

LẬP TRÌNH C/C++

LẬP TRÌNH C/C++

2022

Khoa Công nghệ Thông tin

GIỚI THIỆU

*Giáo trình được biên soạn hoàn toàn dựa trên nhiều tài liệu của các đồng nghiệp. Giáo trình nên dùng với những cuốn sách giáo khoa hoàn chỉnh về ngôn ngữ lập trình C/C++. Giáo trình được chia thành 10 chương theo từng nội dung kiến thức, kèm theo Các đề thi mẫu. Mỗi chương gồm 2 phần: Phần lý thuyết: được tóm tắt ngắn gọn với đầy đủ ví dụ minh họa kèm theo. Phần bài tập: với nhiều bài tập được chia làm hai mức độ cơ bản và luyện tập nâng cao, bài tập có đánh dấu * là bài tập khó dành cho sinh viên luyện tập thêm. Phần tóm tắt: tóm tắt nội dung tóm và các thao tác mà sinh viên cần nắm hay những lưu ý của chương đó.*

Mục lục

GIỚI THIỆU	ii
1 CÁC KHÁI NIỆM CƠ BẢN VỀ LẬP TRÌNH	1
1.1 KHÁI NIỆM LẬP TRÌNH	1
1.2 CÔNG NGHỆ LẬP TRÌNH TRUYỀN THỐNG	4
1.3 CÔNG NGHỆ LẬP TRÌNH HIỆN ĐẠI	4
1.3.1 Công cụ và môi trường lập trình	5
1.4 THUẬT TOÁN	5
2 LƯU ĐỒ THUẬT TOÁN	8
2.1 LÝ THUYẾT	8
2.1.1 Khái niệm	8
2.1.2 Phần mềm công cụ	8
2.1.3 Các ký hiệu	9
2.1.4 Phương pháp vẽ và thực thi	9
2.1.5 Các cấu trúc điều khiển cơ bản	9
2.2 BÀI TẬP	11
2.3 TÓM TẮT	13
3 KIỂU DỮ LIỆU CƠ BẢN VÀ CẤU TRÚC ĐIỀU KHIỂN	14
3.1 LÝ THUYẾT	14
3.1.1 Cấu trúc một chương trình C đơn giản	14
3.1.2 Các kiểu dữ liệu cơ bản trong C	15
3.1.3 Các phép toán	16

3.1.4	Các hàm thư viện cơ bản	18
3.1.5	Cấu trúc tuần tự	19
3.1.6	Cấu trúc rẽ nhánh	20
3.1.7	Cấu trúc lặp	23
3.1.8	Cấu trúc nhảy	26
3.1.9	Thực thi chương trình	27
3.2	BÀI TẬP	28
3.3	TÓM TẮT	36
4	HÀM	37
4.1	LÝ THUYẾT	37
4.1.1	Khái niệm	37
4.1.2	Cấu trúc một chương trình C	38
4.1.3	Cách xây dựng một hàm con	39
4.2	BÀI TẬP	43
4.3	TÓM TẮT	46
5	Kiểu dữ liệu mảng một chiều	47
5.1	LÝ THUYẾT	47
5.2	BÀI TẬP	48
5.3	TÓM TẮT	61
6	Kiểu dữ liệu chuỗi ký tự	62
6.1	LÝ THUYẾT	62
6.2	BÀI TẬP	65
6.3	TÓM TẮT	67
7	Kiểu dữ liệu mảng hai chiều	68
7.1	LÝ THUYẾT	68
7.2	BÀI TẬP	69
7.3	TÓM TẮT	80
8	Kiểu dữ liệu cấu trúc	81
8.1	LÝ THUYẾT	81
8.2	BÀI TẬP	97

8.3	TÓM TẮT	102
9	Kiểu dữ liệu tập tin	103
9.1	LÝ THUYẾT	103
9.1.1	Khái niệm tập tin	103
9.1.2	Thao tác với tập tin	103
9.1.3	Các ví dụ minh hoạ	106
9.2	BÀI TẬP	110
9.3	TÓM TẮT	114
10	Kiểu dữ liệu con trỏ	115
10.1	LÝ THUYẾT	115
10.2	BÀI TẬP	123
10.3	TÓM TẮT	123
11	Kỹ thuật đệ qui	124
11.1	LÝ THUYẾT	124
11.1.1	Các loại đệ qui	125
11.1.2	Các ví dụ	130
11.2	BÀI TẬP	131
11.3	TÓM TẮT	133
12	Lập trình đơn thể	134
12.1	LÝ THUYẾT	134
13	Một số đề thi mẫu	135
	Tài liệu tham khảo	145

CÁC KHÁI NIỆM CƠ BẢN VỀ LẬP TRÌNH

An algorithm must be seen to be believed.

Donald Ervin Knuth

1.1 KHÁI NIỆM LẬP TRÌNH

Định nghĩa.

- CPU của máy tính được thiết kế để có thể thực hiện được các *chương trình mã máy* (**machine code program**) đã được hệ điều hành (HĐH) nạp vào RAM của máy tính.
- Chương trình mã máy thường phải tương thích với từng họ máy cụ thể, bao gồm tập hợp các chỉ thị được viết bằng các lệnh CPU của họ máy đó, được lưu trên đĩa dưới dạng một *tập tin mã thực thi* (**executable program file**) của HĐH cụ thể

Quy trình thực hiện

- Bước 1: *Người sử dụng* (**end user**) ra lệnh thực hiện chạy chương trình.
- Bước 2: HĐH nhận được lệnh sẽ thực hiện:
 - Tìm và nạp tập tin mã thực thi của chương trình (nằm trên đĩa) vào RAM của máy tính.
 - Bộ đếm lệnh của CPU (**CPU program counter**) được trỏ đến lệnh đầu tiên của chương trình hay còn gọi là *ngõ vào chương trình* (**program entry point**)
- Bước 3: CPU thực hiện từng chỉ thị một trong RAM cho đến khi gặp lệnh kết thúc:
 - Chép lệnh mã máy hiện hành vào thanh ghi lệnh.

- Tăng bộ đếm lệnh (để trở đến lệnh kế tiếp).
- Thi hành lệnh mã máy.
- Bước 4: Kết thúc thực hiện chương trình, HĐH chờ nhận lệnh mới.

Đặc điểm

- Mỗi chỉ thị của chương trình là một lệnh mã máy (một dãy các byte chỉ phù hợp với qui ước tập lệnh của một loại CPU nào đó)
- Được cấu trúc hóa theo qui ước của HĐH.
- Được chạy trên một họ CPU và HĐH cụ thể.
- Nội dung rất khó hiểu đối với người dùng máy tính, chỉ có CPU thích hợp với hiểu rõ và thi hành được.

Nhận xét

- Khó có thể sản xuất ra phần mềm bằng cách viết trực tiếp các chương trình mã máy.
- Nếu có làm được theo cách này thì
 - Giá cả sẽ rất đắt do quá khó, tốn quá nhiều thời gian và công sức.
 - Khả năng dùng lại rất giới hạn do không thể bán chon người dùng trên họ máy tính khác hay người dùng sử dụng hệ điều hành khác.

Định nghĩa.

- *Ngôn ngữ lập trình* - NNLT (**programming language**) là ngôn ngữ được lập trình viên sử dụng để viết chương trình cho máy tính.
- Khi một chương trình được viết bằng một NNLT nào đó thì các chỉ thị, câu lệnh trong chương trình phải tuân theo các qui tắc, các luật do NNLT đó qui định.
- Chương trình viết bằng ngôn ngữ lập trình được gọi là *chương trình nguồn* (**source code program**) hay *mã nguồn* (**source code**). Chương trình nguồn được dịch sang chương trình mã máy bằng cách chương trình dịch:
 - *Trình hợp dịch* (**assembler**) để dịch các chương trình hợp ngữ.
 - *Trình thông dịch* (**interpreter**) và *trình biên dịch* (**compiler**) để dịch các chương trình cấp cao.

Ví dụ. Một số ngôn ngữ lập trình thông dụng

- Ngôn ngữ cấp thấp: Hợp ngữ (**assembly language**)
- Ngôn ngữ cấp cao: C/C++, Java, C#, Pascal, Python, PHP, Ruby, Perl, Lisp

Định nghĩa.

- *Chương trình* (**program**) là một dãy các *chỉ thị* (**instruction**) điều khiển sự hoạt động của máy tính nhằm giải quyết một công việc nào đó.
- Người viết chương trình hay còn gọi là lập trình viên hay thảo chương viên (**programmer**) là những người tạo lập ra những chương trình máy tính.

Ví dụ 1.1

Một số chương trình

- Ngôn ngữ assembly

```
.model tiny
.code
org 100h
    main    proc
        mov     ah,9
        mov     dx,offset hello_message
        int     21h
        retn
        hello_message db 'Hello, world!$'
    main    endp
end main
```

- Ngôn ngữ C/C++

```
#include <stdio.h>
void main(void)
{
    printf("Hello world!");
}
```

- Ngôn ngữ Java

```
public class Hello {
    public static void main(String argv[])
    {
        System.out.print("Hello everybody!");
    }
}
```

```
}
}
```

Đặc điểm ngôn ngữ lập trình cấp thấp

- Là NNLT phụ thuộc vào từng họ máy cụ thể, vì vậy không có tính tương thích.
- Dễ viết, đọc, sửa hơn chương trình mã máy.
- Ưu điểm là tận dụng và khai thác được tính năng của mỗi họ máy cụ thể, nhờ vậy chương trình có thể chạy nhanh hơn.

Đặc điểm ngôn ngữ lập trình cấp cao

- Được đề xuất để khắc phục các hạn chế của NNLT cấp thấp.
- Dễ dùng và dễ diễn đạt được các ý tưởng trừu tượng.
- Có tính tương thích cao (khi thay đổi dạng máy tính thì chỉ cần sửa chương trình rất ít hoặc thậm chí không cần sửa mà vẫn đảm bảo chạy đúng).

1.2 CÔNG NGHỆ LẬP TRÌNH TRUYỀN THỐNG

Đối với các NNLT cấp cao truyền thống (trước thế hệ của Java và C#), quá trình viết, dịch và chạy chương trình gồm các công đoạn như sau:

- B1. *Soạn* chương trình nguồn và lưu lên đĩa.
- B2. *Dịch* chương trình nguồn nhờ trình biên dịch.
- B3. *Nối kết* các tập tin mã trung gian tạo ra ở B2.
- B4. *Chạy* chương trình ngôn ngữ máy tạo ra ở B3.

1.3 CÔNG NGHỆ LẬP TRÌNH HIỆN ĐẠI

- Hạn chế của các chương trình cấp cao truyền thống là trình biên dịch của chúng phát sinh trực tiếp mã thực thi phụ thuộc vào mã máy tính của một họ máy tính và hệ điều hành cụ thể nên không thể mang đi sử dụng ở các hệ điều hành khác.
- NNLT hiện đại như Java hay C# trình biên dịch không dịch trực tiếp mã nguồn thành mã thực thi mà được thiết kế để có thể dịch thành *mã thực thi trừu tượng* (**abstract executable code**) độc lập máy và hệ điều hành.

- Do máy tính thật không thể hiểu được mã trừu tượng nên những chương trình dạng mã thực thi trừu tượng chỉ chạy được khi có sẵn máy ảo hỗ trợ cho việc thi hành loại mã thực thi đó.
- Trong các năm gần đây, các ứng dụng chạy trên web phát triển rất mạnh.
 - Chạy trên **Internet** thông qua một trình duyệt web.
 - Được viết bằng các ngôn ngữ như Python, PHP, ASP.NET, JSP, Java Script, VB Script... có tính tương thích cao, hoạt động trên bất kỳ máy tính nào có internet

1.3.1 Công cụ và môi trường lập trình

Toàn bộ qui trình biên dịch được thực một cách dễ dàng và thuận tiện nhờ vào công cụ gọi là *môi trường phát triển phần mềm* - IDE (**integrated development environment**)

- Soạn thảo chương trình.
- Quản lý hệ thống tập tin mã nguồn.
- Quản lý hệ thống các phiên bản của mã nguồn.
- Kiểm tra lỗi cú pháp (**syntax error**), biên dịch (**compile**), liên kết chương trình (**link**).
- Chạy từng dòng lệnh (debug) để tìm lỗi.

Ví dụ. Một số IDE thông dụng

- Eclipse: hỗ trợ nhiều ngôn ngữ lập trình.
- Visual Studio: hỗ trợ nhiều ngôn ngữ lập trình

1.4 THUẬT TOÁN

Các bước thiết kế chương trình

- Xác định bài toán
- Phân tích bài toán
- Thiết kế thuật toán
- Cài đặt chương trình
- Thử nghiệm chương trình

Định nghĩa. *Thuật toán* (**algorithm**) là tập hợp hữu hạn các *chỉ thị* được định nghĩa rõ ràng nhằm giải quyết một vấn đề cụ thể nào đó.

Tính chất. *Một thuật toán phải có các tính chất sau*

- *Tính chính xác: quá trình tính toán hay các thao tác máy tính thực hiện là chính xác.*
- *Tính rõ ràng: các câu lệnh minh bạch được sắp xếp theo thứ tự nhất định.*
- *Tính khách quan: được viết bởi nhiều người trên máy tính nhưng kết quả phải như nhau.*
- *Tính phổ dụng: có thể áp dụng cho một lớp các bài toán có đầu vào tương tự nhau.*
- *Tính kết thúc: hữu hạn các bước tính toán*

Trình bày thuật toán như thế nào?

- Trình bày bằng ngôn ngữ tự nhiên
- Trình bày bằng mã giả
- Trình bày bằng lưu đồ

Ví dụ 1.2

Hãy trình bày cách làm món sữa dâu bằng ngôn ngữ tự nhiên

Cách 1

- Lấy một ít sữa.
- Đổ nước ép dâu vào.
- Trộn hỗn hợp này và làm lạnh.

Cách 2

- Rót một ly sữa vào máy xay
- Đổ thêm vào một ít nước dâu ép
- Đóng nắp máy xay
- Mở điện và bắt đầu trộn
- Dừng máy trộn lại
- Nếu đã trộn đều thì tắt máy, ngược lại thì trộn tiếp
- Khi đã trộn xong, rót hỗn hợp vào tô và đặt vào tủ lạnh
- Để lạnh một lúc rồi lấy ra dùng

Ví dụ 1.3

Hãy trình bày cách sắp xếp một dãy số L theo thứ tự tăng dần bằng ngôn ngữ mã giả

```
MERGESORT( $L$ )
```

```
    if SIZE( $L$ ) > 1
```

`SPLIT`(L , L_1 , L_2) (tách dãy L thành L_1 và L_2)

`MERGESORT`(L_1) (sắp xếp dãy L_1)

`MERGESORT`(L_2) (sắp xếp dãy L_2)

`MERGE`(L_1 , L_2 , L) (trộn dãy L_1 và L_2 thành L)

LƯU ĐỒ THUẬT TOÁN

A picture is worth a thousand words

thành ngữ

Chương này sẽ trình bày các ký hiệu biểu diễn lưu đồ thuật toán, cách biểu diễn các cấu trúc điều khiển rẽ nhánh, cấu trúc lặp và các kỹ thuật liên quan đến lưu đồ thuật toán.

2.1 LÝ THUYẾT

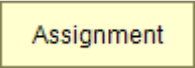
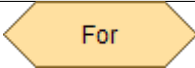
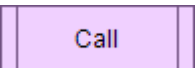
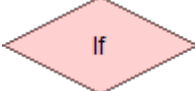

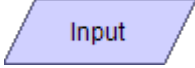
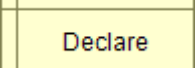
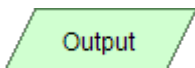
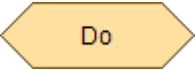

2.1.1 Khái niệm

Lưu đồ thuật toán là công cụ đồ thị dùng để biểu diễn thuật toán, việc mô tả nhập dữ liệu và xuất dữ liệu và xử lý dữ liệu thông qua các ký hiệu hình học.

2.1.2 Phần mềm công cụ

Có rất nhiều công cụ để vẽ lưu đồ từ offline cho đến online. Giáo trình sử dụng phần mềm Flowgorithm để vẽ và thực thi lưu đồ

2.1.3 Các ký hiệu

Lệnh	Ký hiệu	Lệnh	Ký hiệu
Assignment		For	
Call		If	
Comment		Input	
Declare		Output	
Do		While	

2.1.4 Phương pháp vẽ và thực thi

- Vẽ từ trên xuống
- Chạy bắt đầu từ Begin và kết thúc tại End
- Đi theo hướng mũi tên

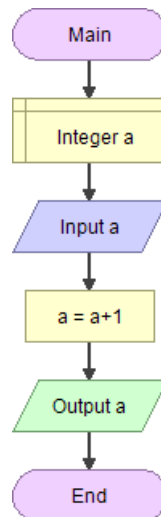
2.1.5 Các cấu trúc điều khiển cơ bản

Cấu trúc tuần tự

Tuần tự thực thi tiến trình. Mỗi lệnh được thực thi theo một chuỗi từ trên xuống, xong lệnh này rồi chuyển xuống lệnh kế tiếp.

Ví dụ 2.1

Nhập vào một số nguyên a và xuất ra màn hình với giá trị tăng lên 1

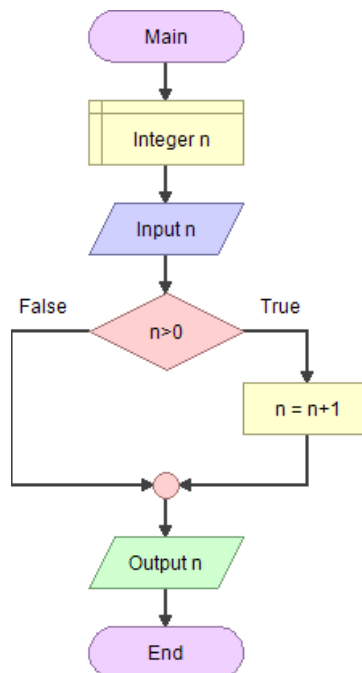


Cấu trúc rẽ nhánh

Điểm quyết định cho phép chọn một trong hai trường hợp đúng hay sai

Ví dụ 2.2

Nhập vào số nguyên n . Kiểm tra nếu $n > 0$ tăng n lên 1 đơn vị

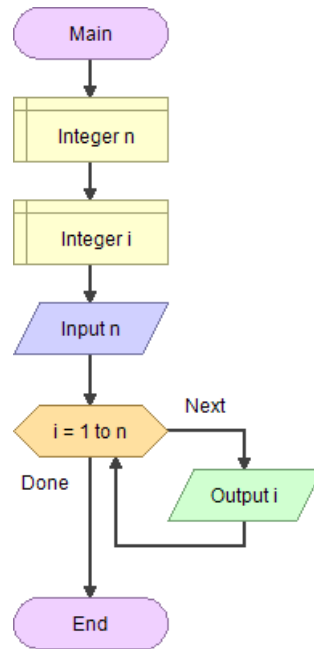


Cấu trúc lặp

Thực hiện liên tục 1 lệnh hay tập lệnh với số lần lặp dựa vào điều kiện. Lặp sẽ kết thúc khi điều kiện được thỏa.

Ví dụ 2.3

Nhập vào số nguyên n . Xuất ra màn hình từ 1 đến n .

**Các ví dụ thực hành**

- 2.1. Giải và biện luận phương trình: $ax + b = 0$.
- 2.2. Tính tổng: $S(n) = 1 + 2 + 3 + \dots + n$ với $n > 0$
- 2.3. Tính tổng: $S(n) = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \dots + \frac{2n+1}{2n+2}$ với $n > 0$
- 2.4. Tính tổng: $S(n) = 1 - 2 + 3 - \dots + (-1)^{n+1}n$ với $n > 0$

2.2 BÀI TẬP**Bài tập cơ bản**

- 2.5. Nhập vào hai số x, y . Xuất ra màn hình tổng, hiệu, tích, thương của hai số trên.
- 2.6. Nhập vào số nguyên n , kiểm tra xem n chẵn hay lẻ và xuất ra màn hình.
- 2.7. Nhập vào ba cạnh a, b, c của tam giác. Xuất ra màn hình tam giác đó thuộc loại tam giác gì? (thường, cân, vuông, đều hay vuông cân).
- 2.8. Nhập vào số nguyên n . Xuất n ra màn hình (nếu n chẵn thì gấp đôi giá trị).
- 2.9. Nhập vào số nguyên n . Nếu $n > 5$ thì tăng n lên 2 đơn vị và trả về giá trị n , ngược lại trả về giá trị 0.

2.10. Tính $n!$, với $n \geq 0$

2.11. Tính $P(n) = 1.3.5...(2n+1)$, với $n \geq 0$

2.12. Tính $S(n) = 1 + 3 + 5 + ... + (2n+1)$, với $n \geq 0$

2.13. Tính $S(n) = 1 - 2 + 3 - 4 + ... + (-1)^{n+1}n$, với $n > 0$

2.14. Tính $S(n) = 1 + 1.2 + 1.2.3 + ... + 1.2.3...n$, với $n > 0$

2.15. Tính $S(n) = 1^2 + 2^2 + 3^2 + ... + n^2$, với $n > 0$

2.16. Tính

$$S(n) = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n}$$

với $n > 0$

2.17. (*) Tính

$$S(n) = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + ... + \frac{1}{1+2+3+...+n}$$

với $n > 0$

2.18. Tính $P(x, y) = x^y$

2.19. Tính $S(n) = 1 + (1+2) + (1+2+3) + ... + (1+2+3+...+n)$ với $n > 0$

2.20. Cho số nguyên n . Tính trị tuyệt đối của n .

2.21. Cho số nguyên dương n gồm k chữ số. Tìm chữ số có giá trị lớn nhất.

2.22. Đếm số lượng ước số chẵn của số nguyên dương n .

2.23. In ra chữ số đầu tiên của số nguyên dương n gồm k chữ số.

2.24. Cho 2 số nguyên dương a, b . Tìm USCLN của a và b .

2.25. Cho 2 số nguyên dương a, b . Tìm BSCNN của a và b .

2.26. Cho số nguyên dương x . Kiểm tra xem x có phải là số nguyên tố không?

2.27. Cho số nguyên dương x . Kiểm tra x có phải là số chính phương không?

2.28. Cho số nguyên dương x . Kiểm tra xem x có phải là số hoàn thiện không?

Luyện tập và nâng cao

2.29. Tính $S(n) = 1 + 2^2 + 3^3 + ... + n^n$, với $n > 0$

2.30. Tính

$$S(n) = \frac{1}{2} + \frac{2}{3} + ... + \frac{n}{n+1}$$

với $n > 0$

2.31. Tính

$$S(n) = 1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}$$

với $n > 0$

2.32. Tính

$$S(n) = 1 + \frac{1+2}{2!} + \frac{1+2+3}{3!} + \dots + \frac{1+2+3+\dots+n}{n!}$$

với $n > 0$

2.33. Giải và biện luận phương trình bậc hai $ax^2 + bx + c = 0$

2.34. Giải và biện luận phương trình trùng phương bậc bốn $ax^4 + bx^2 + c = 0$

2.35. (*) Tính $S(n) = \sqrt{n + \sqrt{n-1 + \sqrt{n-2 + \dots + \sqrt{1}}}}$ với $n > 0$

2.36. (**) Tính $S(n) = \sqrt{1 + \sqrt{2 + \sqrt{3 + \dots + \sqrt{n}}}}$ với $n > 0$

2.3 TÓM TẮT

Lưu đồ thuật toán rất là một công cụ hữu ích trong việc mô tả cách giải quyết của một bài toán. Việc mô tả này rất trực quan thông qua các ký hiệu hình học, đây là giai đoạn đầu tiên trước khi bắt tay vào lập trình trên một ngôn ngữ lập trình cụ thể. Khi xây dựng lưu đồ thuật toán, chúng ta cần chú ý một vài điểm sau: Một lưu đồ phải có điểm bắt đầu và điểm kết thúc. Phải có dữ liệu vào, dữ liệu ra sau khi xử lý tính toán. Tại mỗi vị trí quyết định lựa chọn rẽ nhánh phải ghi rõ điều kiện đúng hoặc sai thì đi theo nhánh nào.

KIỂU DỮ LIỆU CƠ BẢN VÀ CẤU TRÚC ĐIỀU KHIỂN

C is quirky, flawed, and an enormous success.

Dennis M. Ritchie

Chương này trình bày các kiểu dữ liệu cơ bản và các phép toán tương ứng trong ngôn ngữ C, các cấu trúc rẽ nhánh, lặp. Mô tả cách hoạt động và hướng dẫn chạy từng bước chương trình.

3.1 LÝ THUYẾT

3.1.1 Cấu trúc một chương trình C đơn giản

- Khai báo thư viện
- Khai báo biến toàn cục
- Chương trình chính (`main`)

Ví dụ 3.1

Một chương trình đơn giản

```
// Khai bao thu vien
#include <stdio.h>
#include <conio.h>
// Khai bao bien toan cuc
int a, b;
// Chuong trinh chinh
```

```

void main()
{
    printf("Hello world!");
    getch();
}

```

3.1.2 Các kiểu dữ liệu cơ bản trong C

Kiểu số nguyên

Tên kiểu	kích thước (byte)
char	1
unsigned char	1
int	4
unsigned int	4
long	8
unsigned long	8

Kiểu số thực

Tên kiểu	kích thước (byte)
float	4
double	8

Kiểu ký tự

Tên kiểu	kích thước (byte)
char	1
unsigned char	1

Kiểu luận lý

Tên kiểu	kích thước (byte)
bool	1

3.1.3 Các phép toán

Phép toán số học

Phép toán	C/C++	Ví dụ
Cộng	+	a+b
Trừ	-	a-b
Nhân	*	a*b
Chia	/	a/b
Dư	%	a%b

phép toán so sánh

Phép toán	C/C++	Ví dụ
Lớn hơn	>	a>b
Nhỏ hơn	<	a<b
Bằng	==	a==b
Khác	!=	a!=b
Lớn hơn hoặc bằng	>=	a>=b
Nhỏ hơn hoặc bằng	<=	a<=b

phép toán logic

Phép toán	C/C++	Ví dụ
phủ định	!	!a
và	&&	a&&b
hay		a b

phép toán thao tác trên bit

Phép toán	C/C++	Ví dụ
NOT	~	~a
AND	&	a&b
OR		a b
XOR	^	a^b
SHIFT LEFT	<<	a<<b
SHIFT RIGHT	>>	a>>b

phép toán tăng giảm

Phép toán	C/C++	Ví dụ
Tăng biến một đơn vị	++	a++ hoặc ++a
Giảm biến một đơn vị	-	a- hoặc -a

phép toán gán và gán mở rộng

Phép toán	C/C++	Ví dụ
Gán biến một giá trị	=	a=b

Phép toán gán mở rộng	Ví dụ	Ý nghĩa
+=	a+=b	a=a+b
-=	a-=b	a=a-b
=	a=b	a=a*b
/=	a/=b	a=a/b
%=	a%=b	a=a%b
>>=	a>>=b	a=a>>b
<<=	a<<=b	a=a<<b
&=	a&=b	a=a&b
^=	a^=b	a=a^b
=	a =b	a=a b

toán tử điều kiện

Phép toán	C/C++	Ví dụ
điều kiện	?:	a?b:c

độ ưu tiên và kết hợp của các toán tử

Loại	Toán tử	Kết hợp
Postfix	() [] -> . ++ --	trái sang phải
Unary	+ - ! ~ ++ -- (type)* & sizeof	phải sang trái
Multiplicative	* / %	trái sang phải
Additive	+ -	trái sang phải
Shift	<< >>	trái sang phải
Relational	< <= > >=	trái sang phải
Equality	== !=	trái sang phải
Bitwise AND	&	trái sang phải
Bitwise XOR	^	trái sang phải
Bitwise OR		trái sang phải
Logical AND	&&	trái sang phải
Logical OR		trái sang phải
Conditional	?:	phải sang trái
Assignment	= += -= *= /= %= >>= <<= &= ^= =	phải sang trái
Comma	,	trái sang phải

3.1.4 Các hàm thư viện cơ bản

Thư viện xuất nhập chuẩn "stdio.h"

Tên hàm	Ý nghĩa	Ví dụ
printf(...);	xuất dữ liệu ra màn hình	printf("%d %d", a, b);
scanf(...);	nhập dữ liệu từ bàn phím	scanf("%d%d", &a, &b);

Thư viện toán học "math.h"

Tên hàm	Ý nghĩa
<code>abs(x)</code>	$ a $
<code>sin(x)</code>	$\sin(x)$
<code>cos(x)</code>	$\cos(x)$
<code>pow(x,y)</code>	x^y
<code>sqrt(x)</code>	\sqrt{x}
<code>log(x)</code>	$\log(x)$
<code>exp(x)</code>	e^x

3.1.5 Cấu trúc tuần tự**Cú pháp**

<lệnh 1>

<lệnh 2>

...

<lệnh n>

Các câu lệnh được thực hiện từ trên xuống, bắt đầu từ <lệnh 1>, <lệnh 2> và cuối cùng là <lệnh n>

Ví dụ 3.2

Tính tổng hai số nguyên

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a, b, s;
    printf("Nhập a và b = ");
    s = a + b;
    printf("tong là = %d ", s);
    getch();
}
```

3.1.6 Cấu trúc rẽ nhánh

Câu lệnh if

Cú pháp

```
if (<biểu thức điều kiện>
{
    <khối lệnh>
}
```

Nếu <biểu thức điều kiện> đúng thì thực hiện <khối lệnh>. Lưu ý, <biểu thức điều kiện> phải được đặt trong cặp ngoặc

Ví dụ 3.3

Kiểm tra một số có lớn hơn 6

```
#include <stdio.h>
#include <conio.h>
void main()
{
    float number;
    printf("Nhap mot so trong khoang tu 1 den 10 = ");
    scanf("%f", &number);
    if(number>6)
        printf("So ban nhap lon hon 6. \n");
    printf("%f la so ban nhap. ", number);
    getch();
}
```

Câu lệnh if ... else

Cú pháp

```
if (<biểu thức điều kiện>
{
    <khối lệnh 1>
```

```

    }
    else
    {
        <khối lệnh 2>
    }

```

Nếu <biểu thức điều kiện> cho kết quả đúng thì thực hiện <khối lệnh 1>, ngược lại thì cho thực hiện <khối lệnh 2>.

Ví dụ 3.4

Giải và biện luận phương trình $ax + b = 0$

```

#include <stdio.h>
#include <conio.h>
void main()
{
    float a, b;
    printf("Nhap vao a:");
    scanf("%f", &a);
    printf("Nhap vao b:");
    scanf("%f", &b);
    if(a==0)
        if(b==0) printf("Phuong trinh vo so nghiem");
        else printf("Phuong trinh vo nghiem");
    else printf("Nghiem phuong trinh x=%f", -b/a);
    getch();
}

```

Câu lệnh switch

Cú pháp

```

switch (<biểu thức số>)
{
    case n1: <các câu lệnh 1> break;

```

```

    case n2: <các câu lệnh 2>    break;
    ...
    case nk: <các câu lệnh k>    break;
    [default: <các câu lệnh k+1>]
}

```

- Khi giá trị biểu thức bằng `ni` thì thực hiện câu lệnh sau case `ni`.
- Khi giá trị biểu thức không thỏa tất cả các `ni` thì thực hiện câu lệnh sau `default` nếu có, hoặc thoát khỏi câu lệnh `switch`.
- Khi chương trình đã thực hiện xong câu lệnh của case `ni` nào đó thì nó sẽ thực hiện luôn các lệnh thuộc case bên dưới nó mà không xét lại điều kiện (do các `ni` được xem như các nhãn) Vì vậy, để chương trình thoát khỏi lệnh `switch` sau khi thực hiện xong một trường hợp, ta dùng lệnh `break`.

Ví dụ 3.5

Tạo thực đơn và cho phép chọn thực đơn bằng số nhập từ bàn phím.

```

#include<stdio.h>
#include<conio.h>
void main()
{
    int chon;
    printf("Thuc Don");
    printf("\n1. Lau thai!");
    printf("\n2. Nuoc ngot!");
    printf("\n3. Ca loc hap bau!");
    printf("\n4. Chuot dong!");
    printf("\n Xin moi ban chon mon an!");
    scanf("%d",&chon);
    switch(chon)
    {
        case 1: printf("\nBan chon lau thai!"); break;
        case 2: printf("\nBan chon nuoc ngot!"); break;
        case 3: printf("\nBan chon ca loc hap bau!"); break;
    }
}

```

```

        case 4: printf("\Ban chon chuot dong!"); break;
        default: printf("\nBan chon khong dung!"); break;
    }
    getch();
}

```

3.1.7 Cấu trúc lặp

Câu lệnh for

Cú pháp

```

for (<biểu thức 1>; <biểu thức 2>; <biểu thức 3>)
{
    <các câu lệnh>
}

```

Hoạt động của câu lệnh **for** như sau:

- Bước 1: Khởi gán cho <biểu thức 1>
- Bước 2: Kiểm tra điều kiện của <biểu thức 2>
 - Nếu <biểu thức 2> đúng thì cho thực hiện <các câu lệnh> và thực hiện <biểu thức 3> rồi quay trở lại bước 2.
 - Ngược lại thì thoát khỏi lặp.
- Bất kỳ biểu thức nào trong 3 biểu thức nói trên đều có thể vắng nhưng phải giữ dấu chấm phẩy (;)

Ví dụ 3.6

In ra màn hình bảng mã ASCII từ ký tự số 32 đến 255.

```

#include <conio.h>
#include <stdio.h>
void main()
{
    for(int i=32; i<=255; i++)
        printf("Ma ASCII cua %c: %d\t", i, i);
}

```

```
    getch();
}
```

Câu lệnh while

Cú pháp

```
while (<biểu thức điều kiện>)
{
    <các câu lệnh>
}
```

Nếu <biểu thức điều kiện> đúng thì thực hiện <các câu lệnh> và lặp điều này cho đến khi nào <biểu thức điều kiện> sai thì kết thúc

Ví dụ 3.7

In ra các chữ số của số nguyên n

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n, chuso;
    printf("Nhập vào n = ");
    scanf("%d",&n);
    while(n>0)
    {
        chuso = n%10;
        printf("%d ", chuso);
        n = n/10;
    }
    getch();
}
```

Câu lệnh do ... while

Cú pháp

```
do {
    <các câu lệnh>
} while(<biểu thức điều kiện>);
```

Thực hiện <các câu lệnh> cho đến khi <biểu thức điều kiện> có giá trị sai.

Ví dụ 3.8

Nhập ký tự từ bàn phím hiển thị lên màn hình mã ASCII của ký tự đó, thực hiện đến khi nhấn phím ESC (Mã ASCII của phím ESC là 27).

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int ma;
    do{
        ma=getch();
        if(ma !=27)
            printf("Ma ASCII %c:%d\t", ma, ma);
    } while(ma!=27);
    getch();
}
```

Lưu ý. Lệnh lặp while kiểm tra điều kiện trước khi thực hiện lặp, còn lệnh lặp do...while thực hiện lệnh lặp rồi mới kiểm tra điều kiện. Do đó vòng lặp do...while thực hiện <các câu lệnh> ít nhất một lần.

Câu lệnh break và continue

Lệnh break dùng để kết thúc vòng lặp trực tiếp chứa nó

Ví dụ 3.9

Cho phép người dùng nhập liên tục giá trị n cho đến khi nhập số âm thì dừng lại

```
#include<stdio.h>
#include<conio.h>
void main()
{
    while(1)
    {
        printf("Nhap n = ");
        scanf("%d", &n);
        if(n<0) break;
    }
    getch();
}
```

Lệnh continue dùng để bỏ qua một lần lặp.

Ví dụ 3.10

In ra màn hình giá trị từ 1 đến 25 trừ đi số 14 và số 18.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    for(int i=1; i<=25; i++)
    {
        if(i==14 || i==18) continue;
        printf("%d\t", i);
    }
    getch();
}
```

3.1.8 Cấu trúc nhảy

Cú pháp

<nhãn>:


```
...
goto <nhãn>
```

```
...
```

con trỏ lệnh sẽ được nhảy đến vị trí của <nhãn>

Ví dụ 3.11

Kiểm tra một số nguyên là số dương, âm hay không?

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n;
    printf("Nhap vao so nguyen = ");
    scanf("%d", &n);
    if(n>0) goto nhan1;
    if(n<0) goto nhan2;
    if(n==0) goto nhan2;
nhan1:
    printf("so duong");
    goto nhan;
nhan2:
    printf("so am");
    goto nhan;
nhan3:
    printf("so khong");
nhan:
    getch();
}
```

3.1.9 Thực thi chương trình

- Xác định các biến trong chương trình.
- Giá trị ban đầu của mỗi biến.

- Thực hiện chạy chương trình theo đúng trình tự đã được viết

Ví dụ 3.12

Cho biết kết quả của đoạn chương trình sau:

```
void main()
{
    int i, a = 4;
    for(i = 0; i < a; i++)
        printf("%d\n", i);
}
```

3.2 BÀI TẬP

Bài tập cơ bản

Biến, biểu thức và toán tử

3.1. Chuỗi ký tự nào sau đây không thể là tên biến được?

BASICSALARY	_basic	basic-hra
#MEAN	group.	422
population in 2006	over time	mindovermatter
FLOAT	hELLO	queue.
team'svictory	Plot # 3	2015_DDay
nhiệt	van_toc	lai suat

3.2. Chỉ ra biểu thức C/C++ lỗi (nếu có)

- `int = 314.562 * 150;`
- `name = 'Ajay';`
- `varchar = '3';`
- `3.14 * r * r * h = vol_of_cyl;`
- `k = (a * b) (c + (2.5a + b) (d + e));`
- `m_inst = rate of interest * amount in rs;`
- `si = principal * rateofinterest * numberofyears / 100;`

- h) `area = 3.14 * r ** 2;`
- i) `volume = 3.14 * r ^ 2 * h;`
- j) `k = ((a * b) + c) (2.5 * a + b);`
- k) `a = b = 3 = 4;`
- l) `count = count + 1;`
- m) `date = '2 Mar 04';`

3.3. Xác định thứ tự thực hiện các phép toán và tính giá trị của biểu thức

- a) `g = big / 2 + big * 4 / big - big + abc / 3;` (`abc = 2.5`, `big = 2`, giả sử `g` có kiểu `float`)
- b) `on = ink * act / 2 + 3 / 2 * act + 2 + tig;` (`ink = 4`, `act = 1`, `tig = 3.2`, giả sử `on` có kiểu `int`)
- c) `s = qui * add / 4 - 6 / 2 + 2 / 3 * 6 / god;` (`qui = 4`, `add = 2`, `god = 2`, giả sử `s` có kiểu `int`)
- d) `s = 1 / 3 * a / 4 - 6 / 2 + 2 / 3 * 6 / g;` (`a = 4`, `g = 3`, giả sử `s` có kiểu `int`)

3.4. Xác định toán hạng cho các phép toán trong biểu thức

- a) `g = 10 / 5 / 2 / 1;`
- b) `b = 3 / 2 + 5 * 4 / 3;`
- c) `a = b = c = 3 + 4;`

3.5. Chuyển các biểu thức toán sau sang ngôn ngữ C/C++

$$A = \frac{\frac{8.8(a+b)^2}{c} - 0.5 + \frac{2a}{q+r}}{\frac{a+b}{m}}$$

$$B = \frac{-b + b^2 + 2.4ac}{2a}$$

$$C = \frac{2v + 6.22(c + d)}{g + v}$$

$$D = \frac{\frac{7.7b(xy+a)}{c} - 0.8 + 2b}{\frac{x+a}{y}}$$

Cấu trúc rẽ nhánh

3.6. Cho biết kết quả của đoạn chương trình sau:

```
int a=9, b=6;
a++;
a=a+-b;
a=a-+(b);
if(a%2==0) printf("Gia tri cua a la chan");
printf("Tong cua a va b la: %d", a+b);
```

3.7. Cho biết kết quả của đoạn chương trình sau:

```
int a=7, b=8;
a++;
a=a+(-b);-
b;-
a;
a-=(a)-+(b);
if(a%2!=0)
    printf("\n a la so le");
else
    printf("\n a la so chan");
printf("\na = %d",a);
```

3.8. Cho biết kết quả của đoạn chương trình sau:

```
int x=5, y;
y=x++ + 5;
printf("x=%d, y=%d\n", x, y);
y*=6;
x=y%7;
printf("x=%d,y=%d,y/x=%d", x, y, y/x);
```

3.9. Nhập vào hai số nguyên a, b . In ra màn hình giá trị lớn nhất.

3.10. Cho ba số a, b, c đọc vào từ bàn phím. Hãy tìm giá trị lớn nhất của ba số trên và in ra kết quả.

3.11. Cho ba số a, b, c đọc vào từ bàn phím. Hãy in ra màn hình theo thứ tự tăng dần các số. (Chỉ được dùng thêm hai biến phụ).

3.12. Viết chương trình nhập vào một số nguyên n gồm ba chữ số. Xuất ra màn hình chữ số lớn nhất ở vị trí nào? Ví dụ: $n=291$. Chữ số lớn nhất nằm ở hàng chục là 9.

- 3.13. Viết chương trình nhập vào số nguyên n gồm ba chữ số. Xuất ra màn hình theo thứ tự tăng dần của các chữ số. Ví dụ: $n=291$. Xuất ra 129.
- 3.14. Nhập vào ngày, tháng, năm. Kiểm tra xem ngày, tháng, năm đó có hợp lệ hay không? In kết quả ra màn hình.
- 3.15. Nhập vào giờ, phút, giây. Kiểm tra xem giờ, phút, giây đó có hợp lệ hay không? In kết quả ra màn hình.
- 3.16. Viết chương trình nhập vào ngày, tháng, năm hợp lệ. Cho biết năm này có phải là năm nhuận hay không? In kết quả ra màn hình.
- 3.17. Viết chương trình tính diện tích và chu vi các hình: tam giác, hình vuông, hình chữ nhật và hình tròn với những thông tin cần được nhập từ bàn phím.
- 3.18. Viết chương trình tính tiền cước TAXI. Biết rằng:
- a) km đầu tiên giá là 5000đ
 - b) 200m tiếp theo là 1000đ.
 - c) Nếu lớn hơn 30km thì mỗi km thêm sẽ là 3000đ
- Hãy nhập số km sau đó in ra số tiền phải trả.
- 3.19. Nhập vào 3 số nguyên dương a, b, c . Kiểm tra xem 3 số đó có lập thành tam giác không? Nếu có hãy cho biết tam giác đó thuộc loại nào? (cân, vuông, đều, ...).
- 3.20. Viết chương trình nhập vào số nguyên dương n . Kiểm tra xem n có phải là số chính phương hay không? (số chính phương là số khi lấy căn bậc 2 có kết quả là nguyên).

Cấu trúc lặp

- 3.21. Cho biết kết quả của đoạn chương trình sau:

```
int a=18;
for(int i=1; i<=a; i++)
    if(a%i==0) printf("\t %d", i);
```

- 3.22. Cho biết kết quả của đoạn chương trình sau:

```
for(int i=0; i<5; i++)
{
    for(int j=0; j<=i; j++)
        printf("%d\t", j);
    printf("\n");
}
```

3.23. Cho biết kết quả của đoạn chương trình sau:

```
int i=10, s=0;
while(i>0)
{
    if(i%2==0)
        s+=i;
    else
        if(i>5)
            s+=2*i;-
    i;
}
printf("s = %d",s);
```

3.24. Cho biết kết quả của đoạn chương trình sau:

```
int a=18, i=1;
do{
    if(a%i==0)
        printf("\t %d",i);
    i++;
} while(i<=a);
```

3.25. Cho biết kết quả của đoạn chương trình sau:

```
int a=11, b=16, i=a;
while(i<b)
{
    if(i%2==0)
    {
        printf("\t %d", i);
        break;
    }
    i++;
}
```

3.26. Cho biết kết quả của đoạn chương trình sau:

```
int a=10, s=0, i=0;
while(i<a)
```

```
{
    i++;
    if(i%2==0)
        continue;
    else
        s=s+i;
}
printf("s=%d",s);
```

3.27. Cho biết kết quả của đoạn chương trình sau:

```
int i=1,s=0;
while(1)
{
    s=s+i++;
    if(i%2)
        i=i+2;
    else
        i=i+1;
    if(i>20) break;
}
printf("%d",s);
```

3.28. Viết chương trình in ra màn hình hình chữ nhật đặc kích thước $m \times n$. Ví dụ: Nhập $m = 5, n = 4$

```
* * * *
* * * *
* * * *
* * * *
* * * *
```

3.29. Viết chương trình in ra màn hình hình chữ nhật rỗng kích thước $m \times n$. Ví dụ: Nhập $m = 5, n = 4$

```
* * * *
*      *
*      *
*      *
*      *
```

* * * *

3.30. Viết chương trình in ra màn hình tam giác vuông cân đặc có độ cao h . Ví dụ: Nhập $h = 4$

```
*
* *
* * *
* * * *
```

3.31. Viết chương trình in ra màn hình tam giác cân rỗng có độ cao h . Ví dụ: Nhập $h = 4$

```
*
* *
*   *
* * * *
```

3.32. Viết chương trình in ra màn hình tam giác cân đặc có độ cao h . Ví dụ: Nhập $h = 4$

```
      *
    * * *
  * * * * *
* * * * * *
```

3.33. Viết chương trình in ra màn hình tam giác cân rỗng có độ cao h . Ví dụ: Nhập $h = 4$

```
      *
    *   *
  *       *
* * * * * *
```

3.34. Viết chương trình nhập số nguyên dương n . Liệt kê n số nguyên tố đầu tiên.

3.35. Viết chương trình nhập vào hai số nguyên dương a và b . Tìm ước số chung lớn nhất và bội số chung nhỏ nhất của a và b .

3.36. Viết chương trình nhập vào một số nguyên n gồm tối đa 10 chữ số (4 bytes). In ra màn hình giá trị nhị phân của số trên.

3.37. Viết chương trình đếm số ước số của số nguyên dương n . Ví dụ: $n = 12$ số ước số của 12 là 6

3.38. Một số hoàn thiện là một số có tổng các ước số của nó (không kể nó) bằng chính nó. Hãy liệt kê các số hoàn thiện nhỏ hơn 5000. Ví dụ: số 6 là số hoàn thiện vì tổng các ước số là $1+2+3=6$.

3.39. Nhập vào ngày, tháng, năm. Cho biết đó là ngày thứ mấy trong năm.

3.40. In ra dãy số Fibo f_n

$$f_n = \begin{cases} 1 & n = 1 \\ 1 & n = 2 \\ f_{n-1} + f_{n-2} & n > 2 \end{cases}$$

Luyện tập và nâng cao

3.41. Cài đặt tất cả các lưu đồ đã vẽ ở chương 1.

3.42. Nhập vào ngày, tháng, năm. Kiểm tra xem ngày, tháng, năm đó có hợp lệ hay không, nếu hợp lệ cho biết ngày sau đó là bao nhiêu. Ví dụ: Nhập 31/12/2003 Ngày sau đó 01/01/2004

3.43. Nhập vào ngày, tháng, năm. Kiểm tra xem ngày, tháng, năm đó có hợp lệ hay không, nếu hợp lệ cho biết ngày trước đó là bao nhiêu. Ví dụ: Nhập 01/01/2003 Ngày trước đó 31/12/2002

3.44. (*) Nhập vào ngày, tháng, năm của năm 2003. Hãy kiểm tra xem dữ liệu có hợp lệ hay không? Nếu hợp lệ hãy cho biết đó là ngày thứ mấy trong tuần. (hai, ba, tư, ..., CN).(Hướng dẫn: lấy ngày 01 tháng 01 năm 2003 là ngày thứ tư làm mốc).

3.45. Nhập vào giờ, phút, giây. Kiểm tra xem giờ, phút, giây đó có hợp lệ hay không, nếu hợp lệ cho biết giờ sau đó 1 giây là bao nhiêu. Ví dụ: Nhập 01:59:59 Giờ sau đó 1 giây 02:00:00

3.46. Nhập vào giờ, phút, giây. Kiểm tra xem giờ, phút, giây đó có hợp lệ hay không, nếu hợp lệ cho biết giờ trước đó 1 giây là bao nhiêu. Ví dụ: Nhập 02:00:00 Giờ trước đó 1 giây 01:59:59

3.47. Viết chương trình in ra bảng cửu chương từ 2 đến 9.

3.48. (*) Vẽ hình cánh quạt sau với h là chiều dài cánh quạt. Ví dụ $h = 4$

```

*      * * * *
* *    * * *
* * * * *
* * * * * * *
      * * * *
    * * *   * *
* * * *   *
```

3.3 TÓM TẮT

Cấu tuần tự, trúc lặp và rẽ nhánh (lựa chọn) là ba cấu trúc chính hình thành nên chương trình. Dựa vào những cấu trúc điều khiển này ta có thể xây dựng thành những chương trình phức tạp hơn. Vì vậy phải nắm rõ cách hoạt động của những cấu trúc điều khiển này để cài đặt đúng yêu cầu bài toán. Khi sử dụng phải lưu ý điều kiện thực hiện hay kết thúc của một thao tác nào đó. Bên trong một phát biểu điều khiển phải là một lệnh hay một khối lệnh (khối lệnh được đặt bên trong cặp dấu ngoặc {}). Những biến không phụ thuộc vào vòng lặp nên đặt bên ngoài vòng lặp. Khi sử dụng cấu trúc điều khiển lồng nhau phải lưu ý vị trí mở ngoặc hay đóng ngoặc cho hợp lý.

Chương này trình bày cấu trúc của một chương trình, các bước xây dựng cài đặt chương trình theo phương pháp thủ tục hàm và một số kỹ thuật liên quan.

4.1 LÝ THUYẾT

4.1.1 Khái niệm

Hàm là một đoạn chương trình độc lập thực hiện trọn vẹn một công việc nhất định sau đó trả về giá trị cho chương trình gọi nó, hay nói cách khác hàm là sự chia nhỏ của chương trình. Hàm được sử dụng khi

- Có một công việc giống nhau cần thực hiện ở nhiều lúc khác nhau.
- Khi cần chia một chương trình lớn phức tạp thành các chương trình nhỏ để dễ quản lý việc tính toán và giải quyết vấn đề cũng như dễ hiểu.

Ví dụ 4.1

Một chương trình có sử dụng hàm đơn giản

```
// Khai bao thu vien
#include <stdio.h>
#include <conio.h>
// Khai bao bien toan cuc
float x,y;
int n;
// Khai bao ham
```

```

float LuyThua(float x, int n);
// Ham chinh
void main()
{
    printf("Nhap x va n =");
    scanf("%f%d", &x, &n);
    y = LuyThua(x, n);
}
// Cai dat ham
float LuyThua(float x, int n)
{
    float y = 1;
    for(int i=1; i<=n; i++)
        y = y*x;
    return y;
}

```

4.1.2 Cấu trúc một chương trình C

Phần khai báo

Bao gồm các

- Khai báo thư viện sử dụng
- Khai báo hằng số
- Khai báo kiểu dữ liệu tự định nghĩa
- Khai báo các biến toàn cục
- Khai báo các hàm hay nguyên mẫu hàm

Hàm chính main

Thực hiện các công việc và các lời gọi hàm cần thiết.

Các hàm

Thực hiện các công việc xác định.

4.1.3 Cách xây dựng một hàm con

Khai báo hàm

Cú pháp

```
<kiểu dữ liệu trả về> <tên hàm>([danh sách các tham số]);
```

Lưu ý tham số trong nguyên mẫu hàm có thể bỏ phần tên.

Cài đặt hàm

Cú pháp

```
<kiểu dữ liệu trả về> <tên hàm>([danh sách các tham số])
{
    <các câu lệnh>
}
```

Gọi hay thực thi hàm

Cú pháp

```
<tên hàm>([danh sách các tham số]);
```

Có 3 cách truyền tham số:

- Truyền bằng giá trị (**pass by value**)
 - Tham số gọi là tham trị
 - Truyền đối số cho hàm ở dạng giá trị.
 - Có thể truyền hằng, biến, biểu thức.
 - Được sử dụng khi không có nhu cầu thay đổi giá trị của tham số sau khi thực hiện hàm.

```
void Ham(int x)
{
    // ...
    y = x;
    // ...
}

void main()
```

```

{
    int a, b;
    // ...
    Ham(3);
    Ham(a);
    Ham(a+b);
    ...
}

```

- Truyền bằng địa chỉ (**pass by address**)
 - Tham số gọi là tham trỏ
 - Truyền đối số cho hàm ở dạng địa chỉ (con trỏ).
 - Chỉ truyền đối số là biến.
 - Được sử dụng khi có nhu cầu thay đổi hoặc nhận giá trị của đối số sau khi thực hiện hàm.

```

void Ham(int *x)
{
    // ...
    y = *x;
    // ...
}

void main()
{
    int a;
    // ...
    Ham(&a);
    // ...
}

```

- Truyền bằng tham chiếu (**pass by reference**)
 - Tham số gọi là tham biến
 - Truyền đối số cho hàm ở dạng tham chiếu.
 - Chỉ truyền đối số là biến.
 - Được sử dụng khi có nhu cầu thay đổi hoặc nhận giá trị của đối số sau khi thực hiện hàm.

```

void Ham(int &x)
{
    // ...
    y = x;
    // ...
}

void main()
{
    int a;
    // ...
    Ham(a);
    // ...
}

```

Xác định kiểu dữ liệu trả về của hàm

Xác định dựa vào **đầu ra** (output) của hàm. Gồm 2 loại:

- Hàm không trả về giá trị: Những hàm loại này thường rơi vào những nhóm chức năng: nhập/xuất dữ liệu, thống kê, sắp xếp, liệt kê.

Sử dụng

```

void <tên hàm>([danh sách các tham số])
{
    <khai báo các biến cục bộ>
    <các câu lệnh hay lời gọi hàm>
}

```

- Hàm trả về kiểu dữ liệu cơ bản hay kiểu dữ liệu có cấu trúc: Kiểu dữ liệu tùy theo mục đích của hàm cần trả về giá trị gì thông qua việc phân tích bài toán. Những hàm loại này thường được sử dụng trong các trường hợp: Đếm, kiểm tra, tìm kiếm, tính trung bình, tổng, tích, ...

Sử dụng

```

<kiểu dữ liệu> <tên hàm>([danh sách các tham số])
{

```

```

    <kiểu dữ liệu> kq;

    <khai báo các biến cục bộ>

    <các câu lệnh hay lời gọi hàm>

    return kq;

}

```

Đối với những hàm trả về nhiều loại giá trị cho từng trường hợp cụ thể (chẳng hạn như kiểm tra: đúng hay sai, so sánh: bằng, lớn hơn hay nhỏ hơn, ...) thì cần ghi chú rõ giá trị trả về là gì cho từng trường hợp đó.

Xác định tham số

Xác định dựa vào **đầu vào** (output) của hàm. Lưu ý

- Tham số dạng tham trị: Không thay đổi hoặc không cần lấy giá trị mới của tham số sau lời gọi hàm. Tham số dạng này chỉ mang ý nghĩa là dữ liệu đầu vào.
- Tham số dạng tham trở và tham biến: Có sự thay đổi giá trị của tham số trong quá trình thực hiện và cần lấy lại giá trị đó sau khi ra khỏi hàm. Ứng dụng của tham số loại này có thể là dữ liệu đầu ra (kết quả) hoặc cũng có thể vừa là dữ liệu đầu vào vừa là dữ liệu đầu ra.

Xác định biến cục bộ

Là các đại lượng trung gian được sử dụng trong hàm

Xác định tên hàm

Đặt tên theo quy ước đặt tên trong C sao cho tên gọi đúng với chức năng hay mục đích thực hiện của hàm và gọi nhớ.

Các ví dụ

Ví dụ 4.2

Viết hàm in ra màn hình các ước số của n

Phân tích hàm:

- Input: n là số nguyên dương
- Output: không có

- Công việc: In ra các ước số của số của n

```
void LietKeUocSo(unsigned int n)
{
    for(int i=1; i<=n; i++)
        if(n%i == 0)
            printf("%u\t", i);
}
```

Ví dụ 4.3

Viết hàm tính tổng $S(n) = 1 + 2 + 3 + \dots + n$, với n là số nguyên dương

Phân tích bài toán:

- Input: n là số nguyên dương
- Output: S là tổng của $1 + 2 + 3 + \dots + n$
- Công việc: Tính tổng S

```
unsigned int TongS(unsigned int n)
{
    unsigned int S=0;
    int i=1;
    while(i<=n)
    {
        S+=i;
        i++;
    }
    return S;
}
```

4.2 BÀI TẬP

Bài tập cơ bản

- 4.1. Cài đặt lại tất cả các bài tập ở chương 2 theo phương pháp hàm.
- 4.2. Viết chương trình tính diện tích và chu vi của hình chữ nhật với chiều dài và chiều rộng

được nhập từ bàn phím.

4.3. Viết chương trình tính diện tích và chu vi hình tròn với bán kính được nhập từ bàn phím.

4.4. Nhập số nguyên dương n . Liệt kê tất cả các số nguyên tố nhỏ hơn n .

4.5. Nhập số nguyên dương n . Liệt kê n số chính phương đầu tiên.

4.6. Nhập số nguyên dương n . Đếm xem có bao nhiêu số hoàn thiện nhỏ hơn n .

4.7. Nhập số nguyên dương n ($0 < n < 1000$) và in ra cách đọc của n . Ví dụ: Nhập $n = 105$.

In ra màn hình: **mot tram le nam**.

4.8. Viết chương trình tính tiền thuê máy dịch vụ Internet và in ra màn hình kết quả. Với dữ liệu nhập vào là giờ bắt đầu thuê (GBD), giờ kết thúc thuê (GKT), số máy thuê (SoMay).

a) Điều kiện cho dữ liệu nhập: $6 \leq \text{GBD} < \text{GKT} \leq 21$. Giờ là số nguyên.

b) Đơn giá: 2500đ cho mỗi giờ máy trước 17:30 và 3000đ cho mỗi giờ máy sau 17:30.

4.9. Viết chương trình tính tiền lương ngày cho công nhân, cho biết trước giờ vào ca, giờ ra ca của mỗi người. Giả sử rằng:

a) Tiền trả cho mỗi giờ trước 12 giờ là 6000đ và sau 12 giờ là 7500đ.

b) Giờ vào ca sớm nhất là 6 giờ sáng và giờ ra ca trễ nhất là 18 giờ (Giả sử giờ nhập vào nguyên).

4.10. Nhập vào 2 số nguyên p, q và tính biểu thức sau:

$$\left(-\frac{q}{2} + \left(\frac{p^3}{27} + \frac{q^2}{4} \right)^{\frac{1}{2}} \right)^{\frac{1}{3}} + \left(-\frac{q}{2} - \left(\frac{p^3}{27} + \frac{q^2}{4} \right)^{\frac{1}{2}} \right)^{\frac{1}{3}}$$

4.11. Nhập vào 3 số thực a, b, c và kiểm tra xem chúng có thành lập thành 3 cạnh của một tam giác hay không? Nếu có hãy tính diện tích, chiều dài mỗi đường cao của tam giác và in kết quả ra màn hình.

a) Công thức tính diện tích

$$s = \sqrt{(p * (p - a) * (p - b) * (p - c))}$$

với p là nửa chu vi của tam giác

b) Công thức tính các đường cao:

$$h_a = \frac{2s}{a}, h_b = \frac{2s}{b}, h_c = \frac{2s}{c}$$

4.12. Nhập vào 6 số thực a, b, c, d, e, f . Giải hệ phương trình sau:

$$\begin{cases} ax + by = c \\ dx + ey = f \end{cases}$$

4.13. Viết chương trình nhập 2 số nguyên dương a, b . Tìm USCLN và BSCNN của hai số nguyên đó.

4.14. Viết chương trình tính tổng nghịch đảo của n giai thừa.

4.15. Cho 2 số nguyên a, b . Viết hàm hoán vị giá trị 2 số trên.

4.16. (*) Viết chương trình nhập số nguyên dương n gồm 5 chữ số, kiểm tra xem các chữ số n có phải là số đối xứng hay không. Ví dụ: Đối xứng: 13531 Không đối xứng: 13921

4.17. Viết chương trình nhập số nguyên dương n gồm k chữ số, $k \leq 5$, đếm xem n có bao nhiêu chữ số chẵn và bao nhiêu chữ số lẻ.

4.18. Viết chương trình nhập số nguyên dương n gồm k chữ số, $k \leq 5$, đếm xem n có bao nhiêu chữ số là số nguyên tố.

4.19. Viết chương trình nhập số nguyên dương n gồm k chữ số, $k \leq 5$, tính tổng các ước số dương của n . Ví dụ, nhập $n=6$ Tổng các ước số từ 1 đến n : $1+2+3+6=12$.

4.20. Viết chương trình nhập số nguyên dương n gồm k chữ số, $k \leq 5$, tìm ước số lẻ lớn nhất của n . Ví dụ: Ước số lẻ lớn nhất của 27 là 9.

4.21. Viết chương trình nhập số nguyên dương n gồm k chữ số, $k \leq 5$, kiểm tra xem các chữ số của n có toàn lẻ hay toàn chẵn không.

4.22. (*) Viết chương trình nhập số nguyên dương n gồm k chữ số, $k \leq 5$, sắp xếp các chữ số của n theo thứ tự tăng dần. Ví dụ: Nhập $n=1536$ Kết quả sau khi sắp xếp: 1356.

Luyện tập và nâng cao

4.23. Viết chương trình nhập số nguyên dương n gồm k chữ số, $k \leq 5$, sau đó nhập một số nguyên x , tìm vị trí xuất hiện của chữ số có giá trị x trong n . Ví dụ: Nhập $n=1526$, $x=2$ Kết quả: Chu số 2 ở vị trí thứ 3.

4.24. Viết chương trình nhập số nguyên dương n gồm k chữ số, $k \leq 5$, kiểm tra xem các chữ số của n có được sắp thứ tự không. Ví dụ, nhập $n=1569$ hoặc $n=8521$ thì kết quả là có thứ tự.

4.25. Viết chương trình nhập 2 số a, b sao cho: số lớn nhất trong 2 số phải là một số dương và chia hết cho 7. Nếu nhập sai phải yêu cầu nhập lại cho đến khi đúng.

- 4.26. Viết chương trình nhập số nguyên dương n gồm k chữ số, $k \leq 5$, tính giá trị trung bình các chữ số chẵn trong n .
- 4.27. (*) Viết chương trình in ra màn hình ngày/tháng/năm của ngày hiện tại, cho phép sử dụng các phím mũi tên lên, xuống để tăng hoặc giảm một ngày.
- 4.28. (*) Viết chương trình in ra màn hình giờ:phút:giây hiện tại, cho phép sử dụng các phím mũi tên lên, xuống để tăng hoặc giảm một giây.

4.3 TÓM TẮT

Trước khi xây dựng một hàm ta phải xác định mục đích của hàm là dùng để làm gì, trên cơ sở đó, ta mới xác định được các thành phần của hàm và xây dựng nguyên mẫu hàm. Mỗi hàm phải thực hiện một chức năng độc lập và tách biệt với các hàm khác (không được lồng nhau). Đối với hàm có giá trị trả về phải lưu ý kiểu dữ liệu phải tương ứng kiểu dữ liệu cả giá trị trả về và kiểu dữ liệu của biến được gán khi gọi hàm. Trường hợp hàm trả về từ hai loại giá trị trở lên thì phải có dòng chú thích cho trường hợp tương ứng để khi gọi hàm biết được kết quả (chẳng hạn như tìm kiếm, kiểm tra, so sánh, v.v giá trị trả về có 2 trường hợp: có hoặc không có phần tử cần tìm, thỏa điều kiện kiểm tra hay không? Do vậy ta phải quy ước giá trị cho từng trường hợp). Nên đặt tên hàm sao cho gợi nhớ được chức năng, đặt tên theo quy tắc nhất định để tránh việc gọi sai tên hàm do lẫn lộn giữa ký tự hoa và thường, có dấu gạch nối giữa các từ trong hàm hay không? Khi gọi hàm phải truyền đủ tham số, đúng kiểu dữ liệu và đúng thứ tự của tham số.

KIỂU DỮ LIỆU MẢNG MỘT CHIỀU

Chương này trình bày trình bày dữ liệu kiểu mảng, các thao tác nhập xuất, các kỹ thuật thao tác trên mảng. Ứng dụng các kỹ thuật này trong việc cài đặt các hàm tìm kiếm, kiểm tra, xây dựng mảng, tách và ghép mảng.

5.1 LÝ THUYẾT

Định nghĩa 5.1. mảng một chiều là tập hợp các phần tử được lưu trữ liên tục và các phần tử của mảng phải cùng kiểu dữ liệu.

Khai báo biến mảng

Cú pháp

<kiểu dữ liệu> <tên mảng> [<số phần tử tối đa của mảng>];

Khai báo mảng a có 100 số nguyên và mảng b có 50 số thực

```
int a[100];
```

```
float b[50];
```

Truy xuất phần tử của mảng

Với khái niệm và cách khai báo như trên ta có hình dạng của mảng một chiều như sau:

Cú pháp

<tên mảng>[<chỉ số>]

Ví dụ 5.1

Khởi tạo và xuất mảng một chiều

```
#include <conio.h>
#include <stdio.h>
void main()
{
    int a[4] = {5,9,3,8};
    for(int i = 0; i < 4; i++)
        printf(" a [ %d ] = %d \t", i, a[i]);
    getch();
}
```

5.2 BÀI TẬP

Kỹ thuật nhập xuất mảng một chiều**Ví dụ 5.2**

Viết chương trình nhập xuất mảng một chiều các số nguyên.

```
#include <conio.h>
#include <stdio.h>
#define MAX 100

void NhapMang(int a[], int &n)
{
    printf("Nhap so phan tu: ");
    scanf(" %d ", &n);
    for(int i = 0; i < n; i++)
    {
        printf(" a [%d] = ", i);
        scanf(" %d ", &a[i]);
    }
}
```

```

}

void XuatMang(int a[], int n)
{
    printf("\nNoi dung mang: ");
    for(int i = 0; i < n; i++)
        printf(" %d \t ", a[i]);
}

void main()
{
    int a[MAX], n;
    NhapMang(a,n);
    XuatMang(a,n);
    getch();
}

```

- 5.1. Viết chương trình nhập xuất mảng một chiều các số thực.
- 5.2. Viết chương trình khởi tạo giá trị các phần tử là 0 cho mảng một chiều các số nguyên gồm n phần tử.
- 5.3. Viết chương trình phát sinh ngẫu nhiên mảng một chiều các số nguyên âm.
- 5.4. Viết chương trình phát sinh ngẫu nhiên mảng một chiều các số nguyên sao cho mảng có thứ tự tăng dần (không sắp xếp).
- 5.5. Viết chương trình nhập mảng các số thực và xuất các phần tử âm trong mảng.
- 5.6. Viết chương trình nhập mảng các số nguyên và xuất các phần tử lẻ có trong mảng.
- 5.7. Viết chương trình nhập vào mảng một chiều các số nguyên và xuất ra các phần tử chẵn nhỏ hơn 20.
- 5.8. Viết chương trình nhập vào mảng một chiều các số nguyên và xuất ra màn hình các phần tử là số nguyên tố.
- 5.9. Viết chương trình nhập vào số nguyên n và liệt kê các số nguyên tố nhỏ hơn n , nếu mảng không tồn tại số nguyên tố nào nhỏ hơn n thì phải xuất ra một câu thông báo.
- 5.10. Viết chương trình nhập vào mảng một chiều các số nguyên và xuất ra màn hình các phần tử là số chính phương nằm tại những vị trí lẻ trong mảng.

Kỹ thuật đặt cờ hiệu

Kỹ thuật này thường được áp dụng cho những bài toán "kiểm tra" hay "đánh dấu".

Ví dụ 5.3

Viết hàm kiểm tra xem mảng các số nguyên có thứ tự tăng dần không? (Trả về 1: Nếu mảng tăng dần, ngược lại trả về 0).

```
int KiemTraTang(int a[], int n)
{
    int flag = 1;
    for(int i = 0; i < n-1; i++)
        if(a[i] > a[i+1]) // vi pham dieu kien tang dan
        {
            flag = 0;
            break;
        }
    return flag;
}
```

Ví dụ 5.4

Viết hàm kiểm tra xem trong mảng các số nguyên có tồn tại số nguyên lẻ lớn hơn 100 hay không? (Trả về 1: Nếu có tồn tại số lẻ và lớn hơn 100, ngược lại trả về 0).

```
int KiemTraLe(int a[], int n)
{
    int flag = 0;
    for(int i = 0; i < n; i++)
        if(a[i] % 2 != 0 && a[i] > 100) // Gap phan tu thoa
        {
            flag = 1;
            break;
        }
    return flag;
}
```



```
}

```

Kỹ thuật đặt lính canh

Kỹ thuật này thường được áp dụng cho những bài tập về "tìm kiếm", "liệt kê" theo một điều kiện nhất định nào đó.

Ví dụ 5.5

Viết hàm tìm và trả về giá trị lớn nhất trong mảng một chiều các số nguyên.

```
int TimMax(int a[], int n)
{
    int max, i = 1;
    max = a[0];
    while(i < n)
    {
        if(a[i] > max) max = a[i];
        i++;
    }
    return max;
}
```

Kỹ thuật tìm kiếm trên mảng một chiều

Ví dụ 5.6

Viết hàm tìm phần tử có giá trị x xuất hiện đầu tiên trong mảng một chiều. (Nếu tìm thấy trả về vị trí xuất hiện x , ngược lại trả về -1)

```
int TimX(int a[], int n, int x)
{
    for(int i = 0; i < n; i++)
        if(x==a[i])
            return i;
    return -1;
}
```

}

- 5.11. Viết hàm tìm vị trí phần tử có giá trị x xuất hiện cuối cùng trong mảng.
- 5.12. Viết hàm tìm vị trí của phần tử nhỏ nhất trong mảng các số nguyên.
- 5.13. Viết hàm tìm vị trí của phần tử lớn nhất trong mảng các số nguyên.
- 5.14. Viết hàm in vị trí các phần tử nguyên tố trong mảng các số nguyên.
- 5.15. Viết hàm in vị trí các phần tử nguyên tố lớn hơn 23.
- 5.16. Viết hàm tìm vị trí phần tử âm đầu tiên trong mảng. Nếu không có phần tử âm trả về -1.
- 5.17. Viết hàm tìm vị trí phần tử âm lớn nhất trong mảng.
- 5.18. Viết hàm tìm vị trí phần tử dương đầu tiên trong mảng. Nếu không có phần tử âm trả về -1.
- 5.19. Viết hàm tìm vị trí phần tử dương bé nhất trong mảng.
- 5.20. Viết hàm in các phần tử là bội của 3 và 5.
- 5.21. Viết hàm tìm số chẵn cuối cùng có trong mảng, nếu không tồn tại số chẵn hàm trả về -1.
- 5.22. Viết hàm tìm số lẻ lớn nhất có trong mảng, nếu không tồn tại số lẻ hàm trả về -1.
- 5.23. Viết hàm tìm và đổi chỗ phần tử lớn nhất với phần tử nhỏ nhất trong mảng.
- 5.24. Nhập vào x . Viết hàm in ra màn hình những phần tử có giá trị từ 1 đến x có trong mảng.
- 5.25. Viết chương trình nhập vào một dãy số a gồm n số thực và dãy số b gồm m số thực ($m, n \leq 100$).
- In ra những phần tử chỉ xuất hiện trong dãy a mà không xuất hiện trong dãy b .
 - In ra những phần tử xuất hiện ở cả hai dãy.

Kỹ thuật đếm phần tử - tần suất

Ví dụ 5.7

Viết hàm đếm các phần tử chia hết cho 5 trong mảng các số nguyên.

```
int Dem(int a[], int n)
{
    int dem = 0;
    for(int i = 0; i < n; i++)
        if(a[i] % 5 == 0)
            dem++;
}
```

```

    return dem;
}

```

- 5.26. Viết hàm đếm các phần tử âm, dương trong mảng.
- 5.27. Viết hàm đếm các phần tử chẵn, lẻ trong mảng.
- 5.28. Viết hàm đếm số lần xuất hiện của phần tử x trong mảng.
- 5.29. Viết hàm đếm các phần tử nhỏ hơn x trong mảng.
- 5.30. Viết hàm đếm các phần tử là số nguyên tố trong mảng.
- 5.31. Viết hàm đếm các phần tử là số hoàn thiện trong mảng.
- 5.32. Viết hàm đếm các phần tử là bội của 3 và 5 trong mảng các số nguyên.

Kỹ thuật tính tổng - trung bình

Ví dụ 5.8

Viết hàm tính tổng các phần tử trong mảng.

```

int TinhTong(int a[], int n)
{
    int tong = 0;
    for(int i = 0; i < n; i++)
        tong = tong + a[i];
    return tong;
}

```

Ví dụ 5.9

Viết hàm tính giá trị trung bình các phần tử có giá trị âm trong mảng. Đối với hàm tính trung bình có điều kiện phải lưu ý khi chia giá trị (có thể mảng không có phần tử nào thoả điều kiện, nếu ta chia tức là chia cho 0).

```

float TinhTrungBinhAm(int a[], int n)
{
    int tong = 0;
    int spt=0;
    for(int i = 0; i < n; i++)

```

```

        if(a[i]<0)
        {
            tong = tong + a[i];
            spt++;
        }
    if(spt==0) return 0;
    return 1.0*tong/spt;
}

```

- 5.33. Viết hàm tính tổng các phần tử chẵn trong mảng.
- 5.34. Viết hàm tính tổng các phần tử lẻ trong mảng các số nguyên.
- 5.35. Viết hàm tính tổng các phần tử nguyên tố trong mảng.
- 5.36. Viết hàm tính tổng các phần tử nằm ở vị trí chẵn trong mảng các số nguyên.
- 5.37. Viết hàm tính tổng các phần tử nằm ở vị trí nguyên tố trong mảng.
- 5.38. Viết hàm tính tổng các phần tử chia hết cho 5 có trong mảng.
- 5.39. Viết hàm tính tổng các phần tử cực đại trong mảng các số nguyên (phần tử cực đại là phần tử lớn hơn các phần tử xung quanh nó). Ví dụ: 1 **5** 2 **6** 3 **5** 1 **8** 6
- 5.40. Viết hàm tính tổng các phần tử cực tiểu trong mảng các số nguyên (phần tử cực tiểu là phần tử nhỏ hơn các phần tử xung quanh nó). Ví dụ: 6 4 **2** 9 5 **3** 7 **1** 5 8
- 5.41. Viết hàm tính tổng các phần tử là bội của 3 và 5 trong mảng các số nguyên.
- 5.42. Viết hàm tính tổng các phần tử là số hoàn thiện trong mảng các số nguyên.
- 5.43. Viết hàm tính giá trị trung bình của các số hoàn thiện trong mảng các số nguyên.

Kỹ thuật sắp xếp

Ví dụ 5.10

Viết hàm sắp xếp mảng theo thứ tự tăng dần.

```

void HoanVi(int &a, int &b)
{
    int tam = a;
    a = b;
    b = tam;
}

```

```

void SapTang(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]);
}

```

- 5.44. Viết hàm sắp xếp mảng theo thứ tự giảm dần.
- 5.45. Viết hàm sắp xếp mảng theo thứ tự tăng dần của các phần tử là số nguyên tố.
- 5.46. Viết hàm sắp xếp các phần tử lẻ tăng dần.
- 5.47. Viết hàm sắp xếp các phần tử chẵn giảm dần.
- 5.48. Viết hàm sắp xếp các phần tử chẵn nằm bên trái theo thứ tự tăng dần còn các phần tử lẻ bên phải theo thứ tự giảm dần.
- 5.49. Viết hàm sắp xếp các phần tử âm giảm dần từ trái sang phải, phần tử dương tăng dần từ phải sang trái.

Kỹ thuật xoá phần tử

Duyệt mảng từ trái sang phải. Xuất phát từ vị trí cần xoá tiến hành dời lần lượt các phần tử về phía trước cho đến khi kết thúc mảng, sau đó giảm kích thước mảng. Vấn đề đặt ra là tìm vị trí cần xoá theo điều kiện bài toán rồi thực hiện xoá.

Ví dụ 5.11

Viết hàm xoá phần tử đầu tiên của mảng.

```

void XoaDau(int a[], int &n)
{
    for(int i = 0; i < n-1; i++)
        a[i] = a[i+1];-
    n;
}

```

Ví dụ 5.12

Viết hàm xoá phần tử tại vị trí vitri cho trước trong mảng.

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

5.50. Viết hàm xoá phần tử tại vị trí lẻ trong mảng.

5.51. Viết hàm xoá phần tử có giá trị lớn nhất trong mảng.

5.52. Nhập vào giá trị x . Viết hàm xoá tất cả các phần tử có giá trị nhỏ hơn x .

5.53. Nhập vào giá trị x . Viết hàm xoá phần tử có giá trị gần x nhất.

Kỹ thuật thêm/chèn phần tử

Duyệt mảng từ phải sang trái. Xuất phát từ cuối mảng tiến hành đẩy lần lượt các phần tử về phía sau cho đến vị trí cần chèn, chèn phần tử cần chèn vào vị trí chèn và tăng kích thước mảng. Trước khi chèn ta phải xác định vị trí cần chèn theo điều kiện bài toán.

Ví dụ 5.13

Thêm phần tử có giá trị x vào cuối mảng.

```
void ChenCuoi(int a[], int &n, int x)
{
    a[n]=x;
    n++;
}
```

Ví dụ 5.14

Chèn phần tử có giá trị x vào mảng tại vị trí (vitri) cho trước

```
void ChenTaiViTri(int a[], int &n, int x, int vitri)
```

```

{
    for(int i = n; i > vitri; -i)
        a[i] = a[i-1];
    a[vitri] = x;
    n++;
}

```

- 5.54. Viết hàm chèn phần tử có giá trị x vào vị trí đầu tiên của mảng.
- 5.55. Viết hàm chèn phần tử có giá trị x vào phía sau phần tử có giá trị lớn nhất trong mảng.
- 5.56. Viết hàm chèn phần tử có giá trị x vào trước phần tử có giá trị là số nguyên tố đầu tiên trong mảng.
- 5.57. Viết hàm chèn phần tử có giá trị x vào phía sau tất cả các phần tử có giá trị chẵn trong mảng.

Kỹ thuật tách và ghép mảng

Ví dụ 5.15

Cho mảng a kích thước n (n chẵn). Tách mảng a thành 2 mảng b và c sao cho: b có một nửa số phần tử đầu của mảng a , một nửa số phần tử còn lại đưa vào mảng c .

```

void TachMang(int a[], int n, int b[], int &m, int c[], int &l)
{
    int k=n/2;
    m=l=0;
    for(int i=0; i<k; i++)
    {
        b[m++]=a[i];
        c[l++]=a[k+i]
    }
}

```

Ví dụ 5.16

Cho 2 mảng số nguyên a và b kích thước lần lượt là n và m . Viết chương trình nối mảng b

vào cuối mảng a .

```
void GhepMang(int a[], int &n, int b[], int m)
{
    for(int i=0; i<m; i++)
        a[n+i]=b[i];
    n=n+m;
}
```

Ví dụ 5.17

Cho 2 mảng số nguyên a và b kích thước lần lượt là n và m . Viết chương trình nối xen kẻ (đan xen) lần lượt các phần tử mảng a và b vào mảng c .

Đưa lần lượt từng phần tử của mảng a và mảng b vào mảng c , tăng chỉ số tương ứng. Nếu một trong hai mảng hết trước thì chép tất cả các phần tử còn lại của mảng chưa hết vào mảng c . Đặt i là chỉ số của mảng a ; j : chỉ số của mảng b và k là chỉ số của mảng c .

```
void GhepMang(int a[], int &n, int b[], int m, int c[], int &k)
{
    int i=0, j=0;
    k=0;
    while(i<n && j<m)
    {
        c[k++]=a[i++];
        c[k++]=b[j++];
    }
    while(i<n)
        c[k++]=a[i++];
    while(j<m)
        c[k++]=b[j++];
}
```

- 5.58. Viết chương trình tách 1 mảng các số nguyên thành 2 mảng a và b , sao cho mảng a chứa toàn số lẻ và mảng b chứa toàn số chẵn. Ví dụ: mảng ban đầu: 1 3 8 2 7 5 9 0 10 thì mảng a : 1 3 7 5 9 và mảng b : 8 2 10

- 5.59. Cho 2 mảng số nguyên a và b kích thước lần lượt là n và m . Viết chương trình nối 2 mảng trên thành mảng c theo nguyên tắc chèn ở đầu mảng và lẻ ở cuối mảng. Ví dụ: mảng a : 3 2 7 5 9 và mảng b : 1 8 10 4 12 6 thì mảng c : 6 12 4 10 2 8 3 1 7 5 9

Luyện tập và nâng cao

- 5.60. Viết chương trình nhập vào mảng A gồm n phần tử, trong quá trình nhập kiểm tra các phần tử nhập vào không được trùng, nếu trùng thông báo và yêu cầu nhập lại.
- 5.61. Viết hàm tính tổng của từng dãy con giảm có trong mảng.
- 5.62. (*) Cho mảng các số nguyên a gồm n phần tử ($n \leq 30000$) và số dương $k \leq n$. Hãy chỉ ra số hạng lớn thứ k của mảng. Ví dụ: mảng a : 6 3 1 **10** 11 18 và $k = 2$ thì kết quả là 10
- 5.63. (*) Cho 2 dãy A, B các số nguyên (kích thước dãy A nhỏ hơn dãy B). Hãy kiểm tra xem A có phải là con của B hay không?
- 5.64. Viết hàm liệt kê các bộ 4 số a, b, c, d trong mảng các số nguyên (có ít nhất 4 phần tử và đôi một khác nhau) sao cho $a + b = c + d$.
- 5.65. (*) Viết chương trình tính trung bình cộng của các tổng các dãy tăng dần có trong mảng các số nguyên. Ví dụ: **1 2 3 4 2 3 4 5 6 4 5 6** thì trung bình = 15.
- 5.66. Viết chương trình tính tổng tất cả các phần tử xung quanh trên mảng các số nguyên (phần tử xung quanh là hai phần tử bên cạnh cộng lại bằng chính nó, ví dụ: 1 3 2 thì 1, 2 là hai phần tử xung quanh của 3). Ví dụ: 1 3 2 5 3 9 6 tổng 17
- 5.67. (**) Viết chương trình nhập vào hai số lớn a, b nguyên (a, b có từ 20 chữ số trở lên). Tính tổng, hiệu, tích, thương của hai số trên.
- 5.68. Viết hàm tính tổng các phần tử là số Armstrong (số Armstrong là số có đặc điểm như sau: số có k chữ số, tổng của các lũy thừa bậc k của các ký số bằng chính số đó. Ví dụ: 153 là số có các ký số $1^3 + 5^3 + 3^3 = 153$ là một số Armstrong).
- 5.69. Viết hàm tìm và xóa tất cả các phần tử trùng với x trong mảng một chiều các số nguyên, nếu không tồn tại phần tử x trong mảng thì trả về -1.
- 5.70. Viết hàm xóa tất cả những phần tử trùng nhau trong dãy chỉ giữ lại một phần tử trong đó. Ví dụ: 1 6 **2 3 2 4 2** 6 5 thì kết quả 1 6 2 3 4 5
- 5.71. (**) Viết hàm xóa những phần tử sao cho mảng kết quả có thứ tự tăng dần và số lần xóa là ít nhất.
- 5.72. Cho dãy a gồm n số nguyên có thứ tự tăng dần. Nhập vào một phần tử nguyên x , viết hàm chèn x vào dãy sao cho dãy vẫn có thứ tự tăng dần (không sắp xếp).

- 5.73. Viết chương trình tìm số lẻ nhỏ nhất lớn hơn mọi số chẵn có trong mảng.
- 5.74. Viết hàm tìm giá trị chẵn nhỏ nhất nhỏ hơn mọi giá trị lẻ trong mảng các số nguyên.
- 5.75. Viết hàm tìm phần tử xuất hiện nhiều nhất trong mảng các số nguyên.
- 5.76. Viết chương trình đếm và liệt kê các mảng con tăng dần trong mảng một chiều các số nguyên. Ví dụ: 6 5 3 2 3 4 2 7 các dãy con tăng dần là 2 3 4 và 2 7
- 5.77. Viết chương trình tìm mảng con tăng dần có tổng lớn nhất trong mảng một chiều.
- 5.78. (*) Viết chương trình nhập vào một dãy số a gồm n số nguyên ($n \leq 100$). Tìm và in ra dãy con tăng dài nhất. Ví dụ: Nhập dãy a : 1 2 3 6 4 7 8 3 4 5 6 7 8 9 4 5 thì dãy con tăng dài nhất: 3 4 5 6 7 8 9
- 5.79. (**) Viết chương trình tách 1 mảng các số nguyên thành 2 mảng a và b , sao cho kết quả thu được là:
- Mảng a chứa toàn số lẻ tăng dần.
 - Mảng b chứa toàn số chẵn giảm dần
 - Không dùng thuật toán sắp xếp
- Hướng dẫn: Tìm vị trí chèn thích hợp khi trích phần tử từ mảng ban đầu. Ví dụ: mảng ban đầu: 9 3 8 2 7 5 1 0 10 thì Mảng a : 1 3 5 7 9 Mảng b : 10 8 2
- 5.80. (**) Viết chương trình in ra tam giác Pascal (dùng mảng một chiều).
- 5.81. Viết chương trình nhập vào dãy số a gồm n số thực ($n \leq 100$) và dãy số b gồm m số thực ($m \leq 100$).
- Hãy sắp xếp hai dãy theo thứ tự tăng dần.
 - (*) Trộn 2 dãy trên thành dãy c sao cho dãy c vẫn có thứ tự tăng.
 - Xuất dãy a, b, c ra màn hình.
- 5.82. (*) Cho mảng c có n phần tử ($n < 200$), các phần tử là các chữ số trong hệ đếm cơ số 16 (Hexa). Hãy tách mảng c ra các mảng con theo điều kiện sau: các mảng con được giới hạn bởi hai lần xuất hiện thứ hai của con số trong dãy. Ví dụ: 123A4518B23 có các dãy con là 123A451, 23A4518B2, 23A4518B23
- 5.83. (**) Cho hai số nguyên dương A, B . Hãy xác định hai số C, D tạo thành từ hai số A, B sao cho C là số lớn nhất, D là số nhỏ nhất. Khi gạch đi một số chữ số trong C (D), thì các số còn lại giữ nguyên tạo thành A , các chữ số bỏ đi giữ nguyên tạo thành B . Ví dụ: $A = 52568, B = 462384$ thì $C = 54625682384, D = 45256236884$.
- 5.84. Viết chương trình nhập vào dãy số a gồm n số nguyên ($n \leq 100$).
- Hãy đảo ngược dãy đó. Ví dụ: Nhập a : 3 4 5 2 0 4 1 thì dãy sau khi đảo: 1 4 0 2 5 4

3

b) (*) Hãy kiểm tra xem dãy đã cho có thứ tự chưa (dãy được gọi là thứ tự khi là dãy tăng hoặc dãy giảm).

5.85. Cho mảng a có n phần tử hãy cho biết mảng này có đối xứng hay không.

5.86. (**) Hãy viết chương trình phát sinh ngẫu nhiên mảng các số nguyên gồm 10000 phần tử, mỗi phần tử có giá trị từ 0 đến 32000 và xây dựng hàm thống kê số lần xuất hiện các phần tử trong mảng, sau đó cho biết phần tử nào xuất hiện nhiều lần nhất. Ví dụ: mảng: 5 6 11 4 4 5 4 thì 5 xuất hiện 2 lần, 6 xuất hiện 1 lần, 11 xuất hiện 1 lần, 4 xuất hiện 3 lần và 4 xuất hiện nhiều lần nhất

5.87. Cho mảng a có n phần tử. Nhập vào số nguyên dương k , dịch phải xoay vòng mảng a k lần. Ví dụ: mảng a : 5 7 2 3 1 9 và $k = 2$ thì dịch phải xoay vòng mảng a : 1 9 5 7 2 3

5.3 TÓM TẮT

Dữ liệu kiểu mảng dùng cho việc biểu diễn những thông tin có cùng kiểu dữ liệu liên tiếp nhau. Khi cài đặt bài tập mảng một chiều nên xây dựng thành những hàm chuẩn để dùng lại cho các bài tập khác. Các thao tác trên mảng đều theo quy tắc nhất định, chúng ta có thể ứng dụng mảng trong việc biểu diễn số lớn, dùng bảng tra, khử đệ qui, v.v

KIỂU DỮ LIỆU CHUỖI KÝ TỰ

Chuỗi ký tự là trường hợp đặc biệt của mảng một chiều. Chương này mô tả một số hàm thư viện thao tác trên chuỗi và các kỹ thuật cài đặt xử lý trên chuỗi.

6.1 LÝ THUYẾT

Định nghĩa 6.1. Chuỗi ký tự

- Là một dãy các phần tử, mỗi phần tử có kiểu ký tự.
- Được kết thúc bằng ký tự '\0' hay ký tự NULL (do đó khi khai báo độ dài của chuỗi luôn luôn khai báo dư 1 phần tử để chứa ký tự '\0').

Khai báo biến chuỗi

Cú pháp

```
char <tên chuỗi> [<số ký tự tối đa của chuỗi + 1>;
```

Khai báo 1 chuỗi ký tự chuỗi có tối đa 24 ký tự

```
char chuỗi[25];
```

Hàm nhập chuỗi

```
scanf(...)
```

```
char *gets(char *s);
```

Ví dụ 6.1

Nhận các ký tự nhập từ phím cho đến khi nhấn phím ENTER và đưa vào s

```
void main()
{
    char chuoi[80];
    printf("Nhap vao chuoi:");
    gets(chuoi);
    printf("Chuoi vua nhap la: %s\n", chuoi);
}
```

Hàm xuất chuỗi

```
printf(...)
int puts(const char *s);
```

Ví dụ 6.2

Xuất chuỗi ra màn hình.

```
void main()
{
    char chuoi[] = "Vi du xuat chuoi\n";
    puts(string);
}
```

Các hàm về chuỗi trong thư viện "string.h"

Hàm	Chức năng
<code>int strlen(char s[]);</code>	trả về chiều dài của chuỗi s
<code>strcpy(char dest[], char src[]);</code>	sao chép chuỗi src vào chuỗi dest
<code>strncpy(char dest[], char src[], int n);</code>	chép n ký tự đầu tiên từ chuỗi src vào chuỗi dest
<code>strcat(char s1[], char s2[]);</code>	nối chuỗi s2 vào sau chuỗi s1

<code>strncat(char s1[],char s2[],int n);</code>	nối n ký tự đầu tiên của chuỗi s2 vào sau chuỗi s1
<code>int strcmp(char s1[],char s2[]);</code>	so sánh thứ tự chuỗi s1 và s2
<code>int strncmp(char s1[],char s2[], int n);</code>	so sánh n ký tự đầu tiên của s1 và s2
<code>int strcasecmp(char s1[],char s2[], int n);</code>	so sánh không phân biệt chữ hoa và chữ thường
<code>char *strchr(char s[], char c);</code>	tìm xuất hiện đầu tiên của c trong s
<code>char *strstr(char s1[], char s2[]);</code>	tìm xuất hiện đầu tiên của s2 trong s1
<code>char *strtok(char s1[], char s2[]);</code>	tách s1 thành các token dựa vào s2

Ví dụ 6.3

Nhập vào một chuỗi ký tự, xuất ra màn hình chuỗi bị đảo ngược thứ tự các ký tự. Ví dụ, nhập vào: Le Hoang Thai và xuất ra màn hình: iahT gnaoH eL

```
#include<stdio.h>
#include<conio.h>
#include<string.h>

void DaoChuoi(char *s1, char *s2)
{
    int l=strlen(s1);
    for(int i=0; i<l; i++)
        s2[i]=s1[l-i-1];
    s2[i]='\0';
}

void main()
{
    char s1[100], s2[100];
    printf("Nhap vao chuoi ky tu: ");
    gets(s1);
    DaoChuoi(s1, s2);
    printf("Ket qua sau khi dao nguoc chuoi: %s", s2);
}
```

6.2 BÀI TẬP

Bài tập cơ bản

6.1. Cho biết kết quả của đoạn chương trình sau:

```
char s[20]="Truong DHKHTN TPHCM", *p;
p = strtok(s, " ");
while(p != NULL)
{
    printf("%s\n",p);
    p = strtok(NULL, " ");
}
```

6.2. Cho biết kết quả của đoạn chương trình sau:

```
char s1[30]="Truong DHKHTN", s1[30]="Tp. HCM", s3[30], s[30];
strcpy(s, s1);
strcpy(s3,"aeiou");
strcat(s, s2);
int n=strlen(s), k=0;
printf("Chuoi: %s",s);
for(int i=0; i<n; i++)
{
    if(strchr(s3, s[i]))
        k++;
}
printf("\nKet qua: %d", k);
```

6.3. Viết chương trình nhập vào một chuỗi ký tự, đếm số ký tự có trong chuỗi.

6.4. Viết chương trình đếm có bao nhiêu khoảng trắng trong chuỗi.

6.5. Viết chương trình nhập vào một chuỗi, hãy loại bỏ những khoảng trắng thừa trong chuỗi.

6.6. Viết chương trình nhập vào hai chuỗi s_1 và s_2 , nối chuỗi s_2 vào s_1 . Xuất chuỗi s_1 ra màn hình.

6.7. Đổi tất cả các ký tự có trong chuỗi thành chữ thường (không dùng hàm `strlwr`).

6.8. Đổi tất cả các ký tự trong chuỗi sang chữ in hoa (không dùng hàm `struppr`).

- 6.9. Viết chương trình đổi những ký tự đầu tiên của mỗi từ thành chữ in hoa.
- 6.10. Viết chương trình đổi chữ xen kẽ 1 chữ hoa và 1 chữ thường. Ví dụ: nhập ABCDEfgh đổi thành AbCdEfGh
- 6.11. Viết chương trình đảo ngược các ký tự trong chuỗi. Ví dụ: nhập ABCDE, xuất ra màn hình là EDCBA
- 6.12. Viết chương trình tìm kiếm 1 ký tự xem có trong chuỗi hay không, nếu có xuất ra vị trí của từ đó.
- 6.13. Viết 1 chương trình đếm một ký tự xuất hiện bao nhiêu lần trong chuỗi.
- 6.14. Viết chương trình tìm kiếm tên trong chuỗi họ tên. Nếu có thì xuất ra là tên này đã nhập đúng, ngược lại thông báo là đã nhập sai.
- 6.15. Viết chương đảo vị trí của từ đầu và từ cuối. Ví dụ: nhập bo an co xuất ra co an bo
- 6.16. Viết hàm cắt chuỗi họ tên thành chuỗi họ lót và chuỗi tên. Ví dụ: chuỗi họ tên là: Le Hoang Thai cắt ra 2 chuỗi là chuỗi họ lót Le Hoang và chuỗi tên là Thai
- 6.17. Nhập một chuỗi bất kỳ, sau đó hỏi người dùng cần tách bắt đầu từ đâu trong chuỗi trở về sau. Ví dụ: Nhập chuỗi TRUONG DAI HOC KHOA HOC TU NHIEU. Người nhập muốn tách bắt đầu từ chữ KHOA thì sẽ xuất ra chuỗi KHOA HOC TU NHIEU ra màn hình.
- 6.18. Viết hàm kiểm tra xem chuỗi có đối xứng hay không?.
- 6.19. Viết hàm tra xem trong chuỗi có ký tự số hay không nếu có tách ra thành một mảng số riêng.
- 6.20. Nhập một chuỗi bất kì, yêu cầu nhập 1 ký tự muốn xóa. Thực hiện xóa tất cả những ký tự đó trong chuỗi.
- 6.21. Viết chương trình tìm kiếm xem ký tự nào xuất hiện nhiều nhất trong chuỗi.
- 6.22. Viết 1 chương trình xoá một từ nào đó trong chuỗi. Ví dụ: Chuỗi ban đầu KHOA CONG NGHE THONG TIN và từ THONG, và kết quả xuất ra: KHOA CONG NGHE TIN

Luyện tập và nâng cao

- 6.23. Đổi các từ ở đầu câu sang chữ hoa và những từ không phải đầu câu sang chữ thường. Ví dụ: nGuYen vAN a đổi thành: Nguyen Van A
- 6.24. (*) Viết chương trình đảo ngược thứ tự các từ có trong chuỗi Ví dụ: Nhập Truong DHKHTN TpHCM Xuất ra màn hình là: TpHCM DHKHTN Truong
- 6.25. Nhập 1 chuỗi bất kì, liệt kê xem mỗi ký tự xuất hiện mấy lần.
- 6.26. Viết hàm kiểm tra xem trong 2 chuỗi có bao nhiêu ký tự giống nhau.

- 6.27. Viết chương trình mình chạy từ trái qua phải màn hình.
- 6.28. Viết 1 chương trình chèn 1 từ ở bất cứ vị trí nào mà người dùng yêu cầu.
- 6.29. (*) Viết chương trình nhập vào một chuỗi đếm xem chuỗi có bao nhiêu từ. Các từ cách nhau bằng khoảng trắng, dấu chấm câu: dấu chấm (.), dấu phẩy (,), dấu chấm phẩy (;), dấu hỏi (?) và dấu chấm than (!).
- 6.30. (**) Viết chương trình hiển thị một chuỗi ký tự. Chương trình cho phép di chuyển dấu u nháy sang trái, sang phải, lên dòng hay xuống dòng bằng phím mũi tên, chèn hay xóa ký tự tại vị trí dấu nháy.

6.3 TÓM TẮT

Cũng giống như kiểu mảng một chiều, thao tác truy xuất các phần tử trên chuỗi hoàn toàn tương tự. Bên cạnh đó, kiểu dữ liệu này còn được cài đặt sẵn một số hàm thư viện rất hữu ích nên trong quá trình thao tác trên chuỗi nên khi cài đặt ta cố gắng tận dụng tối đa những hàm liên quan. Không nên sử dụng hàm `scanf` để nhập chuỗi trong trường hợp chuỗi dữ liệu nhập vào có chứa khoảng trắng. Nếu nhập chuỗi phía sau hàm `scanf` nên chèn hàm `fflush` hoặc hàm `flushall` giữa `scanf` và `gets` để xóa vùng đệm, tránh trường hợp chương trình bỏ qua hàm `gets` do trong vùng đệm còn lưu ký tự xuống dòng của phím ENTER. Khi thao tác trên chuỗi lưu ý phải đảm bảo chuỗi được kết thúc bằng ký tự kết thúc `'\0'`.

KIỂU DỮ LIỆU MẢNG HAI CHIỀU

Đây là kiểu dữ liệu dùng để biểu diễn dữ liệu kiểu bảng, kiểu dữ liệu này rất thích hợp cho các bài toán liên quan đến ma trận, đồ thị hoặc ảnh và video số

7.1 LÝ THUYẾT

Định nghĩa 7.1. Mảng hai chiều là tập hợp các phần tử được tổ chức theo dòng và cột

Khai báo biến mảng

Cú pháp

<kiểu dữ liệu phần tử> <tên mảng> [<số dòng>] [<số cột>];

Khai báo mảng hai chiều A và b

```
int A[20][10];
```

```
float b[10][10];
```

Truy xuất phần tử của mảng

Cú pháp

<tên mảng> [<chỉ số dòng>] [<chỉ số cột>]

Khai báo kiểu mảng

Cú pháp

```
typedef <kiểu dữ liệu phần tử> <tên kiểu mảng>[<số dòng>][<số cột>];
```

Khai báo kiểu mảng hai chiều MANG2D chứa các số nguyên int có tối đa 100 dòng và 100 cột

```
#define MAX 100
```

```
typedef int MANG2D[100][100];
```

7.2 BÀI TẬP

Bài tập cơ bản

Kỹ thuật nhập/xuất mảng hai chiều

Ví dụ 7.1

Nhập và xuất mảng hai chiều

```
void Nhap(MANG2D a, int &d, int &c)
{
    printf("\nNhap so dong: ");
    scanf(" %d", &d);
    printf("\nNhap so cot: ");
    scanf("%d", &c);
    for(int i = 0; i < d; i++)
        for(int j = 0; j < c; j++)
        {
            printf(" a[%d][%d] = ", i, j);
            scanf("%d", &a[i][j]);
        }
}

void Xuat(MANG2D a, int d, int c)
{
    printf("\nNoi dung ma tran:\n");
    for(int i = 0; i < d; i++)
    {
        for(int j = 0; j < c; j++)
            printf(" \t %d ", a[i][j]);
    }
}
```

```

        printf("\n");
    }
}

```

- 7.1. Viết hàm nhập ma trận các số nguyên dương (nhập sai báo lỗi và không cho nhập).
- 7.2. Viết hàm nhập/xuất ma trận các số thực.
- 7.3. Viết hàm in ra những phần tử có ký số tận cùng là 5.
- 7.4. Viết chương trình khởi tạo giá trị các phần tử là ngẫu nhiên cho ma trận các số nguyên kích thước $m \times n$.
- 7.5. Viết hàm tạo ma trận a các số nguyên gồm 9 dòng 14 cột. Trong đó phần tử $a[i][j] = i * j$
- 7.6. Viết hàm in tam giác Pascal với chiều cao h . Ví dụ: $h = 5$

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

```

Kỹ thuật đặt cờ hiệu

Ví dụ 7.2

Viết hàm kiểm tra xem trong ma trận các số nguyên có tồn tại các số nguyên lẻ lớn hơn 100 không?

```

int KiemTraLe(MANG2D a, int d, int c)
{
    int flag = 0; // tra ve 1 neu co nguoc lai tra ve 0
    for(int i = 0; i < d; i++)
        for(int j = 0; j < c; j++)
            if(a[i][j] % 2 != 0 && a[i][j] > 100)
            {
                flag = 1;
                break;
            }
    return flag;
}

```

```
}

```

Kỹ thuật đặt lính canh

Ví dụ 7.3

Viết hàm tìm phần tử nhỏ nhất trong ma trận.

```
int TimMin(MANG2D a, int d, int c)
{
    int min = a[0][0];
    for(int i = 0; i < d; i++)
        for(int j = 0; j < c; j++)
            if(a[i][j] < min)
                min = a[i][j];
    return min;
}
```

Kỹ thuật tính tổng

Ví dụ 7.4

Viết hàm tính tổng các phần tử trong ma trận.

```
int TinhTong(MANG2D a, int d, int c)
{
    int tong = 0;
    for(int i = 0; i < d; i++)
        for(int j = 0; j < c; j++)
            tong += a[i][j];
    return tong;
}
```

- 7.7. Viết hàm tính tổng các phần tử trên cùng một dòng.
- 7.8. Viết hàm tính tổng các phần tử trên cùng một cột.
- 7.9. Viết hàm tính tổng các phần tử chẵn có trong ma trận.

- 7.10. Viết hàm tính tổng các phần tử là số nguyên tố có trong ma trận.
- 7.11. Viết hàm tính tổng các số hoàn thiện trong ma trận các số nguyên.
- 7.12. Viết hàm tính tổng các giá trị lớn nhất trên mỗi dòng.
- 7.13. Viết hàm tính giá trị trung bình của các phần tử nhỏ nhất trên mỗi cột.

Kỹ thuật tìm kiếm

- 7.14. Viết hàm tìm vị trí phần tử lớn nhất trong ma trận các số nguyên.
- 7.15. Viết hàm tìm vị trí phần tử nhỏ nhất trong ma trận các số nguyên.
- 7.16. Viết hàm tìm vị trí phần tử chẵn cuối cùng trong ma trận các số nguyên.
- 7.17. Viết hàm tìm phần tử âm lẻ lớn nhất trong ma trận.
- 7.18. Viết hàm tìm phần tử chẵn dương và nhỏ nhất trong ma trận.
- 7.19. Viết hàm tìm số hoàn thiện đầu tiên trong ma trận các số nguyên.
- 7.20. Viết hàm tìm số hoàn thiện lớn nhất trong ma trận các số nguyên.
- 7.21. Viết hàm tìm vị trí phần tử nguyên tố cuối cùng trong ma trận các số nguyên.
- 7.22. Viết hàm tìm trong 2 ma trận các số nguyên, những phần tử giống nhau.
- 7.23. Viết hàm tìm và liệt kê những phần tử cực đại trong ma trận (một phần tử được coi là cực đại khi nó lớn hơn các phần tử xung quanh nó).
- 7.24. Viết hàm tìm dòng có tổng lớn nhất trong ma trận các số thực.
- 7.25. Viết hàm tìm cột có tổng nhỏ nhất trong ma trận các số nguyên.

Kỹ thuật đếm

Ví dụ 7.5

Viết hàm đếm các phần tử chẵn trong ma trận.

```
int DemChan(MANG2D a, int d, int c)
{
    int dem = 0;
    for(int i = 0; i < d; i++)
        for(int j = 0; j < c; j++)
            if(a[i][j] % 2 == 0)
                dem++;
    return dem;
}
```

}

- 7.26. Viết hàm đếm các giá trị âm, dương trong ma trận các số thực.
- 7.27. Viết hàm đếm các giá trị chẵn, lẻ trong ma trận các số nguyên.
- 7.28. Viết hàm đếm số lần xuất hiện của phần tử x trong ma trận các số thực.
- 7.29. Viết hàm đếm các giá trị nhỏ hơn x trong ma trận các số thực.
- 7.30. Viết hàm đếm các phần tử nguyên tố trong ma trận các số nguyên.
- 7.31. Viết hàm đếm các giá trị cực đại trong ma trận các số nguyên.
- 7.32. Viết hàm đếm các giá trị cực tiểu trong ma trận các số nguyên.
- 7.33. Viết hàm đếm các cực trị trong ma trận các số nguyên (một phần tử được coi là cực trị khi nó là giá trị cực đại hay cực tiểu).
- 7.34. Viết hàm đếm các giá trị là số hoàn thiện trong ma trận các số nguyên.

Kỹ thuật sắp xếp

Ví dụ 7.6

Viết hàm sắp xếp ma trận tăng dần từ trên xuống dưới và từ trái sang phải không dùng mảng phụ.

```
void SapTang(MANG2D a, int d, int c)
{
    for(int i = 0; i <= d*c-2; i++)
        for(int j = 0; j <= d*c-1; j++)
            if(a[i/c][i%c] < a[j/c][j%c])
            {
                int tmp = a[i/c][i%c];
                a[i/c][i%c] = a[j/c][j%c];
                a[j/c][j%c] = tmp;
            }
}
```

- 7.35. Viết hàm sắp xếp ma trận theo thứ tự tăng dần từ trên xuống dưới và từ trái qua phải theo phương pháp dùng mảng phụ.

Hướng dẫn: Đổ ma trận sang mảng một chiều, sắp xếp trên mảng một chiều theo thứ tự

tăng dần, sau đó chuyển ngược mảng một chiều thành ma trận kết quả.

- 7.36. Viết hàm sắp xếp ma trận theo thứ tự giảm dần từ trên xuống dưới và từ trái sang phải.
 7.37. Viết hàm sắp xếp các dòng trên ma trận theo thứ tự tăng dần.
 7.38. Viết hàm sắp xếp các cột trên ma trận theo thứ tự giảm dần.
 7.39. Viết hàm sắp xếp ma trận theo đường zig zag ngang. Ví dụ:

5	6	3		1	2	3
1	8	7	→	6	5	4
2	4	9		7	8	9

- 7.40. Viết hàm sắp xếp ma trận theo đường zig zag chéo. Ví dụ:

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

- 7.41. Viết hàm sắp xếp ma trận theo đường xoắn ốc từ ngoài vào trong theo chiều kim đồng hồ.
 Ví dụ:

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

Kỹ thuật thêm - xóa - thay thế

- 7.42. Viết hàm xóa một dòng i trên ma trận.
 7.43. Viết hàm xóa một cột j trên ma trận.
 7.44. Viết hàm xóa dòng có tổng lớn nhất trên ma trận.
 7.45. Viết hàm hoán vị dòng có tổng lớn nhất với dòng có tổng nhỏ nhất.
 7.46. Viết hàm tìm và thay thế các phần tử chẵn trong ma trận bằng ước số nhỏ nhất của nó.
 7.47. Viết hàm thay thế những phần tử có giá trị x thành phần tử có giá trị y trong ma trận
 (x, y nhập từ bàn phím)

Mảng vuông và các kỹ thuật xử lý

Định nghĩa 7.2. Mảng vuông là mảng hai chiều có số dòng và số cột bằng nhau.

Định nghĩa 7.3. Cho một mảng vuông

- Đường chéo chính (loại 1): *chỉ số dòng = chỉ số cột*
- Đường chéo song song với đường chéo chính (loại 1): *chỉ số dòng - chỉ số cột = hằng số*
- Đường chéo phụ (loại 2): *chỉ số dòng + chỉ số cột = số dòng (hoặc số cột)*
- Đường chéo song song với đường chéo phụ (loại 2): *chỉ số dòng + chỉ số cột = hằng số*

Ví dụ 7.7

Cho mảng vuông A , in ra các phần tử nằm trên đường chéo song song với đường chéo chính và xuất phát từ (io, jo)

```
for(i = io, j = jo; i < n; i ++, j ++)  
    printf("%4d", A[i][j]);
```

Ví dụ 7.8

Cho mảng vuông A , in ra các phần tử nằm trên đường chéo song song với đường chéo phụ và xuất phát từ (io, jo)

```
for(i = io, j = jo; i < n && j >= 0; i ++, j -)  
    printf("%4d", A[i][j]);
```

- 7.48. Viết chương trình in ra các phần tử nằm trên 2 đường chéo chính và phụ.
- 7.49. Viết hàm in ra các phần tử nằm phía trên đường chéo phụ của ma trận vuông các số nguyên.
- 7.50. Viết hàm in ra các phần tử nằm phía dưới đường chéo phụ của ma trận vuông các số nguyên.
- 7.51. Viết hàm in ra các phần tử nằm phía trên đường chéo chính của ma trận vuông các số nguyên.
- 7.52. Viết hàm in ra các phần tử nằm phía dưới đường chéo chính của ma trận vuông các số nguyên.
- 7.53. Viết hàm tìm phần tử lớn nhất nằm trên đường chéo chính của ma trận vuông.

- 7.54. Viết hàm in các số nguyên tố nằm trên đường chéo phụ của ma trận vuông.
- 7.55. Viết hàm tìm phần tử nhỏ nhất trên mỗi đường chéo loại 2 của ma trận.
- 7.56. Viết hàm tính tổng các phần tử nằm trên đường chéo chính của ma trận vuông.
- 7.57. Viết hàm tìm đường chéo có tổng lớn nhất trong các đường chéo loại 1.
- 7.58. Viết hàm tính tổng các giá trị nhỏ nhất nằm trên từng đường chéo loại 2.
- 7.59. Viết hàm đến các phần tử nguyên tố trên đường chéo chính của ma trận vuông các số nguyên.
- 7.60. Viết hàm đếm các giá trị chẵn trên đường chéo chính của ma trận vuông các số nguyên.
- 7.61. Viết hàm đếm các giá trị là bội của 3 và 5 trên đường chéo chính của ma trận các số nguyên.
- 7.62. Viết hàm đếm các giá trị nguyên tố trên 2 đường chéo (chính, phụ) của ma trận vuông các số nguyên.
- 7.63. Cho ma trận vuông, viết hàm sắp xếp tăng dần các phần tử nằm trên các đường chéo song song với đường chéo chính.
- 7.64. Viết chương trình nhập một ma trận vuông các số nguyên, và thực hiện những công việc sau:
 - a) Sắp xếp các phần tử nằm trên các đường chéo loại 1 tăng dần
 - b) Sắp xếp các phần tử nằm trên các đường chéo loại 2 giảm dần.
 - c) Sắp xếp với điều kiện: các phần tử trên đường chéo chính tăng, các phần tử trên các đường chéo song song với đường chéo chính giảm.

Luyện tập và nâng cao

- 7.65. Viết chương trình tính tổng, tích của hai ma trận các số nguyên.
- 7.66. Viết hàm kiểm tra xem ma trận vuông các số nguyên có đối xứng qua đường chéo chính hay không.
- 7.67. Viết hàm kiểm tra xem trong ma trận vuông cấp n có hàng nào trùng nhau hay không, nếu có thì chỉ rõ những hàng nào. (trùng giá trị và vị trí).
- 7.68. Viết chương trình nhập vào ma trận vuông kích thước $n \times n$. Hãy viết hàm thực hiện những công việc sau:
 - a) In ra các phần tử trên 4 đường biên của ma trận.
 - b) Tính tổng các phần tử trên biên.
- 7.69. (*) Viết chương trình xoay ma trận các số thực 90° ngược chiều kim đồng hồ. Ví dụ:

$$\begin{array}{cccc}
 1 & 2 & 3 & 4 \\
 5 & 6 & 7 & 8 \\
 9 & 10 & 11 & 12 \\
 13 & 14 & 15 & 16
 \end{array}
 \rightarrow
 \begin{array}{cccc}
 4 & 8 & 12 & 16 \\
 3 & 7 & 11 & 15 \\
 2 & 6 & 10 & 14 \\
 1 & 5 & 9 & 13
 \end{array}$$

7.70. Viết chương trình dịch phải xoay vòng một cột trong ma trận các số thực.

7.71. Viết chương trình dịch xuống xoay vòng một dòng trong ma trận các số thực.

7.72. (*) Cho ma trận $A_{m \times n}$ các số nguyên hãy phát sinh ma trận B sao cho B là ma trận lật ngược của ma trận A . Ví dụ:

$$\begin{array}{cccc}
 1 & 2 & 3 & 4 \\
 5 & 6 & 7 & 8 \\
 9 & 10 & 11 & 12 \\
 13 & 14 & 15 & 16
 \end{array}
 \rightarrow
 \begin{array}{cccc}
 4 & 3 & 2 & 1 \\
 8 & 7 & 6 & 5 \\
 12 & 11 & 10 & 9 \\
 16 & 15 & 14 & 13
 \end{array}$$

7.73. (**) Cho ma trận $A_{m \times n}$ hãy phát sinh ma trận B sao cho phần tử $B(i, j)$ là trung bình cộng của các phần tử trong hình vuông 3×3 tâm tại (i, j) . Ví dụ

$$\begin{array}{cccc}
 1 & 5 & 2 & 6 \\
 4 & 2 & 3 & 6 \\
 8 & 7 & 9 & 1 \\
 10 & 2 & 12 & 13
 \end{array}
 \rightarrow
 \begin{array}{cccc}
 3 & 2 & 4 & 4 \\
 4 & 4 & 4 & 4 \\
 5 & 6 & 6 & 7 \\
 6 & 8 & 7 & 8
 \end{array}$$

7.74. (**) Cho ma trận các số nguyên dương $A_{m \times n}$. Hãy xây dựng ma trận $B_{m \times n}$. Sao cho phần tử $B(i, j)$ là số lớn nhất trong ô vuông 3×3 tâm tại (i, j) của A . Ví dụ:

$$\begin{array}{cccc}
 1 & 5 & 2 & 6 \\
 4 & 2 & 3 & 6 \\
 8 & 7 & 9 & 1 \\
 10 & 2 & 12 & 13
 \end{array}
 \rightarrow
 \begin{array}{cccc}
 5 & 5 & 6 & 6 \\
 8 & 9 & 9 & 9 \\
 10 & 12 & 13 & 13 \\
 10 & 12 & 13 & 13
 \end{array}$$

7.75. (**) Cho ma trận $A_{m \times n}$. Hãy xây dựng ma trận $B_{m \times n}$ với phần tử $B(i, j)$ được xác định theo qui tắc sau: tại vị trí (i, j) trên mảng A kẻ hai tia vuông góc với nhau, tạo thành với trục hoành một góc 45° từ trên xuống dưới; $B(i, j)$ là tổng của tất cả các số của vùng mặt phẳng tạo bởi hai tia này và các cạnh của bảng. Ví dụ:

1 3 2 25 30 23
 6 4 3 → 13 18 15
 2 5 7 2 5 7

7.76. (**) Cho ma trận vuông $A_{m \times n}$. Hãy xây dựng mảng $B_{m \times n}$ bằng cách: phần tử $B(i, j)$ là số lớn nhất trong tam giác vuông vẽ từ $A(i, j)$ tới đường chéo chính. Ví dụ:

1 3 2 1 4 7
 6 4 3 → 6 4 7
 2 5 7 7 7 7

7.77. (*) Viết chương trình hiển thị đồng hồ điện tử (gồm giờ phút), với giờ lấy từ hệ thống và đồng hồ được cập nhật theo phút.

Hướng dẫn: Tạo 1 ma trận giá trị gồm 0 hoặc 1, vị trí nào cần hiển thị thì gán giá trị là 1, ngược lại có giá trị là 0. Sau mỗi phút cập nhật lại ma trận và hiển thị lên màn hình.

Ví dụ: 01 giờ 25 phút

1111 11 11111111 11111111
 11 11 11 11 11 11
 11 11 11 11111111 11111111
 11 11 11 11 11 11
 1111 11 11111111 11111111

7.78. Nhập vào mảng hai chiều gồm n dòng và m cột các số nguyên. Hãy tìm phần tử lớn nhất trên mỗi dòng và đồng thời nhỏ nhất trên mỗi cột, hoặc lớn nhất trên mỗi cột và đồng thời nhỏ nhất trên mỗi dòng. Có bao nhiêu phần tử như thế? Ví dụ:

3 **6** 2 1
 4 7 6 9
5 15 8 7

7.79. Viết chương trình tạo ngẫu nhiên một ma trận các số nguyên trong khoảng $[0, \dots, 50]$, tìm những phần tử cực đại (là phần tử lớn hơn các phần tử xung quanh). Ví dụ:

2 6 **8** 4
9 7 5 3
 6 2 **8** 1

7.80. (**) Cho ma trận các số nguyên $A_{m \times n}$ ($m, n \geq 3$). Hãy tìm ma trận con (3×3) có tổng lớn nhất. Ví dụ:

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 11 \\ 13 & 14 & 15 & 16 \end{array} \rightarrow \begin{array}{ccc} 6 & 7 & 8 \\ 10 & 11 & 11 \\ 14 & 15 & 16 \end{array}$$

7.81. Nhập ma trận vuông cấp $n \times n$ ($n < 10$). In ra các phần tử của ma trận này theo hướng của đường chéo chính:

7.82. (**) Hãy điền các số từ 1 đến n^2 vào ma trận cấp n ($n > 2$), chỉ xét trường hợp n là số lẻ với tính chất P là tổng các số bằng nhau.

Hướng dẫn: Ma phương của một bảng vuông cấp n , trong mỗi ô nhận một giá trị sao cho, mỗi hàng, mỗi cột và mỗi đường chéo đều thoả mãn một tính chất P nào đó cho trước.
Ví dụ:

$$\begin{array}{ccc} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{array}$$

7.83. (*) Viết hàm in ma trận các số nguyên dương theo qui luật được mô tả như sau: các phần tử phía trên đường chéo phụ là giá trị bình phương, các giá trị từ đường chéo phụ trở xuống là các số nguyên tố. Ví dụ:

$$\begin{array}{ccccc} 1 & 9 & 36 & 100 & 31 \\ 4 & 25 & 81 & 37 & 17 \\ 16 & 64 & 41 & 19 & 7 \\ 49 & 43 & 23 & 11 & 3 \\ 47 & 29 & 13 & 5 & 2 \end{array}$$

7.84. Cho ma trận vuông a cấp n (n lẻ và $3 \leq n \leq 15$), mỗi phần tử đều có giá trị nguyên dương. Hãy xây dựng hàm kiểm tra xem ma trận a có phải là ma phương hay không?

7.85. (**) Viết chương trình giải bài toán 8 hậu. Hãy đặt 8 con hậu trên bàn cờ 8×8 sao cho chúng không ăn nhau (2 hậu ăn nhau khi cùng hàng, cùng cột và cùng nằm trên đường chéo).

Hướng dẫn: Dùng ma trận 8×8 để lưu bàn cờ. Mỗi ô có 3 trạng thái:

- a) Có hậu 1
- b) Ô trống 0
- c) Ô không được đi -1

7.86. (**) Viết chương trình giải bài toán mã đi tuần. Hãy đi con mã 64 lượt đi trên bàn cờ 8×8 sao cho mỗi ô chỉ đi qua một lần (xuất phát từ một ô bất kỳ)

Hướng dẫn: Đứng tại một ô trên bàn cờ con mã có thể đi được 1 trong 8 hướng sau. Khai báo 8 hướng đi của mã như sau:

```
typedef struct DIEM
{
    int x, y;
};
DIEM huongdi[8]={{-2,-1},{-2,1},{-1,2},{1,2},
                 {2,1},{2,-1},{1,-2},{-1,-2}};
```

Trong đó mỗi thành phần của huongdi là độ lệch của dòng và cột so với vị trí của con mã. Ví dụ: huongdi[0] có độ lệch 2 dòng và 1 cột. (Giá trị âm biểu thị độ lệch về bên trái cột hay hướng lên của dòng). Chọn vị trí đi kế tiếp sao cho vị trí đó phải gần với biên hay góc nhất (tức số đường đi có thể đi là ít nhất).

7.87. Viết chương trình giải bài toán 8-puzzle. Cho ma trận vuông 3×3 gồm các số nguyên từ 0 đến 8 trong đó 0 là ô trống. Bài toán đặt ra là hãy đưa ma trận ở một trạng thái đầu về trạng thái đích, mỗi lần chỉ dịch chuyển được 1 ô. Ví dụ:

$$\text{Trạng thái đầu} \begin{bmatrix} 2 & 1 & 0 \\ 3 & 8 & 7 \\ 6 & 4 & 5 \end{bmatrix} \rightarrow \text{Trạng thái đích} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$$

7.3 TÓM TẮT

Kiểu dữ liệu mảng hai chiều được ứng dụng rộng rãi trong các bài toán về tìm đường đi trong đồ thị, xử lý ảnh, xử lý những dữ liệu dạng bảng.

KIỂU DỮ LIỆU CẤU TRÚC

Cung cấp cơ chế cho phép khai báo các kiểu dữ liệu mới để giải quyết theo yêu cầu của bài toán dựa vào những kiểu dữ liệu cơ bản được cài đặt sẵn trong ngôn ngữ lập trình.

8.1 LÝ THUYẾT

Định nghĩa 8.1. Kiểu dữ liệu cấu trúc (**struct**) là gom nhóm các phần tử có thể không cùng kiểu dữ liệu.

Khái báo kiểu dữ liệu cấu trúc

Cú pháp

```
struct <tên cấu trúc> {
    <các dữ liệu thành phần>
};
```

Ngoài ra, ta có thể dùng từ khoá **typedef** để định nghĩa một tên mới cho kiểu dữ liệu đã có.

```
typedef struct {
    <các dữ liệu thành phần>
} <tên kiểu dữ liệu cấu trúc>;
```

Ví dụ 8.1

Kiểu dữ liệu về ngày DATE gồm các thành phần

- Thứ (thu): chuỗi có tối đa 4 ký tự.
- Ngày (ngay): số nguyên 1 byte.
- Tháng (thang): số nguyên 1 byte.
- Năm (nam): số nguyên 2 bytes.

Ta định nghĩa cấu trúc SDATE

```
struct SDATE {
    char thu[5];
    unsigned char ngay;
    unsigned char thang;
    int nam;
};
```

hoặc kiểu dữ liệu cấu trúc DATE

```
typedef struct SDATE {
    char thu[5];
    unsigned char ngay;
    unsigned char thang;
    int nam;
} DATE;
```

Khai báo biến cấu trúc

Khi ta định nghĩa kiểu dữ liệu tức là ta có một kiểu dữ liệu mới, muốn sử dụng ta phải khai báo biến. Cú pháp khai báo kiểu dữ liệu cũng giống như cách khai báo của các kiểu dữ liệu chuẩn.

Cú pháp

```
struct <tên cấu trúc> <tên biến>;
```

hoặc

```
<tên cấu trúc> <tên biến>;
```

hoặc

```
<tên kiểu dữ liệu cấu trúc> <tên biến>;
```


Khai báo biến `x` có kiểu cấu trúc `SDATE`

```
struct SDATE x;
```

Khai báo biến `x` có kiểu `DATE`

```
DATE x;
```

Lưu ý. Kiểu các thành phần của một cấu trúc cũng có thể là kiểu dữ liệu cấu trúc, kiểu dữ liệu mảng hoặc bất cứ kiểu dữ liệu nào.

Ví dụ 8.2

Định nghĩa kiểu dữ liệu của học sinh `HOCSINH` gồm:

- Mã số học sinh (`MSHS`): chuỗi có tối đa 5 ký tự.
- Họ tên (`hoten`): chuỗi có tối đa 30 ký tự.
- Ngày tháng năm sinh (`ngaysinh`): kiểu `DATE`.
- Địa chỉ (`diachi`): chuỗi có tối đa 50 ký tự.
- Giới tính (`phai`): chuỗi có tối đa 3 ký tự.
- Điểm trung bình (`diemtb`): số thực.

Ta định nghĩa kiểu `HOCSINH` như sau:

```
typedef struct HOCSINH {
    char MSHS[6];
    char hoten[31];
    DATE ngaysinh;
    char diachi[51];
    unsigned char phai[4];
    float diemtb;
};
```

Truy xuất thành phần

Cú pháp

<tên biến cấu trúc>.<tên thành phần>

Truy xuất thành phần ngay của biến `x`

```
DATE x; // khai bao bien x kieu DATE
x.ngay = 5; // gan ngay bang 5
```

Ví dụ 8.3

Viết chương trình nhập vào tọa độ hai điểm trong mặt phẳng và tính tổng hai tọa độ này.

```
#include <conio.h>
#include <stdio.h>
//khai bao mot kieu du lieu DIEM gom toa do x va y
typedef struct {
    int x;
    int y;
} DIEM;

void Nhap (DIEM &d)
{
    printf("\nNhap vao tao do diem\n");
    printf("Hoanh do: ");
    scanf("%d", &d.x);
    printf("Tung do: ");
    scanf("%d", &d.y);
}

void Xuat(DIEM d)
{
    printf("\nToa do diem: (%d, %d)",d.x,d.y);
}

DIEM Tong(DIEM d1,DIEM d2)
{
    DIEM temp;
    temp.x = d1.x + d2.x;
    temp.y = d1.y + d2.y;
    return Temp;
}
```

```

}

void main()
{
    DIEM A, B, AB; // khai bao 3 diem A, B, AB;
    Nhap (A);
    Xuat (A);
    Nhap (B);
    Xuat (B);
    printf("\n Tong cua hai diem vua nhap la: ");
    AB = Tong(A, B);
    Xuat(AB);
    getch();
}

```

Mảng cấu trúc

- Cách khai báo tương tự như mảng một chiều hay ma trận (Kiểu dữ liệu bây giờ là kiểu dữ liệu có cấu trúc).
- Cách truy cập phần tử trong mảng cũng như truy cập trên mảng một chiều hay ma trận. Nhưng do từng phần tử có kiểu cấu trúc nên phải chỉ định rõ cần lấy thành phần nào, tức là phải truy cập đến thành phần cuối cùng có kiểu là dữ liệu cơ bản (xem lại bảng các kiểu dữ liệu cơ bản).

Kỹ thuật viết chương trình có mảng cấu trúc

Do kiểu dữ liệu có cấu trúc thường chứa rất nhiều thành phần nên khi viết chương trình loại này ta cần lưu ý:

- Xây dựng hàm xử lý cho một kiểu cấu trúc.
- Muốn xử lý cho mảng cấu trúc, ta gọi lại hàm xử lý cho một kiểu cấu trúc đã được xây dựng bằng cách dùng vòng lặp.

Ví dụ 8.4

Cho một lớp học gồm n học sinh ($n \leq 50$). Thông tin của một học sinh được mô tả trong ví

dự trước. Hãy viết chương trình nhập và xuất danh sách học sinh sau đó đếm xem có bao nhiêu học sinh được lên lớp (Điều kiện được lên lớp là điểm trung bình ≥ 5.0). Cách làm:

- Trước hết ta phải xây dựng hàm nhập và xuất cho 1 học sinh.
- Xây dựng hàm nhập và xuất ngày tháng năm (Kiểu dữ liệu DATE).
- Sau đó mới xây dựng hàm nhập và xuất cho danh sách học sinh.

```
#define MAX 50
typedef struct {
    char thu[5];
    unsigned char ngay;
    unsigned char thang;
    int nam;
} DATE;

typedef struct {
    char MSHS[6];
    char hoten[31];
    struct DATE ngaysinh;
    char diachi[51];
    unsigned char phai[4];
    float diemtb;
} HOCSINH;

void NhapNamSinh(DATE &d);
void XuatNamSinh(DATE d);
void Nhap1HS(HOCSINH &hs);
void Xuat1HS(HOCSINH hs);
void NhapDSHS(HOCSINH lh[], int &n);
void XuatDSHS(HOCSINH lh[], int n);
int DemHSLenLop(HOCSINH lh[], int n);

void main()
{
    HOCSINH lh[MAX]; // Khai báo mảng có tối đa 50 học sinh
    int n, sohsdau;
    NhapDSHS(lh, n);
    XuatDSHS(lh, n);
}
```

```

    sohsdau = DemHSLenLop(lh, n);
    printf("\nSố lượng học sinh được lên lớp là: %d", sohsdau);
    getch();
}

void NhapNamSinh(DATE &d)
{
    printf("\nNhập vào ngày: ");
    scanf("%u", &d.ngay);
    printf("\nNhập vào tháng: ");
    scanf("%u", &d.thang);
    printf("\nNhập vào năm: ");
    scanf("%d", &d.nam);
}

void XuatNamSinh(DATE d)
{
    printf("%02u / %02u / %4d", d.ngay, d.thang, d.nam);
}

void Nhap1HS(HOCSINH &hs)
{
    flushall(); // Xóa vùng đệm
    printf("\nNhập mã số học sinh: ");
    gets(hs.MSHS);
    printf("\nNhập họ tên học sinh: ");
    gets(hs.hoten);
    printf("\nNhập ngày tháng năm sinh: ");
    flushall(); // Xóa vùng đệm
    NhapNamSinh(hs.ngaysinh);
    printf("\nNhập vào địa chỉ: ");
    flushall(); // Xóa vùng đệm
    gets(hs.diachi);
    printf("\nPhái: ");
    gets(hs.phai);
}

```

```

    printf("\nNhap vao diem trung binh: ");
    fflush();
    scanf("%f", &hs.diemtb);
}

void NhapDSHS(HOCSINH lh[], int &n)
{
    printf("\nNhap vao so luong hoc sinh: ");
    scanf("%d", &n);
    for(int i=0; i<n; i++)
    {
        printf("\nNhap vao thong tin cua hoc sinh thu %d:\n", i+1);
        Nhap1HS(lh[i]); //Goi ham nhap thong tin 1 hoc sinh
    }
}

void Xuat1HS(HOCSINH hs)
{
    printf("\nMa so hoc sinh: %s", hs.MSHS);
    printf("\nHo ten hoc sinh: %s", hs.hoten);
    printf("\nNgay thang nam sinh: ");
    XuatNamSinh(hs.ngaysinh);
    printf("\nDia chi: %s", hs.diachi);
    printf("\nPhai: %s", hs.phai);
    printf("\nDiem trung binh: %2.2f", hs.diemtb);
}

void XuatDSHS(HOCSINH lh[], int n)
{
    for(int i=0; i<n; i++)
    {
        printf("\n\nThong tin hoc sinh thu %d:", i+1);
        Xuat1HS(lh[i]); //Goi ham xuat thong tin 1 hoc sinh
    }
}

```

```

int DemHSLenLop(HOCSINH lh[], int n)
{
    int d=0;
    for(int i=0; i<n; i++)
        if(lh[i].diemtb>=5.0)
            d++;
    return d;
}

```

Kết quả ví dụ khi chạy chương trình:

```

Nhap vao thong tin cua hoc sinh thu 1:
Nhap ma so hoc sinh: 02313
Nhap ho ten hoc sinh: Nguyen Van A
Nhap ngay thang nam sinh:
Nhap vao ngay: 12
Nhap vao thang: 03
Nhap vao nam: 1980
Nhap vao dia chi: 60 Phan Dang Luu Q.Phu Nhuan
Phai: Nam Nhap vao diem trung binh: 6.5
Nhap vao thong tin cua hoc sinh thu 2:
Nhap ma so hoc sinh: 03852
Nhap ho ten hoc sinh: Ly Thi B
Nhap ngay thang nam sinh:
Nhap vao ngay: 05
Nhap vao thang: 12
Nhap vao nam: 1981
Nhap vao dia chi: 24 Ly Tu Trong Q.1
Phai: Nu Nhap vao diem trung binh: 3.5
Thong tin hoc sinh thu 1:
Ma so hoc sinh: 02313
Ho ten hoc sinh: Nguyen Van A
Ngay thang nam sinh: 12 / 03 / 1980

```

Dia chi: 60 Phan Dang Luu Q.Phu Nhuan
 Phai: Nam
 Diem trung binh: 6.50
 Thông tin học sinh thu 2:
 Ma so hoc sinh: 03852
 Ho ten hoc sinh: Ly Thi B
 Ngay thang nam sinh: 05 / 12 / 1981
 Dia chi: 24 Ly Tu Trong Q.1
 Phai: Nu Diem trung binh: 3.50
 So luong hoc sinh duoc len lop la: 1

Ví dụ 8.5

Cho một mảng các phân số (PHANSO) gồm n phần tử ($n \leq 50$). Hãy viết chương trình nhập và xuất danh sách các phân số sau đó tìm phân số có giá trị lớn nhất, tổng và tích các phân số và nghịch đảo giá trị các phân số trong mảng. Cách làm:

- Trước hết ta phải xây dựng hàm nhập và xuất cho 1 phân số.
- Xây dựng hàm tính tổng, hiệu, tích, thương, rút gọn, so sánh và nghịch đảo cho 2 phân số.
- Sau đó mới xây dựng hàm nhập, xuất, tính tổng, tích cho mảng các phân số.

```

#define MAX 100
typedef struct {
    int tu, mau;
} PHANSO;

void NhapPS(PHANSO &ps);
void XuatPS(PHANSO ps);
void NhapMangPS(PHANSO dsps[], int &n);
void XuatMangPS(PHANSO dsps[], int n);
PHANSO TimMax(PHANSO dsps[], int n);
int KiemTra(PHANSO ps);
//Tra ve 0: Neu khong hop le
//Tra ve 1: Neu hop le
int USCLN(int a, int b); PHANSO RutGon(PHANSO ps);
  
```



```

PHANSO NghichDao(PHANSO ps);
PHANSO Nhan(PHANSO ps1, PHANSO ps2);
PHANSO Chia(PHANSO ps1, PHANSO ps2);
PHANSO Tru(PHANSO ps1, PHANSO ps2);
PHANSO Cong(PHANSO ps1, PHANSO ps2);
int SoSanh(PHANSO ps1, PHANSO ps2);
//Tra ve 0: ps1=ps2
//Tra ve 1: ps1>ps2
//Tra ve -1: ps1<ps2
PHANSO TongCacPS(PHANSO dsps[], int n);
PHANSO TichCacPS(PHANSO dsps[], int n);
void NghichDaoCacPS(PHANSO dsps[], int n);

void main()
{
    int n;
    PHANSO a[MAX], max, s, p;
    NhapMangPS(a, n);
    printf("\nMang cac phan so vua nhap: ");
    XuatMangPS(a, n);
    max=TimMax(a, n);
    printf("\nPhan so co gia tri lon nhat: ");
    XuatPS(max);
    s=TongCacPS(a, n);
    printf("\nTong gia tri cac phan so co trong mang: ");
    XuatPS(s);
    p=TichCacPS(a, n);
    printf("\nTich gia tri cac phan so co trong mang: ");
    XuatPS(p);
    NghichDaoCacPS(a, n);
    printf("\nMang phan so sau khi nghich dao cac phan tu: ");
    XuatMangPS(a, n);
    getch();
}

```

```

void NhapPS(PHANSO &ps)
{
    do {
        printf("\nNhap tu so: ");
        scanf("%d", &ps.tu);
        printf("\nNhap mau so: ");
        scanf("%d", &ps.mau);
        if(!KiemTra(ps))
            printf("\nMau so khong duoc bang 0, nhap lai phan so\n");
        else
            break;
    } while(1);
    ps=RutGon(ps);
}

void XuatPS(PHANSO ps)
{
    printf("%d", ps.tu);
    if(ps.tu&&ps.mau!=1)
        printf("/%d", ps.mau);
}

void NhapMangPS(PHANSO dsps[], int &n)
{
    printf("\nNhap so luong phan so: ");
    scanf("%d", &n);
    for(int i=0; i<n; i++)
    {
        printf("\nNhap vao phan so thu %d: ", i+1);
        NhapPS(dsps[i]);
    }
}

void XuatMangPS(PHANSO dsps[], int n)
{

```

```

    for(int i=0; i<n; i++)
    {
        XuatPS(dsps[i]);
        printf("\t");
    }
}

int KiemTra(PHANSO ps)
{
    if(ps.mau==0) return 0;
    return 1;
}

int USCLN(int a, int b)
{
    a=abs(a);
    b=abs(b);
    while(a!=b)
    {
        if(a>b) a=a-b;
        else b=b-a;
    }
    return a;
}

PHANSO RutGon(PHANSO ps)
{
    int us;
    if(ps.tu==0) return ps;
    us=USCLN(ps.tu, ps.mau);
    ps.tu=ps.tu/us;
    ps.mau=ps.mau/us;
    return ps;
}

```

```

PHANSO NghichDao(PHANSO ps)
{
    PHANSO kq;
    kq.tu=ps.mau;
    kq.mau=ps.tu;
    return kq;
}

PHANSO Nhan(PHANSO ps1, PHANSO ps2)
{
    PHANSO kq;
    kq.tu=ps1.tu*ps2.tu;
    kq.mau=ps1.mau*ps2.mau;
    kq=RutGon(kq);
    return kq;
}

PHANSO Chia(PHANSO ps1, PHANSO ps2)
{
    PHANSO kq;
    kq=Nhan(ps1, NghichDao(ps2));
    return kq;
}

PHANSO Tru(PHANSO ps1, PHANSO ps2)
{
    PHANSO kq;
    kq.tu=ps1.tu*ps2.mau-ps1.mau*ps2.tu;
    kq.mau=ps1.mau*ps2.mau; kq=RutGon(kq);
    return kq;
}

PHANSO Cong(PHANSO ps1, PHANSO ps2)
{
    PHANSO kq;

```

```

    kq.tu=ps1.tu*ps2.mau+ps1.mau*ps2.tu;
    kq.mau=ps1.mau*ps2.mau;
    kq=RutGon(kq);
    return kq;
}

int SoSanh(PHANSO ps1, PHANSO ps2)
{
    ps1=RutGon(ps1);
    ps2=RutGon(ps2);
    if(ps1.tu==ps2.tu&&ps1.mau==ps2.mau)
        return 0;
    if(ps1.tu*ps2.mau>ps2.tu*ps1.mau)
        return 1;
    return -1;
}

PHANSO TimMax(PHANSO dsps[], int n)
{
    PHANSO max;
    max=dsps[0];
    for(int i=1; i<n; i++)
        if(SoSanh(dsps[i], max)==1)
            max=dsps[i];
    return max;
}

PHANSO TongCacPS(PHANSO dsps[], int n)
{
    PHANSO s=dsps[0];
    for(int i=1; i<n; i++)
    {
        s=Cong(s, dsps[i]);
    }
    return s;
}

```

```

}

PHANSO TichCacPS(PHANSO dsps[], int n)
{
    PHANSO p=dsps[0];
    for(int i=1; i<n; i++)
    {
        p=Nhan(p, dsps[i]);
    }
    return p;
}

void NghichDaoCacPS(PHANSO dsps[], int n)
{
    for(int i=0; i<n; i++)
    {
        dsps[i]=NghichDao(dsps[i]);
    }
}

```

Kết quả ví dụ khi chạy chương trình:

```

Nhap so luong phan so: 5
Nhap vao phan so thu 1:
Nhap tu so: 1
Nhap mau so: 3
Nhap vao phan so thu 2:
Nhap tu so: 7
Nhap mau so: 4
Nhap vao phan so thu 3:
Nhap tu so: 9
Nhap mau so: 7
Nhap vao phan so thu 4:
Nhap tu so: 5
Nhap mau so: 6

```

```

Nhap vao phan so thu 5:
Nhap tu so: 4
Nhap mau so: 7
Mang cac phan so vua nhap: 1/3 7/4 9/7 5/6 4/7
Phan so co gia tri lon nhat: 7/4
Tong gia tri cac phan so co trong mang: 401/84
Tich gia tri cac phan so co trong mang: 5/14
Mang phan so sau khi nghich dao cac phan tu: 3 4/7 7/9 6/5 7/4

```

8.2 BÀI TẬP

Bài tập cơ bản

- 8.1. Viết chương trình sử dụng con trỏ cấu trúc để hiển thị giờ, phút, giây ra màn hình, và tính khoảng cách giữa 2 mốc thời gian.
- 8.2. Viết chương trình sử dụng con trỏ cấu trúc thể hiện ngày, tháng, năm ra màn hình, và tính khoảng cách giữa 2 ngày.
- 8.3. Viết chương trình khai báo kiểu dữ liệu thể hiện một số phức. Sử dụng kiểu này để viết hàm tính tổng, hiệu, tích của hai số phức.
- 8.4. Viết chương trình khai báo kiểu dữ liệu để biểu diễn một phân số. Hãy viết hàm thực hiện những công việc sau:
 - a) Tính tổng, hiệu, tích, thương hai phân số.
 - b) Rút gọn phân số.
 - c) Qui đồng hai phân số.
 - d) So sánh hai phân số.
- 8.5. Viết chương trình khai báo kiểu dữ liệu để biểu diễn một hỗn số. Hãy viết hàm thực hiện những công việc sau:
 - a) Đổi hỗn số sang phân số
 - b) Tính tổng, tích hai hỗn số
- 8.6. Viết chương trình khai báo kiểu dữ liệu để biểu diễn một điểm trong hệ tọa độ Oxy. Hãy viết hàm thực hiện các công việc sau:
 - a) Tìm những điểm đối xứng của nó qua tung độ, hoành độ, tọa độ tâm.
 - b) Hãy tính tổng, hiệu, tích của hai điểm trong mặt phẳng tọa độ Oxy.

c) Tính khoảng cách giữa hai điểm.

8.7. Cho một hình trụ có các thông tin sau: BánKính (bán kính hình trụ kiểu số thực), ChiềuCao (chiều cao hình trụ kiểu số thực). Hãy thực hiện các công việc sau.

a) Nhập dữ liệu cho hình trụ trên.

b) Tính diện tích xung quanh, diện tích toàn phần, thể tích hình trụ.

Luyện tập và nâng cao

8.8. Viết chương trình tạo một mảng các số phức. Hãy viết hàm tính tổng, tích các số phức có trong mảng.

8.9. Viết chương trình tạo một mảng các phân số. Hãy viết hàm thực hiện các công việc sau:

a) Tính tổng tất cả các phân số (kết quả dưới dạng phân số tối giản)

b) Tìm phân số lớn nhất, phân số nhỏ nhất.

c) Sắp xếp mảng tăng dần.

8.10. Viết chương trình khai báo kiểu dữ liệu STACK (cơ chế LIFO). Viết hàm làm những công việc sau:

a) Kiểm tra STACK rỗng

b) Kiểm tra STACK đầy

c) Thêm phần tử vào STACK

d) Lấy phần tử ra khỏi STACK

8.11. Tổ chức dữ liệu để quản lý sinh viên bằng cấu trúc mẫu tin trong một mảng N phần tử, mỗi phần tử có cấu trúc như sau:

- Mã sinh viên.
- Tên.
- Năm sinh.
- Điểm toán, lý, hoá, điểm trung bình.

Viết chương trình thực hiện những công việc sau:

a) Nhập danh sách các sinh viên cho một lớp học.

b) Xuất danh sách sinh viên ra màn hình.

c) Tìm sinh viên có điểm trung bình cao nhất.

d) Sắp xếp danh sách lớp theo thứ tự tăng dần của điểm trung bình.

e) Sắp xếp danh sách lớp theo thứ tự giảm dần của điểm toán.

f) Tìm kiếm và in ra các sinh viên có điểm trung bình lớn hơn 5 và không có môn nào

dưới 3.

g) Tìm sinh viên có tuổi lớn nhất.

h) Nhập vào tên của một sinh viên. Tìm và in ra các thông tin liên quan đến sinh viên đó (nếu có).

8.12. Tổ chức dữ liệu quản lí danh mục các bộ phim VIDEO, các thông tin liên quan đến bộ phim này như sau:

- Tên phim (tựa phim).
- Thể loại (3 loại: hình sự, tình cảm, hài).
- Tên đạo diễn.
- Tên diễn viên nam chính.
- Tên diễn viên nữ chính.
- Năm sản xuất.
- Hãng sản xuất

Viết chương trình thực hiện những công việc sau:

- a) Nhập vào bộ phim mới cùng với các thông tin liên quan đến bộ phim này.
- b) Nhập một thể loại: In ra danh sách các bộ phim thuộc thể loại này.
- c) Nhập một tên nam diễn viên. In ra các bộ phim có diễn viên này đóng.
- d) Nhập tên đạo diễn. In ra danh sách các bộ phim do đạo diễn này dàn dựng.

8.13. Một thư viện cần quản lí thông tin về các đầu sách. Mỗi đầu sách bao gồm các thông tin sau: MaSSach (mã số sách), TenSach (tên sách), TacGia (tác giả), SL (số lượng các cuốn sách của đầu sách). Viết chương trình thực hiện các chức năng sau:

- a) Nhập vào một danh sách các đầu sách (tối đa là 100 đầu sách)
- b) Nhập vào tên của quyển sách. In ra thông tin đầy đủ về các sách có tên đó, nếu không có thì tên của quyển sách đó thì báo là: Không Tìm Thấy.
- c) Tính tổng số sách có trong thư viện.

8.14. Viết chương trình tạo một mảng danh sách các máy tính của một cửa hàng, thông tin của một máy tính bao gồm:

- Loại máy
- Nơi sản xuất
- Thời gian bảo hành

a) Viết hàm nhập một dãy các loại máy tính có thông tin như trên.

b) Hãy viết hàm thống kê xem có bao nhiêu máy có thời gian bảo hành là 1 năm.

c) In ra danh sách các máy tính có xuất xứ từ Mỹ.

8.15. Để lắp ráp một máy vi tính hoàn chỉnh cần phải có tối thiểu 10 linh kiện loại A và có thể lắp bổ sung thêm vào khoảng tối đa 8 linh kiện loại B. Tại một cửa hàng vi tính cần quản lý bán hàng các loại linh kiện tại cửa hàng. Thông tin về một loại linh kiện gồm có: Tên linh kiện, quy cách, loại, đơn giá loại 1 (chất lượng tốt - số nguyên), đơn giá loại 2 (chất lượng thường - số nguyên). Viết chương trình thực hiện những công việc sau:

a) Nhập vào thông tin về các linh kiện có ở cửa hàng.

b) Xuất danh sách các linh kiện đã nhập theo thứ tự tăng dần của loại linh kiện và tên linh kiện.

c) Cho biết đã có đủ 10 linh kiện loại A cần thiết lắp ráp máy hay chưa?

8.16. Một cửa hàng cần quản lý các mặt hàng, thông tin một mặt hàng bao gồm:

- Mã hàng.
- Tên mặt hàng.
- Số lượng.
- Đơn giá.
- Số lượng tồn.
- Thời gian bảo hành (tính theo đơn vị tháng).

a) Hãy nhập vào một danh sách các mặt hàng.

b) Tìm mặt hàng có số lượng tồn nhiều nhất.

c) Tìm mặt hàng có số lượng tồn ít nhất.

d) Tìm mặt hàng có giá tiền cao nhất.

e) In ra những mặt hàng có thời gian bảo hành lớn hơn 12 tháng.

f) Sắp xếp các mặt hàng theo thứ tự tăng dần của số lượng tồn.

8.17. Viết chương trình quản lý hồ sơ nhân viên trong một công ty, chương trình thực hiện những công việc sau:

- Họ và tên.
- Giới tính.
- Ngày sinh.
- Địa chỉ.
- Lương cơ bản.
- Bảo hiểm xã hội.
- Thưởng.

- Phạt.
- Lương thực lĩnh = Lương cơ bản + Thưởng - Bảo hiểm xã hội - Phạt.

a) Nhập vào hồ sơ của các nhân viên trong công ty.

b) Xuất danh sách các nhân viên theo lương thực lĩnh giảm dần bằng 2 cách sau:

i. Cấp phát vùng nhớ tĩnh.

ii. Cấp phát vùng nhớ động.

8.18. (*) Viết chương trình quản lý lớp học của một trường. Các thông tin của một lớp học như sau:

- Tên lớp.
 - Sĩ số.
 - Danh sách các sinh viên trong lớp.
- a) Nhập vào danh sách các lớp với thông tin yêu cầu như trên.
- b) In danh sách các lớp có trên 5 sinh viên có điểm trung bình loại giỏi.
- c) Tìm lớp có nhiều sinh viên nhất.
- d) Tìm lớp có ít sinh viên nhất.
- e) Tìm sinh viên có điểm trung bình cao nhất.
- f) Tìm lớp có số lượng sinh viên đạt điểm trung bình loại giỏi nhiều nhất.

8.19. Viết chương trình quản lý vé tàu, thông tin một vé tàu như sau:

- Ngày giờ khởi hành, ngày giờ đến.
 - Ga đi, ga đến.
 - Loại tàu, loại chỗ ngồi (ngồi, nằm, cứng, mềm).
 - Số toa, số ghế.
- a) Viết hàm nhập vào danh sách các vé tàu.
- b) In danh sách các vé tàu có ga đến là Huế.
- c) In danh sách các vé tàu có ga đến là Hà Nội và đi ngày 8/6/2005.
- d) Đếm xem có bao nhiêu khách đi tàu loại chỗ ngồi là nằm cứng.

8.20. Viết chương trình tính tiền điện hàng tháng của các hộ gia đình, thông tin các khách hàng như sau:

- Kỳ thu, từ ngày... đến ngày...
- Tên khách hàng, mã khách hàng.
- Địa chỉ.
- Điện năng tiêu thụ (Kwh).

- a) Nhập vào danh sách các khách hàng.
- b) Xuất danh sách hoá đơn theo thứ tự tăng dần của điện năng tiêu thụ.
- c) Tính tiền điện của các khách hàng theo quy định sau.
 - i. 100 kw đầu tiên là 550 đ / kw - 50 kw tiếp theo là 900 đ / kw
 - ii. 50 kw tiếp theo là 1210 đ / kw
 - iii. Thuế 10% trên tổng số tiền phải trả
- d) Tính tổng số tiền thu được của các khách hàng.

8.3 TÓM TẮT

Kiểu dữ liệu có cấu trúc cho phép ta định nghĩa những kiểu dữ liệu bất kỳ trên cơ sở là những kiểu dữ liệu cơ bản có sẵn trong ngôn ngữ lập trình. Khi xây dựng xong kiểu dữ liệu mới ta phải định nghĩa những thao tác cho kiểu dữ liệu đó. Những kiểu dữ liệu tự định nghĩa này thông thường có rất nhiều thành phần, mỗi thành phần cũng có thể là một kiểu dữ liệu tự định nghĩa, vấn đề là ta chọn kiểu dữ liệu cơ bản nào để xây dựng nên chúng sao cho phù hợp về mặt kiểu dữ liệu và phù hợp về kích thước lưu trữ (vừa đủ). Cách sử dụng những kiểu dữ liệu tự định nghĩa cũng giống như các kiểu dữ liệu cơ bản. Muốn sử dụng phải khai báo biến, khi truy cập các thành phần phải truy cập theo quy ước. Nếu thành phần cấu trúc có kiểu dữ liệu là số thực thì khi sử dụng hàm scanf() phải thông qua biến trung gian rồi gán lại cho thành phần cấu trúc đó. Đối với mảng các kiểu dữ liệu có cấu trúc ta nên xử lý cho từng thành phần cấu trúc rồi mới xử lý cho mảng cấu trúc bằng cách dùng vòng lặp.

KIỂU DỮ LIỆU TẬP TIN

Chương này sẽ trình bày cấu trúc tập tin, cài đặt các thao tác, một số hàm thư viện và ứng dụng trong việc tổ chức dữ liệu trên tập tin.

9.1 LÝ THUYẾT

9.1.1 Khái niệm tập tin

Trong các chương trình trước thì các dữ liệu đưa vào chương trình chỉ được tồn tại trong RAM, khi thoát chương trình thì tất cả dữ liệu đều bị mất. Để khắc phục tình trạng này C/C++ cung cấp cho ta các hàm thư viện để lưu trữ và truy xuất tập tin, đó là kiểu FILE. Và ở đây ta chỉ đề cập đến 2 loại tập tin:

- Tập tin văn bản: là tập tin lưu trữ dưới dạng ký tự.
- Tập tin nhị phân: là tập tin dùng lưu trữ dưới dạng nhị phân.

9.1.2 Thao tác với tập tin

Quá trình thao tác trên tập tin thông qua 4 bước:

- Khai báo con trỏ cho tập tin.
- Mở tập tin.
- Xử lý tập tin.
- Đóng tập tin.

Khai báo

Sử dụng

```
FILE *<tên biến>;
```

Khai báo biến con trỏ file `f`

```
FILE *f;
```

Mở tập tin**Sử dụng**

```
fopen(<đường dẫn tên tập tin>, <kiểu truy nhập>);
```

Các kiểu truy nhập tập tin thông dụng:

- `t` là kiểu truy nhập tập tin đối với dạng tập tin văn bản.
- `b` là kiểu truy nhập tập tin đối với dạng tập tin nhị phân.
- `r` mở ra để đọc (ready only).
- `w` mở ra để ghi (create / write).
- `a` mở ra để thêm vào (append).
- `r+` mở ra để đọc và ghi (modify).

Mở tập tin văn bản tên DATA.TXT để đọc

```
// Khai bao bien con tro f
FILE *f;
// Mo tap tin van ban va f quan ly tap tin nay
f = fopen("C:\\DATA.TXT", "rt");
```

Đọc tập tin nhị phân

```
fread(&ptra, size, len, FILE *);
```

Ghi tập tin nhị phân

```
fwrite(&ptra, size, len, FILE *);
```

Đóng tập tin

Sau khi không còn làm việc với tập tin, để đảm bảo an toàn cho dữ liệu thì nhất thiết ta phải đóng tập tin lại.

- Đóng một tập tin

```
fclose(<biến con trỏ tập tin>);
```

- Đóng tất cả tập tin

```
fcloseall();
```

Ví dụ 9.1

Đóng tập tin f và tập tin g

```
fclose(f);
```

```
fclose(g);
```

Xoá tập tin

```
remove(<đường dẫn + tên tập tin>);
```

Đổi tên tập tin

```
rename(<tên tập tin cũ>, <tên tập tin mới>);
```

Di chuyển con trỏ tập tin:

```
fseek(FILE *, <độ dời>, <mốc>);
```

mốc:

- SEEK_SET dời đến đầu tập tin (giá trị 0).
- SEEK_CUR dời vị trí hiện hành (giá trị 1).
- SEEK_END dời đến cuối tập tin (giá trị 2).

Ví dụ 9.2

Di chuyển vị trí con trỏ về sau 5 bytes, sau đó di chuyển về trước 4 bytes

```
fseek(f, +5, SEEK_CUR);
```

```
fseek(f, -4, SEEK_CUR);
```

Cho biết vị trí con trỏ file

```
ftell(FILE *);
```

9.1.3 Các ví dụ minh họa**Ví dụ 9.3**

Kiểm tra tập tin một tập tin có thể mở được không?

```
void KiemTra(char *tenfile)
{
    FILE *f = fopen(tenfile, "rt");
    if(f == NULL)
    {
        printf("Khong mo duoc tap tin %s", tenfile);
        return;
    }
    printf("Tap tin %s da duoc mo", tenfile);
    fclose(f);
}
```

Tập tin văn bản**Ví dụ 9.4**

Viết chương trình phát sinh ngẫu nhiên ma trận a kích thước 5×6 , lưu ma trận này vào file data.txt. Đọc lại file data.txt đưa dữ liệu vào ma trận b và xuất ra màn hình xem kết quả lưu đúng không? Cấu trúc của file data.txt như sau:

- Dòng đầu lưu 2 số nguyên: m, n thể hiện số dòng và số cột của ma trận.
- m dòng tiếp theo, mỗi dòng gồm n phần tử là giá trị các phần tử trên một dòng của ma trận.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
```



```

#define MAX 100

void LuuFile(char *tenfile, int a[MAX][MAX], int m, int n)
{
    FILE *f;
    f=fopen(tenfile, "wt");
    if(f==NULL)
    {
        printf("\nKhong tao duoc file.");
        getch();
        exit(0);
    }
    fprintf(f, "%d %d\n", m, n);
    for(int i=0; i<m; i++)
    {
        for(int j=0; j<n; j++)
            fprintf(f, "%d\t", a[i][j]);
        fprintf(f, "\n");
    }
    fclose(f);
}

void DocFile(char *tenfile, int a[MAX][MAX], int &m, int &n)
{
    FILE *f;
    f=fopen(tenfile, "rt");
    if(f==NULL)
    {
        printf("\nKhong doc duoc file.");
        getch();
        exit(0);
    }
    fscanf(f, "%d%d", &m, &n);
    for(int i=0; i<m; i++)
    {

```

```

        for(int j=0; j<n; j++)
            fscanf(f, "%d", &a[i][j]);
    }
    fclose(f);
}

void main()
{
    int i, j;
    int a[MAX][MAX], m=5, n=6;
    int b[MAX][MAX], x, y;
    randomize();
    for(i=0; i<m; i++)
        for(j=0; j<n; j++)
            a[i][j]=random(1000);
    LuuFile("data.txt", a, m, n);
    DocFile("data.txt", b, x, y);
    for(i=0; i<x; i++)
    {
        for(j=0; j<y; j++)
            printf("%d\t", b[i][j]);
        printf("\n");
    }
}

```

Ví dụ 9.5

Viết chương trình tạo tập tin văn bản OUT.TXT gồm n số nguyên, các số của dãy được tạo ngẫu nhiên có giá trị tuyệt đối không vượt quá M (n, M đọc từ tập tin IN.TXT). Kết quả chương trình là 1 tập tin văn bản có dòng thứ nhất ghi số n ; n dòng tiếp theo ghi các số tạo được, mỗi số trên một dòng.

```

void NhapFile(char *tenfile, int &n, int &M)
{

```

```

    FILE *fi;
    fi = fopen(tenfile, "rt");
    fscanf(fi, "%d%d", &n, &M);
    fclose(fi);
}

void XuatFile(char *tenfile, int n, int M)
{
    FILE *fo;
    fo = fopen(tenfile, "wt");
    fprintf(fo, " %d\n", n);
    randomize ();
    for(; n > 0; n -)
        fprintf(fo, "%d\n", random((2 * M + 1) - M));
    fclose(fo);
}

void main()
{
    int n, M;
    Nhap("IN.TXT", n, M);
    Xuat("OUT.TXT", n, M);
}

```

Tập tin nhị phân

Ví dụ 9.6

Viết hàm đọc/ghi một danh sách sinh viên của một lớp vào tập tin

```

void Doc(char *tenfile, SINHVIEN ds[], int &siso)
{
    FILE *fi;
    fi = fopen("SV.DAT", "rb");
    fseek(fi, 0, SEEK_END);
    siso = (ftell(fi) + 1) / sizeof(SINHVIEN);
}

```

```

    fseek(fi, 0, SEEK_SET);
    fread(ds, sizeof(SINHVIEN), siso, fi);
    fclose(fi);
}

void Ghi(char *tenfile, SINHVIEN ds[], int siso)
{
    FILE *fo;
    fo = fopen(tenfile, "wb");
    fwrite(ds, sizeof(SINHVIEN), siso, fo);
    fclose(fo);
}

```

9.2 BÀI TẬP

Bài tập cơ bản

- 9.1. Viết chương trình tạo tập tin văn bản chứa 1 dãy số nguyên bất kỳ.
- 9.2. Viết chương trình tạo tập tin nhị phân chứa 10000 số nguyên bất kỳ ghi vào file SONGUYEN.INP. Mỗi dòng 10 số, sau đó viết chương trình đọc file SONGUYEN.INP, sắp xếp theo thứ tự tăng dần và lưu kết quả vào file SONGUYEN.OUT.
- 9.3. Viết chương trình tạo một file chứa 10000 số nguyên ngẫu nhiên đôi một khác nhau trong phạm vi từ 1 đến 32767 và đặt tên là "SONGUYEN.INP".
- 9.4. Viết chương trình tạo một file chứa các số nguyên có tên SONGUYEN.INP. Sau đó đọc file SONGUYEN.INP và ghi các số chẵn vào file SOCHAN.OUT và những số lẻ vào file SOLE.OUT.
- 9.5. Viết chương trình ghi vào tập tin SOCHAN.DAT các số nguyên chẵn từ 0 đến 100.
- 9.6. Viết chương trình đọc tập tin SOCHAN.DAT và xuất ra màn hình, mỗi dòng 30 số.
- 9.7. Viết chương trình giả lập lệnh COPY CON để tạo tập tin văn bản. Khi kết thúc tập tin nhấn phím F6 để lưu
- 9.8. Viết chương trình giả lập lệnh TYPE để in nội dung của tập tin văn bản ra màn hình.
- 9.9. Viết chương trình kiểm tra một tập tin nào đó có trong một thư mục được chỉ định hay không?
- 9.10. Viết chương trình giả lập lệnh DEL để xóa tập tin. Yêu cầu nhập đường dẫn và tên tập

tin, kiểm tra sự tồn tại của tập tin, nếu có thì xoá tập tin được chỉ định.

9.11. Viết chương trình giả lập lệnh RENAME để đổi tên một tập tin.

9.12. Viết chương trình tạo file văn bản có tên là "MATRIX.INP" có cấu trúc như sau:

a) Dòng đầu ghi hai số m, n .

b) Trong m dòng tiếp theo mỗi dòng ghi n số và các số cách nhau một khoảng cách.

Hãy kiểm tra xem trong file đó có bao nhiêu số nguyên tố. Kết quả cần ghi vào file "MATRIX.OUT" có nội dung là một số nguyên đó là số lượng các số nguyên tố trong file "MATRIX.INP".

9.13. Cho số nguyên n , hãy in tam giác PASCAL gồm n dòng

- Dữ liệu vào: tập tin văn bản PAS.INP gồm 1 dòng chứa giá trị n .
- Kết quả: đưa ra tập tin văn bản PAS.OUT thể hiện một tam giác PASCAL n dòng.

9.14. Cho mảng các số nguyên, hãy sắp xếp mảng theo thứ tự tăng dần.

Dữ liệu vào: tập tin văn bản ARRAY.INP gồm 2 dòng

a) Dòng 1 chứa số nguyên n ($n \leq 100$).

b) Dòng 2 chứa n số nguyên.

Kết quả: Đưa ra tập tin văn bản ARRAY.OUT gồm hai dòng

- a) Dòng 1 chứa n phần tử của mảng các số nguyên.
- b) Dòng 2 chứa n số nguyên được xếp tăng dần.

9.15. Cho mảng các số nguyên, tìm phần tử lớn nhất của mảng.

Dữ liệu vào: tập tin văn bản ARRAY.INP gồm hai dòng:

- a) Dòng 1 chứa số nguyên n ($n \leq 100$).
- b) Dòng 2 chứa n số nguyên.

Kết quả: Đưa ra tập tin văn bản ARRAY.OUT gồm 1 dòng ghi 2 giá trị x, y trong đó x là giá trị lớn nhất, y là vị trí của x trong mảng.

Luyện tập và nâng cao

9.16. Cho mảng các số nguyên, tính tổng các phần tử của mảng.

Dữ liệu vào: tập tin văn bản ARRAY.INP gồm hai dòng

- a) Dòng 1 chứa số nguyên n ($n \leq 10$)
- b) Dòng 2 chứa n số nguyên

Kết quả: Đưa ra tập tin văn bản ARRAY.OUT gồm một dòng ghi tổng các phần tử trong mảng.

9.17. Cho mảng các số nguyên, hãy liệt kê các phần tử là số nguyên tố

Dữ liệu vào: tập tin văn bản NT.INP gồm hai dòng

- a) Dòng 1 chứa số nguyên n ($n \leq 100$)
- b) Dòng 2 chứa n số nguyên

Kết quả: đưa ra tập tin văn bản NT.OUT gồm hai dòng:

- a) Dòng 1 chứa số lượng các phần tử nguyên tố trong mảng.
- b) Dòng 2 liệt kê các số nguyên tố đó.

9.18. (*) Tạo file văn bản có tên là "INPUT.TXT" có cấu trúc như sau:

- Dòng đầu tiên ghi N (N là số nguyên dương nhập từ bàn phím).
- Trong các dòng tiếp theo ghi N số nguyên ngẫu nhiên trong phạm vi từ 0 đến 100, mỗi dòng 10 số (các số cách nhau ít nhất một khoảng trắng).

Hãy đọc dữ liệu của file "INPUT.TXT" và lưu vào mảng một chiều A . Thực hiện các công việc sau:

- a) Tìm giá trị lớn nhất của mảng A .
- b) Đếm số lượng số chẵn, số lượng số lẻ của mảng A .
- c) Hãy sắp xếp các phần tử theo thứ tự tăng dần n .

Hãy ghi các kết quả vào file văn bản có tên OUTPUT.TXT

9.19. (*) Viết chương trình nhập và lưu hồ sơ của sinh viên vào một file có tên là "DSSV.TXT".

Sau đó đọc file "DSSV.TXT" và cất vào mảng, hãy sắp xếp các hồ sơ sinh viên theo thứ tự giảm dần theo điểm trung bình môn học rồi in ra màn hình hồ sơ các sinh viên theo thứ tự đó ra màn hình có thông tin như sau:

- a) Mã số sinh viên.
- b) Họ và tên sinh viên.
- c) Điểm trung bình kiểm tra.
- d) Điểm thi hết môn.
- e) Điểm trung bình môn học tính bằng (Điểm trung bình kiểm tra + điểm thi hết môn)/2.

9.20. (*) Tạo một file text có tên là "INPUT.TXT" có cấu trúc như sau:

- Dòng đầu tiên ghi hai số M và N (M, N là hai số nguyên dương nhập từ bàn phím).
- Trong M dòng tiếp theo mỗi dòng ghi N số nguyên ngẫu nhiên trong phạm vi từ 0 đến 100 (các số này cách nhau ít nhất một khoảng trắng).

Hãy đọc dữ liệu từ file trên và lưu vào mảng hai chiều. Rồi thực hiện các công việc sau:

- a) Tìm giá trị lớn nhất của ma trận.
- b) Đếm số lượng số chẵn, lẻ, nguyên tố có trong ma trận.
- c) Hãy tính tổng các phần tử trên mỗi dòng của ma trận.

Hãy ghi kết quả này vào file văn bản có tên là "OUTPUT.TXT"

- 9.21. (*)** Xét dãy số $\{a_1, a_2, \dots, a_N\}$. Một đoạn con của dãy là dãy các phần tử liên tiếp nhau được xác định bởi chỉ số của số bắt đầu (L) và chỉ số của số cuối cùng (R). Tổng các số trên đoạn được gọi là tổng đoạn. Yêu cầu: Cho dãy $\{a_1, a_2, \dots, a_N\}$, hãy tìm đoạn con có tổng đoạn lớn nhất (T).

Dữ liệu được cho trong tập tin văn bản SUMMAX.INP

- a) Dòng thứ nhất chứa số nguyên N ($0 < N \leq 30000$)
 - b) N dòng tiếp theo, mỗi dòng chứa một số là các số của dãy đã cho theo đúng thứ tự.
- Giá trị tuyệt đối của mỗi số không vượt quá 30000 Kết quả tìm được ghi vào tập tin văn bản SUMMAX.OUT

- 9.22. (*)** Cho dãy $\{a_1, a_2, \dots, a_N\}$, hãy tìm đoạn con tăng dần có tổng lớn nhất.

Dữ liệu: được cho trong tập tin AMAX.INP

- a) Dòng 1 chứa số nguyên N ($0 < N \leq 30000$).
 - b) N dòng tiếp theo, mỗi dòng chứa một số là các số của dãy đã cho theo đúng thứ tự.
- Giá trị tuyệt đối của mỗi số không vượt quá 30000.

Kết quả tìm được ghi vào tập tin văn bản AMAX.OUT gồm hai dòng:

- a) Dòng 1 ghi tổng của dãy con.
- b) Dòng 2 ghi mảng con tăng dần có tổng lớn nhất.

- 9.23.** Viết chương trình nhập lý lịch một nhân viên vào danh sách các nhân viên. Khi không nhập nữa bấm phím ESC và ghi vào tập tin NHANVIEN.DAT sau đó:

- a) Đọc từ tập tin NHANVIEN.DAT vừa tạo và in danh sách các nhân viên lên màn hình.
- b) Tìm và in lý lịch một nhân viên bằng các nhập và họ tên hoặc mã số nhân viên.

- 9.24. (**)** Để lắp ráp một máy vi tính hoàn chỉnh cần phải có tối thiểu 10 linh kiện loại A và có thể lắp bổ sung thêm vào khoảng tối đa 8 linh kiện loại B. Tại một cửa hàng vi tính cần quản lý bán hàng các loại linh kiện tại cửa hàng. Thông tin về một loại linh kiện gồm có: Tên linh kiện, quy cách, loại, đơn giá loại 1 (chất lượng tốt - số nguyên), đơn giá loại 2 (chất lượng thường - số nguyên). Viết chương trình thực hiện những công việc sau:

- a) Nhập vào thông tin của các loại linh kiện có ở cửa hàng. Xuất danh sách các linh

kiện đã nhập theo thứ tự tăng dần của loại linh kiện và tên linh kiện. Cho biết đã có đủ 10 linh kiện loại A cần thiết để lắp ráp máy tính hay chưa?

- b) Với giả định là cửa hàng đã có đủ 10 linh kiện loại A để lắp ráp máy. Nhập vào một số tiền để lắp ráp một máy tính. Có thể lắp được một máy tính hoàn chỉnh với các linh kiện toàn bộ theo đơn giá loại 1 hay đơn giá loại 2 hay không? Nếu số tiền trong khoảng giữa thì hãy tìm một phương án gồm những linh kiện theo đơn giá 1 và linh kiện theo đơn giá 2 để lắp?
- c) Tất cả dữ liệu phải lưu ở tập tin.

9.3 TÓM TẮT

Mục đích của kiểu dữ liệu tập tin cho phép chúng ta lưu lại những thông tin cần thiết tương đối lớn: những dữ liệu đầu vào, những kết quả của chương trình hoặc những dữ liệu dùng để kiểm tra chương trình. Khi thao tác trên tập tin phải thông qua 4 bước: Khai báo con trỏ trỏ đến tập tin, Mở tập tin, Xử lý trên tập tin và cuối cùng là Đóng tập tin. Lưu ý khi mở tập tin để ghi thì phải cẩn thận với thao tác tạo mới hay chỉnh sửa nội dung tập tin, di chuyển con trỏ hợp lý để tránh mất thông tin. Sử dụng hàm thao tác trên tập tin phải dùng đúng loại hàm cho tập tin kiểu nhị phân hay kiểu văn bản.

10

KIỂU DỮ LIỆU CON TRỎ

Chương này cung cấp một kiểu dữ liệu mới dùng để quản lý địa chỉ của các biến.

10.1 LÝ THUYẾT

Khái niệm

- Bộ nhớ máy tính bao gồm các ô nhớ, mỗi ô nhớ có kích thước 1 byte
- Mỗi ô nhớ có địa chỉ duy nhất là một số nguyên không âm
- Mỗi biến sẽ được gán với một số ô nhớ liên tục
- Địa chỉ của biến là địa chỉ của ô nhớ đầu tiên
- Địa chỉ NULL là địa chỉ 0

Định nghĩa 10.1. Biến lưu địa chỉ là biến con trỏ

Khai báo con trỏ

Cú pháp

<kiểu dữ liệu> *<tên biến con trỏ>

Khai báo biến con trỏ và gán địa chỉ

```
char   *p1;  
int     *p2;  
float   *p3;  
p1 = 0x0010;  
p2 = 0x0030;  
p3 = NULL;
```

Toán tử lấy địa chỉ**Cú pháp**

`&<tên biến>`

Lấy địa chỉ của một biến và lưu lại địa chỉ này vào biến con trỏ

```
char c;
int n;
char *p1;
int *p2;
p1 = &c;
p2 = &n;
```

Toán tử truy xuất đến ô nhớ con trỏ quản lý**Cú pháp**

`*<tên biến con trỏ>`

Truy xuất đến biến do biến con trỏ quản lý

```
int n;
int *p;
p = &n;
*p = 5;
```

Thao tác với biến con trỏ

```
int a;
int *pa;
a = 5;
pa = &a;
printf("%d\n", a);
printf("%d\n", pa);
printf("%d\n", *pa);
printf("%d\n", &pa);
```

Con trỏ nhiều cấp

Biến con trỏ cũng có địa chỉ

Định nghĩa 10.2.

- Biến con trỏ cấp 1 là con trỏ quản lý địa chỉ của biến thông thường
- Biến con trỏ cấp k quản lý địa chỉ của con trỏ cấp $k - 1$

Sử dụng

Con trỏ cấp 1 p

```
int *p;
```

Con trỏ cấp 2 q

```
int **q;
```

Con trỏ cấp 3 r

```
int ***r;
```

Ví dụ 10.1

Thao tác với biến con trỏ nhiều cấp

```
int a;
int *p;
int **q;
a = 12;
p = &a;
q = &p;
```

Con trỏ và tham số**Ví dụ 10.2**

Hàm hoanvi sử dụng hai tham số x, y là con trỏ

```
void hoanvi(int *x, int *y)
{
```

```

    int t = *x;
    *x = *y;
    *y = t;
}

void main()
{
    int a = 4, b = 7;
    hoanvi(&a, &b);
    printf("a = %d, b = %d", a, b);
}

```

Diễn giải. hai biến a,b sẽ hoán đổi giá trị sau khi hàm hoanvi thực hiện

Toán tử cộng số nguyên

Cú pháp

<con trỏ> + <số nguyên>

Được hiểu là thực hiện phép tính số học <địa chỉ>+<số nguyên>*<kích thước kiểu dữ liệu> và kết quả trả về cũng là <địa chỉ>

Sử dụng

```

int a[200];
int *p,*q;
p = a;
q = &a[100];
p = p+2;
q = q+5;

```

p sẽ chứa địa chỉ của phần tử a[2], q sẽ chứa địa chỉ của phần tử a[105]

Toán tử trừ số nguyên

Cú pháp

<con trỏ> - <số nguyên>

Được hiểu là thực hiện phép tính số học <địa chỉ> - <số nguyên>*<kích thước kiểu dữ liệu> và kết quả trả về cũng là <địa chỉ>

Sử dụng

```
int a[200];
int *p,*q;
p = &a[50];
q = &a[100];
p = p-2;
q = q-6;

p sẽ chứa địa chỉ của phần tử a[48], q sẽ chứa địa chỉ của phần tử a[94]
```

Toán tử trừ con trỏ

Cú pháp

<con trỏ 1> - <con trỏ 2>

<con trỏ 1> và <con trỏ 2> phải cùng kiểu giá trị trả về là một số nguyên cho biết sự chênh lệch địa chỉ giữa hai con trỏ

Sử dụng

```
int a[100];
int *p,*q;
p = &a[50];
q = &a[100];
d = q - p;

Giá trị của d = 50*4 = 200
```

Con trỏ và mảng 1 chiều tĩnh

Tên của mảng chính là địa chỉ của phần tử đầu tiên, các phần tử còn lại của mảng được sắp liên tục và tăng dần

Sử dụng

Khai báo biến mảng a

```
int a[100];
```

Khai báo biến con trỏ p, q

```
int *p,*q;
```

Gán địa chỉ cho p, q

```
p = a;
```

```
q = &a[0];
```

p và q sẽ có cùng giá trị địa chỉ

Con trỏ và cấu trúc**Cú pháp**

Truy xuất thành phần của biến cấu trúc qua biến con trỏ

```
(*<biến con trỏ>).<tên thành phần>
```

```
<biến con trỏ>-><tên thành phần>
```

Ví dụ 10.3

```
typedef struct {
    int tu, mau;
} PhanSo;
PhanSo a;
PhanSo *p;
p = &a;
p->tu = 3;
(*p).mau = 4;
```

Con trỏ hàm

Lưu ý.

- Con trỏ hàm dùng để quản lý địa chỉ của hàm

- Có thể gọi hàm thực thi thông qua con trỏ hàm

Cú pháp

Khai báo con trỏ hàm

<kiểu trả về> (*<biến con trỏ hàm>)(<danh sách tham số>)

```
int (*pf1)(int);
```

```
int (*pf2)(double, double);
```

pf1 quản lý địa chỉ của hàm có một tham số int và kiểu trả về int, pf2 quản lý địa chỉ của hàm có hai tham số double và kiểu trả về int

Gọi thực thi con trỏ hàm

Cú pháp

<biến con trỏ hàm>(<danh sách tham số>)

Các thao tác trên con trỏ hàm

```
int Cong(int a, int b)
```

```
{
```

```
    return a+b;
```

```
}
```

```
void main()
```

```
{
```

```
    int s;
```

```
    int (*f)(int, int);
```

```
    f = Cong;
```

```
    s = f(5, 3);
```

```
}
```

Cấp phát bộ nhớ trong C

Cú pháp

Sử dụng các hàm trong thư viện malloc.h

- Xin cấp phát vùng nhớ

`<địa chỉ> malloc(<kích thước vùng nhớ>)`

`<kích thước vùng nhớ>` được tính theo đơn vị byte

```
int *p = (int *)malloc(8);
```

```
PhanSo *q = (PhanSo *)malloc(2*sizeof(PhanSo));
```

- Xin cấp phát vùng nhớ theo khối

`<địa chỉ> calloc(<số khối>, <kích thước 1 khối>)`

```
int *p = (int *)calloc(2, sizeof(int));
```

```
PhanSo *q = (PhanSo *)calloc(2, sizeof(PhanSo));
```

Thu hồi bộ nhớ đã cấp phát trong C

Cú pháp

Ngôn ngữ C/C++ không tự động thu hồi vùng nhớ được cấp phát động. Người lập trình phải làm công việc thu hồi bộ nhớ động đã cấp phát.

Sử dụng hàm trong thư viện malloc

`free(<biến con trỏ>)`

```
float *p = (float *)calloc(2, sizeof(float));
```

```
free(p);
```

Cấp phát bộ nhớ trong C++

C++ bổ sung thêm các toán tử quản lý bộ nhớ

Cú pháp

`new <kiểu dữ liệu>`

`new <kiểu dữ liệu>[<số phần tử>]`

Biến `q` chứa địa chỉ của một vùng nhớ cho một số nguyên `int`, biến `q` chứa địa chỉ của một vùng nhớ cho một dãy 10 số nguyên `int`

```
int *p = new int;
```



```
int *q = new int[10];
```

Thu hồi bộ nhớ đã cấp phát trong C++

Cú pháp

```
delete <biến con trỏ>
```

```
delete []<biến con trỏ>
```

Thu hồi vùng nhớ đã cấp phát đang được p, q quản lý

```
int *p = new int;
```

```
int *q = new int[10];
```

```
delete p;
```

```
delete [] q;
```

10.2 BÀI TẬP

10.3 TÓM TẮT

11

KỸ THUẬT ĐỆ QUY

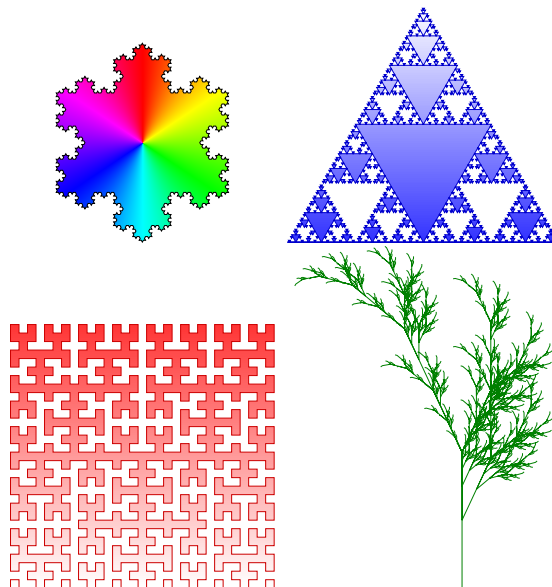
Chương này giới thiệu phương pháp lập trình theo kỹ thuật đệ quy, phân loại, cách hoạt động và cách cài đặt các hàm đệ quy.

11.1 LÝ THUYẾT

Định nghĩa 11.1.

- Một khái niệm là khái niệm đệ quy nếu trong định nghĩa của nó có sử dụng lại chính nó
- Một hàm là hàm đệ quy nếu trong thân của hàm đó có những lệnh gọi lại chính nó một cách tường minh hay tiềm ẩn.

Ví dụ. Những hình tự đồng dạng



11.1.1 Các loại đệ qui

Đệ qui tuyến tính

Sử dụng

Trong mỗi lần thực hiện, hàm *gọi lại chính nó không quá một lần*

```
<Kiểu dữ liệu hàm> <Tên hàm>(<danh sách tham số>)
{
    if(<điều kiện dừng>)
    {
        ...
    }
    else
    {
        ...
        <Tên hàm>(<danh sách tham số>);
        ...
    }
}
```

Ví dụ 11.1

Tính tổng $S_n = 1 + 2 + \dots + n$

Phân tích bài toán

- Phần dừng: $S_0 = 0$
- Phần đệ qui: $S_n = S_{n-1} + n$ với $n > 0$

```
int TinhTong(int n)
{
    if(n==0)
        return 0;
    else
        return (TinhTong(n-1) + n);
}
```

Ví dụ 11.2

Tính $P_n = n!$

Phân tích bài toán

- Phần dừng: $P_0 = 1$
- Phần đệ qui: $P_n = P_{n-1} \cdot n$ với $n > 0$

```
int TinhGiaiThua(int n)
{
    if(n==0)
        return 1;
    else
        return (TinhGiaiThua(n-1) * n);
}
```

Đệ qui nhị phân**Sử dụng**

Trong mỗi lần thực hiện, hàm *gọi lại chính nó không quá hai lần*

```
<Kiểu dữ liệu hàm> <Tên hàm>(<danh sách tham số>)
{
    if(<điều kiện dừng>)
    {
        ...
    }
    else
    {
        ...
        <Tên hàm>(<danh sách tham số>);
        ...
        <Tên hàm>(<danh sách tham số>);
        ...
    }
}
```

}

Ví dụ 11.3

Tính số f_n của dãy Fibonacci $\{1, 1, 2, 3, 5, \dots\}$

- Phần dừng: $f_0 = 1$ và $f_1 = 1$.
- Phần đệ qui: $f_n = f_{n-1} + f_{n-2}$ với $n > 1$

```
int Fibo(int n)
{
    if(n==0 || n==1)
        return 1;
    else
        return Fibo(n-1) + Fibo(n-2);
}
```

Ví dụ 11.4

Bài toán tháp Hà Nội

- Phần dừng: $n = 1$ thì $A \rightarrow C$
- Phần đệ qui:
 - Bước 1: Di chuyển $n - 1$ đĩa trên cùng từ cọc A sang cọc B .
 - Bước 2: $A \rightarrow C$.
 - Bước 3: Di chuyển $n - 1$ đĩa trên cùng từ cọc B sang cọc C .

```
void ThapHaNoi(int n, char A, char B, char C)
{
    if(n==1)
        printf("Di chuyen dia tren cung tu %d den %d\n", A, C);
    else
    {
        ThapHaNoi(n-1, A, C, B);
        printf("Di chuyen dia tren cung tu %d den %d\n", A, C);
        ThapHaNoi(n-1, B, A, C);
    }
}
```

```
}

```

Đệ qui phi tuyến

Sử dụng

Trong mỗi lần thực hiện hàm, có thể gọi lại chính nó hơn hai lần

```
<Kiểu dữ liệu hàm> <Tên hàm>(<danh sách tham số>)
{
    if(<điều kiện dừng>)
    {
        ...
    }
    else
    for (int i = 1; i<=n; i++)
    {
        ...
        <Tên hàm>(<danh sách tham số>);
        ...
    }
}
```

Ví dụ 11.5

Tính số hạng thứ n của dãy x_n được định nghĩa như sau:

$$x_0 = 1$$

$$x_n = n^2 x_0 + (n-1)^2 x_1 + \dots + 1^2 x_n$$

- Phần dừng: dựa trên định nghĩa
- Phần đệ qui: dựa trên định nghĩa

```
int TinhXn(int n)
{
    if(n==0) return 1;
```

```

int s = 0;
for (int i=1; i<=n; i++)
    s = s + i * i * TinhXn(n-i);
return s;
}

```

Đệ qui hỗ tương

Sử dụng

Hàm thứ nhất và hàm thứ hai gọi qua lại lẫn nhau.

```

<Kiểu dữ liệu hàm> <Tên hàm 1>(<danh sách tham số>);
<Kiểu dữ liệu hàm> <Tên hàm 2>(<danh sách tham số>);
...
<Kiểu dữ liệu hàm> <Tên hàm 1>(<danh sách tham số>)
{
    ...
    <Tên hàm 2>(<danh sách tham số>);
    ...
}
<Kiểu dữ liệu hàm> <Tên hàm 2>(<danh sách tham số>)
{
    ...
    <Tên hàm 1>(<danh sách tham số>);
    ...
}

```

Ví dụ 11.6

Tính số hạng thứ n của hai dãy x_n và y_n được định nghĩa như sau:

$$x_0 = 1$$

$$y_0 = 1$$

$$x_n = x_{n-1} + y_{n-1}$$

$$y_n = n^2 x_{n-1} + y_{n-1}$$

- Phần dừng: dựa trên định nghĩa
- Phần đệ qui: dựa trên định nghĩa

```

int TinhXn(int n);
int TinhYn(int n);
int TinhXn(int n)
{
    if(n==0) return 1;
    return TinhXn(n-1) + TinhYn(n-1);
}
int TinhYn(int n)
{
    if(n==0) return 1;
    return n*n*TinhXn(n-1) + TinhYn(n-1);
}

```

11.1.2 Các ví dụ

Ví dụ 11.7

Viết hàm tính lũy thừa x^n bằng kỹ thuật đệ qui

```

float TinhLuyThua(float x, int n)
{
    if(n==0) return 1;
    else return x*TinhLuyThua(x,n-1);
}

```

Ví dụ 11.8

Viết hàm tính tổng các phần tử có giá trị chẵn của một dãy số kỹ thuật đệ qui.

```
int TongChan(int a[], int n)
{
    if(n==0) return 0;
    int s = TongChan(a, n-1);
    if(a[n-1]%2==0) s+=a[n-1];
    return s;
}
```

Ví dụ 11.9

Cho dãy số nguyên a gồm n phần tử có thứ tự tăng dần. Tìm phần tử có giá trị x có xuất hiện trong mảng không bằng kỹ thuật đệ qui.

```
int TimNhiPhan(int a[], int l, int r, int x)
{
    int m = (l+r)/2;
    if(l>r) return -1; // Không có phần tử x
    if(a[m]==x)
        return m; // Tra về vị trí tìm thấy
    if(a[m]>x)
        return TimNhiPhan(a, l, m-1, x);
    if(a[m]<x)
        return TimNhiPhan(a, m+1, r, x);
}
```

11.2 BÀI TẬP

Bài tập cơ bản

- 11.1. Cài đặt bằng đệ qui lại những bài tập ở chương mảng một chiều.
- 11.2. Tìm chữ số có giá trị lớn nhất của số nguyên dương n .
- 11.3. Hãy xây dựng một dãy gồm N số có giá trị từ 1 đến K cho trước, sao cho không có hai

dãy con liên tiếp đứng kề nhau. Ví dụ, $N = 6$ và $K = 3$ thì kết quả: 121312

11.4. Tìm ước số chung lớn nhất của hai số nguyên dương a và b .

11.5. Tìm chữ số đầu tiên của số nguyên dương n .

11.6. Tìm dãy nhị phân dài nhất sao cho trên dãy này không có hai bộ k bất kỳ trùng nhau.

Bộ k là dãy con có k số liên tiếp nhau trên dãy tìm được. Ví dụ, $k = 3$ thì kết quả là 0001011100

11.7. Tính $P(n) = 1.3.5...(2n+1)$ với $n > 0$

11.8. Tính $P(n) = 1 + 3 + 5 + ... + (2n+1)$ với $n > 0$

11.9. Tính $P(n) = 1 - 2 + 3 - ... + (-1)^{n+1}n$ với $n > 0$

11.10. Tính $P(n) = 1 + 1.2 + 1.2.3 + ... + 1.2.3...n$ với $n > 0$

11.11. Tính $P(n) = 1 + (1+2) + (1+2+3) + ... + (1+2+3+...+n)$ với $n > 0$

11.12. Tính $P(n) = 1^2 + 2^2 + 3^2 + ... + n^2$ với $n > 0$

11.13. Tính

$$P(n) = 1 + \frac{1}{2} + \frac{1}{3} + ... \frac{1}{n}$$

với $n > 0$

11.14. Tính

$$P(n) = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + ... + \frac{1}{1+2+3+...+n}$$

với $n > 0$

11.15. Tính $P(x, n) = x^n$ với n là số nguyên không âm

Luyện tập và nâng cao

11.16. Cho số nguyên dương n . In ra biểu diễn nhị phân của n .

11.17. (*) Tính

$$S(n) = \sqrt{n + \sqrt{n-1 + \sqrt{n-2 + ... + \sqrt{1}}}}$$

với $n > 0$

11.18. (*) Tính

$$S(n) = \sqrt{1 + \sqrt{2 + \sqrt{3 + ... + \sqrt{n}}}}$$

với $n > 0$

11.19. (*) Tính

$$\frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \dots + \frac{1}{1 + \frac{1}{1}}}}}$$

với $n > 0$

11.20. (**) Cài đặt bài toán mã đi tuần.

11.21. (**) Cài đặt bài toán tám hậu.

11.22. (*) Cài đặt hàm tính $x.y$ với x, y là các số nguyên không âm mà chỉ được dùng các tính cộng và shift left và shift right thông qua công thức sau

$$x.y = \begin{cases} 0 & \text{nếu } x = 0 \\ (x \gg 1) \cdot (y \ll 1) & \text{nếu } x \text{ là số chẵn} \\ (x \gg 1) \cdot (y \ll 1) + y & \text{nếu } x \text{ là số lẻ} \end{cases}$$

11.23. Cài đặt hàm in ra tất cả các chỉnh hợp chập k của n phần tử, với $k \in [0, \dots, n]$.

11.24. Cài đặt hàm in ra tất cả các hoán vị của n phần tử.

11.25. Cài đặt hàm in ra tất cả các tổ hợp chập k của n phần tử, với $k \in [0, \dots, n]$.

11.3 TÓM TẮT

Đệ qui cung cấp cho ta cơ chế giải quyết các bài toán phức tạp một cách đơn giản hơn. Xây dựng hàm đệ qui thông qua việc xác định điều kiện dừng và bước thực hiện tiếp theo. Chỉ nên cài đặt bằng phương pháp đệ qui khi không còn cách giải quyết bằng cách lặp thông thường.

12

LẬP TRÌNH ĐƠN THỂ

12.1 LÝ THUYẾT

Khái niệm

- *Lập trình đơn thể* là một phương pháp thiết kế phần mềm mà tập trung vào việc phân tách các chức năng của một chương trình thành thành các đơn thể độc lập
- Chia một chương trình lớn thành các tập tin nhỏ hơn, mỗi tập tin chứa các khai báo nguyên mẫu hàm, cài đặt các hàm và dữ liệu thực hiện một số chức năng nhất định. Việc phân chia này giúp quá trình lập trình: Dễ kiểm soát các lệnh và kiểm lỗi. Tránh được giới hạn kích thước tập tin quá lớn của ngôn ngữ lập trình.
- Lập trình đơn thể liên quan chặt chẽ đến lập trình cấu trúc và lập trình hướng đối tượng, tất cả đều có cùng một mục tiêu tạo thuận lợi cho xây dựng các chương trình phần mềm lớn và hệ thống bằng cách phân rã thành những mảnh nhỏ

Phương pháp

13

MỘT SỐ ĐỀ THI MẪU

ĐỀ SỐ 01

Thời gian: 120 phút

Bài 1) Viết chương trình tính tổng: $S(n) = 1! + 2! + \dots + n!$

Bài 2) Viết chương trình thực hiện các yêu cầu sau:

- a) Nhập mảng một chiều các số nguyên.
- b) Đếm số lượng giá trị chẵn âm trong mảng.
- c) Tìm số lẻ cuối cùng trong mảng.

Bài 3) Cho ma trận các số thực. Viết hàm tìm giá trị trong ma trận xa giá trị x nhất.

```
float xanhat(float a[][100], int m, int n, float x);
```

Bài 4) Hãy khai báo kiểu dữ liệu biểu diễn khái niệm điểm trong mặt phẳng Oxy (DIEM).

- a) Viết hàm nhập tọa độ điểm.

```
void nhap(DIEM &P);
```

- a) Viết hàm xuất tọa độ điểm.

```
void xuat(DIEM P);
```

- a) Viết hàm tính khoảng cách giữa 2 điểm.

```
float khoangcach(DIEM P, DIEM Q);
```

ĐỀ SỐ 02

Thời gian: 120 phút

Bài 1) Viết chương trình tính tổng: $S(n) = x + x^2 + \dots + x^n$

Bài 2) Viết chương trình thực hiện các yêu cầu sau:

- a) Nhập mảng một chiều các số nguyên.
- b) Đếm số lượng giá trị lẻ dương trong mảng.
- c) Tìm số chẵn cuối cùng trong mảng.

Bài 3) Cho ma trận các số thực. Viết hàm tìm giá trị trong ma trận gần giá trị x nhất.

```
float gannhat(float a[][100], int m, int n, float x);
```

Bài 4) Hãy khai báo kiểu dữ liệu biểu diễn khái niệm phân số (PHANSO)

- a) Viết hàm nhập phân số.

```
void nhap(PHANSO &x);
```

- b) Viết hàm xuất phân số.

```
void xuat(PHANSO x);
```

- c) Viết hàm tính tổng hai phân số.

```
PHANSO tong(PHANSO x, PHANSO y);
```

ĐỀ SỐ 03

Thời gian: 120 phút

Bài 1) Cho $S_n = \frac{1}{2} + \frac{3}{4} + \dots + \frac{2n+1}{2n+2}$ ($n > 0$)

- Vẽ lưu đồ thuật toán tính tổng trên.
- Viết hàm tính tổng trên bằng phương pháp đệ quy.

Bài 2) Cho mảng một chiều các số thực A kích thước $n \leq 100$. Hãy xây dựng hàm thực hiện các yêu cầu sau:

- Nhập giá trị các phần tử vào mảng.
- Tìm và trả về vị trí của phần tử có giá trị âm đầu tiên trong mảng. Nếu không có giá trị âm thì trả về -1.
- Tìm và trả về giá trị phần tử âm lớn nhất trong mảng A . Nếu mảng không có phần tử chứ A giá trị âm thì trả về 0.

Bài 3) Cho ma trận vuông các số nguyên A kích thước $n \times n$ ($5 < n < 10$). Hãy xây dựng các hàm cho phép thực hiện các yêu cầu sau:

- Nhập giá trị các phần tử vào ma trận.
- Đếm và trả về số lượng các phần tử là số nguyên tố trong ma trận.
- Tính trung bình cộng các phần tử trên đường chéo chính.

Bài 4) Hãy khai báo kiểu dữ liệu để biểu diễn thông tin của một nhân viên (NHANVIEN). Biết một nhân viên gồm:

- Mã nhân viên (MaNV): Chuỗi tối đa 5 ký tự.
- Tên nhân viên (TenNV): Chuỗi tối đa 30 ký tự.
- Chức vụ (ChucVu): Chuỗi tối đa 20 ký tự (gồm các chức vụ: "Truong phong", "Nhan vien", "Giam doc", "Pho giam doc", ...).
- Số năm làm việc (SoNam): Số nguyên 1 byte.
- Hệ số lương (HeSo): Kiểu số thực.

Cho danh sách gồm n ($n > 0$) nhân viên. Viết các hàm sau:

- Liệt kê các nhân viên có số năm làm việc từ 3 năm trở lên.
- Đếm số nhân viên có chức vụ là "Truong phong". Sắp xếp danh sách nhân viên tăng dần theo hệ số lương nhân viên.

ĐỀ SỐ 04

Thời gian: 120 phút

Bài 1) $S_n = 1 - 2 + 3 - \dots + (-1)^{n+1}n$ với $n > 0$

- Vẽ lưu đồ thuật toán tính tổng trên.
- Viết hàm tính tổng trên bằng phương pháp đệ quy.

Bài 2) Cho mảng một chiều các số nguyên A kích thước $n \leq 100$. Hãy xây dựng hàm thực hiện các yêu cầu sau:

- Nhập giá trị các phần tử vào mảng.
- Tìm và trả về vị trí của phần tử có giá trị là số nguyên tố đầu tiên trong mảng. Nếu không có giá trị là số nguyên tố thì trả về -1.
- Tìm và trả về giá trị phần tử là số nguyên tố lớn nhất trong mảng A . Nếu mảng không có phần tử là số nguyên tố thì trả về 0.

Bài 3) Cho ma trận vuông các số thực A kích thước $n \times n$ ($5 < n < 10$). Hãy xây dựng các hàm cho phép thực hiện các yêu cầu sau:

- Nhập giá trị các phần tử vào ma trận.
- Liệt kê những phần tử tại những dòng lẻ trong ma trận.
- Tính và trả về giá trị trung bình cộng của những phần tử âm trong ma trận.

Bài 4) Hãy khai báo kiểu dữ liệu để biểu diễn thông tin của một mặt hàng (MATHANG). Biết một mặt hàng gồm:

- Mã hàng (MaHang): Chuỗi tối đa 5 ký tự.
- Tên hàng (TenHang): Chuỗi tối đa 30 ký tự.
- Số lượng (SoLuong): Số nguyên 2 byte.
- Đơn vị tính (DonViTinh): Chuỗi tối đa 5 ký tự.
- Đơn giá (DonGia): Kiểu số thực.

Cho danh sách gồm n ($n > 0$) mặt hàng. Viết các hàm sau:

- Liệt kê các mặt hàng có số lượng lớn hơn 100.
- Tìm và trả về mặt hàng có thành tiền lớn nhất (thành tiền=số lượng*đơn giá). Sắp xếp danh sách các mặt hàng theo thứ tự giảm dần của đơn giá.

ĐỀ SỐ 05

Thời gian: 120 phút

Bài 1) Nhập số nguyên n ($0 < n \leq 20$). Viết chương trình xuất n phần tử đầu tiên của hai mảng A và B , cho biết các giá trị được xác định như sau:

$$A_1 = 1$$

$$B_1 = 1$$

$$A_i = \sqrt{A_{i-1}^2 + B_{i-1}^2}$$

$$B_i = 2A_{i-1} + \frac{B_{i-1}}{2}$$

Bài 2) Viết chương trình nhập vào ma trận vuông cấp n với n nhập từ bàn phím. Hãy kiểm tra ma trận này có phải là ma trận tam giác dưới hoặc tam giác trên theo đường chéo phụ không?

Bài 3) Mỗi hồ sơ nhân viên gồm:

- họ tên
- năm sinh
- lương cơ bản

Viết chương trình thực hiện các công việc sau:

- a) Nhập n hồ sơ với n nhập từ bàn phím.
- b) In ra họ tên và lương cơ bản của nhân viên có lương cơ bản thấp nhất và nhân viên có lương cơ bản cao nhất.
- c) Ghi xuống file văn bản (với tên file là hoso.txt) danh sách gồm họ tên, lương cơ bản, phụ cấp và thực lãnh của các nhân viên (mỗi nhân viên một dòng) biết rằng: Phụ cấp = 30% lương cơ bản và Thực lãnh = lương cơ bản + phụ cấp

ĐỀ SỐ 06

Thời gian: 120 phút

Bài 1) Nhập vào một dãy số thực kết thúc bởi 0 hoặc đã đủ 20 phần tử

- a) Sắp xếp dãy theo thứ tự tăng dần.
- b) Cho biết dãy có hội tụ không? (Dãy được gọi là hội tụ khi có nửa phần tử trở lên nhỏ hơn trung bình cộng của dãy).

Bài 2) Nhập vào ma trận $m \times n$ với m và n nhập từ bàn phím. Hãy kiểm tra xem ma trận có cân bằng theo cột hay không? (Ma trận cân bằng theo cột khi tổng các giá trị của các cột bên trái bằng tổng các giá trị của các cột bên phải, nếu số cột lẻ thì không tính cột giữa). Ví dụ: 8 4 5 8 9 3 5 7 4 6 4 9 7 5 1 Tổng bên trái = 33 Tổng bên phải = 33 Kết luận: Ma trận cân bằng theo cột.

Bài 3) Một Album ca nhạc MP3 gồm tối đa 150 ca khúc. Thông tin mỗi ca khúc gồm:

- Tên ca khúc
- Tên nhạc sỹ
- Tên ca sỹ
- Thời gian (tính bằng giây)

Viết chương trình thực hiện các công việc sau:

- a) Nhập n ca khúc với n nhập từ bàn phím.
- b) Xuất tổng thời gian của các ca khúc (hiển thị theo dạng hh:mm:ss) và cho biết tên ca khúc nào có thời gian dài nhất. Ghi xuống file văn bản (với tên file là mp3.txt) danh sách gồm tên ca khúc, tên nhạc sỹ, tên ca sỹ và thời gian (hiển thị theo dạng hh:mm:ss), mỗi ca khúc chiếm một dòng.

ĐỀ SỐ 07

Thời gian: 120 phút

Bài 1) Tính số hạng thứ n của hệ thức truy hồi như sau

$$f_n = \begin{cases} 1 & n = 1 \\ 2 & n = 2 \\ 3f_{n-1} + 2f_{n-2} & n > 2 \end{cases}$$

bằng hai cách

- a) Dùng đệ qui
- b) Khử đệ qui, dùng vòng lặp

Bài 2) Xây dựng một cấu trúc có các thành phần sau

- Mã số học sinh
- Họ và tên học sinh
- Điểm Toán
- Điểm Văn
- Điểm trung bình=(Điểm Toán+Điểm Văn)/2

Viết chương trình nhập dữ liệu của n học sinh và lưu vào một tập tin có tên là HOSOHS.DOC (hay mảng 1 chiều có cấu trúc). Sau đó đọc dữ liệu từ tập tin HOSOHS.DOC (hay mảng 1 chiều có cấu trúc), sắp xếp theo thứ tự Điểm trung bình giảm dần và xuất dữ liệu của từng học sinh ra màn hình

Bài 3) Cho n là một số nguyên dương, tính giá trị biểu thức sau bằng cách viết chương trình sử dụng vòng lặp và tối ưu vòng lặp.

$$S_n = 1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}$$

với n là số nguyên dương

ĐỀ SỐ 08

Thời gian: 120 phút

PHẦN I: (Trắc nghiệm) Chọn câu trả lời đúng nhất

Bài 1) Đoạn chương trình sau sẽ cho giá trị của t :

```
for(t=i=0; (i<10) && (t<100); i++, t += 2*i);
```

- a) 90
- b) 100
- c) 110
- d) 120

Bài 2) Cho dãy gồm 12 phần tử a như sau: $\{-9 -9 -5 -2 0 3 7 7 10 15\}$. Dùng thuật toán tìm nhị phân để tìm vị trí phần tử $x = -9$, vị trí tìm được sẽ là:

- a) -1
- b) 0
- c) 1
- d) 2

Bài 3) Chương trình sau:

```
#include <conio.h>
#include <stdio.h>
int a=1, b=2, c=3;
int A(int &a, int b)
{
    a += c + 2;
    b += 3;
    return a;
}
void main()
{
    printf(" %d %d", A(b, c), a+c);
}
```

sẽ in ra:

- a) 7 5
- b) 7 4

c) 7 3

d) 7 2

Bài 4) Chương trình sau:

```

#include <conio.h>
#include <stdio.h>
int A(int a, int &b)
{
    a += b + 2;
    b -= a;
    return a;
}
void main()
{
    int x = 5;
    printf(" %d %d", A(A(3, x), x), x);
}

```

Sẽ in ra:

a) 5 5

b) 5 7

c) 7 7

d) 7 5

Bài 5) Đoạn chương trình dưới đây khi thực thi sẽ:

```

char buf1[100], buf2[100], *strptr1, *strptr2;
strcpy(buf1, "abcdefghijklmnopqrstuvwxyz");
strcpy(buf2, "Hello");
strptr1 = buf1 + 6;
strcpy(strptr1, buf2); strptr2 = (strptr1 + 4);
strncpy(strptr2, buf2, 4);
printf("%s\n", buf1);

```

Sẽ in ra màn hình:

a) abcdefHellHelloabcdefghijklmnopqrstuvwxyz

b) ghijklmnHellotuvwxyz

c) abcdefghijklmnopqrstuvwxyz

d) `abcdefHellolmnopqrstuvwxyz`

PHẦN II: (Tự luận) Lập trình

Bài 1) Hãy viết hàm kiểm tra một số nguyên không n có phải là số nguyên tố hay không, hàm thực hiện sẽ trả về: 1 nếu n là số nguyên tố, 0 nếu n không là số nguyên tố

```
int LaSNT(unsigned int n);
```

Bài 2) Hãy viết hàm tìm tổng các số nguyên tố nằm trong mảng một chiều a có n phần tử

Bài 3) Hãy viết hàm xác định vị trí của số nguyên tố lớn nhất trên mảng a có n phần tử

Tài liệu tham khảo

- [1] JOHN R. HUBBARD. *455 Bài tập cấu trúc dữ liệu cài đặt bằng C++*. Nhà Xuất Bản Thống Kê.
- [2] TẤN DŨNG HUỖNH and ĐỨC HẢI HOÀNG. *Bài tập ngôn ngữ C từ A đến Z*. Nhà Xuất Bản Lao Động - Xã Hội.
- [3] HOÀI BẮC LÊ, HOÀNG THÁI LÊ, TẤN TRẦN MINH KHANG NGUYỄN, and PHƯƠNG THẢO NGUYỄN. *Giáo trình ngôn ngữ C*. Nhà Xuất Bản Đại Học Quốc Gia Tp. Hồ Chí Minh, 2003.
- [4] THANH SƠN NGUYỄN. *Tập bài giảng Kỹ thuật lập trình*. 2004.
- [5] TẤN TRẦN MINH KHANG NGUYỄN. *Bài tập Kỹ thuật lập trình - Tập 1*. Nhà Xuất Bản Đại Học Quốc Gia Tp. Hồ Chí Minh, 2004.
- [6] ĐÌNH TÊ NGUYỄN and ĐỨC HẢI HOÀNG. *Giáo trình lý thuyết & Bài tập ngôn ngữ C*. Nhà Xuất Bản Mũi Cà Mau.
- [7] SANFORD LEESTMA LARRY NYHOFF. *Pascal Programming and Solving*. Macmillan Publishing Company, 1990.
- [8] VĂN ẮT PHẠM. *Kỹ thuật lập trình C: cơ sở và nâng cao*. Nhà Xuất Bản Khoa Học Kỹ Thuật, 1996.
- [9] MINH THÁI TRẦN. *Tập bài giảng Kỹ thuật lập trình*. 2005.