn, cho đến khi thu được lời giải của bài toán con xuất phát (là bài toán con có kích thước lớn nhất).

Kỹ thuật giải các bài toán con của quy hoạch động là quá trình đi từ dưới lên (bottom-up) là điểm khác quan trọng với phương pháp chia để trị, trong đó các bài toán con được trị một cách đệ quy (top-down).

Khi sử dụng phương pháp quy hoạch động, có thể gặp hai khó khăn sau đây:

Thứ nhất, Không phải lúc nào sự kết hợp lời giải của các bài toán con cũng cho ra lời giải của bài toán lớn hơn.

Thứ hai, Số lượng các bài toán con cần giải quyết và lưu trữ có thể rất lớn, không thể chấp nhận được.

Hiện tại, chưa ai xác định được một cách chính xác những bài toán nào có thể được giải quyết hiệu quả bằng phương pháp quy hoạch động.

Thuật toán quy hoạch động như đã mô tả trên, có thể sử dụng các biến đơn để lưu trữ kết quả của các bài toán trung gian, cũng có thể sử dụng mảng 1 chiều hoặc 2 chiều,…

11.3.4. Một số bài toán quy hoạch động điển hình

Ví dụ 11.7.

Cho dãy số nguyên $A = a_1, a_2,...,a_n$ ($n \le 1000$, $-10000 \le a_i \le 10000$). Một dãy con của A là một cách chọn ra trong A một số phần tử giữ nguyên thứ tự. Hãy tìm một dãy con tăng dài nhất.

Hướng dẫn giải:

Với $\forall$ i: $1 \le i \le n$. Gọi $L[i]$ là độ dài của dãy con tăng dài nhất kết thúc tại a_i.

Cơ sở quy hoạch động

$L[1]$ là độ dài dãy con tăng dài nhất kết thúc tại a_1. Dãy con này chỉ gồm một phần tử nên $L[1] = 1$.

Công thức truy hồi

Giả sử với i từ 2 đến n, ta cần tính L[i]: độ dài dãy con tăng dài nhất kết thúc tại a_i. L[i] được tính trong điều kiện các L[i -1], L[i - 2], ..., L[1] đã biết.

Dãy con tăng dài nhất kết thúc tại a_i sẽ được thành lập bằng cách lấy a_i ghép vào cuối một trong số những dãy con tăng dài nhất kết thúc tại vị trí a_j đứng trước a_i. Ta sẽ chọn dãy nào để ghép a_i vào cuối ? Tất nhiên là chỉ được ghép a_i vào cuối những dãy con kết thúc tại a_j nào đó nhỏ hơn a_i (để đảm bảo tính tăng) và dĩ nhiên ta sẽ chọn dãy dài nhất để ghép a_i vào cuối (để đảm bảo tính dài nhất). Và do đó L[i] được tính như sau: Xét tất cả các chỉ số j trong khoảng từ 1 đến i-1 mà $a_j < a_i$, chọn ra chỉ số *jmax* có L[*jmax*] lớn nhất. Đặt L[i]= L[*jmax*] + 1.

Truy vết

Tại bước xây dựng dãy L, mỗi khi tính L[i] = L[*jmax*] + 1, ta đặt P[i] = *jmax*. Để lưu lại rằng: Dãy con dài nhất kết thúc tại a_i sẽ có phần tử kế tiếp trước đó là a_{jmax}. Dãy con cần tìm sẽ kết thúc tại trong những vị trí mà giá trị L[i] là lớn nhất.

Chẳng hạn cho dãy *A:*

2 8 11 3 5 9 10 4 17 6

Ta sẽ lập bảng minh họa thuật toán vừa mô tả qua dãy số trên.

i	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
A[i]	2	8	11	3	5	9	10	4	17	6
L[i]	1	2	3	2	3	4	5	3	6	4
P[i]	0	1	2	1	4	5	6	4	7	8

Dãy con tăng dài nhất tìm đươc là: 2 3 5 9 10 17

Ví dụ 11.8.

Cho *n* món hàng, món thứ i có khối lượng w[i] và có giá trị là v[i] (trong đó w_i,v_i <=500 và là các số nguyên). Cần chọn những món hàng nào để bỏ vào ba lô sao cho khối lượng các vật được chọn không vượt quá *M* (số nguyên dương <=1000) và có giá trị là lớn nhất. Mỗi vật chỉ có thể chọn tối đa một lần.

Hướng dẫn giải:

Nếu gọi *F*[i][j] là giá trị lớn nhất có thể có bằng cách chọn trong các món hàng {1, 2, ..., i} với giới hạn trọng lượng j thì giá trị lớn nhất khi được chọn trong số *n* món hàng với giới hạn trọng lượng *M* chính là *F*[*n*][*M*].

Công thức truy hồi tính *F*[i]][j]

Với giới hạn trọng lượng j, việc chọn tối ưu trong số các món hàng {1, 2, ...,i - 1, i} để có giá trị lớn nhất sẽ có hai khả năng:

- Nếu không chọn món hàng thứ i thì *F*[i][j] là giá trị lớn nhất có thể bằng cách chọn trong số các món hàng {1, 2, ..., i - 1} với giới hạn trọng lượng là j. Tức là *F*[i][j] = *F*[i – 1][j].
- Nếu có chọn món hàng thứ *i* (tất nhiên chỉ xét tới trường hợp này khi mà $W_i \leq j$) thì *F*[i][j] bằng giá trị món hàng thứ i là V_i cộng với giá trị lớn nhất có thể có được bằng cách chọn trong số các món hàng {1, 2, ..., i - 1} với giới hạn trọng lượng j - W_i. Tức là về mặt giá trị thu được: *F*[i][j] = C_i + *F*[i – 1][j - W_i].

Vì theo cách xây dựng *F*[i][j] là giá trị lớn nhất có thể, nên *F*[*i*][*j*] sẽ là *max* trong hai giá trị thu được ở trên.

Cơ sở quy hoạch động

Dễ thấy *F*[0][j] = giá trị lớn nhất có thể bằng cách chọn trong số 0 món hàng = 0.

Tính bảng phương án

Bảng phương án *F* gồm *n* + 1 dòng, *M* + 1 cột, trước tiên được điền cơ sở quy hoạch động: Dòng 0 gồm toàn số 0. Sử dụng công thức truy hồi, dùng dòng 0 tính dòng 1, dùng dòng 1 tính dòng 2,... đến khi tính hết dòng *n*.

Truy vết

Tính xong bảng phương án thì ta quan tâm đến *F*[*n*][*M*] đó chính là giá trị lớn nhất thu được khi chọn trong cả n món hàng với giới hạn trọng lượng *M*. Nếu *F*[*n*][*M*] = *F*[*n* – 1][*M*] thì tức là không chọn món hàng thứ *n*, ta truy tiếp *F*[*n* – 1][*M*]. Còn nếu *F*[*n*][*M*] ≠ *F*[*n* – 1][*M*] thì ta thông báo rằng phép chọn tối ưu có chọn món hàng thứ n và truy tiếp *F*[*n* – 1][*M* - W_n]. Cứ tiếp tục cho tới khi truy lên tới hàng 0 của bảng phương án.

Chẳng hạn, cho bài toán ba lô với *n*=6 *M*=12 và các dãy *a*[i], *c*[i] như sau:

a[i]	5	1	6	4	9	5
c[i]	6	7	4	2	11	7

Ta sẽ lập bảng minh họa thuật toán vừa mô tả qua các số liệu trên.

k \ v	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	6	6	6	6	6	6	6	6
2	7	7	7	7	7	13	13	13	13	13	13	13
3	7	7	7	7	7	13	13	13	13	13	13	17
4	7	7	7	7	9	13	13	13	13	15	15	17
5	7	7	7	7	9	13	13	13	13	18	18	18
6	7	7	7	7	9	14	14	14	14	18	20	20

Kết quả: Các món hàng ở các vị trí sau được chon: 1,2,6

Khi đó tổng khối lượng là 11 và tổng giá trị là: 20.

Một điểm cốt lõi không thể bỏ qua được là phương pháp quy hoạch động giải bài toán cái ba lô không thể thực hiện được nếu trọng lượng của ba lô và và những kích thước hay giá trị của các đồ vật là số thực thay vì số nguyên. Đây không phải là một rắc rối nhỏ mà là một khó khăn chính. Chưa có một giải pháp tốt nào cho vấn đề được biết đến, và đến nay, một giải pháp như vậy hầu như không tồn tại.

Ví dụ 11.9.

Cho một bảng *A* kích thước *m* x *n* ô (bắt đầu từ 1); mỗi ô chứa một số nguyên. Từ ô *A*[i,j] chỉ có thể di chuyển sang một trong 3 ô *A*[i,j+1], *A*[i-1,j+1] và *A*[i+1,j+1]. Hãy tìm vị trí ô xuất phát từ cột 1 sáng cột *n* sao cho tổng các số ghi trên đường đi là lớn nhất.

Gọi *B*[i,j] là giá trị lớn nhất có thể có khi di chuyển đến ô *A*[i,j]. Rõ ràng với những ô ở cột 1 thì *B*[i,1]=*A*[i,1].

Với những ô (i,j) ở các cột khác. Vì chỉ những ô (i,j-1), (i-1,j-1), (i+1,j-1) là có thể sang được ô (i,j), và khi sang ô (i,j) thì giá trị được cộng them *A*(i,j) nữa. Chúng ta cần *B*[i,j] là số điểm lớn nhất có thể nên *B*[i,j]=max(*B*[i,j-1], *B*[i-1,j-1], *B*[i+1,j-1])+*A*[i,j]. Ta dung công thức truy hồi này tính tất cả các *B*[i,j]. Cuối cùng chọn ra *B*[i,*n*] là phần tử lớn nhất trên cột *n* của bảng *B* và từ đó truy vết tìm ra đường đi có giá trị lớn nhất.

Ví dụ 11.10. Dãy con chung dài nhất

Cho hai số nguyên dương $M,N(0<M,N<=100)$ và hai dãy số nguyên: A_1, $A_2,\ldots,A_M$ và $B_1, B_2,..., B_N$. Tìm một dãy *C* là con chung dài nhất của hai dãy *A* và *B*, nhận được từ *A* bằng cách xoá đi một số số hạng và cũng nhận được từ *B* bằng cách xoá đi một số số hạng.

Hướng dẫn giải:

Cần xây dựng mảng $L[0..M, 0..N]$ với ý nghĩa: $L[i, j]$ là độ dài của dãy chung dài nhất của hai dãy $A[0.. i]$ và $B[0..j]$.

Đương nhiên nếu một dãy là rỗng (số phần tử là 0) thì dãy con chung cũng là rỗng vì vậy $L[0, j] = 0\ \forall j, j = 1.. N$,

$L[i, 0] = 0\ \forall i, i = 1.. M$.

Với $M \geq i > 0$ và $N \geq j > 0$ thì $L[i, j]$ được tính theo công thức truy hồi sau:

$L[i,j] = \text{Max}\{L[i, j-1], L[i-1, j], L[i-1, j-1] + x\}$

(với $x = 0$ nếu $A[i] \neq B[j]$, $x=1$ nếu $A[i]=B[j]$)

Ví dụ 11.11. Chia đa giác.

Cho một đa giác lồi N đỉnh. Bằng các đường chéo không cắt nhau, ta có thể chia đa giác thành N–2 tam giác. Hãy xác định cách chia có tổng các đường chéo ngắn nhất.

Hướng dẫn giải:

Để đơn giản ta coi mọi đoạn thẳng nối 2 đỉnh đều là "đường chéo" (nếu nối 2 đỉnh trùng nhau hoặc 2 đỉnh liên tiếp thì có độ dài bằng 0).

Gọi $F(i,j)$ là tổng độ dài các đường chéo khi chia đa giác gồm các đỉnh từ i đến j thành các tam giác. Nếu $j < i+3$ thì đa giác đó có ít hơn 4 đỉnh, không cần phải chia nên $F(i,j)=0$. Ngược lại ta xét cách chia đa giác đó bằng cách chọn một đỉnh k nằm giữa i,j và nối i,j với k. Khi đó $F(i,j)=F(i,k)+F(k,j)+d(i,k)+d(k,j)$; $d(i,k)$ là độ dài đường chéo (i,k).

Tóm lại công thức quy hoạch động như sau:

$F(i,j)=0$ với $j < i+3$.

$F(i,j)=\min(F(i,k)+F(k,j)+d(i,k)+d(k,j)$ với $k=i+1,...j–1)$.

$F(1,n)$ là tổng đường chéo của cách chia tối ưu.

11.4. THUẬT TOÁN THAM LAM

Các thuật toán tham lam (greedy method) là một cách tiếp cận trực tiếp để xây dựng thuật toán giải nhiều lớp bài toán khác nhau. Chúng ta sẽ đưa ra quyết định dựa ngay vào thông tin đang có, và trong tương lai sẽ không xem xét lại tác động của các quyết định đã nhận trong quá khứ. Chính vì thế các thuật toán dạng này rất dễ đề xuất, và thông thường chúng không đòi hỏi nhiều thời gian tính. Tuy nhiên các thuật toán

dạng này thông thường không cho kết quả đúng. Thuật toán tham lam thường sắp xếp các đối tượng theo một thứ tự tăng/giảm hoặc chọn các phần tử lớn nhất/bé nhất theo một ngữ cảnh cụ thể nào đó.

Lưu đồ của thuật toán tham lam

Lời giải cần tìm có thể mô tả như là bộ gồm hữu hạn các thành phần thõa mãn các điều kiện nhất định. Ta phải giải quyết bài toán một cách tối ưu: Tìm lời giải với giá trị hàm mục tiêu lớn nhất (nhỏ nhất). Có tập các ứng cử viên có thể lựa chọn làm các thành phần của lời giải. Xuất phát từ một lời giải rỗng, thuật toán xây dựng lời giải của bài toán theo từng bước, ở mỗi bước sẽ chọn một phần tử từ tập ứng cử viên và bổ sung vào lời giải hiện có.

Hàm sulution(*S*) nhận biết tính chấp nhận được của lời giải S.

Hàm select(*C*) chọn từ tập *C* ứng cử viên có triển vọng nhất để bổ sung vào lời giải hiện có.

Hàm feasible ($S \cup x$) kiểm tra tính chấp nhận được của lời giải bộ phận ($S \cup x$).

void greedy

(*Giả sử *C* là tập các ứng cử viên*)

{

S={} ; (**S* là lời giải cần xây dựng của thuật toán *)

while (*C*<>0) and not solution(*S*)

{

x←select(*C*) ;

C=*C*\x ;

if feasible ($S \cup x$)

S:=*S*$\cup x$;

}

if solution(*S*)

return *S*;

}

Thuật toán tham lam tìm được lời giải tốt có thể chấp nhận; thời gian thực hiện nhanh. Thuật toán đơn giản, dễ cài đặt. Thuật toán này có tính ứng dụng trong các bài toán có kích thước lớn.

Ví dụ 11.12.

Trong mặt phẳng hệ trục tọa độ OXY cho *n* điểm (đỉnh), giữa các cặp đỉnh có thể có các đoạn thẳng (cạnh) nối chúng hoặc không. Hãy tô màu các đỉnh này; mỗi đỉnh một màu sao cho không có bất kỳ hai đỉnh nào chung một cạnh được tô cùng một màu và số màu được tô là ít nhất có thể.

Có nhiều thuật toán dạng tham lam để giải quyết bài toán tô màu, sau đây là một thuật toán trong số đó.

Bước 1:

Bậc của mỗi đỉnh là số cạnh kề với đỉnh đó; sắp xếp các đỉnh theo bậc giảm dần.

Số màu=1;

Bước 2:

Dùng màu *Số màu* tô cho đỉnh có bậc cao nhất chưa được tô màu và các đỉnh khác còn lại chưa được tô màu mà có thể tô được (các đỉnh được duyệt theo thứ tự đã được sắp).

Bước 3:

Nếu tất cả các đỉnh đều được tô màu thì *Số màu* chính là kết quả cần tìm; dừng thuật toán. Nếu không gán *Số màu= Số màu+1* và quay lại *bước 2.*

Ví dụ 11.13: **Bài toán tập các đoạn thẳng không giao nhau**

Đầu vào: Cho họ các đoạn thẳng mở $C = \{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$

Đầu ra: Tập các đoạn thẳng không giao nhau có lực lượng lớn nhất.

Ví dụ thực tế: Bài toán xếp thời gian biểu cho các hội thảo. Bài toán lựa chọn hành động. Bài toán phục vụ khách hành trên một máy.

Thuật toán 1:

Sắp xếp các đoạn thẳng theo thứ tự tăng dần của đầu mút trái. Bắt đầu từ tập *S* là tập rỗng, ta lần lượt bổ sung các đoạn trong danh sách theo thứ tự đã sắp xếp vào *S* nếu nó không có điểm chung với bất cứ đoạn nào trong *S*.

Độ phức tạp của thuật toán là O(nlog n).

Thuật toán 2:

Sắp xếp các đoạn thẳng theo thứ tự không giảm của độ dài. Bắt đầu từ tập *S* là tập rỗng, ta lần lượt bổ sung các đoạn thẳng theo thứ tự đã sắp vào *S* nếu nó không có điểm chung với bất cứ đoạn nào trong *S*.

Thuật toán 3:

Sắp xếp các đoạn thẳng theo thứ tự không giảm của đầu mút phải. Bắt đầu từ tập *S* là tập rỗng, ta lần lượt xét các đoạn trong danh sách theo thứ tự đã sắp vào *S* nếu nó không có điểm chung với bất cứ đoạn nào trong *S*.

Thuật toán 3 cho lời giải tối ưu của bài toán về các đoạn thẳng không giao nhau.

Ví dụ 11.14: Bài toán ba lô

Có n đồ vật. Đồ vật i có trọng lượng w_i và giá trị c, i=1..n. Cần tìm cách chất các đồ vật này vào ba lô có dung lượng là b sao cho tổng trọng lượng của các đồ vật được chất vào túi là không quá *b*, đồng thời tổng giá trị của chúng là lớn nhất.

Ký hiệu $C=\{1,2,\ldots,n\}$ tập chỉ số các đồ vật.

Thuật toán 1:

Sắp xếp theo thứ tự không tăng của giá trị.

Lần lượt xét các đồ vật theo thứ tự đã sắp, và chất đồ vật đang xét vào túi nếu như dung lượng còn lại ba lô đủ chứa nó (tức là tổng trọng lượng của các đồ vật đã xếp vào túi và trọng lượng của đồ vật đang xét là không vượt quá *b*).

Thuật toán tham lam này không cho lời giải tối ưu.

Thuật toán 2:

Sắp xếp các đồ vật theo thứ tự không giảm của trọng lượng.

Lần lượt xét các đồ vật theo thứ tự đã sắp , và chất đồ vật đang xét vào túi nếu như dung lượng còn lại ba lô đủ chứa nó (tức là tổng trọng lượng của các đồ vật đã xếp vào túi và trọng lượng của đồ vật đang xét là không vượt quá *b*).

Thuật toán tham lam này không cho lời giải tối ưu.

Thuật toán 3:

Sắp xếp các đồ vật theo thứ tự không tăng của giá trị một đơn vị trọng lượng (c_i/w_i), lần lượt xét các đồ vật theo thứ tự đã sắp, và chất đồ vật đang xét vào túi nếu như dung lượng còn lại ba lô đủ chứa nó (tức là tổng trọng lượng của các đồ vật đã xếp vào túi và trọng lượng của đồ vật đang xét là không vượt quá *b*).

Thuật toán tham lam này không cho lời giải tối ưu.

Greedy3 cũng không cho lời giải tối ưu.

Ví dụ 11.15. Bài toán người du lịch (Traveling Saleman Problem-TSP)

Cho *n* thành phố (được đánh số từ 1 đến *n*), biết ma trận chi phí đi lại giữa các thành phố là C_{ij}.

người du lịch xuất phát từ một thành phố bất kỳ, muốn đi qua tất cả các thành phố, mỗi thành phố đúng một lần rồi quay về thành phố xuất phát. Cách đi như vậy gọi là một hành trình. Chi phí của hành trình được tính như là tổng các chi phí của các đoạn đường của nó.

Cần tìm hành trình với chi phí nhỏ nhất

Thuật toán tham lam giải bài toán TSP

Thuật toán GTS1 (Greedy Traveling Saleman)

Input: n là số thành phố, u là đỉnh xuất phát u và c ,là ma trận chi phí.

Output: tour (thứ tự các thành phố đi qua; tính cả thanh phố xuất phát ở cuối của hành trình).

cost – chí phí ứng với tour tìm được

```
v=u;
tour={u};
cost=0;
for i=1 to n
{
        đặt w là thành phố kề sau thành phố v.
        tour=tour + {w};
        cost=cost+c[v,w]
        v=w;
        }
tour=tour + {u};
cost=cost+c[v,u]
```

Thuật toán GTS2 (Greedy Traveling Saleman)

Input: n, c, p,v_i (i = 1..p) // v_i là các thành phố cho trước hoặc cũng có thể được chọn ngẫu nhiên trong tập 1..p

Output: besttour, bestcost

bestcost=0;

besttour={}

for i=1 to p

{

GTS1(v_k); // suy ra được tour(v_k) và cost(v_k)

If cost(v_k)<bestcost

{

bestcost=cost(v_k)

besttour=tour(v_k)

}

}

BÀI TẬP

BT11-1

Cho một bảng vuông kích thước 100×100 gồm 10.000 ô vuông có cạnh bằng đơn vị. Một đường đi giữa hai ô vuông A và B là dãy các ô vuông kề nhau mà các ô đầu và cuối của dãy là A và B. Chiều dài của đường đi từ A đến B là số các ô vuông của dãy tạo thành đường đi từ A đến B (kể cả A và B). Hai ô vuông gọi là kề nhau nếu có chung cạnh hay chung đỉnh. Vậy một ô vuông không nằm trên cạnh của bảng vuông sẽ có 8 ô vuông của bảng vuông kề với nó.

Nhiệm vụ: *Cho biết chỉ số dòng và cột của hai ô vuông A và B, tìm chiều dài đường đi ngắn nhất từ A đến B.*

Dữ liệu: *Cho trong tập tin văn bản SROAD.INP, gồm 4 số nguyên dương trên cùng một dòng, lần lượt là chỉ số dòng và cột của hai ô vuông A và B.*

Kết quả: *Cho trong tập tin văn bản SROAD.OUT, gồm một số nguyên duy nhất là chiều dài của đường đi ngắn nhất từ A đến B.*

Ví dụ 1:

SROAD.INP	***SROAD.OUT***
10 11 9 11	*2*

Ví dụ 2:

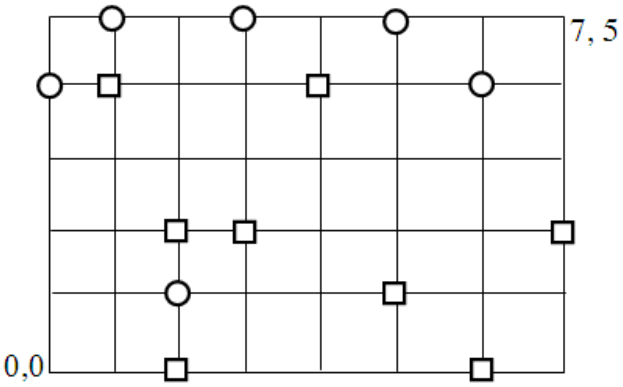
SROAD.INP	***SROAD.OUT***
4 99 100 1	*99*

BT11-2.

Trong các cuộc kháng chiến chống xâm lược, cha ông ta đã xây dựng các địa đạo rất lớn dưới lòng đất với các tuyến đường giao thông ngầm chằng chịt, vừa bảo đảm an toàn, vừa giữ bí mật tuyệt đối. Trong địa đạo này giao thông đi lại phải tuân thủ các qui định chặt chẽ, tất cả mọi người đều phải di chuyển dọc theo các tuyến đường và thực hiện nghiêm ngặt các chỉ dẫn giao thông trên đường.

Một trong các địa đạo như vậy bao gồm N đường dọc và M đường ngang được mô tả như một lưới ô vuông kích thước N x M. Các đường ngang đánh số từ 0 đến M-1 từ dưới lên trên, các đường dọc đánh số từ 0 đến N-1 từ trái sang phải. Tại một số vị trí giao giữa các đường người ta đặt

các biển chỉ dẫn dạng □ hoặc O với ý nghĩa như sau: khi di chuyển theo các đường tại các nút giao thông, nếu gặp chỉ dẫn □ thì bắt buộc rẽ trái, nếu gặp chỉ dẫn O thì bắt buộc rẽ phải, còn nếu không có chỉ dẫn thì phải đi thẳng. Sơ đồ sau cho ta một hình ảnh các đường của địa đạo và các chỉ dẫn.



Bạn có nhiệm vụ dẫn một đoàn khách tham quan đi theo các đường của địa đạo, xuất phát từ vị trí (0,0) và tuân thủ chỉ dẫn tại các nút giao thông. Từ vị trí ban đầu bạn có thể đi theo một trong hai hướng (ngang hoặc dọc). Đường đi của bạn sẽ dừng lại nếu xảy ra một trong hai tình huống sau:

-Không thể đi tiếp được nữa.

-Nút giao thông sắp đến theo hành trình là nút giao thông đã đi qua.

Độ dài của đoạn đường đã đi là tổng số các nút giao thông đã đi qua kể cả vị trí xuất phát và vị trí kết thúc.

Yêu cầu: *tính độ dài của đoạn đường có thể đi được trong địa đạo.*

Dữ liệu: *vào từ file văn bản PIPE.INP có dạng sau:*

- *Dòng đầu tiên chứa 3 số tự nhiên là N, M và K với N, M là kích thước của lưới mô tả địa đạo. N, M < 100. K là số lượng các chỉ dẫn có tại các nút giao thông trong địa đạo. K < 1000. Các số cách nhau bởi dấu cách.*
- *K dòng tiếp theo mô tả tọa độ và tính chất của các chỉ dẫn tại các nút giao thông tương ứng. Mỗi dòng bao gồm 3 số nguyên dạng X Y Z. Ở đây X, Y là toạ độ của vị trí biển chỉ dẫn (0 ≤ X ≤ N-1, 0 ≤ Y ≤ M-1), Z nhận giá trị 0 hoặc 1 với ý nghĩa: 0 là chỉ dẫn rẽ phải và 1 là chỉ dẫn rẽ trái. Các số cách nhau bởi dấu cách.*

Kết quả: *ghi ra file văn bản PIPE.OUT bao gồm một dòng chứa đúng 2 số tự nhiên theo thứ tự tăng dần là độ dài của hai đường đi trong địa đạo xuất phát từ vị trí ban đầu.*

Ví dụ:

PIPE.INP	PIPE.OUT
8 6 14 0 4 0 1 4 1 1 5 0 2 0 1 2 1 0 2 2 1 3 2 1 3 5 0 4 4 1 5 1 1 5 5 0 6 0 1 6 4 0 7 2 1	13 19

BT11-3.

Chèn các phép toán "+" "-" "" "/" (cộng, trừ, nhân, chia) vào chuỗi 123456789 sao cho sau khi tính toán ta được kết quả biểu thức bằng M (nếu có lời giải).*

BT11-4.

Cho một số nguyên dương n (n≤30), hãy tìm tất cả các cách phân tích số n thành tổng của các số nguyên dương, các cách phân tích là hoán vị của nhau chỉ tính là một cách.

BT11-5.

a. *Cho dãy n phần tử. Tìm tất cả các dãy con có tổng bằng S cho trước.*

b. *Cho dãy n phần tử. Tìm một cách chia n số trên thành hai phần sao cho tổng các phần tử của mỗi phần chênh lệch nhau là ít nhất.*

c. *Cho dãy n phần tử. Tìm các bộ k số có tổng bằng m cho trước (k,m là các số nguyên dương).*

d. *Cho dãy n phần tử. Tìm một tập con có tổng lớn nhất không lớn hơn m cho trước (m là số nguyên dương).*

BT11-6.

Trong mặt phẳng tọa độ OXY, cho n điểm (x_i,y_i), i=1..n. Giữa các cặp điểm có thể có các đoạn nối hoặc không.

a. *Hãy kiểm tra xem giữa hai điểm u,v (1<=u,v<=n) có đường nối với nhau hay không (trực tiếp hoặc gián tiếp).*

b. *n điểm này được chia thành ít nhất bao nhiêu cụm mà các điểm trong mỗi cụm đều có thể kết nối được với nhau (trực tiếp hay gián tiếp).*

c. *Tìm tất cả các cách nối n điểm này với nhau sao cho các đoạn không tạo thành một chu trình nào.*

BT11-7.

Bài toán người du lịch (Traveling Saleman Problem-TSP)
Cho n thành phố (được đánh số từ 1 đến n), biết ma trận chi phí đi lại giữa các thành phố là C_{ij}.
người du lịch xuất phát từ một thành phố bất kỳ, muốn đi qua tất cả các thành phố, mỗi thành phố đúng một lần rồi quay về thành phố xuất phát. Cách đi như vậy gọi là một hành trình. Chi phí của hành trình được tính như là tổng các chi phí của các đoạn đường của nó.
Cần tìm hành trình với chi phí nhỏ nhất.

BT11-8.

Có n cuộc họp, cuộc họp thứ i bắt đầu vào thời điểm a_i và kết thúc ở thời điểm b_i. Do chỉ có một phòng hội thảo nên hai cuộc họp bất kì sẽ được cùng bố trí phục vụ nếu khoảng thời gian làm việc của chúng chỉ giao nhau tại đầu mút.
Hãy bố trí phòng họp để phục vụ được nhiều cuộc họp nhất.

BT11-9.

Cho dãy n số nguyên. Tìm dãy con liên tiếp có tổng lớn nhất. Cho biết giá trị tổng lớn nhất tìm được này.

BT11-10.

Cho một ma trận M x N gồm các số tự nhiên. Đường đi trong ma trận là một đường gấp khúc không tự cắt xuất phát từ một ô bất kỳ trong ma trận, sau đó có thể đi theo các hướng: lên trên, xuống dưới, rẽ trái, rẽ phải sao cho các số ghi trong các ô trên đường đi tạo thành một dãy số không giảm. Độ dài của đường đi tính bằng số các ô mà đường đi qua. Lập chương trình tìm đường đi dài nhất trong ma trận.

BT11-11.

Cho n món hàng, món thứ i có khối lượng là a[i] (i=1..n, a[i] là số nguyên). Cần chọn những món hàng nào để bỏ vào một ba lô sao cho tổng khối lượng của các món hàng được chọn là lớn nhất nhưng không vượt quá khối lượng w cho trước. Mỗi món hàng chỉ được chọn đúng một hoặc là không được chọn.

BT11-12.

a. *Trong mặt phẳng tọa độ OXY, cho n điểm có tọa độ x_i,y_i (i=1..n). Hãy tìm một đường nối đi qua tất cả các điểm và có tổng chiều dài các đoạn nối là nhỏ nhất.*

b. *Trong mặt phẳng tọa độ OXY, cho n điểm có tọa độ x_i,y_i (i=1..n). Hãy tìm một đường đi từ điểm u đến điểm v có tổng chiều dài các đoạn nối là ngắn nhất (1<=u,v<=n).*

c. *Trong mặt phẳng tọa độ OXY, cho n điểm có tọa độ x_i,y_i (i=1..n). Hãy tìm một đường đi qua tất cả các điểm mỗi điểm đúng một lần và quay lại điểm khởi đầu.*

BT11-13.

a. *Viết chương trình tính a^n; trong đó a là mảng có n dòng n cột các số nguyên,1<=n<=1000.*

b. *Viết chương trình nhân hai số tự nguyên lớn (mỗi số có hàng ngàn chữ số).*

BT11-14.

Cho mảng n dòng n cột chứa các số nguyên (n<=15). Tìm bộ n số có tổng lớn nhất sao cho n số này không nằm trên cùng dòng, không nằm trên cùng cột.

BT11-15.

Trong mặt phẳng tọa độ OXY cho n điểm có tọa độ x_i,y_i (i=1..n). Tìm một cặp điểm gần nhau nhất bằng thuật toán chia để trị.

BT11-16.

Cho một ma trận vuông A cấp N (N <=50) kiểu integer. Chứa các số đôi một khác nhau từ 1 đến N^2. Bằng cách dùng một số phép biến đổi đối với mảng A thuộc một trong hai loại :

-Đổi chổ hai phần tử thuộc cùng một dòng cho nhau

-Đổi chổ hai phần tử thuộc cùng một cột cho nhau

Hãy biến đổi mảng A về một mảng sao cho tổng các phần tử của mỗi dòng đều bằng nhau.

BT11-17.

Cho N số nguyên H_i ($1 \le N \le 20.000$, $1 \le H_i \le 10.000$) và một số nguyên dương B ($1 \le B \le S < 2.000.000.000$), trong đó S là tổng của N số nguyên H_1, ..., H_N. Tìm số ít nhất các con số trong N số nguyên H_1, ..., H_N sao cho tổng của chúng không nhỏ hơn B.

Dữ liệu: *Vào từ tập tin văn bản TONGSO.INP. Dòng đầu gồm hai số N, B, trên các dòng từ 2 đến N+1, dòng thứ i+1 chứa số H_i.*

Kết quả: *Ghi ra tập tin văn bản TONGSO.OUT số nguyên duy nhất cần tìm.*

Ví dụ:

TONGSO.INP	***TONGSO.OUT***
6 40 *6* *18* *11* *13* *19* *11*	*3*

Giải thích: Tổng của hai số bất kỳ trong 6 số 6, 18, 11, 13, 19, 11 đều nhỏ hơn 40, có 3 số có tổng lớn hơn hay bằng 40, chẳng hạn 18 + 13 + 11 ≥ 40.

ĐỀ THI THAM KHẢO

ĐỀ THI CƠ SỞ LẬP TRÌNH 01

VIẾT CHƯƠNG TRÌNH HOÀN CHỈNH THỰC HIỆN CÁC YÊU CẦU SAU:

Câu 1 (2 điểm)

Liệt kê tất cả các số tự nhiên *k* thỏa mãn đồng thời các điều kiện sau:

-*k* là số có 5 chữ số;

-*k* là số nguyên tố;

-*k* là số đối xứng (*k* là số đối xứng nếu đọc xuôi hay đọc ngược các chữ số của *k* ta đều nhận được một số như nhau. Ví dụ số: 10001 là số đối xứng).

Câu 2 (2 điểm)

Dãy A_n được cho như sau :

$A_1=1$; $A_n=n(A_1+A_2+\ldots+A_{n-1})$

Hãy viết hàm tính A_n theo hai cách: có sử dụng đệ qui và không sử dụng đệ qui.

Câu 3 (3 điểm)

Cho mảng một chiều chứa *n* số nguyên dương và một số nguyên dương *k*.

a. Tìm giá trị lớn nhất của mảng

b. Tìm giá trị lớn thứ *k* của mảng

c. Hãy sắp xếp sao cho các số chẵn về đầu mảng, các số lẻ về cuối mảng

d. Tìm dãy con liên tiếp dài nhất chứa toàn các số nguyên tố.

Ví dụ: *n*=9; 5,5,7,8,9,7,5,3,2

Kết quả là:

a. 9

b. *k*=5; kết quả là 3

c. 8,2,5,5,7,9,7,5,3

d. 7,5,3,2

Câu 4 (3 điểm)

Cho dãy *n* khối lập phương. Trên mỗi mặt của mỗi khối lập phương ghi một giá trị nguyên từ 1 đến 9.

a. Tổng giá trị của mỗi khối lập phương bằng tổng các giá trị trên các mặt của khối lập phương đó. Hãy tìm tổng lớn nhất.

b. Hãy cho biết có bao nhiêu khối lập phương có giá trị của các mặt là các số khác nhau ?

c. Hãy cho biết giá trị nào xuất hiện nhiều lần nhất ? Bao nhiêu lần ?

HẾT

ĐỀ THI CƠ SỞ LẬP TRÌNH 02

VIẾT CHƯƠNG TRÌNH HOÀN CHỈNH THỰC HIỆN CÁC YÊU CẦU SAU:

Câu 1 (3 điểm)

Viết chương trình nhập vào số nguyên dương *n* và số thực *x*. Hãy tính giá trị của biểu thức sau đây:

$$S(x,n) = 1 - \frac{x}{1+2} + \frac{x^2}{2+3} - \ldots + (-1)^n \frac{x^n}{n+(n+1)}$$

Câu 2 (3 điểm)

Dãy A_n được cho như sau :

$A_1=1$; $A_{2n}=n + A_n + 2$;$A_{2n+1}=n^2 + A_n.A_{n+1} +1$

a. Viết hàm tính A_n bằng đệ quy

b. Viêt hàm tính A_n bằng cách không sử dụng đệ quy.

Câu 3 (4 điểm)

Cho dãy *a* gồm *n* phần tử là các số nguyên. Hãy viết các hàm thực hiện các công việc sau:

a. Đếm số các đoạn con liên tiếp tăng.

b. Đếm tần số xuất hiện của các giá trị.

Ví dụ: $n = 9$

4 7 7 2 2 34 34 7 7

Thì kết quả câu a là: 3

Kết quả câu b là: 4 xuất hiện 1

7 xuất hiện 4

2 xuất hiện 2

34 xuất hiện 2

HẾT

ĐỀ THI CƠ SỞ LẬP TRÌNH 03

VIẾT CHƯƠNG TRÌNH HOÀN CHỈNH THỰC HIỆN CÁC YÊU CẦU SAU:

Câu 1 (3 điểm)

Giả sử $n \geq 1$ và x là số thực. Hãy viết chương trình tính giá trị của biểu thức sau:

$$S(n,x) = \frac{x}{1} - \frac{x^2}{1+\frac{1}{2}} + \frac{x^3}{1+\frac{1}{2}+\frac{1}{3}} - \ldots + (-1)^{n-1} \frac{x^n}{1+\frac{1}{2}+\ldots+\frac{1}{n}}$$

Câu 2 (3 điểm)

Nhập vào số nguyên dương *n*.

Hãy thực hiện các công việc sau:

a. Đếm số lượng số nguyên tố nhỏ hơn hoặc bằng *n*.

b. Tìm số nguyên tố thứ *n* (dãy các số nguyên tố: 2,3,5,7,11,...)

c. Tìm số nguyên tố gần *n* nhất (có thể lớn hơn hoặc nhỏ hơn *n*).

Ví dụ: *n*=8 thì kết quả là:

a. 4

b. 19

c. 7

Câu 3 (4 điểm)

Cho dãy *n* số nguyên dương $a_o, a_1, \ldots, a_{n-1}$. Hãy viết các hàm thực hiện các công việc sau:

a. Đếm xem trong mảng có bao nhiêu số nguyên tố có hai chữ số?

b. Biến đổi các số của mảng về mảng toàn số nguyên tố theo nguyên tắc: Các số không phải là số nguyên tố thì được biến đổi thành số nguyên tố gần nó nhất

Ví dụ:

với *n*=8 và dãy 42, 6, 5, 7, 15, 17, 2010, 13

kết quả câu a là: 2

kết quả câu b là: 43, 7, 5, 7, 17, 17, 2011, 13

HẾT

ĐỀ THI CƠ SỞ LẬP TRÌNH 04

VIẾT CHƯƠNG TRÌNH HOÀN CHỈNH THỰC HIỆN CÁC YÊU CẦU SAU:

Câu 1 (3 điểm)

Cho S sẽ được tính theo công thức S=1+3+5+...+(2n+1) với $n \geq 0$ và số nguyên M.

Hãy tìm giá trị n nhỏ nhất sao cho $S > M$ với M được cho trước.

Ví dụ: $M = 20$ thì n tìm được là 4.

Câu 2 (3 điểm)

Với mỗi $n \geq 1$, dãy số Y_n được định nghĩa như sau:

$Y_1 = 1$, $Y_2 = 2$, $Y_3 = 3$. $Y_n = Y_{n-1} (5Y_{n-2} + 6Y_{n-3})$ với mọi $n \geq 4$

a. Viết hàm tính Y_n bằng phương pháp đệ quy.

b. Viết hàm tính Y_n bằng cách không dùng đệ quy và cũng không dùng biến mảng để lưu giá trị tạm.

Câu 3 (4 điểm)

Cho dãy n số nguyên dương. Hãy viết các hàm thực hiện các công việc sau:

a. Sắp xếp các số nguyên tố trong dãy tăng, còn các số khác vẫn giữ nguyên giá trị và vị trí.

b. Xóa tất cả các số nguyên tố trong dãy.

Ví dụ : n=8

12 2 7 11 3 100 5 3

Kết quả câu a là : 12 2 3 3 5 100 7 11

Kết quả câu b là : 12 100

HẾT

ĐỀ THI CƠ SỞ LẬP TRÌNH 05

CÂU 1 (1.0 ĐIỂM)

Cho biết kết quả của chương trình sau:

```
#include <iostream.h>
int test(int a, int &b, int &c);
void main()
{       int a=3,b=2,c=1;
        cout<<test(a,b,c)+a+b+c;
}
int test(int a, int &b, int &c)
{       a=a+1;b=b+2;c=c+3;
        return a+b+c;
}
```

CÂU 2 (3 điểm)

Với mỗi $n \geq 1$, dãy số Y_n được định nghĩa như sau :

$Y_1 = 1$, $Y_2 = 2$, $Y_3 = 3$. $Y_n = Y_{n-1} + 2Y_{n-2} + 3Y_{n-3}$ với mọi $n \geq 4$

a.Tính Y_7.

b.Hãy viết hàm tính Y_n bằng phương pháp đệ quy.

c.Hãy viết hàm tính Y_n bằng cách không sử dụng đệ quy mà cũng không sử dụng biến mảng để lưu trữ biến tạm.

Câu 3 (4 điểm)

Cho một dãy gồm n phần tử nguyên dương. Viết các hàm thực hiện các công việc sau:

a.Tìm giá trị nhỏ nhất của dãy.

b.Hãy đếm xem trong dãy có bao nhiêu số hoàn chỉnh ? (số hoàn chỉnh là số bằng tổng các ước số thực sự của nó, chẳng hạn 6 là số hoàn chỉnh vì 6 = 1 + 2 + 3).

c.Tìm số nguyên dương lớn nhất là ước số của tất cả các số của dãy.

d.Tìm chiều dài của dãy con tăng dài nhất.

CÂU 4 (2 điểm)

Hãy tìm các số p,q ($0<p<q<n$) sao cho tổng các ước số thực sự của p bằng q và tổng các ước số thực sự của q bằng p. Tìm một thuật toán hiệu quả khi n lớn.

Hết

ĐỀ THI CƠ SỞ LẬP TRÌNH 06

CÂU 1 (3 điểm)

Giả sử $n \geq 1$ và x là số thực. Hãy viết hàm tính giá trị của biểu thức sau đây :

Tính tổng $S(x,n) = x + \frac{x^2}{1+2} + \frac{x^3}{1+2+3} + ... + \frac{x^n}{1+2+3+...+n}$

CÂU 2 (2 điểm)

Dãy A_n được cho như sau :

$A_1=1$

$A_{2n}=n + A_n + 2$

$A_{2n+1}=n^2 + A_n.A_{n+1} +1$

a. Viết hàm tính A_n bằng đệ quy

b. Viêt hàm tính A_n bằng cách không sử dụng đệ quy.

CÂU 3 (2 điểm)

Tìm các dãy nhị phân chiều dài n, với n là số nguyên dương.

Ví dụ n=3 thì có kết quả là: 0 0 0; 0 0 1; 0 1 0; 0 1 1; 1 0 0; 1 0 1; 1 1 0; 1 1 1;

CÂU 4 (3 điểm)

Cho dãy gồm n khối lập phương. Trên mỗi mặt của khối lập phương ghi một giá trị nguyên từ 1 đến 9.

a.Hãy cho biết có bao nhiêu khối lập phương có giá trị của các mặt là các số khác nhau ?

b.Hãy cho biết giá trị nào xuất hiện nhiều lần nhất ? Bao nhiêu lần ?

Hết

ĐỀ THI CƠ SỞ LẬP TRÌNH 07

CÂU 1 (3 điểm)

Giả sử $n \geq 1$ và x là số thực. Hãy viết hàm tính giá trị của biểu thức sau đây :

$$S(x,n) = -x + \frac{x^2}{1+2} - \frac{x^3}{1+2+3} + ... + (-1)^n \frac{x^n}{1+2+3+...+n}$$

CÂU 2 (3 điểm)

Cho dãy số x_n được định nghĩa như sau:

$x_0 = 1$;

$x_1 = 2$;

$x_n = nx_0 + (n - 1)x_1 + \ldots + x_{n-1}$

a. Tính giá trị của x_5.

b. Với $n \geq 0$, tính x_n bằng cách có sử dụng đệ qui.

CÂU 3 (4 điểm)

Cho dãy n số nguyên dương $a_1, a_2, ..., a_n$. Hãy viết hàm thực hiện các công việc sau:

a. Tìm chiều dài của đoạn con liên tiếp dài nhất chứa toàn số chẵn. Nếu không có thì trả về giá trị 0

b. Tìm các số nguyên tố sao cho khi đảo ngược các chữ số của nó ta cũng thu được một số nguyên tố (ví dụ số 13, số 149,...).

Hết

ĐỀ THI CƠ SỞ LẬP TRÌNH 08

CÂU 1 (3 điểm)

Viết chương trình nhập một thông tin ngày (ngày/tháng/năm) - là các số nguyên dương). Hãy cho biết ngày đó là ngày thứ bao nhiêu trong năm ? (ngày 1 tháng 1 là ngày thứ nhất của năm).

Ví dụ: Ngày 3/2/2012 là ngày thứ 34 trong năm.

CÂU 2 (3 điểm)

Cho S_n=1.2 + 2.3.4 + 3.4.5.6 +...+ n.(n+1)...(2n)

Hãy viết chương trình hoàn chỉnh để tính S_n.

Ví dụ: với n =4 thì kết quả S_n= 7106

CÂU 3 (4 điểm)

Cho mảng một chiều gồm n phân số, trong đó tử số và mẫu số là các số nguyên. Hãy viết chương trình thực hiện các công việc sau:

a.Hãy tìm phân số có giá trị lớn nhất

b.Hãy sắp xếp các phân số theo chiều tăng dần.

Hết

ĐỀ THI CƠ SỞ LẬP TRÌNH 09

CÂU 1 (2 điểm)

Ngày hiện tại được lưu vào ba biến nguyên: dd/mm/yy. Giả sử các giá trị này là đã hợp lệ - không cần kiểm tra.

Hãy viết hàm xác định ngày mai (tức là ngày kề sau của ngày dd/mm/yy)

CÂU 2 (2 điểm)

Cho mảng một chiều gồm n số thực a_0,a_1,a_{n-1}. Hãy viết các hàm thực hiện các yêu cầu sau:

a.Sắp xếp theo thứ tự tăng dần

b.Sắp xếp theo trị tuyệt đối tăng dần

c.Tìm các cặp số nguyên tố sinh đôi trong dãy (là các cặp số mà khoảng cách giá trị của chúng bằng 2, ví dụ 17 và 19 là cặp số sinh đôi).

CÂU 3 (3 điểm)

Tính tổng S= $1+\frac{2}{1+\frac{1}{2}}+\frac{3}{1+\frac{1}{2}+\frac{1}{3}}+...+\frac{n}{1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n}}$ (n nguyên dương).

Dãy số {hn} được cho theo quy tắc h1=1; h2=2; h3=3 và hn= 23*h(n-1) + 4*h(n-2) + 2016 với mọi n>3. Hãy tìm số hạng thứ n của dãy{hn}.

Đếm xem trong các số nguyên từ 1 triệu đến 1 tỷ có bao nhiêu số đối xứng ? (số đối xứng là số mà khi viết các chữ số của nó theo chiều từ trái qua phải hay từ phải qua trái thì thu được cùng kết quả. Ví dụ 1425241, 73422437 là các số đối xứng).

CÂU 4 (3 điểm)

Cho một bảng a có m dòng n cột; mỗi phần tử của bảng là một số nguyên dương.

a. Tìm giá trị lớn nhất của bảng a.
b. Tìm tổng các phần tử của bảng a.
c. Tìm tổng các phần tử nằm trên các dòng, các cột ngoài cùng của bảng a (dòng đầu, dòng cuối, cột đầu, cột cuối; trong đó các phần tử nằm ở các góc của bảng chỉ được tính một lần vào tổng).
d. Đếm số lượng số nguyên tố có trên mỗi dòng của bảng a.
e. Từ bảng a, hãy tạo bảng b có m dòng n cột, biết bij bằng tổng các chữ số của số aij tương ứng.

Hết

ĐỀ THI CƠ SỞ LẬP TRÌNH 10

CÂU 1 (3 điểm)

Viết chương trình nhập số tự nhiên *n*. Hãy in ra chữ số thứ *n* của dãy vô hạn các số chính phương 149162536496481100121….

CÂU 2 (3 điểm)

Với mỗi số nguyên $n \geq 1$, số X_n được tính như sau :

$X_1=1$, $X_2=1$

$X_n=X_{n-1} + (n-1)X_{n-2}$ với $n \geq 3$

Viết hàm tính X_n bằng phương pháp đệ quy.

Viết hàm tính X_n không dùng đệ quy.

CÂU 3 (4 điểm)

Cho mảng gồm *n* số nguyên dương. Hãy thực hiện các yêu cầu sau:

a.Kiểm tra xem trong mảng có tồn tại số chính phương nào hay không ? Nếu có trả về giá trị 1, nếu không có trả về giá trị 0.

b.Tìm k giá trị khác nhau lớn nhất của mảng.

c.Hãy tìm hạng của các phần tử.

Hết

ĐỀ KIỂM TRA THỰC HÀNH CƠ SỞ LẬP TRÌNH 01

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Nhập từ bàn phím số nguyên dương *n*.

CÂU 1 (2.0 điểm)

Có bao nhiêu số nguyên tố nhỏ hơn hoặc bằng *n* ?

CÂU 2 (2.0 điểm)

Số *n* có bao nhiêu ước số nguyên dương ?

CÂU 3 (2.0 điểm)

Tính tổng các chữ số của số *n.*

CÂU 4 (1.5 điểm)

Xuất các chữ số của số *n* theo chiều ngược lại.

CÂU 5 (1.5 điểm)

Số nguyên tố Palindrome là số nguyên tố mà khi viết các chữ số của nó theo chiều từ trái qua phải hay từ phải qua trái thì ta cùng được một kết quả. Ví dụ 2, 3, 151, 19891, 19991 là các số nguyên tố Palindrome. Tìm các số nguyên tố Palindrome nhỏ hơn hoặc bằng *n*.

CÂU 6 (1.0 điểm)

Số nguyên tố đối xứng là một số nguyên tố bằng trung bình cộng của hai số nguyên tố liền trước và liền sau nó. Với p_n là số nguyên tố thứ *n*, một số nguyên tố là đối xứng khi thoả: $p_n = \frac{p_{n-1} + p_{n+1}}{2}$.

Ví dụ 5 là số nguyên tố đối xứng, 7 không phải là số nguyên tố đối xứng.

Tìm các số nguyên tố đối xứng nhỏ hơn hoặc bằng *n*.

Xuất kết quả tìm được của các yêu cầu trên lên màn hình.

Ví dụ:

n=53 thì kết quả là:

CÂU 1: 16

CÂU 2: 2

CÂU 3: 8

CÂU 4: 35

CÂU 5: 2 3 5 7 11

CÂU 6: 5 53

Hết

ĐỀ KIỂM TRA THỰC HÀNH CƠ SỞ LẬP TRÌNH 02

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Cho hai tập A và B. Tập hợp A có *n* phần tử, tập hợp B có *m* phần tử; các phần tử là các số nguyên dương (mỗi tập hợp chứa các số khác nhau).

CÂU 1 (2 điểm)

Nhập từ bàn phím các số *n*, $a_0,a_1,\ldots,a_{n-1}$ cho tập hợp *A*, và các số *m*, $b_0,b_1,\ldots,b_{m-1}$ cho tập hợp *B*.

CÂU 2 (2 điểm)

Tìm các phần tử vừa thuộc tập *A* vừa thuộc tập *B*.

CÂU 3 (2 điểm)

Tìm các phần tử ít nhất thuộc tập *A* hoặc tập *B*.

CÂU 4 (2 điểm)

Tìm các phần tử chỉ thuộc tập *A* mà không thuộc tập *B*.

CÂU 5 (2 điểm)

Tìm trong tập *A* các phần tử *x* mà tồn tại ít nhất một cặp phần tử *y,z* thuộc tập *B* sao cho $x = y + z$.

Xuất kết quả tìm được của các yêu cầu trên lên màn hình.

Ví dụ:

CÂU 1:

Nếu dữ liệu nhập là:

A= 5

3 6 5 4 14

B= 6

2 4 10 3 11 19

Thì kết quả các câu tiếp theo là:

CÂU 2: 3 4

CÂU 3: 3 6 5 4 14 2 10 11 19

CÂU 4: 6 5 14

CÂU 5: 6 5 14

Hết

ĐỀ KIỂM TRA THỰC HÀNH CƠ SỞ LẬP TRÌNH 03

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Cho mảng một chiều *a* chứa *n* số nguyên dương. Hãy viết một chương trình hoàn chỉnh thực hiện các công việc sau:

CÂU 1 (2 điểm).

Tìm giá trị lớn nhất, giá trị nhỏ nhất của mảng *a*.

CÂU 2 (2 điểm).

Tìm tổng các chữ số của tất cả các số của mảng *a*.

CÂU 3 (2 điểm).

Tìm số nguyên tố lớn nhất của mảng *a*; nếu mảng *a* không có số nguyên tố nào thì xuất giá trị 0.

CÂU 4 (2 điểm).

Tính tổng các ước số (không kể chính nó) của mỗi số của mảng *a*. Ví dụ 33 có tổng các ước số là 1+3+11=15.

CÂU 5 (2 điểm).

Tìm 3 số lớn nhất khác nhau của mảng.

Bộ test mẫu tham khảo

INPUT

12

102 101 11 33 44 55 66 77 88 97 101 102

OUTPUT

Cau 1: 102 11

Cau 2: 94

Cau 3: 101

Cau 4: 114 1 1 15 40 17 78 19 92 1 1 114

Cau 5: 102 101 97

Hết

ĐỀ KIỂM TRA THỰC HÀNH CƠ SỞ LẬP TRÌNH 04

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Cho dãy *a* gồm *n* số nguyên dương $a_o,a_1,a_2,...,a_{n-1}$. Viết một chương trình hoàn chỉnh thực hiện các công việc sau:

CÂU 1 (2 điểm).

Tính tổng các phần tử trong dãy *a*. Tính trung bình cộng các phần tử trong dãy *a*.

CÂU 2 (2 điểm).

Đếm số lượng số đối xứng có từ hai chữ số trở lên trong dãy *a*. Ví dụ 11, 121, 64446,… là các số đối xứng.

CÂU 3 (2 điểm).

Tìm dãy *b*, biết số b_i là viết đảo ngược của số a_i; ví dụ a_i =5334 thì b_i tương ứng là 4335. Lưu ý rằng số 100 khi viết ngược lại thì được biểu diễn là 1.

CÂU 4 (2 điểm).

Đếm số lượng số của dãy *a* thỏa điều kiện: 2 số liền kề trái/phải với nó là số nguyên tố.

CÂU 5 (2 điểm).

Tìm dãy *c*, biết c_i là số lượng các số nguyên tố *nhỏ hơn* a_i; ví dụ a_i =18 thì c_i tương ứng là 7.

Bộ test mẫu tham khảo

INPUT

12

7 18 101 242 102 101 122 31 5 41 121 102

OUTPUT

Cau 1 : 993 82.75

Cau 2 : 4

Cau 3 : 7 81 101 242 201 101 221 13 5 14 121 201

Cau 4 : 3

Cau 5 : 3 7 25 53 26 25 30 10 2 12 30 26

Hết

ĐỀ KIỂM TRA THỰC HÀNH CƠ SỞ LẬP TRÌNH 05

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Cho mảng một chiều a chứa n số nguyên dương (kiểu unsigned long). Hãy thực hiện các công việc sau:

CÂU 1 (2 điểm).

Tìm giá trị trung bình cộng các số của mảng a.

CÂU 2 (2 điểm).

Tìm tổng các chữ số của tất cả các số của mảng a.

CÂU 3 (2 điểm).

Tìm số nguyên tố nhỏ nhất của mảng a; nếu mảng a không có số nguyên tố nào thì xuất giá trị 0.

CÂU 4 (2 điểm).

Cho biết số lượng các ước số của từng số của mảng a. 6 có 4 ước số là 1,2,3,6.

CÂU 5 (2 điểm).

Biến đổi mỗi số của mảng a về số nguyên tố nhỏ nhất lớn hơn hoặc bằng nó . Thời gian thực hiện câu 5 tối đa là 60 giây.

Bộ test mẫu tham khảo.

INPUT

10

101 11 33 44 55 66 10 496 97 28

OUTPUT

Cau 1: 94.1

Cau 2: 86

Cau 3: 11

Cau 4: 2 2 4 6 4 8 4 10 2 6

Cau 5: 101 11 37 47 59 67 11 499 97 29

Hết

ĐỀ THI KỸ THUẬT LẬP TRÌNH 01

THỜI GIAN LÀM BÀI : 90 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

VIẾT CÁC CHƯƠNG TRÌNH HOÀN CHỈNH GIẢI CÁC BÀI TOÁN SAU:

CÂU 1 (1 điểm)

Tìm các dãy hoán vị của tập gồm n phần tử {1,2,3,…,n} với n là số nguyên dương.

Ví dụ n=3 thì kết quả là: 1 2 3; 1 3 2; 2 1 3; 2 3 1; 3 1 2; 3 2 1;

CÂU 2 (2 điểm)

Cho một chuỗi s chứa tối đa 256 ký tự gồm chữ cái và khoảng trắng, các từ cách nhau đúng một khoảng trắng, đầu và cuối chuỗi s không có khoảng trắng.

Tạo chuỗi con chứa các từ thứ 2,3,4 của chuỗi s; đầu và cuối chuỗi con này không có khoảng trắng, giữa các từ có một khoảng trắng.

Đảo ngược các từ trong chuỗi s, trong đó thứ tự các ký tự trong mỗi từ giữ nguyên.

Ví dụ:

s=”what good is money if it can not buy happiness”

Kết quả là:

good is money

happiness buy not can it if money is good what

CÂU 3 (3 điểm)

Cho mảng hai chiều a gồm m dòng n cột, các phần tử là các số nguyên dương. Hãy tạo mảng hai chiều b từ mảng a, sao cho $b_{ij} = a_{ij} \times k_i$ với k_i là giá trị trung bình cộng của số lớn nhất và số nhỏ nhất trên dòng i. Giá trị k_i được lấy đến phần nguyên.

Ví dụ:

m=3, n=4

45	11	22	10
31	14	23	4
13	12	11	14

kết quả là:

1215	297	594	270
527	238	391	68
156	144	132	168

CÂU 4 (2 điểm)

Xây dựng cấu trúc cho bài toán quản lý các số thuê bao điện thoại bằng cấu trúc mảng. Mỗi phần tử chứa các thông tin: họ tên (chuỗi, tối đa 32 ký tự), địa chỉ (chuỗi, tối đa 48 ký tự), số điện thoại (chuỗi, tối đa 11 ký tự).

Viết các hàm nhập, xuất và tìm họ tên, địa chỉ của chủ thuê bao khi biết số điện thoại là X.

CÂU 5 (2 điểm)

Cho file văn bản songuyen.inp chứa các số nguyên dương; các số cách nhau ít nhất một khoảng trắng.

Tìm số lớn nhất và số lớn thứ nhì trong file songuyen.inp

Số nguyên tố *x* được gọi là số nguyên tố đối xứng nếu nó bằng trung bình cộng của hai số nguyên tố kề trước (ký hiệu là *p*) và kề sau của nó (ký hiệu là *q*), tức là *p*+*q*=2**x*. Đếm xem trong file songuyen.inp có bao nhiêu số nguyên tố đối xứng ?

Ghi kết quả vào file ketqua.out gồm hai dòng:

-Dòng thứ nhất ghi số lớn nhất và số lớn thứ nhì.

-Dòng thứ hai ghi số lượng số nguyên tố đối xứng tìm được.

Ví dụ:

songuyen.inp

4

5 62 10 62

53 47 8 53 62 3

ketqua.out

62 53

3

Hết

ĐỀ THI KỸ THUẬT LẬP TRÌNH 02

THỜI GIAN LÀM BÀI : 90 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

VIẾT CÁC CHƯƠNG TRÌNH HOÀN CHỈNH GIẢI CÁC BÀI TOÁN SAU:

CÂU 1 (1 điểm)

Tìm các dãy nhị phân chiều dài *n*, với *n* là số nguyên dương.

Ví dụ *n*=3 thì có kết quả là: 0 0 0; 0 0 1; 0 1 0; 0 1 1; 1 0 0; 1 0 1; 1 1 0; 1 1 1;

CÂU 2 (2 điểm)

Cho một chuỗi *s* chứa tối đa 256 ký tự gồm chữ cái và khoảng trắng, các từ của chuỗi *s* cách nhau đúng một khoảng trắng, đầu và cuối chuỗi *s* không có khoảng trắng.

Tìm một từ dài nhất của chuỗi *s*.

Xem mỗi từ trong chuỗi *s* là một chuỗi con. Hãy sắp xếp các chuỗi con tăng dần từ trái qua phải.

Ví dụ:

s=”what good is money if it can not buy happiness”

Kết quả là:

happiness

buy can good happiness if is it money not what

CÂU 3 (3 điểm)

Cho mảng hai chiều *a* gồm *m* dòng *n* cột, các phần tử là các số nguyên dương. Hãy tạo mảng hai chiều *b* từ mảng *a*, sao cho b_{ij} là số nguyên tố nhỏ nhất lớn hơn hoặc bằng a_{ij} tương ứng.

Ví dụ:

m=3, n=4

45 11 22 10

31 14 23 4

14 12 11 14

Kết quả là:

47 11 23 11

31 17 23 5

17 13 11 17

CÂU 4 (2 điểm)

Thông tin cho *n* thí sinh trong một kỳ thi tuyển sinh cao học như sau:

- Số báo danh (chuỗi, tối đa 8 ký tự)

- Họ và tên (chuỗi, tối đa 32 ký tự)
- Điểm ngoại ngữ (số- Điểm môn cơ sở (số nguyên)
- Điểm môn chuyên ngành (số nguyên)

Hãy thực hiện các công việc sau đây :

a. Hãy mô tả cấu trúc cho bài toán.

b. Viết hàm nhập thông tin cho n thí sinh

c. Viết hàm tìm những thí sinh có điểm thi môn ngoại ngữ ≥ 5 và có tổng điểm môn cơ sở và môn chuyên ngành ≥ 12.

CÂU 5 (2 điểm)

Cho file văn bản songuyen.inp chứa các số nguyên dương; các số cách nhau ít nhất một khoảng trắng.

Hãy tìm số lớn nhất, có bao nhiêu số bằng số lớn nhất ?

Đếm xem có bao nhiêu bộ 3 số x,y,z thỏa mãn điều kiện $x^2 + y^2 = z^2$?

Ghi kết quả vào file ketqua.out gồm hai dòng:

-Dòng thứ nhất ghi số lớn nhất và số lương số bằng số lớn nhất tương ứng với câu a.

-Dòng thứ hai ghi kết quả tìm được tương ứng với câu b.

Ví dụ:

songuyen.inp

121 10 8 7 12 6

4 5

121 3 121

ketqua.out

121 3

2

Hết

ĐỀ THI KỸ THUẬT LẬP TRÌNH 03

THỜI GIAN LÀM BÀI : 90 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

VIẾT CÁC CHƯƠNG TRÌNH HOÀN CHỈNH GIẢI CÁC BÀI TOÁN SAU:

CÂU 1 (2.0 điểm)

Cho dãy số x_n được định nghĩa như sau:

$x_0 = 1; x_1 = 2;$

$x_n = nx_0 + (n - 1)x_1 + \ldots + x_{n-1}$

Tính giá trị của x_5.

Với $n \geq 0$, tính x_n bằng cách có sử dụng đệ qui.

CÂU 2 (2.0 điểm)

Cho chuỗi *S* gồm các từ, mỗi từ cách nhau đúng một ký tự trắng, đầu chuỗi và cuối chuỗi không có khoảng trắng.

Đếm số lượng ký tự của mỗi từ trong chuỗi *S*.

Mã hóa chuỗi: Các ký tự *a*,*b*,*c*,….,*x*,*y*,*z* theo thứ tự đó được xếp thành vòng tròn. Hãy thay mỗi ký tự khác khoảng trắng của *S* bằng ký tự đứng sau nó đúng 5 vị trí theo thứ tự trên (nghĩa là ký tự *a* được thay bằng ký tự *f*, nghĩa là ký tự *b* được thay bằng ký tự *g*,…, ký tự *z* được thay bằng ký tự *e*).

Ví dụ:

Dữ liệu nhập	Kết quả
Que huong la chum khe ngot	Cau a. 3 5 2 4 3 4
	Cau b. *vzj mztsl qf hmzr pmj slty*

Câu 3 (2 điểm)

Cho mảng một chiều các phân số (tử số và mẫu số là số nguyên), hãy thực hiện các công việc sau đây:

a. Viết hàm tìm phân số có giá trị lớn nhất.

b. Viết hàm sắp xếp các phân số theo chiều tăng dần.

Câu 4 (2 điểm)

Trong kỳ thi tốt nghiệp hệ đại học có n thí sinh dự thi. Mỗi thí sinh cần quản lý các thông tin được mô tả như sau :

- Mã số sinh viên (kiểu chuỗi, 10 ký tự)
- Họ và tên (kiểu chuỗi, tối đa 36 ký tự)
- Điểm môn cơ bản (số nguyên)
- Điểm môn chuyên ngành 1 (số nguyên)

- Điểm môn chuyên ngành 2 (số nguyên)

a. Khai báo kiểu dữ liệu có cấu trúc như mô tả trên. Nhập thông tin cho n thí sinh.

b. Tìm những thí sinh có ít nhất một môn thi có điểm nhỏ hơn 5.

c. Tìm những thí sinh có tổng điểm ba môn thi lớn nhất.

d. Tìm những thí sinh có điểm của cả ba môn thi đều lớn hoặc bằng 5. Danh sách các thí sinh cần được sắp xếp giảm dần theo tổng điểm của ba môn thi.

CÂU 5 (2.0 điểm)

Cho file văn bản songuyen.inp chứa các số nguyên dương.

a. Định nghĩa số đối xứng là số mà khi viết các chữ số của nó theo chiều ngược lại thì ta được chính số đó; chẳng hạn 6, 131, 1991 là các số đối xứng. Đếm xem file songuyen.inp chứa bao nhiêu số đối xứng ?

b. Đếm xem trong file songuyen.inp có bao nhiêu số vừa là số nguyên tố vừa là số đối xứng ?

Kết quả xuất ra file songuyen.out ghi hai số nguyên là kết quả tìm được.

Ví dụ:

songuyen.inp				songuyen.out
4	11	4		4
100	101	34	45	2
23				

Hết

ĐỀ THI KỸ THUẬT LẬP TRÌNH 04

THỜI GIAN LÀM BÀI : 90 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

VIẾT CÁC CHƯƠNG TRÌNH HOÀN CHỈNH GIẢI CÁC BÀI TOÁN SAU:

CÂU 1 (2.0 điểm)

Cho dãy số x_n được định nghĩa như sau:

$x_0 = 1$;

$x_1 = 2$;

$x_n = nx_0 + (n - 1)x_1 + \ldots + x_{n-1}$

Cho biết giá trị của x_6

Với $n \geq 0$, tính x_n bằng cách có sử dụng đệ qui.

CÂU 2 (2.0 điểm)

Cho chuỗi s chứa các ký tự chữ cái thường và các ký tự khoảng trắng. Mỗi nhóm ký tự liên tiếp nhau trong s không chứa ký tự khoảng trắng gọi là một từ.

a. Đếm xem chuỗi s có bao nhiêu ký tự nguyên âm (a,e,i,o,u là các nguyên âm) ? Bao nhiêu ký tự phụ âm ?

b. Đếm xem mỗi từ trong chuỗi s có bao nhiêu ký tự ?

c. Tìm chuỗi gồm k từ bên trái của chuỗi s (giả sử chuỗi có nhiều hơn k từ).

d. Loại bỏ các khoảng trắng ở đầu và cuối chuỗi s; giữa các từ của chuỗi s chỉ giữ lại đúng một ký tự khoảng trắng. Tìm chuỗi còn lại sau khi đã loại bỏ các khoảng trắng như đã mô tả.

CÂU 3 (3.0 điểm)

Cho mảng hai chiều a gồm *m* dòng *n* cột, các phần tử là các số nguyên dương.

Tìm mảng *b*, biết $b_{ij}=a_{ij}*k$ với *k* là giá trị nhỏ nhất của dòng *i* nếu i là số lẻ và $b_{ij}=a_{ij}*k$ với k là giá trị lớn nhất của dòng *i* nếu i là số chẵn.

Tìm mảng c, biết $c_{ij}=l$, với *l* là tổng số lượng số nguyên tố trên dòng i và trên cột j (nếu a_{ij} là số nguyên tố thì chỉ được đếm một lần trên dòng hoặc trên cột).

CÂU 4 (3.0 điểm)

Cho file chuoi.inp chứa một chuỗi S. Chuỗi S gồm các từ cách nhau đúng một khoảng trắng, đầu và cuối chuỗi S không có khoảng trắng. Các từ trong S gồm các ký tự chữ cái từ a đến z. Giả thiết chuỗi S có ít nhất là 3 từ.

Từ chuỗi S, hãy tạo một chuỗi S_1 có cấu trúc như sau: Từ đầu tiên của chuỗi S_1 là một từ bên trái của chuỗi S, từ tiếp theo của chuỗi S_1 là một từ bên phải của chuỗi S, tiếp theo là các từ còn

lại của chuỗi S. Chuỗi S_1 cũng không có khoảng trắng đầu chuỗi, không có khoảng trắng cuối chuỗi, và giữa các từ chỉ có duy nhất một khoảng trắng.

Mỗi từ trong chuỗi S được xem là một chuỗi con, hãy sắp xếp các chuỗi con này theo thứ tự tăng.

Hết

ĐỀ THI KỸ THUẬT LẬP TRÌNH 05

THỜI GIAN LÀM BÀI : 90 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

VIẾT CÁC CHƯƠNG TRÌNH HOÀN CHỈNH GIẢI CÁC BÀI TOÁN SAU:

Anh/Chị hãy sử dụng ngôn ngữ lập trình *C*/*C*++ viết các chương trình hoàn chỉnh giải quyết các bài toán sau:

CÂU 1 (2 điểm)

Cho dãy số A_n được biểu diễn theo công thức đệ qui sau:

$A_1=1;A_2=2;A_3=3;A_n=24y_{n-1}+25y_{n-2}+10y_{n-3}+2015;$

a.Hãy tính A_n bằng cách sử dụng đệ qui.

b.Hãy tính A_n bằng cách không sử dụng đệ qui và cũng không sử dụng cấu trúc dữ liệu mảng.

CÂU 2 (2 điểm)

Nhập một bảng *a* có *m* dòng *n* cột; mỗi phần tử là một số nguyên dương.

a.Tìm giá trị nhỏ nhất của bảng *a*.

b.Hãy đếm số lượng số nguyên tố, số lượng số đối xứng có trong bảng *a* (số đối xứng là số mà khi viết các chữ số của nó theo chiều từ trái qua phải hay từ phải qua trái thì thu được cùng kết quả).

c.Hãy tạo bảng *b* có *m* dòng *n* cột biết rằng $b_{ij}=a_{ij} \times k_i$ với k_i là số có giá trị nhỏ nhất trên dòng i. Xuất bảng *b* lên màn hình.

Ví dụ:

3 4

101 14 111 18

2 16 46 1221

3 8 15 24

Thì kết quả câu a là: 2

kết quả câu b là: 3 6

kết quả câu c là:

1414 196 1554 252

4 32 92 2442

9 24 45 72

CÂU 3 (2 điểm)

Nhập vào một chuỗi *s* chứa các ký tự chữ cái (chữ thường hoặc chữ hoa) và các ký tự khoảng trắng. Giả thiết đầu chuỗi *s*, cuối chuỗi *s* không có khoảng trắng và giữa các từ có đúng một khoảng trắng.

a.Hãy đưa các ký tự đầu của mỗi từ thành chữ hoa, còn các ký tự khác thành chữ thường.

b.Hãy đếm xem mỗi từ trong chuỗi có bao nhiêu ký tự ?

Ví dụ nếu s=”truong dai hoc sai gon khoa cong nghe thong tin”

thì kết quả câu a là: Truong Dai Hoc Sai Gon Khoa Cong Nghe Thong Tin

kết quả câu b là: 6 3 3 3 3 4 4 4 5 3

CÂU 4 (2 điểm)

Trong kỳ thi tuyển sinh cao học có *n* thí sinh tham gia; mỗi thí sinh cần quản lý các thông tin sau:

- Số báo danh (kiểu chuỗi, 8 ký tự)
- Họ và tên (kiểu chuỗi, tối đa 36 ký tự)
- Điểm *môn ngoại ngữ* (số nguyên)
- Điểm *môn cơ sở* (số nguyên)
- Điểm *môn chuyên ngành* (số nguyên)

a.Nhập thông tin *n* thí sinh theo mô tả trên.

b.Tìm những thí sinh có điểm *môn chuyên ngành* là cao nhất.

c.Tìm những thí sinh có điểm thi *môn ngoại ngữ* ≥ 5 và có tổng điểm *môn cơ sở* và *môn chuyên ngành* ≥ 12.

CÂU 5 (2 điểm)

Cho file văn bản songuyen.inp chứa các số nguyên dương; các số cách nhau ít nhất một khoảng trắng.

a.Tìm số lớn nhất và số lớn thứ nhì trong file songuyen.inp

b.Số nguyên tố *x* được gọi là số nguyên tố đối xứng nếu nó bằng trung bình cộng của hai số nguyên tố kề trước (ký hiệu là *p*) và kề sau của nó (ký hiệu là *q*), tức là *p*+*q*=2**x*. Đếm xem trong file songuyen.inp có bao nhiêu số nguyên tố đối xứng ?

Ghi kết quả vào file ketqua.out gồm hai dòng:

-Dòng thứ nhất ghi số lớn nhất và số lớn thứ nhì.

-Dòng thứ hai ghi số lượng số nguyên tố đối xứng tìm được.

Ví dụ:

songuyen.inp

```
4
5  62    10    62
53 47    8     53    62    3
```

ketqua.out

```
62 53
3
```

Hết

ĐỀ THI KỸ THUẬT LẬP TRÌNH 06

THỜI GIAN LÀM BÀI : 90 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

VIẾT CÁC CHƯƠNG TRÌNH HOÀN CHỈNH GIẢI CÁC BÀI TOÁN SAU:

CÂU 1 (2.0 điểm)

Cho S_n=1.2 + 2.3.4 + 3.4.5.6 +...+ n.(n+1)...(2n)

Hãy viết chương trình hoàn chỉnh để tính S_n.

Ví dụ: với n =4 thì kết quả S_n= 7106

CÂU 2 (2.0 điểm)

Trong mặt phẳng tọa độ OXY, cho hai hình chữ nhật có các cạnh song song với các trục tọa độ: Hình chữ nhật thứ nhất có đỉnh dưới trái là A và đỉnh trên phải là B; hình chữ nhật thứ hai có đỉnh dưới trái là C và đỉnh trên phải là D. Giả sử hoành độ và tung độ của các đỉnh A,B,C,D là các số nguyên,

Hãy viết chương trình hoàn chỉnh để tìm tọa độ góc dưới trái (E) và tọa độ góc trên phải (F) của hình chữ nhật bé nhất chứa cả hai hình chữ nhật trên.

Ví dụ: với dữ liệu nhập là A(1;0), B(5;3), C(3;-2), D(6;1) thì kết quả là E(1;-2), F(6;3)

CÂU 3 (2.0 điểm)

Cho chuỗi s gồm các từ, mỗi từ gồm các ký tự chữ cái (giả sử chuỗi s không có khoảng trắng dư thừa ở đầu và cuối chuỗi; giữa các từ có thể có một hoặc nhiều khoảng trắng). Hãy viết các hàm thực hiện các công việc sau:

a. Đếm xem chuỗi s có bao nhiêu từ ?

b. Đếm số lượng ký tự của mỗi từ trong chuỗi ?

c. Đếm xem chuỗi s có bao nhiêu từ có đúng k ký tự ?

Ví dụ: với chuỗi s là "Truong Dai Hoc Sai Gon" và k =3 thì kết quả câu a là 5, kết quả câu b là 6, 3, 3, 3, 3 và kết quả câu c là 4.

CÂU 4 (2.0 điểm)

Cho dãy n số nguyên dương a_o,a_1,…,a_{n-1}. Hãy viết các hàm thực hiện các công việc sau:

a. Đếm xem trong mảng có bao nhiêu số nguyên tố có hai chữ số?

b. Tìm giá trị nguyên tố lớn nhất, nếu không có trả về giá trị 0.

c. Biến đổi các số của mảng về mảng toàn số nguyên tố theo nguyên tắc: Các số không phải là số nguyên tố thì được biến đổi thành số nguyên tố gần nó nhất.

Ví dụ: với n=8 và dãy 42, 6, 5, 7, 15, 17, 2010, 13 thì kết quả câu a là 2, kết quả câu b là 17, kết quả câu c là: 43, 5, 5, 7, 13, 17, 2011, 13.

CÂU 5 (2.0 điểm)

Cho file văn bản songuyen.inp chứa các số nguyên dương.

a.Định nghĩa số đối xứng là số mà khi viết các chữ số của nó theo chiều ngược lại thì ta được chính số đó; chẳng hạn 6, 131, 1991 là các số đối xứng. Đếm xem file songuyen.inp chứa bao nhiêu số đối xứng ?

b.Đếm xem trong file songuyen.inp có bao nhiêu số vừa là số nguyên tố vừa là số đối xứng ?

Kết quả xuất ra file songuyen.out ghi hai số nguyên là kết quả tìm được.

Ví dụ:

songuyen.inp	songuyen.out
4 11 4 4 100 101 34 45 23	2

Hết

ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 01

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Cho file Bangso.inp có cấu trúc như sau:

-Dòng đầu ghi 2 số m,n.

-Trong m dòng tiếp theo mỗi dòng ghi n số nguyên dương; các số cách nhau ít nhất một khoảng trắng (bảng có tất cả là m*n phần tử). Vị trí dòng, cột bắt đầu từ 1.

Viết một chương trình thực hiện các công việc sau (các câu là độc lập với nhau):

CÂU 1. Đếm số lượng số đối xứng lớn hơn hoặc bằng 10 có trong bảng (3 điểm).

CÂU 2. Tìm tích của 3 giá trị lớn nhất của bảng (3 giá trị này không nhất thiết khác nhau) (3 điểm).

CÂU 3. Tìm tổng các phần tử của bảng *b*, biết $b_{ij}=a_{ij}*k$ với *k* là giá trị nhỏ nhất của dòng *i* nếu i là số lẻ và $b_{ij}=a_{ij}*k$ với k là giá trị lớn nhất của dòng *i* nếu i là số chẵn. (2 điểm).

CÂU 4. Tìm tổng các phần tử của mảng c, biết c_{ij}=l, với l là tổng số lượng số nguyên tố trên dòng i và trên cột j (nếu a_{ij} là số nguyên tố thì chỉ được đếm một lần trên dòng hoặc trên cột). (2 điểm).

Ví dụ:

Bangso.inp

```
4   5
16  171  5   7   29
28  2    2   11  31
121 19   5   7   37
6   11   3   23  25
```

Bangso.out

Cau 1: 4

Cau 2: 765567

Cau 3: 6079

Cau 4: 112

Hết

ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 02

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Cho file chuoi.inp chứa một chuỗi S. Chuỗi S gồm các từ cách nhau đúng một khoảng trắng, đầu và cuối chuỗi S không có khoảng trắng. Các từ trong S gồm các ký tự chữ cái từ a đến z. Giả thiết chuỗi S có ít nhất là 3 từ.

Viết một chương trình thực hiện các công việc sau (các câu hỏi là độc lập với nhau):

CÂU 1. (4 điểm) Cho biết số lượng ký tự của mỗi từ trong chuỗi S.

CÂU 2. (3 điểm) Từ chuỗi S, hãy tạo một chuỗi S_1 có cấu trúc như sau: Từ đầu tiên của chuỗi S_1 là một từ bên trái của chuỗi S, từ tiếp theo của chuỗi S_1 là một từ bên phải của chuỗi S, tiếp theo là các từ còn lại của chuỗi S. Chuỗi S_1 cũng không có khoảng trắng đầu chuỗi, không có khoảng trắng cuối chuỗi, và giữa các từ chỉ có duy nhất một khoảng trắng.

CÂU 3. (3 điểm) Mã hóa chuỗi: Thay mỗi ký tự khác khoảng trắng bằng ký tự đứng sau nó đúng 5 vị trí (vòng tròn: ký tự a được thay bằng ký tự f, ký tự z được thay bằng ký tự e,…)

Ví dụ:

chuoi.inp

que huong la chum khe ngot

chuoi.out

Câu 1: 3 5 2 4 3 4

Câu 2: que ngot huong la chum khe

Câu 3: vzj mztsl qf hmzr pmj slty

Hết

ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 03

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Cho file THI.INP có nhiều chuỗi, mỗi chuỗi trên một dòng, mỗi chuỗi chứa các ký tự chữ cái thường tiếng Anh và khoảng trắng, giữa các từ có đúng một khoảng trắng, đầu và cuối mỗi chuỗi không có khoảng trắng.

Hãy lập trình thực hiện các công việc sau:

CÂU 1 (2.0 đ):Đếm xem file THI.INP có tất cả bao nhiêu từ ?

CÂU 2 (2.0 đ):Tìm tần số xuất hiện của mỗi ký tự (khác khoảng trắng) trong file THI.INP? (Ký tự a xuất hiện bao nhiêu lần ? Ký tự b xuất hiện bao nhiêu lần ?...)

CÂU 3 (2.0 đ):Tìm một từ dài nhất cho mỗi chuỗi trong file THI.INP.

CÂU 4 (2.0 đ):Đảo ngược các từ trong mỗi chuỗi của file THI.INP, trong đó thứ tự các ký tự trong mỗi từ giữ nguyên.

CÂU 5 (2.0 đ):Tìm các chuỗi con chứa các từ thứ 2,3,4 của mỗi chuỗi trong file THI.INP; đầu và cuối chuỗi con này không có khoảng trắng, giữa các từ của chuỗi con có một khoảng trắng.

Kết quả ghi vào file THI.OUT theo cấu trúc như minh họa ở bộ test tham khảo sau.

BỘ TEST THAM KHẢO

THI.INP

what good is money if it can not buy happiness
do not waste your time on a man who is not willing to waste their time on you
it is what is in yourself that makes you happy or unhappy
a true friend is someone who reaches for your hand and touches your heart

THI.OUT

CAU 1:
54
CAU 2:
a 17
b 1
c 3
d 5
e 17
… // yêu cầu liệt kê hết
y 10

CAU 3:

happiness

willing

yourself

someone

CAU 4:

happiness buy not can it if money is good what

you on time their waste to willing not is who man a on time your waste not do

unhappy or happy you makes that yourself in is what is it

heart your touches and hand your for reaches who someone is friend true a

CAU 5:

good is money

not waste your

is what is

true friend is

Hết

ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 04

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Cho file THI.INP chứa các số nguyên trong phạm vi 0 đến 99999. Các số cách nhau ít nhất một khoảng trắng.

Hãy lập trình thực hiện các công việc sau:

CÂU 1 (2.0 đ). Số nguyên tố Palindrome là số mà khi viết các chữ số của nó theo chiều từ trái qua phải hay từ phải qua trái thì ta cùng được một kết quả. Ví dụ 2, 3, 151, 19891, 19991 là các số nguyên tố Palindrome. Đếm xem file THI.INP có bao nhiêu số nguyên tố ? Bao nhiêu số nguyên tố Palindrome?

CÂU 2 (2.0 đ). Hai số nguyên dương a,b được gọi là nguyên tố cùng nhau nếu ước số chung lớn nhất của chúng bằng 1. Chẳng hạn các cặp (3,4), (8,9), (1,6) là nguyên tố cùng nhau.

Đếm xem file THI.INP có bao nhiêu cặp số nguyên tố cùng nhau ?

CÂU 3 (2.0 đ). Số nguyên tố đối xứng là một số nguyên tố bằng trung bình cộng của hai số nguyên tố liền trước và liền sau nó. Với p_n là số nguyên tố thứ *n*, một số nguyên tố là đối xứng khi thoả: $p_n = \frac{p_{n-1} + p_{n+1}}{2}$.

(Lưu ý nếu THI.INP chứa số 5 thì sẽ được tính là một số nguyên tố đối xứng mà không cần kiểm tra điều kiện 3 và 7 có thuộc THI.INP hay không ?).

Đếm xem file THI.INP có bao nhiêu số nguyên tố đối xứng ?

CÂU 4 (2.0 đ). Đếm xem file THI.INP có bao nhiêu số nguyên tố khác nhau (mỗi giá trị nguyên tố chỉ được tính là một lần).

CÂU 5 (2.0 đ). Số Mersenne là số biểu diễn được dưới dạng $2^n - 1$. Số nguyên tố Mersenne là số thỏa hai tính chất: là số Mersenne và là số nguyên tố. Ví dụ 31 là số nguyên tố Mersenne vì $31 = 2^5 - 1$, và 31 là số nguyên tố. Lưu ý rằng, trong biểu diễn số Mersenne, nếu *n* là số nguyên tố thì $2^n - 1$ là số nguyên tố Mersenne. Đếm xem file THI.INP có bao nhiêu số nguyên tố Mersenne ?

Kết quả ghi vào file THI.OUT theo cấu trúc như minh họa ở bộ test tham khảo sau.

BỘ TEST THAM KHẢO

THI.INP

123 23 121 223 4 41 127 8191 199 19 103 43 3 1191

17 211 47 59 6 31 213 31 3 1011 213 41 53 801

127 19 101 11 5 101 31013 111 117

THI.OUT

CAU 1: 26 7

CAU 2: 602

CAU 3: 3

CAU 4: 20

CAU 5: 7

Hết

ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 05

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Cho file THI.INP chứa các số nguyên trong phạm vi 0 đến 99999. Các số cách nhau ít nhất một khoảng trắng.

Hãy lập trình thực hiện các công việc sau:

CÂU 1 (2.0 đ).THI.INP có bao nhiêu số chính phương ? Bao nhiêu số hoàn chỉnh ? Bao nhiêu số Armstrong ? Bao nhiêu số nguyên tố ? Bao nhiêu số đối xứng ?

CÂU 2 (2.0 đ).Trong THI.INP tìm số chính phương lớn nhất, số hoàn chỉnh lớn nhất, số Armstrong lớn nhất, số nguyên tố lớn nhất, số đối xứng lớn nhất (số có tính chất nào không được tìm thấy thì giá trị lớn nhất được tính là 0).

CÂU 3 (2.0 đ).Số nào trong THI.INP xuất hiện nhiều lần nhất và xuất hiện bao nhiêu lần ?

CÂU 4 (2.0 đ).THI.INP chứa bao nhiêu số có giá trị khác nhau ?

CÂU 5 (2.0 đ).THI.INP có bao nhiêu bộ 3 số x,y,z ($x < y < z$) thỏa mãn $x^2 + y^2 = z^2$?

Kết quả câu 1 gồm 5 số nguyên được ghi trên một dòng; là kết quả tương ứng tìm được.

Kết quả câu 2 gồm 5 số nguyên được ghi trên một dòng; là kết quả tương ứng tìm được.

Kết quả câu 3 gồm 2 số nguyên được ghi trên một dòng; là kết quả tương ứng tìm được.

Kết quả câu 4 gồm 1 số nguyên; là kết quả tương ứng tìm được.

Kết quả câu 5 gồm 1 số nguyên; là kết quả tương ứng tìm được.

Kết quả ghi vào file THI.OUT.

BỘ TEST THAM KHẢO

THI.INP

```
407   121   28    8128  49    25    36    153   370   101   153   97    71
      79
32    34    93    111   4     3     1     29    13    5     19    103   16
      10
28    6     496   28    11    31013 5     8     6
```

THI.OUT

```
CAU 1:   7   7   12   13   13
CAU 2:  121  8128  407 31013 31013
CAU 3:   28   3
CAU 4:   32
CAU 5:   4
```

Hết

ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 06

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Cho file THI.INP có cấu trúc như sau:

-Dòng đầu ghi hai số nguyên dương *m*,*n*.

-Trong *m* dòng tiếp theo, mỗi dòng ghi *n* số nguyên trong phạm vi 0 đến 99999.

Các số cách nhau ít nhất một khoảng trắng. Đặt mảng *a* ứng với dữ liệu trong *m* dòng *n* cột này.

Hãy lập trình thực hiện các công việc sau:

CÂU 1 (2.0 đ).Tìm số lớn thứ nhì của mảng *a*, mảng *a* có bao nhiêu số bằng số lớn thứ nhì này ?

CÂU 2 (2.0 đ).Tạo mảng *b* từ mảng *a* với b_{ij} là số lượng các số nguyên tố bao quanh a_{ij} (8 hướng, không kể chính nó). Có bao nhiêu số trong mảng *b* lớn hơn hoặc bằng 4 ?

CÂU 3 (2.0 đ).Tìm một bộ 3 số cùng thuộc một dòng nào đó của mảng *a* sao cho tổng các chữ số của 3 số này là lớn nhất. Hãy cho biết chỉ số dòng (được tính từ 0,1,2,3,...) chứa 3 số tìm được và tổng các chữ số của 3 số đó.

CÂU 4 (2.0 đ).Đếm xem trong mảng *a* có bao nhiêu bộ 3 số nguyên tố *x,y,z* cùng thuộc một dòng nào đó của mảng *a* (trong đó *x,y,z* là khác nhau) thỏa tính chất: (*x*+*z*)=2**y* ?

CÂU 5 (2.0 đ).Tìm một số xuất hiện nhiều lần nhất trong mảng *a* ? Xuất hiện bao nhiêu lần ?

Kết quả ghi vào file THI.OUT gồm 5 dòng, trong đó mỗi dòng 1,3,5 ghi hai số nguyên, mỗi dòng 2,4 ghi một số nguyên là kết quả của các câu tương ứng.

BỘ TEST THAM KHẢO

THI.INP

```
4       5
101     12      21      42      49
5       59      33      22      33
101     7       33      5       3
59      101     53      47      10
```

THI.OUT

```
CAU 1:59 2
CAU 2:7
CAU 3:3 33
CAU 4:2
CAU 5:33 3
```

Hết

ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 07

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Cho file THI.INP chứa các chuỗi; mỗi chuỗi trên một dòng; đầu và cuối mỗi chuỗi không có khoảng trắng, giữa các từ có duy nhất một khoảng trắng. Mỗi chuỗi có tối đa 256 ký tự là chữ cái thường hoặc khoảng trắng.

Hãy lập trình thực hiện các công việc sau:

CÂU 1 (2điểm)**.** Có bao nhiêu chuỗi chứa chuỗi "que huong" ?

CÂU 2 (2điểm). Mỗi chuỗi chứa bao nhiêu từ có 5 ký tự ?

CÂU 3 (2điểm). Cho biết số lượng nguyên âm của mỗi từ trong mỗi chuỗi ?

CÂU 4 (2điểm). Tìm một chuỗi có nhiều từ nhất; và cho biết chuỗi tìm được này có bao nhiêu từ ?

CÂU 5 (1 điểm). Sắp xếp các chuỗi tăng theo thứ tự chiều dài của các chuỗi.

CÂU 6 (1 điểm)**.** Mã hóa bậc k: Mỗi ký tự nằm trong từ có k ký tự thì sẽ được thay thế bằng ký tự đứng sau nó k vị trí trong bảng chữ cái tiếng Anh (giả thiết bảng các chữ cái này được sắp theo vòng tròn; nghĩa là ký tự z sẽ được thay bằng ký tự e nếu k=5; rõ ràng cùng một ký tự có thể được mã hóa thành các ký tự khác nhau).

Kết quả ghi vào file THI.OUT

BỘ TEST THAM KHẢO

THI.INP

que huong la gi ho me ma co giao day phai yeu que huong la gi ho me ai di xa cung nho nhieu

que huong la chum khe ngot cho con treo hai moi ngay

que huong la duong di hoc con ve rop buom vang bay

que huong la con dieu biec tuoi tho con tha tren dong que huong la con do nho em dem khua nuoc ven song

que huong la vang hoa bi la hong tim giau mong toi

la do doi bo dam but mau hoa sen trang tinh khoi

que huong moi nguoi chi mot nhu la chi mot me thoi que huong neu ai khong nho se khong lon noi thanh nguoi

THI.OUT

CAU 1: 6

CAU 2: 3 1 2 2 1 1 7

CAU 3: 2 2 1 1 1 1 1 1 3 1 2 2 2 2 1 1 1 1 2 1 1 1 1 3

(câu này gồm 7 dòng dạng như trên)

CAU 4: que huong la gi ho me ma co giao day phai yeu que huong la gi ho me ai di xa cung nho nhieu

24

CAU 5: la do doi bo dam but mau hoa sen trang tinh khoi

que huong la duong di hoc con ve rop buom vang bay

que huong la vang hoa bi la hong tim giau mong toi

que huong la chum khe ngot cho con treo hai moi ngay

que huong la gi ho me ma co giao day phai yeu que huong la gi ho me ai di xa cung nho nhieu

que huong la con dieu biec tuoi tho con tha tren dong que huong la con do nho em dem khua nuoc ven song

que huong moi nguoi chi mot nhu la chi mot me thoi que huong neu ai khong nho se khong lon noi thanh nguoi

CAU 6: (gồm 7 dòng)

txh mztsl… // đây là kết quả mã hóa của hai từ que huong

Hết

ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 08

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Trong một kỳ thi tuyển sinh cao học có *n* thí sinh tham gia; giả sử rằng mỗi thí sinh cần quản lý các thông tin sau:

- Số báo danh (số nguyên)
- Họ và tên (kiểu chuỗi, tối đa 36 ký tự)
- Điểm *môn ngoại ngữ* (số nguyên)
- Điểm *môn cơ sở* (số nguyên)
- Điểm *môn chuyên ngành* (số nguyên)

Nhập từ bàn phím thông tin *n* thí sinh theo mô tả trên ; mỗi thí sinh nhập lần lượt các thông tin Số báo danh, Họ và tên, điểm môn ngoại ngữ, điểm môn cơ sở, điểm môn chuyên ngành.

Hãy thực hiện các công việc sau :

CÂU 1 (2 điểm)

Tìm kết quả thi các môn của thí sinh có số báo danh là 1094.

CÂU 2 (2 điểm)

Tìm điểm trung bình cộng của từng môn thi *ngoại ngữ, cơ sở, chuyên ngành*.

CÂU 3 (2 điểm)

Tìm các thí sinh có tổng điểm hai môn thi cơ sở và chuyên ngành là cao nhất.

CÂU 4 (2 điểm)

Tìm các thí sinh có điểm thi môn cơ sở và chuyên ngành đều ≥ 8 và điểm môn Ngoại ngữ ≥ 5.

CÂU 5 (2 điểm)

Tìm kết quả thi của tất cả thí sinh có điểm thi ngoại ngữ lớn hơn hoặc bằng 5. Yêu cầu danh sách được sắp xếp giảm dần theo tổng điểm của hai môn thi cơ sở và chuyên ngành.

Xuất kết quả tìm được của các yêu cầu trên lên màn hình.

Ví dụ:

Dữ liệu nhập từ bàn phím:

n=5

1093	Phung Quoc Nguyen	4	9	9
1094	Le Thi Thanh Nhan	8	8	10
1095	Tran Trong Nhan	7	5	9
1096	Bui Pham Ngoc Nhi	8	8	8
1097	Phuong Minh Nhi	5	6	7

Kết quả xuất ra màn hình:

CÂU 1: 1094 Le Thi Thanh Nhan

Diem thi mon ngoai ngu: 8

Diem thi mon co so: 8

Diem thi mon chuyen nganh: 10

CÂU 2: Môn ngoại ngữ: 6.4

Môn cơ sở: 7.2

Môn chuyên ngành: 8.6

CÂU 3:	1093	Phung Quoc Nguyen	9	9	
	1094	Le Thi Thanh Nhan	8	10	
CÂU 4:	1094	Le Thi Thanh Nhan	8	8	10
	1096	Bui Pham Ngoc Nhi	8	8	8
CÂU 5:	1094	Le Thi Thanh Nhan	8	8	10
	1096	Bui Pham Ngoc Nhi	8	8	8
	1095	Tran Trong Nhan	7	5	9
	1097	Phuong Minh Nhi	5	6	7

Hết

ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 09

THỜI GIAN LÀM BÀI : 120 PHÚT (KHÔNG SỬ DỤNG TÀI LIỆU)

Cho mảng có *n* phân số, giả thiết tử số và mẫu số của mỗi phần số là các số nguyên dương.

CÂU 1 (2 điểm).

In các phân số của mảng ở dạng tối giản.

CÂU 2 (2 điểm).

Có bao nhiêu phân số có giá trị nhỏ nhất trong mảng.

CÂU 3 (2 điểm).

Đếm xem trong mảng có bao nhiêu phân số có giá trị lớn hơn 0 và nhỏ hơn 1.

CÂU 4 (2 điểm).

Tìm 3 phân số có giá trị lớn nhất khác nhau trong mảng; xuất kết quả ở dạng tối giản, xuất theo thứ tự giảm.

CÂU 5 (2 điểm).

Đếm xem trong mảng có bao nhiêu phân số (ở dạng tối giản) mà tử số và mẫu số của phân số đó đều là các số nguyên tố ?

Bộ test mẫu tham khảo

INPUT

10

2/3 8/6 10/12 7/2 10/4 5/2 7/10 13/11 8/12 4/6

OUTPUT (thời gian thực hiện tất cả các câu không quá 1 giây)

Cau 1 : 2/3 4/3 5/6 7/2 5/2 5/2 7/10 13/11 2/3 2/3

Cau 2 : 3

Cau 3 : 5

Cau 4 : 7/2 5/2 13/11

Cau 5 : 7

Hết

ĐỀ KIỂM TRA THỰC HÀNH KỸ THUẬT LẬP TRÌNH 10

Trong một kỳ thi tuyển sinh cao học có *n* thí sinh tham gia; giả sử rằng mỗi thí sinh cần quản lý các thông tin sau:

- Số báo danh (số nguyên)
- Họ và tên (kiểu chuỗi, tối đa 36 ký tự)
- Điểm *môn ngoại ngữ* (số nguyên)
- Điểm *môn cơ sở* (số nguyên)
- Điểm *môn chuyên ngành* (số nguyên)

Nhập từ bàn phím thông tin *n* thí sinh theo mô tả trên ; mỗi thí sinh nhập lần lượt các thông tin Số báo danh, Họ và tên, điểm môn ngoại ngữ, điểm môn cơ sở, điểm môn chuyên ngành.

Hãy thực hiện các công việc sau :

CÂU 1 (2 điểm)

Tìm kết quả thi các môn của thí sinh có số báo danh là 1094.

CÂU 2 (2 điểm)

Tìm điểm trung bình cộng của từng môn thi *ngoại ngữ, cơ sở, chuyên ngành*.

CÂU 3 (2 điểm)

Tìm các thí sinh có tổng điểm hai môn thi cơ sở và chuyên ngành là cao nhất.

CÂU 4 (2 điểm)

Tìm các thí sinh có điểm hai môn thi cơ sở và chuyên ngành đều ≥ 8 và điểm môn Ngoại ngữ ≥ 5.

CÂU 5 (2 điểm)

Tìm kết quả thi của tất cả thí sinh có điểm thi ngoại ngữ lớn hơn hoặc bằng 5. Yêu cầu danh sách được sắp xếp giảm dần theo tổng điểm của hai môn thi cơ sở và chuyên ngành.

Xuất kết quả tìm được của các yêu cầu trên lên màn hình.

Ví dụ:

Dữ liệu nhập từ bàn phím:

n=5

1093	Phung Quoc Nguyen	4	9	9
1094	Le Thi Thanh Nhan	8	8	10
1095	Tran Trong Nhan	7	5	9
1096	Bui Pham Ngoc Nhi	8	8	8
1097	Phuong Minh Nhi	5	6	7

Kết quả xuất ra màn hình:

CÂU 1:	1094	Le Thi Thanh Nhan			
	Diem thi mon ngoai ngu:		8		
	Diem thi mon co so:		8		
	Diem thi mon chuyen nganh:		10		
CÂU 2:	Môn ngoại ngữ:		6.4		
	Môn cơ sở:		7.2		
	Môn chuyên ngành:		8.6		
CÂU 3:	1093	Phung Quoc Nguyen	9	9	
	1094	Le Thi Thanh Nhan	8	10	
CÂU 4:	1094	Le Thi Thanh Nhan	8	8	10
	1096	Bui Pham Ngoc Nhi	8	8	8
CÂU 5:	1094	Le Thi Thanh Nhan	8	8	10
	1096	Bui Pham Ngoc Nhi	8	8	8
	1095	Tran Trong Nhan	7	5	9
	1097	Phuong Minh Nhi	5	6	7

Hết

PHỤ LỤC. MỘT SỐ HÀM CHUẨN THƯỜNG SỬ DỤNG

(trong môi trường C chuẩn)

conio.h

void clrscr();

Xoá màn hình

void gotoxy(int x, int y);

Đưa điểm nháy đến toạ độ x,y trên màn hình

void textcolor(int color)

Đặt màu chữ mới

void textbackground (int color)

Đặt màu nền mới

void getch();

Nhập ký tự (không hiện ra màn hình)

void getche();

Nhập ký tự (có hiện ra màn hình)

stdlib.h

void randomize();

Khởi động cơ chế tạo số ngẫu nhiên

void random(int n)

Trả về số nguyên trong khoảng 0 đến n-1.

unsigned int rand();

Cho một giá trị ngẫu nhiên trong khoảng 1.. 32767

void flushall();

Xoá vùng đệm bàn phím, lệnh này thường được sử dụng trước các lệnh nhập liệu như gets hoặc scanf

math.h

double sqrt(double x) ;

Căn bậc hai

double exp(double x) ;

Tính e^x

double pow(double x, double y);

Tính x^y

int abs(int x) ;

Trị tuyệt đối của kiểu số int

double fabs(double x) ;

Trị tuyệt đối của kiểu số float,double

long int labs(long int x) ;

Trị tuyệt đối của kiểu số long

double ceil(double x) ;

Làm tròn số lên (đối với số thực)

double floor(double x) ;

Làm tròn số xuống(đối với số thực)

double log(double x) ;

Tính logarit tụ nhiên của x

double sin(double x) ;

Tính sin của x

double cos(double x) ;

Tính cos của x

double tan(double x) ;

Tính tg của x

asin,acos,atan

Các hàm lượng giác ngược (tham số các hàm lượng giác là radian)

ctype.h

void putchar (char c);

Xuất ký tự

void getchar (char c);

Nhập ký tự

int isalnum(int c);

Kiểm tra c là ký tự (chữ cái hoặc chữ số) hay không ?

int isalpha(int c);

Kiểm tra c là chữ cái hay không ?

int isdigit(int c);

Kiểm tra c là chữ số hay không ?

int islower (int ct);

Kiểm tra c là chữ cái thường hay không ?

isupper(int c);

Kiểm tra c là chữ cái hoa hay không ?

int isspace(int c);

Kiểm tra xem ký tự c là ký tự trống ?

int tolower(int c);

Chuyển ký tự c thành chữ thường

int toupper(int c);

Chuyển ký tự c thành chữ hoa.

int toascii(int c);

Chuyển ký tự c thành mã ASCII

Bảng mã ASCII

Bộ mã ASCII gồm 256 ký tự được phân bố như sau:

-32 ký tự đầu tiên là các ký tự điều khiển không in được như ký tự enter (mã 13), ký tự ESC (mã 27), 32 là ký tự khoảng trắng.

-Các mã ASCII 33-47,58-64,91-96 và 123-127 là các ký tự đặc biệt như dấu chấm, dấu phẩy, dấu cách, dấu ngoặc, dấu móc, dấu hỏi,…

-Các mã ASCII 48-57 là 10 chữ số.

-Các mã ASCII 65-90 là các chữ cái hoa từ A đến Z.

-Các mã ASCII 97-122 là các chữ cái thường từ a đến z.

-Các mã ASCII 128-255 là các ký tự đồ họa.

string.h

*char *gets(char * str);*

Chờ nhập vào từ bàn phím chuỗi ký tự, hàm trả về con trỏ trỏ tới địa chỉ đầu chuỗi str. Chú ý rằng để đảm bảo bộ đệm bàn phím được làm rỗng trước khi nhập dữ liệu chuỗi, chúng ta nên sử dụng hàm fflush(stdin).

*char *puts (char * str);*

Xuất chuỗi str lên màn hình.

*unsigned *strlen(const char *str);*

Xác định chiều dài của chuỗi str.

char strcpy(char *dest, const char * src);*

Sao chép nội dung trong src vào trong dest. Hàm trả về con trỏ trỏ tới địa chỉ đầu của chuỗi đích (con trỏ dest)

*int strcmp(char *s1, char *s2);*

(so sánh có phân biệt chữ hoa và chữ thường)

Trả về kết quả so sánh hai chuỗi s1 và s2.

Nếu giá trị trả về > 0 thì chuỗi s1 chứa chuỗi s2

Nếu giá trị trả về < 0 thì chuỗi s2 chứa chuỗi s1

Nếu giá trị trả về = 0 thì chuỗi s2 giống chuỗi s1

*int stricmp(char *s1, char *s2);*

(so sánh không phân biệt chữ hoa và chữ thường)

*char * strrev(char *st);*

Đảo ngược các kỹ tự trong chuỗi st, hàm trả về con trỏ st.

*strlwr(char *s);*

Đổi chuỗi s thành chữ thường.

*strupr(char *s);*

Đổi chuỗi s thành chữ hoa.

*strcat(char *s1, char *s2);*

Ghép chuỗi s2 vào sau chuỗi s1.

*strchr(char *s,char c);*

Tìm ký tự c trong chuỗi s, không có trả về NULL; tìm từ bên trái.

*strrchr(char *s,char c);*

Giống hàm trên nhưng bắt đầu tìm từ bên phải.

*strstr(char *s1, char *s2);*

Tìm chuỗi s2 trong chuỗi s1 trả về vị trí chuỗi s2 trong chuỗi s1.

*int atoi(char *s);*

Đổi chuỗi s thành số kiểu int.

*int atol(char *s);*

Đổi chuỗi s thành số kiểu long.

*int atof(char *s);*

Đổi chuỗi s thành số kiểu float.

void flushall();

Xoá vùng đệm bàn phím, lệnh này thường được sử dụng trước các lệnh nhập liệu như gets hoặc scanf.

Tọa độ màn hình

Hệ toạ độ trên màn hình nhận điểm ở gốc trên bên trái làm điểm gốc, trục hoành là trục nằm ngang chạy từ trái sang phải, trục tung là trục thẳng đứng từ trên xuống dưới. Các tọa độ màn hình thường được ký kiệu là x và y, với x là chỉ số cột; còn y chỉ số hàng. Góc trái bên trên của màn hình có tọa độ (1:1); góc phải bên dưới có tọa độ (80:25) – giới hạn này tùy thuộc loại màn hình. Ta có thể di chuyển con trỏ tới một vị trí mới trên màn hình bằng lệnh gotoxy(x,y).

Giá trị bảng màu

BLACK	=	0
BLUE	=	1
GREEN	=	2
CYAN	=	3
RED	=	4
MAGENTA	=	5
BROWN	=	6
LIGHTGRAY	=	7
DARKGRAY	=	8
LIGHTBLUE	=	9
LIGHTGREEN	=	10
LIGHTCYAN	=	11
LIGHTRED	=	12
LIGHTMAGENTA	=	13
YELLOW	=	14
WHITE	=	15
BLINK	=	128

Cách ghi giá trị màu là chuỗi in hoa như trên hay số tương ứng là tương đương.

TÀI LIỆU THAM KHẢO

[1]. **Brian W. Kernighan, Dennis M. Ritchie**, "*The C Programming Language*", 1998.

[2]. **Jon Bentley**, "*Programming Pearls*", Addison-Wesley, 2002.

[3]. **Tim Bailey**, "*An Introduction to the C Programming Language and Software Design"*, 2005.

[4]. **John M.Harris, Jeffry L.Hirst, Michael J.Mossinghoff**. "*Combinatorics and Graph Theory*". Springer, pp.1-353.2008.

[5]. **Michael T. Goodrich, Roberto Tamassia, David M. Mount**. "*Data Structures and Algorithms in C++*". John Wiley & Sons, pp.194-654. 2011.

[6]. **Robert Sedgewick, Kevin Wayne**. "*Algorithms"*. Fourth edition, Addsion-Wesley, pp.518-700. 2011.

[7]. **Phạm Văn Ất**, "*Kỹ thuật lập trình C - Cơ sở và nâng cao*", NXB khoa học kỹ thuật, 2007.

[8]. **Nguyễn Đức Nghĩa, Nguyễn Tô Thành,"***Toán rời rạc",NXB ĐHQG Hà Nội,* 2007.

[9]. **Nguyễn Đức Nghĩa**, "*Cấu trúc dữ liệu và giải thuật*", ĐH Bách Khoa Hà Nội, 2012.

[10]. **Trần Đan Thư, Nguyễn Thanh Phương, Đinh Bá Tiến, Trần Minh Triết**,"*Nhập môn lập trình*", Trường ĐH KHTN ĐHQG TPHCM, 2011.

[11]. **Trần Văn Hạo, Huỳnh Minh Trí, Phan Tấn Quốc**, "*Giáo trình tuyển tập các bài tập lập trình căn bản*", Trường Đại Học Sài Gòn, 2011.

[12]. www.olp.vn