

HƯỚNG DẪN THỰC HÀNH CƠ SỞ LẬP TRÌNH

HƯỚNG DẪN THỰC HÀNH CƠ SỞ LẬP TRÌNH

Tác giả: **Đào Quốc Thắng**

Trịnh Hoàng Nam

Thái Thị Thu Thủy

TRƯỜNG ĐẠI HỌC NGÂN HÀNG TP. HỒ CHÍ MINH

KHOA HỆ THỐNG THÔNG TIN QUẢN LÝ

ĐÀO QUỐC THẮNG - TRỊNH HOÀNG NAM (đồng chủ biên)
THÁI THỊ THU THỦY

**HƯỚNG DẪN
THỰC HÀNH
cơ sở
LẬP TRÌNH**

TP. HỒ CHÍ MINH, 2020

LỜI GIỚI THIỆU

Cơ sở lập trình (tiếng Anh: *Fundamentals of Programming*) là một môn học cơ sở ngành Hệ thống thông tin quản lý, Trường Đại học Ngân hàng TP. Hồ Chí Minh với mục tiêu trang bị cho sinh viên các kiến thức, kỹ năng cơ bản về lập trình, làm cơ sở cho một số môn học tiếp theo. Nhằm hỗ trợ cho việc dạy và học môn học được nêu, các tác giả đã tiến hành biên soạn cuốn tài liệu dưới đây theo hướng trình bày tóm tắt các nội dung lý thuyết quan trọng, bổ sung nhiều ví dụ minh họa cụ thể dễ hiểu, tập trung hướng dẫn, phát triển hệ thống bài tập mẫu và bài tập tự làm, giúp sinh viên nhanh chóng nắm bắt vấn đề, hình thành, phát triển tư duy, phong cách lập trình cho bản thân. Tài liệu cũng được biên soạn với các nội dung bám sát theo đề cương môn học, giúp giảng viên gấp thuận lợi trong quá trình giảng dạy.

Nội dung tài liệu được chia thành 9 chương với các chủ đề sau đây:

Chương 1: CÁC KHÁI NIỆM CƠ BẢN

Chương 2: CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ LẬP TRÌNH C

Chương 3: CÁC CẤU TRÚC ĐIỀU KHIỂN

Chương 4: HÀM

Chương 5: MẢNG

Chương 6: CON TRỎ

Chương 7: KIẾU CHUỖI KÝ TỰ

Chương 8: KIẾU CẤU TRÚC VÀ HỢP

Chương 9: KIẾU TẬP TIN

Do tài liệu được biên soạn lần đầu nên chắc chắn không tránh khỏi một số sai sót, mong được các quý độc giả, quý thầy cô và các em sinh viên đóng góp ý kiến, nhận xét,

giúp nhóm biên soạn bổ sung, chỉnh sửa, nâng cao chất lượng tài liệu, giúp phục vụ việc dạy và học cơ sở lập trình tại trường ngày một tốt hơn.

Xin chân thành cảm ơn

CÁC TÁC GIẢ

MỤC LỤC

| | |
|---|----|
| CHƯƠNG 1. CÁC KHÁI NIỆM CƠ BẢN | 4 |
| 1.1 Thuật toán và biểu diễn thuật toán | 4 |
| 1.2 Chương trình và ngôn ngữ lập trình..... | 7 |
| 1.3 Câu hỏi ôn tập | 10 |
| CHƯƠNG 2. CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ LẬP TRÌNH C..... | 11 |
| 2.1 Giới thiệu tổng quan về ngôn ngữ lập trình C | 11 |
| 2.2 Cấu trúc chung chương trình C | 13 |
| 2.3 Bộ ký tự, từ khóa và định danh | 14 |
| 2.4 Các kiểu dữ liệu cơ sở..... | 16 |
| 2.5 Hằng và biến | 20 |
| 2.6 Biểu thức và toán tử | 21 |
| 2.7 Câu lệnh | 26 |
| 2.8 Xuất – nhập | 27 |
| 2.9 Hướng dẫn sử dụng Dev C++ | 30 |
| 2.10 Câu hỏi ôn tập | 32 |
| 2.11 Bài tập thực hành | 32 |
| 2.12 Bài tập đề nghị | 33 |
| CHƯƠNG 3. CÁC CẤU TRÚC ĐIỀU KHIỂN | 35 |
| 3.1 Giới thiệu | 35 |
| 3.2 Cấu trúc rẽ nhánh | 36 |
| 3.3 Cấu trúc lặp | 40 |
| 3.4 Câu lệnh break và continue, goto..... | 45 |
| 3.5 Câu hỏi ôn tập | 46 |
| 3.6 Bài tập thực hành | 50 |
| 3.7 Bài tập đề nghị | 55 |
| CHƯƠNG 4. HÀM | 59 |
| 4.1 Giới thiệu hàm..... | 59 |
| 4.2 Dạng tổng quát của hàm | 60 |

| | | |
|----------------------------------|---|-----|
| 4.3 | Hàm main..... | 61 |
| 4.4 | Quy tắc cài đặt hàm..... | 62 |
| 4.5 | Quy tắc về phạm vi của hàm..... | 63 |
| 4.6 | Tham số và đối số | 63 |
| 4.7 | Nguyên mẫu hàm | 66 |
| 4.8 | Xây dựng chương trình với các hàm..... | 67 |
| 4.9 | Hàm main có tham số | 69 |
| 4.10 | Câu hỏi ôn tập | 70 |
| 4.11 | Bài tập thực hành | 71 |
| 4.12 | Bài tập đề nghị | 81 |
| CHƯƠNG 5. MẢNG | | 83 |
| 5.1 | Giới thiệu mảng..... | 83 |
| 5.2 | Mảng một chiều | 84 |
| 5.3 | Mảng nhiều chiều..... | 90 |
| 5.4 | Chương trình minh họa sử dụng mảng | 92 |
| 5.5 | Câu hỏi ôn tập | 94 |
| 5.6 | Bài tập thực hành | 95 |
| 5.7 | Bài tập đề nghị | 100 |
| CHƯƠNG 6. CON TRỎ | | 104 |
| 6.1 | Khái niệm con trỏ..... | 104 |
| 6.2 | Biến con trỏ..... | 105 |
| 6.3 | Các toán tử con trỏ | 106 |
| 6.4 | Các phép toán liên quan đến con trỏ | 107 |
| 6.5 | Con trỏ và mảng nhiều chiều | 110 |
| 6.6 | Cấp phát bộ nhớ | 111 |
| 6.7 | Câu hỏi ôn tập | 115 |
| 6.8 | Bài tập thực hành | 116 |
| 6.9 | Bài tập đề nghị | 122 |
| CHƯƠNG 7. KIỀU CHUỖI KÝ TỰ | | 125 |
| 7.1 | Khai báo chuỗi ký tự | 125 |

| | | |
|---|--|-----|
| 7.2 | Các thao tác đọc, ghi chuỗi ký tự..... | 126 |
| 7.3 | Các hàm xử lý chuỗi ký tự | 127 |
| 7.4 | Một số hàm xử lý chuỗi ký tự khác | 129 |
| 7.5 | Câu hỏi ôn tập | 130 |
| 7.6 | Bài tập có lời giải | 132 |
| 7.7 | Bài tập đề nghị | 138 |
| CHƯƠNG 8. KIỀU CÂU TRÚC VÀ KIỀU HỢP | | 139 |
| 8.1 | Kiểu câu trúc | 139 |
| 8.2 | Mảng câu trúc..... | 143 |
| 8.3 | Con trỏ câu trúc..... | 143 |
| 8.4 | Câu trúc và hàm | 144 |
| 8.5 | Hợp..... | 147 |
| 8.6 | Câu hỏi ôn tập | 148 |
| 8.7 | Bài tập có lời giải | 151 |
| 8.8 | Bài tập đề nghị | 154 |
| CHƯƠNG 9. KIỀU TẬP TIN | | 156 |
| 9.1 | Một số khái niệm..... | 156 |
| 9.2 | Thao tác trên tập tin | 157 |
| 9.3 | Một số hàm xử lý tập tin thông dụng..... | 163 |
| 9.4 | Hàm có tập tin là tham số | 165 |
| 9.5 | Câu hỏi ôn tập | 166 |
| 9.6 | Bài tập có lời giải | 168 |
| 9.7 | Bài tập đề nghị | 172 |

CHƯƠNG 1. CÁC KHÁI NIỆM CƠ BẢN

Trong chương này người học sẽ làm quen với một số khái niệm cơ bản về thuật toán, cách biểu diễn thuật toán, khái niệm chương trình, ngôn ngữ lập trình, các bước phát triển chương trình và môi trường lập trình.

Nội dung:

- Thuật toán và biểu diễn thuật toán
- Chương trình và ngôn ngữ lập trình
- Các bước xây dựng chương trình
- Môi trường lập trình

1.1 Thuật toán và biểu diễn thuật toán

1.1.1 Khái niệm vấn đề - bài toán

Theo cách hiểu thông thường của con người, *vấn đề* (*Problem*) là những khó khăn, vướng mắc cần giải quyết để có thể đạt được một số mục tiêu nào đó, còn *bài toán* (*Problem*) là những vấn đề mà khi giải quyết cần có sự tính toán.

Trong toán học, bài toán thường được mô hình hóa dưới dạng mô hình:

$$A \rightarrow B,$$

trong đó:

- A: Yêu tố có sẵn/giả thuyết
- B: Kết quả cần đạt (mục tiêu);
- : Chuỗi công việc (hành động) cần thực hiện.

Các bước giải quyết bài toán:

- Phân tích bài toán, tìm cách giải (chuỗi các phép toán cần thực hiện) và thu thập các dữ liệu cần thiết
- Thực hiện các phép toán với các dữ liệu đã thu thập
- Thông báo kết quả

1.1.2 Thuật toán (Algorithm)

Thuật toán là dãy *hữu hạn* các chỉ thị được mô tả rõ ràng và *thực hiện* được nhằm giải quyết một bài toán cụ thể nào đó. Thuật toán là một cơ sở nền tảng, là phương pháp cơ bản để mô tả lời giải cho các bài toán trong toán học và tin học.

Các tính chất của thuật toán:

- *Tính xác định*: các chỉ thị phải rõ ràng, không mơ hồ, nhập nhằng.
- *Tính đúng*: kết quả thực hiện phải đúng theo yêu cầu bài toán.
- *Tính hữu hạn (tính dừng)*: Việc thực hiện thuật toán phải kết thúc và cho kết quả sau một số bước xác định.
- *Tính phổ dụng*: thuật toán có thể được sử dụng để giải quyết một lớp bài toán tương tự.
- *Tính khách quan*: Thuật toán mang tính khách quan, độc lập vào ý chí chủ quan của người thực hiện.
- *Tính hiệu quả*: Thuật toán cần tối ưu về sử dụng bộ nhớ và đáp ứng yêu cầu về thời gian thực hiện bài toán (thực tế, rất khó cùng lúc đạt được cả 2 tiêu chí trên mà phải tìm cách dung hòa).

1.1.3 Biểu diễn thuật toán

Có thể biểu diễn thuật toán theo 4 cách: (1) Sử dụng *ngôn ngữ tự nhiên*, (2) Sử dụng *lưu đồ*, (3) Sử dụng *mã giả* và (4) Sử dụng *ngôn ngữ lập trình*.

1.1.3.1 Sử dụng ngôn ngữ tự nhiên

Biểu diễn thuật toán bằng ngôn ngữ tự nhiên là phương pháp được sử dụng rộng rãi trong toán học. Các bước thực hiện thuật toán được mô tả một cách đơn giản, ngắn gọn, rõ ràng, dễ hiểu bằng ngôn ngữ tự nhiên, không có qui định cụ thể nào về cách viết.

Ví dụ 1.1. Thuật toán giải phương trình $ax + b = 0$ (ngôn ngữ tự nhiên)

Dữ liệu: Hai số a, b.

Kết quả: Số nghiệm, giá trị nghiệm

B1: Xác định 2 số a, b.

B2: **Nếu** $a = 0$:

B2.1: **Nếu** $b = 0$: Thông báo “Vô số nghiệm”

B2.2: **Nếu** $b \neq 0$: Thông báo “Vô nghiệm”

B3: **Nếu** $a \neq 0$:

B3.1: Tính $x = -b/a$ (nghiệm phương trình)

B3.2: Thông báo x

B4: Kết thúc

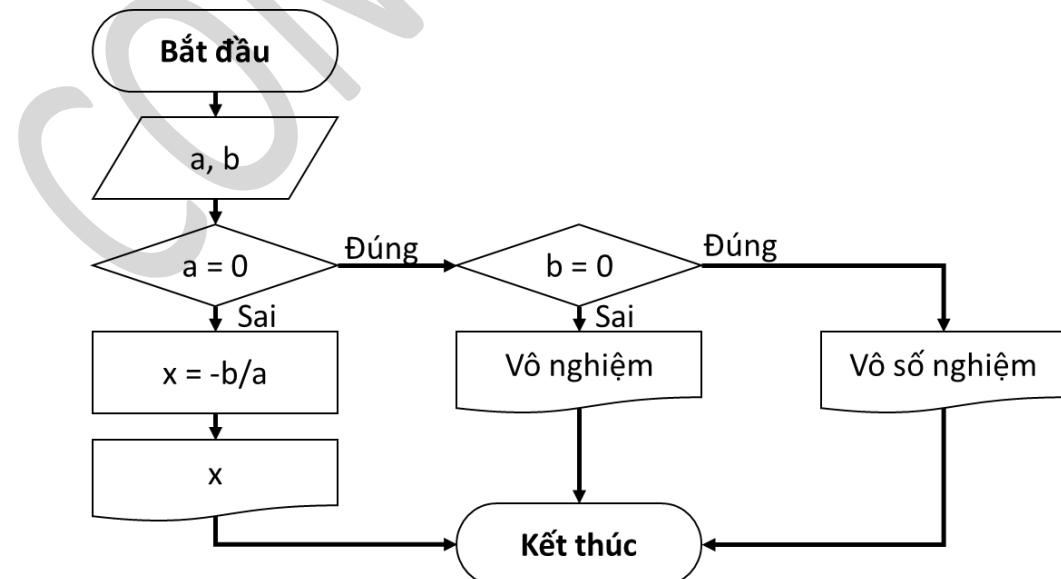
1.1.3.2 Sử dụng lưu đồ

Biểu diễn thuật toán bằng lưu đồ (flow chart) là công cụ cho phép biểu diễn thuật toán một cách trực quan với các khối chức năng cơ bản (*Bảng 1.1*). Lưu đồ thường chỉ dùng được cho các thuật toán tương đối ngắn, có thể biểu diễn trong một trang giấy.

Bảng 1.1. Các khối chức năng thường được sử dụng

| Khối | Tên | Ý nghĩa |
|------|----------------|-------------------------------|
| | Hình Oval | Điểm xuất phát, điểm kết thúc |
| | Hình chữ nhật | Hành động cần thực hiện |
| | Hình bình hành | Nhập/xuất dữ liệu |
| | Hình thoi | Kiểm tra điều kiện |
| | Tờ giấy | Đưa ra máy in |
| | Mũi tên | Hướng di chuyển của thao tác |

Ví dụ 1.2. Lưu đồ thuật toán giải phương trình $ax + b = 0$



1.1.3.3 Sử dụng mã giả

Biểu diễn thuật toán bằng mã giả (pseudo code) gần giống với phương pháp sử dụng ngôn ngữ tự nhiên, nhưng có sử dụng các cấu trúc chuẩn hóa (khai báo biến, cấu trúc điều khiển, ...) do người thiết kế quy định, thường gần với một ngôn ngữ lập trình nào đó.

Ví dụ 1.3. Thuật toán giải phương trình $ax + b = 0$ (mã giả tựa Pascal)

Thuật toán Giai_PTB1

Biến a, b, x: số thực;

Khởi đầu

Nhập (a, b)

Nếu ($a == 0$) **thì**

Nếu ($b == 0$) **thì** xuất (“Vô số nghiệm”)

Nếu không xuất (“Vô nghiệm”)

Nếu không

Khởi đầu

$x := -b/a$

 xuất (“Một nghiệm $x =$ “, x)

Kết thúc

Kết thúc

1.1.3.4 Sử dụng ngôn ngữ lập trình

Thuật toán được mô tả bằng một ngôn ngữ lập trình cụ thể, quen thuộc nào đó (chẳng hạn *Pascal*, *C*, *C++*, *Java*, ...).

1.2 Chương trình và ngôn ngữ lập trình

1.2.1 Chương trình máy tính

Chương trình máy tính (gọi tắt là *chương trình - program*) là tập hợp các chỉ thị được biểu thị qua ngôn ngữ lập trình nhằm mục đích thực hiện một số thao tác máy tính nào đó. *Lập trình máy tính* (gọi tắt là *lập trình - programming*) là quá trình cài đặt một hoặc nhiều thuật toán trừu tượng có liên quan với nhau bằng một ngôn ngữ lập trình để tạo ra một chương trình máy tính phục vụ cho việc giải quyết bài toán.

1.2.2 Ngôn ngữ lập trình

Ngôn ngữ lập trình (*programming language*) là ngôn ngữ chuyên dùng để viết chương trình cho máy tính.

Các thành phần cơ bản của ngôn ngữ lập trình gồm:

- *Bộ ký tự (character set)* hay bảng chữ cái dùng để viết chương trình.
- *Cú pháp (syntax)* là bộ quy tắc để viết chương trình.
- *Ngữ nghĩa (semantic)* xác định ý nghĩa các thao tác, hành động cần phải thực hiện, ngữ cảnh (context) của các câu lệnh trong chương trình.

Sau đây là các yêu cầu cơ bản đối với một ngôn ngữ lập trình:

- Dễ hiểu và dễ sử dụng, có thể dùng để giải quyết nhiều bài toán khác nhau.
- Miêu tả đầy đủ và rõ ràng các tiến trình (*process*) để chạy được trên các hệ máy tính khác nhau.

Các ngôn ngữ lập trình được phân loại thành các *thế hệ (Generations)*:

- *Thế hệ 1 (1GL) - Ngôn ngữ máy*: gồm một hệ lệnh và bộ mã dữ liệu biểu diễn dưới dạng chuỗi 0, 1, sử dụng trực tiếp trong máy tính. Mỗi loại máy tính có một hệ lệnh (ngôn ngữ máy) riêng, mỗi chương trình viết bằng ngôn ngữ máy chỉ có thể chạy được trên một loại máy nhất định.
- *Thế hệ 2 (2GL) - Ngôn ngữ cấp thấp (Hợp ngữ - Assembly)* biểu diễn các lệnh và dữ liệu dưới dạng các ký hiệu dễ nhớ hơn đối với con người. Các chương trình viết bằng hợp ngữ phải được dịch sang ngôn ngữ máy trước khi thực hiện bằng các chương trình đặc biệt gọi là *bộ hợp dịch (Assembler)*.
- *Thế hệ 3 (3GL) - Ngôn ngữ cấp cao* (như FORTRAN, COBOL, PASCAL...) biểu diễn các lệnh và dữ liệu dưới dạng gần giống với ngôn ngữ tự nhiên của con người. Các chương trình nguồn viết bằng ngôn ngữ cấp cao cần phải được dịch sang ngôn ngữ máy trước khi thực hiện bằng các *bộ thông dịch (Interpreter)* hay các *bộ hợp dịch (Compiler)*.
- *Thế hệ 4 (4 GL) - Ngôn ngữ hệ quản trị CSDL* (SQL Server, Oracle...).
- *Thế hệ 5 (5GL) - Ngôn ngữ trí tuệ nhân tạo* (Mathematica, Matlab, R...).

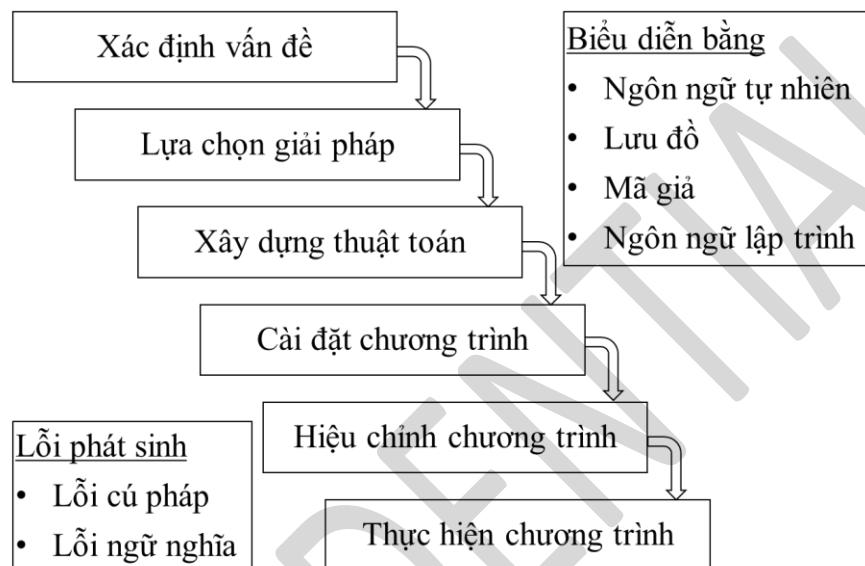
Các phương pháp tiếp cận lập trình và ngôn ngữ:

- *Lập trình tuần tự*: Assembly.
- *Lập trình hướng cấu trúc*: PASCAL, C, C++...
- *Lập trình hướng đối tượng*: Java, C#, C++...

- *Lập trình hàm*: R, Mathematica, Matlab, J...
- *Lập trình logic*: Prolog, Datalog...

1.2.3 Các bước xây dựng chương trình

Toàn bộ quá trình xây dựng chương trình nhằm giải quyết một bài toán được thực hiện theo 6 bước như sau:



Bước 1: Xác định yêu cầu của bài toán: Cần xác định rõ mục tiêu, phạm vi, các giới hạn, ràng buộc, các giả thiết của bài toán.

Bước 2: Lựa chọn phương pháp giải: Cần có kiến thức liên quan đến vấn đề đang giải quyết. Cần xác định rõ thuật toán sẽ tác động trên những đối tượng (thông tin) nào? Có bao nhiêu đối tượng (biến) cần xử lý? Mỗi biến có thể được lưu trữ dưới dạng nào, kiểu gì? Giá trị ban đầu của các biến? Hình dung trước kết xuất dữ liệu sau khi xử lý sẽ như thế nào?

Bước 3: Thể hiện thuật giải bằng **lưu đồ thuật toán** (nếu được).

Bước 4: **Cài đặt chương trình:** Dùng một trình soạn thảo văn bản để tạo chương trình nguồn (source code) theo một ngôn ngữ nào đó.

Bước 5: **Hiệu chỉnh chương trình:** Biên dịch, chạy thử và sửa lỗi chương trình (lỗi cú pháp, lỗi thực hiện). Các lỗi về mặt cú pháp (syntax error) thường đơn giản, có thể phát hiện và chỉnh sửa được ngay khi biên dịch còn lỗi xảy ra trong lúc thực hiện chương trình (run-time error) như tràn stack, không đủ bộ nhớ, lỗi logic...

khó phát hiện hơn nhiều. Đặc biệt, các lỗi về thuật toán hoặc nghiêm trọng hơn, xác định sau bài toán hoặc lựa chọn phương pháp giải không phù hợp có thể dẫn tới việc phải xây dựng chương trình lại từ đầu.

Bước 6: Kết thúc việc xây dựng, chuyển giao thực hiện chương trình.

1.2.4 Môi trường phát triển tích hợp

Môi trường phát triển tích hợp (*Integrated Development Environment - IDE*) là công cụ cho phép soạn thảo, biên dịch, gỡ lỗi, thực hiện các chương trình. Các thành phần cơ bản của một IDE gồm:

- Bộ biên tập (trình soạn thảo chương trình nguồn - EDITOR)
- Bộ biên dịch chương trình (Trình COMPILER)
- Bộ thực hiện chương trình nguồn (Trình RUNTIME)
- Bộ sửa lỗi chương trình nguồn (Trình DEBUG)

Một số môi trường phát triển tích hợp IDE hỗ trợ ngôn ngữ lập trình C, C++ bao gồm DevC++, CodeBlocks C++, Microsoft Visual C#, Visual C++, ...

1.3 Câu hỏi ôn tập

- 1) Mô hình chung của các bài toán và các bước giải quyết bài toán là gì?
- 2) Thuật toán là gì? Trình bày các tính chất cơ bản của một thuật toán.
- 3) Các cách biểu diễn thuật toán? Ưu và nhược điểm của từng phương pháp? Ví dụ.
- 4) Ngôn ngữ lập trình là gì, các thể hệ ngôn ngữ lập trình, các phương pháp lập trình, ví dụ ngôn ngữ lập trình tương ứng mỗi loại?
- 5) Các bước xây dựng chương trình?
- 6) Môi trường phát triển tích hợp IDE là gì? Các thành phần cơ bản của một IDE?
- 7) Biểu diễn thuật toán giải phương trình $a.x^2 + b.x + c = 0$ theo các hệ số a, b, c dưới dạng ngôn ngữ tự nhiên và lưu đồ.
- 8) Biểu diễn thuật toán tính tổng và tích của dãy số a_1, a_2, \dots, a_n nhập từ bàn phím dưới dạng ngôn ngữ tự nhiên và lưu đồ.
- 9) Biểu diễn thuật toán tìm số lớn nhất và số nhỏ nhất trong dãy số a_1, a_2, \dots, a_n nhập từ bàn phím dưới dạng ngôn ngữ tự nhiên và dạng lưu đồ.

CHƯƠNG 2. CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ LẬP TRÌNH C

Trong chương này, người học làm quen với một số khái niệm tổng quan về ngôn ngữ lập trình C, cấu trúc chương trình C và các môi trường phát triển tích hợp C/C++.

Nội dung:

- Giới thiệu tổng quan về ngôn ngữ lập trình C
- Cấu trúc chung chương trình C
- Bộ ký tự, từ khóa và định danh
- Các kiểu dữ liệu cơ sở
- Hằng và biến
- Biểu thức và toán tử
- Câu lệnh
- Xuất – nhập dữ liệu

2.1 Giới thiệu tổng quan về ngôn ngữ lập trình C

Ngôn ngữ lập trình C (với tiền thân là *ngôn ngữ lập trình B*) được Dennis Ritchie phát triển tại Phòng thí nghiệm Bell thuộc tập đoàn Telephone (AT&T), Hoa Kỳ, năm 1972 nhằm mục đích viết hệ điều hành Unix. Từ khi ra đời tới nay, C không ngừng được phát triển, bổ sung thêm các khả năng mới và được chuẩn hóa. Năm 1989, Viện tiêu chuẩn quốc gia Hoa Kỳ (ANSI) công bố một tiêu chuẩn cho C, mang mã ANSI X3.159-1989. Phiên bản này thường được gọi là "ANSI C" hay "C89". Năm 1990, Tiêu chuẩn ANSI C (với một vài chi tiết về định dạng được chỉnh lại) đã được tiêu chuẩn hóa bởi Tổ chức Quốc tế về Tiêu chuẩn hóa (ISO), trở thành chuẩn ISO/IEC 9899:1990, thường được gọi là "C90" hay "ISO C". Ngày nay, ANSI C được hỗ trợ bởi hầu hết các trình dịch C, và hầu hết các mã đều được viết trên ANSI C. Tiêu chuẩn gần đây nhất của ngôn ngữ C được phát hành là ISO/IEC 9899:2011 ("C11", được phát hành năm 2011). Các trình dịch hiện tại chỉ hỗ trợ một phần chuẩn này.

Một số đặc điểm của ngôn ngữ lập trình C:

- Là ngôn ngữ lập trình có cấu trúc, phân biệt chữ HOA - thường (case sensitive).
- Là ngôn ngữ bậc trung (*medium-level language*), kết hợp được các tính năng ngôn ngữ bậc cao với ngôn ngữ bậc thấp.
- Mạnh mẽ trong việc xử lý bit, địa chỉ ô nhớ, thích hợp lập trình hệ thống

Ưu điểm của ngôn ngữ lập trình C

- Rất mạnh và mềm dẻo, có khả năng thể hiện bất cứ ý tưởng nào, dùng viết hệ điều hành, các trình điều khiển, soạn thảo văn bản, chương trình dịch...
- Được sử dụng rộng rãi bởi các nhà lập trình chuyên nghiệp. Chương trình viết bởi C rất hiệu quả (có thể đạt 80% tính năng của chương trình viết bằng mã máy)
- Có tính khả chuyển, dễ thích nghi, ít thay đổi trên các máy tính khác nhau.
- Có ít từ khoá.
- Có cấu trúc module, sử dụng hàm, có thể sử dụng nhiều lần.

Nhược điểm của ngôn ngữ lập trình C

- Cú pháp lạ và khó học.
- Một số ký hiệu có nhiều nghĩa khác nhau (ví dụ ký hiệu * là toán tử nhân, toán tử không định hướng, thay thế...)
- Quá mềm dẻo (truy nhập tự do vào dữ liệu, trộn lẫn toán tử...)

Từ nền tảng C, người ta đã phát triển thêm nhiều ngôn ngữ mới như C++, C#, Java.

Đặc biệt, C++, được tạo ra bởi Bjarne Stroustrup, là một ngôn ngữ lập trình đa năng như một phần mở rộng của C với các lớp (*Class*). Ngôn ngữ C++ hiện đại có các tính năng lập trình cấu trúc và lập trình hướng đối tượng. Từ thập niên 1990, C++ đã trở thành một trong những ngôn ngữ thương mại được ưa thích và phổ biến của lập trình viên trong các lĩnh vực *lập trình hệ thống* và phần mềm nhúng. Người lập trình cũng có thể viết các chương trình thuần C trong môi trường phát triển tích hợp (IDE) C++.

Một số IDE C/C++ được sử dụng phổ biến hiện nay:

- GNU C (Linux).
- Turbo C/C++ (MS OS, MS Windows).

- Dev-C++, CodeBlocks-C++ (Windows).
- Microsoft Visual C++ (Microsoft.NET, Windows).

2.2 Cấu trúc chung chương trình C

Một chương trình C bao gồm những phần sau đây:

- Các lệnh tiền xử lý
- Các hàm
- Các biến
- Các lệnh và biểu thức
- Các chú thích

Cấu trúc tổng quát chương trình:

```
#include      /* Gọi các tập tin tiền xử lý */
#define       /* Định nghĩa */
typedef     /* Định nghĩa kiểu */
int         /* Khai báo biến toàn cục */
const       /* Khai báo hằng */
/*Khai báo các hàm, có thể có hoặc không*/
Kiểu_dữ_liệu tên_hàm (các tham số)
{
    /* Khai báo các biến, hằng*/
    /* Các lệnh của hàm*/
    return();   /* Trả lại giá trị */
}
int main()
{
    /* Bắt buộc phải có */
    /* Khai báo các biến, hằng*/
    /* Các lệnh của hàm*/
    return();   /* Có thể có hoặc không */
}
```

Ví dụ 2.1. Ghi ra màn hình chuỗi “Hello, C!”

```
#include <stdio.h>
int main(){
    printf("HELLO, C !");
    getchar();
    return 0;
}
```

2.3 Bộ ký tự, từ khóa và định danh

2.3.1 Bộ ký tự

Chương trình C là một tập các từ được tạo từ các ký tự sau:

- Các chữ cái hoa: A, B, C, ..., Z.
- Các chữ cái thường: a, b, c, ..., z.
- Các chữ số: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- Các ký hiệu toán học: + - * / = < > ().
- Các ký tự đặc biệt: . , : ; [] % \# \$ ‘ ^ & @ .
- Ký tự gạch nối ‘ _ ’ và dấu cách (khoảng trắng), dấu tab, ký tự xuống dòng.

2.3.2 Từ khóa

Từ khóa (keyword) là các từ dành riêng trong ngôn ngữ, mỗi từ có tác dụng và ý nghĩa cụ thể. Người lập trình không được sử dụng từ khóa để đặt tên cho biến, hàm, tên chương trình con.

Một số từ khóa thông dụng:

- o const, enum, signed, struct, typedef, unsigned,
- o char, double, float, int, long, short, void,
- o case, default, else, if, switch,
- o do, for, while,
- o break, continue, goto, return.

2.3.3 Định danh

Tên hay định danh (Identifier) là chuỗi ký tự liền nhau gồm các chữ cái a...z, A...Z, các chữ số 0...9, và dấu gạch nối.

Chú ý :

- Mọi tên đều phải khai báo trước khi sử dụng.
- Tên trong C phân biệt chữ HOA, thường.
- Độ dài tối đa mặc định là 32 ký tự.

Quy tắc đặt tên :

- Tên không được trùng với các từ khoá.
- Không được bắt đầu bằng chữ số.

- Không chứa ký tự đặc biệt như dấu cách, dấu chấm.
- Tên phải gợi nhớ về đối tượng được đặt tên.
- Cùng phạm vi không được đặt 2 tên trùng nhau.
- Phân biệt chữ hoa chữ thường, do đó các tên sau đây khác nhau A, a, BaiTap, baitap, BAITAP, bAltaP, ...
- Thường dùng chữ HOA đặt tên cho hằng, chữ thường cho các đối tượng khác.

Ví dụ 2.2. Quy tắc đặt tên trong ngôn ngữ C

Các tên hợp lệ: GiaiPhuongTrinh, Bai_Tap1, PI

Các tên không hợp lệ:

| | |
|---------|---------------------|
| 1A | bắt đầu bằng chữ số |
| PI\$ | chứa kí hiệu \$ |
| Giai pt | chứa dấu cách |
| char | trùng từ khoá char |

2.3.4 Phân cách và ghi chú

Trong C, mỗi lệnh phải được kết thúc bởi một dấu chấm phẩy (';'). Các ghi chú được đặt vào giữa cặp dấu /* ... */ hoặc sau ký hiệu //

Ví dụ 2.3. Phân cách và ghi chú trong ngôn ngữ C

```
/*Yeu cau: Ghi ra man hinh dong chu Hello World! va xuong hang moi*/
#include <stdio.h> // Su dung thu vien xuat nhap chuan
int main(){
    //...
    printf("Hello World!"); printf("\n"); //cau lenh phai ket thuc voi dau ;
    //...
    return 0;
}
```

2.3.5 Hằng ký tự và hằng chuỗi

Trong C, các hằng ký tự được đặt giữa hai dấu nháy đơn ('A'), hằng chuỗi ký tự được đặt giữa hai dấu nháy kép ("A"). Chú ý, hằng ký tự 'A' khác hằng chuỗi ký tự "A".

Một số quy tắc khi viết chương trình C

- Mỗi câu lệnh có thể viết trên một hay nhiều dòng, nhưng phải kết thúc bằng dấu chấm phẩy (;
- Khi chuỗi ký tự nằm trên nhiều dòng, kết thúc mỗi dòng (trừ dòng cuối cùng) bằng dấu \
- Lời chú thích có thể viết trên một hoặc nhiều dòng, đặt giữa cặp dấu /*...*/
- Các câu trong cùng khối phải thăng hàng theo chiều dọc.

Ví dụ 2.4. Một số quy tắc khi viết chương trình C

Xem xét hai câu lệnh sau đây:

```
(1) printf("Hello world!\n");  
(2) printf("Hello world! This is my first \  
C program!\n");
```

Nhận xét:

- Câu lệnh (1) ghi nội dung trên một dòng rồi xuống hàng
- Câu lệnh (2) ghi nội dung trên hai hàng sau đó xuống hàng

2.4 Các kiểu dữ liệu cơ sở

2.4.1 Khái niệm kiểu dữ liệu

Kiểu dữ liệu (*data type*) là một tập hợp các giá trị mà một biến thuộc kiểu đó có thể nhận cùng một số phép toán có thể thực hiện.

Các kiểu dữ liệu trong C gồm:

- Kiểu vô hướng
- Kiểu cơ sở (số nguyên, số thực, kiểu ký tự, kiểu logic)
- Kiểu dữ liệu có cấu trúc (thời gian, chuỗi ký tự, tập tin)
- Kiểu do người dùng định nghĩa

2.4.2 Kiểu số nguyên

Trong C, *số nguyên* (*integer*) được biểu diễn dưới dạng chuỗi nhị phân với độ dài 1 byte (char), 2 bytes (short hoặc integer), 4 bytes (long) và được chia thành 2 loại: *số nguyên có dấu* (*signed integer*) và *số nguyên không dấu* (*unsigned integer*) với các miền giá trị xác định.

2.4.2.1 Các kiểu số nguyên có dấu

Số nguyên có dấu được biểu diễn dưới dạng chuỗi nhị phân với độ dài n bit, trong đó 1 bit (bit 0) được sử dụng để xác định dấu (+/-). Miền giá trị của số nằm trong khoảng từ -2^{n-1} tới $+2^{n-1} - 1$. *Bảng 2.1* thể hiện các kiểu số nguyên có dấu trong C:

Bảng 2.1. Các kiểu số nguyên có dấu trong C

| Kiểu | Kích thước (số byte) | Miền giá trị |
|-------|----------------------|-----------------------------------|
| char | 1 | -128 ... +127 |
| int | 2 | -32.768 ... +32.767 |
| short | 2 | -32.768 ... +32.767 |
| long | 4 | -2.147.483.648 ... +2.147.483.647 |

Ghi chú: kiểu *int* trong C++, có kích thước 4 bytes như kiểu *long*.

2.4.2.2 Các kiểu số nguyên không dấu

Số nguyên không dấu n bit có miền giá trị từ 0 tới $2^n - 1$. Các kiểu số nguyên không dấu trong C (*Bảng 2.2*):

Bảng 2.2. Các kiểu số nguyên không dấu trong C

| Kiểu | Kích thước (số byte) | Miền giá trị |
|----------------|----------------------|---------------------|
| unsigned char | 1 | 0 ... 255 |
| unsigned int | 2 | 0 ... 65.535 |
| unsigned short | 2 | 0 ... 65.535 |
| unsigned long | 4 | 0 ... 4.294.967.295 |

2.4.2.3 Các phép tính số học với số nguyên

Các phép toán số học có thể thực hiện trên số nguyên bao gồm: phép *cộng*, phép *trừ*, phép *nhân*, phép *chia lấy phần nguyên* và *chia lấy số dư* được nêu trong

Bảng 2.3 dưới đây.

Bảng 2.3. Các phép tính số học với số nguyên

| Phép toán | Toán tử | Ví dụ | Kết quả |
|----------------------|---------|----------|--------------|
| Cộng | + | $x + y$ | $5 + 2 = 7$ |
| Trừ | - | $x - y$ | $5 - 2 = 3$ |
| Nhân | * | $x * y$ | $5 * 2 = 10$ |
| Chia lấy phần nguyên | / | x / y | $5 / 2 = 2$ |
| Chia lấy số dư | % | $x \% y$ | $5 \% 2 = 1$ |

Chú ý:

- Kết quả phép chia hai số nguyên là số nguyên, muốn là số thực phải ép kiểu dạng (*float*) x / y
- Thận trọng tránh hiện tượng tràn số
- Biểu diễn số nguyên dạng hệ đếm 16 (Hexa) và hệ đếm 8 (Octa)
- Ngoài hệ thập phân (hệ đếm 10), số nguyên còn có thể được biểu diễn dưới các dạng hệ nhị phân (hệ đếm cơ số 2), hệ Hexa (hệ đếm cơ số 16) và Octa (cơ số 8). Trong C, các số Hexa được bắt đầu bằng ký tự 0x hoặc 0X, các số Octa được biểu diễn bắt đầu bằng ký tự 0.

Ví dụ 2.5. Hệ đếm 8 và 16

Số 65 (hệ 10) được viết là 0x41 (hoặc 0X41) trong hệ 16 và 0101 trong hệ 8

Số 15 (hệ 10) được viết là 0xF (hoặc 0XF) trong hệ 16 và 017 trong hệ 8

2.4.2.4 Hằng số nguyên định trước kiểu

Ngôn ngữ lập trình C cho phép người lập trình định các kiểu cụ thể cho hằng bằng cách thêm một ký tự cuối vào số: **L** (long), **U** (unsigned integer), **UL** (unsigned long). Trong trường hợp không nêu rõ kiểu, C sẽ tự gán kiểu cho các số. Chẳng hạn số 1000L có kiểu *long*, 1000UL có kiểu *unsigned long* trong khi số 1000 có kiểu *int*.

2.4.3 Kiểu số thực

Các số thực trong ngôn ngữ lập trình C có thể được viết dưới 2 dạng:

- Dạng viết bình thường (dạng dấu chấm cố định), ví dụ: 3.14, -453174, -0.543

- Dạng khoa học (dạng dấu chấm động), gồm 3 phần: (1) dấu, (2) phần định trị, (3) phần mũ viết sau chữ **E** (hoặc **e**), giữa chúng không có khoảng cách, ví dụ: 7.52144E+02 (=752.144), -3.2672E-2 (=−0.032672)

Chú ý:

- Nếu không có phần mũ thì phần định trị bắt buộc phải có dấu chấm (.)
- Có thể không cần số 0 ở đầu (ví dụ: .3216)
- KHÔNG tồn tại phép % cho số thực
- Hàng số thực cũng có thể được định trước kiểu bằng cách thêm ký tự F (float), L (long double) vào cuối số thực, ví dụ: 0.1234567E-20L, 32.76F

Bảng 2.4. Các kiểu số thực trong ngôn ngữ lập trình C

| Kiểu | Kích thước (số byte) | Miền giá trị |
|-------------|----------------------|---|
| float | 4 | 1.24E-38 ... 3.4E38 Độ chính xác khoảng 7 chữ số |
| double | 8 | 2.2E-308 ... 1.8E308 Độ chính xác khoảng 15 chữ số |
| long double | 10 | 3.4E4932...3.4E4932 Độ chính xác khoảng 19 chữ số |

2.4.4 Kiểu ký tự

Kiểu ký tự (*kiểu char*) trong ngôn ngữ lập trình C gồm 256 ký tự mã hóa theo bảng mã ASCII. Đây cũng chính là kiểu số nguyên 1 byte. Các ký tự này có thể được biểu diễn dưới dạng \xHHH hoặc \DDD với HHH là giá trị số Hexa, DDD là giá trị số Octa.

Bảng 2.5. Một số ký tự đặc biệt và cách biểu diễn trong C

| Ký tự | Dãy mã | Giá trị trong bảng ASCII |
|-----------------|--------|--------------------------|
| Xoá trái | \b | X08 |
| Nhảy cách ngang | \t | X09 |
| Xuống dòng | \n | X0A |
| Dấu “ | ” | X22 |
| Dấu ‘ | ’ | X27 |
| Dấu \ | \\\ | X5C |
| Null | \0 | X00 |

Ví dụ 2.6. Kiểu ký tự trong ngôn ngữ lập trình C

Ký tự ‘A’ được lưu trong bộ nhớ với mã 65, ký tự ‘a’ được lưu với mã 97.

Phép toán ‘A’ + 2 cho kết quả là ‘C’ (mã 67).

Phép toán ‘A’ – ‘a’ cho kết quả là -32.

2.4.5 Kiểu Boolean

Ngôn ngữ lập trình C không khai báo kiểu Boolean (kiểu logic) mà ngầm định một cách không tường minh:

false (sai): giá trị 0.

true (đúng): giá trị khác 0, thường là 1.

2.5 Hằng và biến

2.5.1 Hằng

Hằng (constant) là đại lượng có giá trị không đổi, được khai báo trong chương trình với cú pháp sau đây:

```
const <kiểu dữ liệu> <tên hằng> = <giá trị>; //Hằng thường, có dấu = và ;
#define <tên hằng> <chuỗi thay thế>           //Hằng tượng trưng, không dấu ;
```

Ví dụ 2.7. Khai báo hằng trong ngôn ngữ lập trình C

```
const int A = 200;
const float D = 15.06e-3;
const char T='t';
#define MAX 100
#define PI 3.14
#define TRUE 1
#define FALSE 0
```

2.5.2 Biến

Biến (variable) là đại lượng có thể thay đổi được giá trị, được khai báo trong chương trình với cú pháp sau đây:

```
<kiểu dữ liệu> <danh sách các biến>; //Chỉ khai báo, không khởi tạo
<kiểu dữ liệu> <tên biến> = <giá trị>; //Khai báo và khởi tạo
```

Ví dụ 2.8. Khai báo biến trong ngôn ngữ lập trình C

```
int i;  
int j, k;  
int i=1, j;  
int i, j =1;
```

2.6 Biểu thức và toán tử

2.6.1 Biểu thức

Biểu thức là công thức tính toán bao gồm các *toán hạng (operand)* và *toán tử (operator)* tương ứng. Các toán hạng có thể là một biến, hằng, lời gọi hàm, một biểu thức con khác đặt trong cặp ngoặc đơn, hình thành nên một biểu thức phức hợp. Trong ngôn ngữ lập trình C, biểu thức luôn trả về một giá trị.

Các loại biểu thức thông dụng:

- Biểu thức số học, ví dụ: $3 + 5$, $x + y$
- Biểu thức logic, ví dụ $x > 3$, $x > y$

2.6.2 Toán tử

2.6.2.1 Toán tử số học

Toán tử số học bao gồm các phép toán cộng, trừ, nhân, chia và chia lấy số dư, được mô tả trong *Bảng 2.6*

Bảng 2.6. Các toán tử số học trong ngôn ngữ lập trình C

| Ký hiệu | Ý nghĩa | Số ngôi | Toán hạng |
|---------|------------------|---------|--------------------------|
| + | Cộng | 2 | int, float, double, char |
| - | Trừ | 2 | int, float, double, char |
| * | Nhân | 2 | int, float, double, char |
| / | Chia | 2 | int, float, double, char |
| % | Chia lấy phần dư | 2 | int, char |

2.6.2.2 Toán tử quan hệ

Toán tử quan hệ bao gồm các phép toán so sánh lớn hơn, bé hơn, bằng, khác, được mô tả trong *Bảng 2.7*

Bảng 2.7. Các toán tử quan hệ trong ngôn ngữ lập trình C

| Ký hiệu | Ý nghĩa | Số ngôi | Toán hạng |
|---------|-------------------|---------|--------------------------|
| < | Nhỏ hơn | 2 | int, float, double, char |
| <= | Nhỏ hơn hoặc bằng | 2 | int, float, double, char |
| > | Lớn hơn | 2 | int, float, double, char |
| >= | Lớn hơn hoặc bằng | 2 | int, float, double, char |
| == | Bằng | 2 | int, float, double, char |
| != | Khác | 2 | int, float, double, char |

2.6.2.3 Toán tử logic

Toán tử logic trong ngôn ngữ lập trình C gồm các toán tử **&&** (and), **||** (or), **!** (not), được mô tả trong *Bảng 2.8*.

Bảng 2.8. Bảng giá trị của các toán tử logic

| | | | | | | | | |
|-------------------|---|---|-----------|---|---|----------|---|---|
| && | 0 | 1 | | 0 | 1 | ! | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | | | |

2.6.2.4 Toán tử gán

Toán tử gán (ký hiệu **=**) dùng để thay đổi giá trị của một biến bằng một hằng hoặc bằng giá trị của một biểu thức, có dạng sau đây:

<biến> = <hằng>; //biến và hằng cùng kiểu

<biến> = <biểu thức>; //biểu thức có kết quả trả về cùng kiểu với biến

Chú ý:

- Ngôn ngữ lập trình C cho phép biểu thức gồm nhiều phép gán
- Ngôn ngữ lập trình C cho phép một số toán tử gán mở rộng (*Bảng 2.9*)

Ví dụ 2.9. Phép gán kép trong ngôn ngữ lập trình C

int a=b=c=3; //c=3; b=c; a=b;

int e = a + b + (d=2); //d=2; e=a+b+2;

int f = (d==2); //f=1, nếu d=2, ngược lại, f=0;

Bảng 2.9. Một số toán tử gán mở rộng

| | |
|--------------------------|--------------------------|
| $x = x + y$ hay $x+=y$ | $x = x >> y$ hay $x>>=y$ |
| $x = x - y$ hay $x-=y$ | $x = x << y$ hay $x<<=y$ |
| $x = x * y$ hay $x*=y$ | $x = x \& y$ hay $x\&=y$ |
| $x = x / y$ hay $x/=y$ | $x = x y$ hay $x =y$ |
| $x = x \% y$ hay $x\%=y$ | $x = x ^ y$ hay $x^=y$ |

2.6.2.5 Các toán tử trên bit

Các toán tử trên bit tác động lên các bit của toán hạng (nguyên), bao gồm các toán tử $\&$ (and), $|$ (or), \wedge (xor), \sim (not hay lấy số bù 1), $>>$ (shift right), $<<$ (shift left), toán tử gộp: $\&=$, $|=$, $\wedge=$, $\sim=$, $>>=$, $<<=$.

Bảng 2.10. Một số toán tử trên bit

| $\&$ | 0 | 1 | $ $ | 0 | 1 | \wedge | 0 | 1 |
|------|---|---|-----|---|---|----------|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

| \sim | 0 | 1 | $>>$ | $a >> n = a/2^n$ |
|--------|---|---|------|--------------------|
| 1 | 1 | 0 | $<<$ | $a << n = a * 2^n$ |

Ví dụ 2.10. Các toán tử trên bit trong ngôn ngữ lập trình C

```

int a = 5;      // 0000 0000 0000 0101
int b = 6;      // 0000 0000 0000 0110
int z1, z2, z3, z4, z5, z6;
z1 = a & b;    // 0000 0000 0000 0100
z2 = a | b;    // 0000 0000 0000 0111
z3 = a ^ b;    // 0000 0000 0000 0011
z4 = ~a;       // 1111 1111 1111 1010
z5 = a >> 2;  // 0000 0000 0000 0001
z6 = a << 2;  // 0000 0000 0001 0100
a >>= 1;       // 0000 0000 0000 0010

```

2.6.2.6 Toán tử điều kiện

Toán tử điều kiện là toán tử ba ngôi (gồm ba toán hạng) với cú pháp sau đây:

<biểu thức 1> ? <biểu thức 2> : <biểu thức 3>

Trong đó, giá trị trả về là giá trị <biểu thức 2> nếu <biểu thức 1> đúng hoặc <biểu thức 3> nếu <biểu thức 1> sai.

Ví dụ 2.11. Toán tử điều kiện

```
int x = 1, y = 2, s1, s2;  
s1 = x>y? 1:2;      //s1 = 2  
s2 = y>1? 10:20;    //s2 = 10
```

2.6.2.7 Toán tử tăng (++), giảm (--)

Cú pháp biểu thức:

<biến>++ : tăng giá trị biến thêm 1 sau khi sử dụng.

++<biến> : tăng giá trị biến thêm 1 trước khi sử dụng.

<biến>-- : giảm giá trị biến đi 1 sau khi sử dụng.

--<biến> : giảm giá trị biến đi 1 trước khi sử dụng.

2.6.2.8 Toán tử phẩy

Toán tử phẩy cho phép tạo biểu thức liên kết dưới dạng chuỗi các biểu thức đặt cách nhau bằng dấu phẩy (,) với thứ tự tính toán và kết quả trả về như sau:

- Các biểu thức con lần lượt được tính từ trái sang phải.
- Biểu thức mới nhận được là giá trị của biểu thức bên phải cùng.

Ví dụ 2.12. Toán tử phẩy

```
int x, a=1, b=2;  
x = (a++, b = b + 2);  
//x nhận giá trị tương đương với dãy biểu thức dưới đây  
a++; b = b + 2; x = b; //x=4
```

2.6.2.9 Toán tử sizeof()

Toán tử `sizeof(<biểu thức>)` trả về kích thước (số bytes) tương ứng với giá trị trả về của `<biểu thức>`. Trong khi đó, `sizeof(<kiểu>)` trả về kích thước (số bytes) của kiểu.

Ví dụ 2.13. Toán tử sizeof()

```
printf("%d\n", sizeof(long));      //Ghi ra 4, ứng với kích thước của kiểu long  
float x;  
printf("%d\n", sizeof(x+2));      //Ghi ra 4, ứng với kích thước của kiểu float
```

2.6.2.10 Độ ưu tiên của các toán tử

Thứ tự thực hiện các phép toán trong biểu thức phụ thuộc vào độ ưu tiên (Bảng 2.11) và trật tự kết hợp của các toán tử, cụ thể như sau:

- Toán tử có độ ưu tiên cao nhất được thực hiện trước.
- Trong trường hợp toán hạng ở giữa 2 toán tử có cùng độ ưu tiên thì trật tự kết hợp (phải hoặc trái) sẽ quy định thứ tự thực hiện các toán tử.

Bảng 2.11. Độ ưu tiên của các toán tử

| Mức | Toán tử | Trật tự kết hợp |
|-----|-------------------------------|-----------------|
| 1 | () [] -> | → |
| 2 | ! ~ ++ -- * & (type) sizeof() | ← |
| 3 | * / | → |
| 4 | + | → |
| 5 | <<>> | → |
| 6 | < <= > >= | → |
| 7 | == != | → |
| 8 | & | → |
| 9 | ^ | → |
| 10 | | → |
| 11 | && | → |
| 12 | | → |
| 13 | ? : | ← |
| 14 | = += -= *= /= %= . . . | ← |

Ví dụ 2.14. Độ ưu tiên của các toán tử

```
n = 2 + 3 * 5;          //n = (2 + (3 * 5));  
a > 1 && b < 2         //((a > 1) && (b < 2))  
a>0 && b>0 || a<0 && b<0 //((a>0) && (b>0)) || ((a<0) && (b<0))
```

2.6.3 Chuyển đổi kiểu

Chuyển đổi kiểu ngầm định: Trong cùng một biểu thức, nếu các toán hạng không cùng kiểu với nhau thì trước khi tính toán giá trị của biểu thức, chương trình dịch sẽ thực hiện việc chuyển đổi kiểu ngầm định (nếu được) theo nguyên tắc “Kiểu có phạm vi giá trị biểu diễn nhỏ hơn sẽ được chuyển sang kiểu có phạm vi giá trị biểu diễn lớn hơn”. Sơ đồ chuyển đổi kiểu ngầm định:

char → int → long → float → double → long double

Phép ép kiểu: Một số trường hợp bắt buộc phải sử dụng đến toán tử ép kiểu để tạo ra một biểu thức hợp lệ theo cú pháp như sau:

<biểu thức> hoặc (<biểu thức> <biểu thức>)

Ví dụ 2.15. Chuyển đổi kiểu

Phép ép kiểu sau đây cần thực hiện để tạo biểu thức số học hợp lệ

(int) 8.0 % 3 hoặc int (8.0 % 3)

2.7 Câu lệnh

Câu lệnh (*statement, instruction*) là một chỉ thị trực tiếp, hoàn chỉnh nhằm ra lệnh cho máy tính thực hiện một số tác vụ nhất định nào đó. Trong ngôn ngữ lập trình C, các câu lệnh cách nhau bằng dấu chấm phẩy (;). Trình biên dịch bỏ qua các khoảng trắng, dấu tab và ký tự xuống dòng chen giữa lệnh.

Ví dụ 2.16. Các câu lệnh tương đương

x=a+4;

x = a + 4;

x=a

+4;

Có 3 loại câu lệnh trong C:

- *Câu lệnh đơn* chỉ gồm một câu lệnh được kết thúc bởi dấu chấm phẩy. Câu lệnh đơn có dạng: <biến>=<biểu thức>; hay <biểu thức>; hay <lời gọi hàm>;
- *Câu lệnh phức* (*câu lệnh ghép, khối lệnh*) gồm nhiều câu lệnh đơn được bao bởi cặp mốc nhọn ‘{’ và ‘}’.

- Câu lệnh điều khiển được xây dựng từ các cấu trúc điều khiển (rẽ nhánh, lựa chọn, vòng lặp).

Ví dụ 2.17. Các loại câu lệnh trong ngôn ngữ lập trình C

```
x = a + 4; y++; printf ("HELLO, C !"); //cau lenh don
{x = 1; y = 2;} //cau lenh phuc (khoi lenh)
if (a > b) max = a; else max = b; //cau lenh dieu khien
```

2.8 Xuất – nhập

2.8.1 Xuất kết quả

Để xuất kết quả ra màn hình, có thể sử dụng hàm `printf()` trong thư viện nhập xuất chuẩn `stdio` (*standard input/output*) theo cú pháp sau đây:

`printf ("dãy mã quy cách", dãy các biểu thức);`

trong đó:

- *Dãy mã quy cách*: là dãy các định dạng được đặt trong cặp nháy kép “”, bao gồm các chuỗi *văn bản thường* (*literal text*), ký tự *điều khiển* (*escape sequence*) và *đặc tả* (*conversion specifier*) cho kết quả xuất.
- *Dãy các biểu thức*: các kết quả được xuất theo định dạng được đặc tả.
- *Ký tự điều khiển* gồm dấu \ và một ký tự như trong *Bảng 2.12*:

Bảng 2.12. Các ký tự điều khiển trình xuất kết quả trong ngôn ngữ lập trình C

| Ký tự điều khiển | Ý nghĩa |
|------------------|-----------------------------|
| \a | Tiếng chuông |
| \b | Lùi lại một bước |
| \n | Xuống dòng |
| \t | Nhảy đến vị trí tab kế tiếp |
| \\\ | Ghi dấu \ |
| \? | Ghi dấu ? |
| \” | Ghi dấu “ |

Các *đặc tả* (*conversion specifier*) gồm dấu %' và một ký tự xác định dạng của giá trị xuất như trong *Bảng 2.13*.

Bảng 2.13. Các đặc tả chuyển đổi kiểu thành chuỗi trong ngôn ngữ lập trình C

| Đặc tả | Ý nghĩa |
|-------------|---------------------|
| %c | Ký tự |
| %d hoặc %ld | Số nguyên có dấu |
| %f hoặc %lf | Số thực |
| %s | Chuỗi ký tự |
| %u hoặc %lu | Số nguyên không dấu |
| %x hoặc %X | Số nguyên dạng Hexa |
| %o hoặc %O | Số nguyên dạng Octa |
| %e hoặc %E | Số thực dạng mũ |

Ví dụ 2.18. Xuất kết quả ra màn hình

```

printf("Hello "); printf("World"); // Xuất "Hello World"
printf("Hello World");           // Xuất "Hello World"

int a = 10, b = 20;
printf("%d", a);                // Xuất "10"
printf("b = %d", b);            // Xuất "b = 20"
printf("a = %d, b = %d\n", a, b); // Xuất "a = 10, b = 20 <xuống dòng>"
float x = 15.06;
printf(x = "%f", x);           // Xuất "x = 15.060000"
printf("%f", 1.0/3);            // Xuất "0.333333"

```

Định dạng độ dài kết quả xuất:

- Định dạng xuất số nguyên: %nd (độ dài n chữ số)
- Định dạng xuất số thực: %n.kf (độ dài n chữ số với k chữ số phần thập phân)

Ví dụ 2.19. Định dạng độ dài kết quả xuất

```

int a = 1234; float x = 123.456;
printf ("a=%10d", a);           // Xuất "a =      1234" (6 khoảng trắng trước số)
printf ("%10.2f", x);          // Xuất "   123.46" (4 khoảng trắng trước số)
printf ("%,.2f", x);           // Xuất "123.46" (0 khoảng trắng trước số)

```

2.8.2 Nhập dữ liệu

Để nhập dữ liệu từ bàn phím, có thể sử dụng hàm `scanf()` trong thư viện xuất nhập chuẩn `stdio` (*standard input/output*) theo cú pháp sau đây:

`scanf` (“*dãy mã quy cách*”, *dãy các địa chỉ các biến*);

trong đó:

- *Dãy mã quy cách*: là dãy các định dạng được đặt trong cặp nháy kép, bao gồm các chuỗi *văn bản thường* (*literal text*), *ký tự điều khiển* (*escape sequence*) và *đặc tả* (*conversion specifier*) dữ liệu nhập.
- *Dãy địa chỉ các biến*: các dữ liệu nhập được lưu vào các biến theo dạng được nêu trong dãy mã quy cách.

Bảng 2.14. Bảng mã đặc tả dữ liệu nhập

| Đặc tả | Ý nghĩa |
|--------------|--|
| %c | Ký tự |
| %d | Số nguyên có dấu |
| %u | Số nguyên không dấu |
| %hd hoặc %hu | Số nguyên nhỏ (short) có dấu/không dấu |
| %ld hoặc %lu | Số nguyên dài (long) có dấu/không dấu |
| %f hoặc %e | Số thực |
| %lf | Số thực double |
| %s | Chuỗi ký tự (không chứa dấu cách) |

Trong quá trình thực thi, sau khi người sử dụng nhấn ENTER, hàm `scanf` nhận và phân tích chuỗi ký tự từ bộ đệm bàn phím để trích lấy dữ liệu và lưu vào các biến theo nguyên tắc sau đây:

- *Số*: nhảy qua các khoảng trắng (dấu cách), dấu tab, ký tự xuống dòng cho đến khi lấy đủ số chữ số được nêu hoặc gặp ký tự là chữ số, đọc đến khi gặp ký tự không là chữ số.
- *Ký tự*: lấy một ký tự tại vị trí được đặc tả.
- *Chuỗi ký tự*: đọc cho tới khi gặp khoảng trắng hoặc đủ số ký tự theo yêu cầu

- Nhảy qua các dấu cách như trong mã định dạng để tiếp tục đọc dữ liệu.

Ví dụ 2.20. Nhập dữ liệu từ bàn phím

```

int a, b, m, n;
float x, y;
char ch;
printf ("Nhập 2 số m, n: ");
scanf ("%3d %3d",&m,&n);
printf ("Nhập 2 số nguyên a, b: ");
scanf ("%d %d",&a,&b);
printf ("Nhập 2 số thực x, y: ");
scanf ("%f %0.2f",&x,&y);
fflush(stdin); //Phải xóa bộ nhớ đệm khi nhập ký tự hoặc chuỗi ký tự
printf("Nhập 1 ký tự: ");
scanf("%c", &ch);

```

Ngôn ngữ lập trình C cung cấp một số hàm (không bao gồm scanf) để nhập dữ liệu từ bàn phím.

Bảng 2.15. Một số hàm nhập dữ liệu từ bàn phím

| Hàm | Chức năng | Thư viện |
|---------------|---|----------|
| fflush(stdin) | Xoá bộ nhớ đệm | stdio.h |
| getchar() | Đọc một ký tự từ bàn phím | stdio.h |
| getch() | Đọc ký tự từ bàn phím ngay khi gõ vào không đợi ấn phím Enter và không hiển thị ra màn hình | conio.h |
| getche() | Giống getch(), nhưng hiển thị ký tự lên màn hình | conio.h |
| gets() | Đọc một chuỗi ký tự cho đến khi gặp Enter | stdio.h |

2.9 Hướng dẫn sử dụng Dev C++

Dev C++ là một môi trường phát triển tích hợp (IDE) dùng để soạn thảo, lập trình ngôn ngữ C/C++ chạy trên nền hệ điều hành Windows. So với nhiều IDE C/C++ khác, Dev

C++ có ưu điểm là khá nhẹ, dễ cài đặt, dễ sử dụng, tích hợp sẵn các tính năng như highlight, gợi ý code, tự động lưu code, dịch và chạy trực tiếp. Người sử dụng có thể tải Dev C++ về máy để cài đặt tại: <https://sourceforge.net/projects/orwelldvcpp/>. Giao diện tiêu chuẩn của Dev C++ như hình dưới đây:

The screenshot shows the Dev-C++ IDE interface. The title bar reads "F:\BT\Example_1.cpp - Dev-C++ 5.11". The menu bar includes File, Edit, Search, View, Project, Execute, Tools, AStyle, Window, Help. The toolbar has various icons for file operations like Open, Save, Print, and Build. The status bar at the bottom shows "Line: 9 Col: 23 Sel: 0 Lines: 21 Length: 424 Insert Done parsing in 0.094 seconds". The main window displays the code for "Example_1.cpp". The code includes a function to trim whitespace from the start of a string and a main function that prints the original and trimmed strings. The code is annotated with comments explaining its purpose.

```

1 #include <stdio.h>
2 char *ltrim(char *s)
3 {
4     while (*s == ' ' && *s != '\0') s++;
5     return s;
6 }
7 void ltrim_1(char *s1, char *s2) // Loại khoảng trắng trước s3, ghi vào s1
8 {
9     while (*s2 == ' ' && *s2 != '\0') s2++;
10    while (*s2 != '\0') {
11        *s1 = *s2; s1++; s2++;
12    }
13    *s2 = '\0';
14 }
15 main()
16 {
17     char s[] = "      abcd      ";
18     printf("s=%s\n", ltrim(s));
19     ltrim_1(t,s);
20     printf("t=%s\n", t);
21 }

```

Một số thao tác cơ bản trên Dev-C++:

- Tạo file mới:
 - o File → New → Source File
- Mở file có sẵn:
 - o File → Open → chọn file
- Biên dịch chương trình:
 - o Execute → Compile (F9)
- Chạy chương trình:
 - o Execute → Run (F10)
- Dịch và chạy chương trình:
 - o Execute → Compile & Run (F11)
- Gõ rồi:
 - o Execute → Debug (F5)

2.10 Câu hỏi ôn tập

- 1) Trình bày các kiểu dữ liệu cơ sở trong C và cho ví dụ.
- 2) Trình bày khái niệm về biến và cách sử dụng lệnh gán.
- 3) Trình bày khái niệm về biểu thức.
- 4) Tại sao nên sử dụng cặp ngoặc đơn khi viết biểu thức.
- 5) Trình bày cách định dạng xuất, nhập.

2.11 Bài tập thực hành

- 1) Viết chương trình C tính tuổi của một người theo năm sinh và năm hiện tại được nhập từ bàn phím.

```
#include <stdio.h>
int main() {
    int ns, nam, tuoi; // nam sinh, nam hien tai, tuoi
    printf("Nam hien tai: "); scanf("%d", &nam);
    printf("Nam sinh: "); scanf("%d", &ns);
    tuoi = nam - ns;
    printf("Tuoi: %d\n", tuoi);
    return 0;
}
```

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

```
#include <stdio.h>
#define Pi 3.1416
int main()
{
    float r,cv,dt;
    printf("Nhập bán kính: "); scanf("%f",&r);
    cv=2*Pi*r; dt=Pi*r*r;
    printf("Chu vi: %0.2f Dien tich: %0.2f\n",cv,dt);
    return 0;
}
```

- 3) Viết chương trình số tiền phải trả trong tương lai biết số tiền vay hiện tại, lãi vay, và thời hạn vay (tính theo năm). Gợi ý: sử dụng hàm pow(x,y) trong thư viện math.h

```
#include <stdio.h>
#include <math.h>
int main()
{
    float p, r, f;
```

```

int n;
printf("Nhập số tiền vay, lãi suất, số năm: "); scanf("%f %f %n",&p,&r,&n);
f = p*pow(1+r,n);
printf("Phải trả %.2f cho khoản vay %.2f (lãi suất %.2f, thời hạn %d)\n",f,p,r,n);
return 0;
}

```

2.12 Bài tập đề nghị

1) Cho chương trình sau đây:

```

#include <stdio.h>
int main(){
    int a, b, t;
    printf("Bắt đầu: Nhập 2 số a, b: "); scanf("%d%d", &a, &b);
    t = a; a = b; b = t;
    printf("Kết thúc: a = %d, b = %d\n", a, b);
    return 0;
}

```

Cho biết chức năng chương trình và kết quả xuất ra màn hình khi thực hiện chương trình và nhập $a = 3, b = 5$.

2) Cho chương trình sau đây:

```

#include <stdio.h>
int main()
{
    int a, b, c, m1, m2, m;
    printf("Nhập 3 số a, b, c: "); scanf("%d%d%d", &a, &b, &c);
    m1 = a>b?a:b; m2 = a<b?a:b; c = c>0?c:-c; m = (m1 - m2) % c;
    printf("%d %d %d\n", m1, m2, m);
    return 0;
}

```

Cho biết chức năng chương trình và kết quả xuất ra màn hình khi thực hiện chương trình và nhập $a = 3, b = 10, c = -3$.

3) Cho chương trình sau đây:

```

#include <stdio.h>
int main(){
    int a, b, m=1, n = 2;
    printf("Nhập 2 số a, b: "); scanf("%d%d", &a, &b);
    m += a++ + ++b; n *= --m;
}

```

```
printf("%d %d %d %d\n", a, b, m, n);
int th, nm, nh;
printf("Nhập số nm: "); scanf("%d", &nm);
nh = nm%400 ==0 || nm%4 ==0 && nm% 100 != 0;
printf("%d\n", nh);
}
```

Cho biết c kết quả xuất ra màn hình khi thực hiện chương trình khi nhập a = 3, b = 5, nm = 2000, 2100, 2019, 2020. Nếu nm là năm thì th cho biết gì ?

- 4) Viết chương trình nhập điểm thi và hệ số 3 môn Toán, Lý, Hóa của một sinh viên. Tính điểm trung bình của sinh viên đó.
- 5) Viết chương trình tính tổng, hiệu, tích, thương của hai số thực a, b nhập từ bàn phím.
- 6) Viết chương trình nhập số lượng, đơn giá của một loại sản phẩm mua, tính thành tiền, thuế giá trị gia tăng và tổng số tiền phải trả, biết:
 - Thành tiền = số lượng * đơn giá
 - Thuế GTGT = 10% Thành tiền
 - Tổng số tiền phải trả, biết = Thành tiền + Thuế GTGT

CHƯƠNG 3. CÁC CẤU TRÚC ĐIỀU KHIỂN

Trong chương này người học được cung cấp các kiến thức, kỹ năng cơ bản về việc sử dụng cấu trúc điều khiển trong lập trình.

Nội dung:

- Cấu trúc tuần tự
- Cấu trúc rẽ nhánh
- Cấu trúc lặp

3.1 Giới thiệu

Tất cả các chương trình máy tính dù đơn giản hay phức tạp đều được viết bằng cách sử dụng các cấu trúc điều khiển. Có ba loại cấu trúc điều khiển cơ bản là cấu trúc tuần tự (sequence), cấu trúc rẽ nhánh (selection) và cấu trúc lặp (loop). Các cấu trúc này điều khiển thứ tự thực thi các câu lệnh của chương trình:

- **Cấu trúc tuần tự:** thực hiện các lệnh theo thứ tự từ trên xuống dưới.
- **Cấu trúc rẽ nhánh:** dựa vào kết quả của biểu thức điều kiện. Tùy theo sự định trị của biểu thức này mà những lệnh tương ứng sẽ được thực hiện. Các cấu trúc lựa chọn gồm cấu trúc if, switch.
- **Cấu trúc lặp:** lặp lại một hay nhiều lệnh cho đến khi biểu thức điều kiện là sai.
Các cấu trúc lặp gồm for, while, do ... while

Câu lệnh (statement) là một biểu thức kết thúc bởi dấu chấm phẩy (;), câu lệnh phổ biến trong ngôn ngữ lập trình C là lệnh gán. **Khối lệnh** (block) là một hoặc nhiều câu lệnh được bao quanh (giới hạn) bởi một cặp dấu mốc nhọn ({ ... }). Một khi khối lệnh nhận trình điều khiển, tất cả các câu lệnh trong khối đều được thực thi theo trình tự, sau đó khối lệnh trả trình điều khiển cho khối (câu) lệnh kế tiếp.

Ví dụ 3.1. Câu lệnh và khối lệnh

```
int a, b, temp;           //cau lenh (khai bao)
a = 10; b = a + 5;        //cau lenh (gan)
```

```

if (a < 0)          //cau truc re nhanh
{
    //khoi lenh
    temp = a; a = b; b = temp;
}

```

3.2 Cấu trúc rẽ nhánh

3.2.1 Câu lệnh if

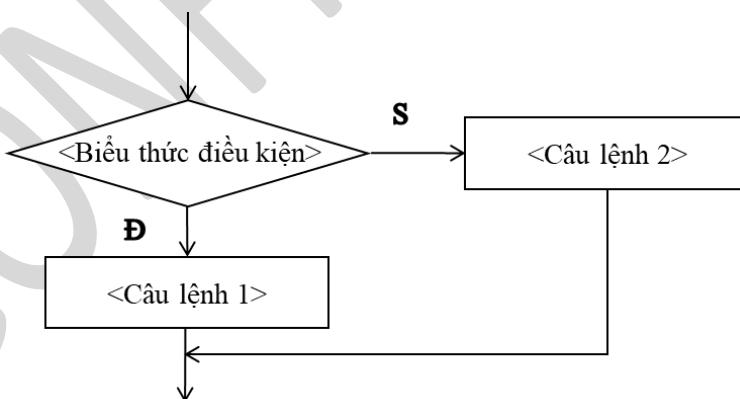
Cấu trúc rẽ nhánh sử dụng câu lệnh ***if*** cho phép lựa chọn thực hiện hành động dựa trên giá trị của biểu thức điều kiện theo cú pháp sau đây:

if (<biểu thức điều kiện>) <câu lệnh 1> [else <câu lệnh2>]

trong đó:

- <biểu thức điều kiện> là biểu thức logic có kết quả trả về là sai (0, false) hoặc đúng (khác 0, true).
- Phần *else* là không bắt buộc phải có; câu lệnh *if* không có phần *else* được gọi là câu lệnh ***if thiếu***, câu lệnh *if* có phần *else* gọi là câu lệnh ***if đầy đủ***
- <câu lệnh 1>, <câu lệnh 2> là câu lệnh hợp lệ bất kỳ; <câu lệnh 1> được thực hiện khi <biểu thức điều kiện> cho kết quả đúng, ngược lại, <câu lệnh 2> được thực hiện

Lưu đồ câu lệnh ***if đầy đủ*** như hình sau:



Trình tự thực hiện câu lệnh ***if đầy đủ***:

Bước 1: Tính giá trị <Biểu thức điều kiện>

Bước 2: Nếu <Biểu thức điều kiện> có giá trị đúng (true hoặc khác 0) thì thực hiện <câu lệnh 1> trong phần *if*, ngược lại (<Biểu thức điều kiện> có giá trị sai (false hoặc bằng 0) thì thực hiện <câu lệnh 2> trong phần *else* (nếu có)).

Bước 3: Chuyển quyền điều khiển sang câu lệnh kế tiếp sau lệnh ***if* đầy đủ**.

Ví dụ 3.2. Thuật toán giải phương trình $ax + b = 0$

```
float a, b, x;  
printf("Nhập hai số a, b: ");  
scanf("%f%f", &a, &b);  
if (a == 0)  
    if (b == 0) printf("Vô số nghiệm.\n");  
    else printf("Vô nghiệm.\n");  
else {  
    x = -b/a;  
    printf("Một nghiệm x = %.2f\n", x);  
}
```

Chú ý:

- Câu lệnh *if* thiếu và câu lệnh *if* đầy đủ là một câu lệnh đơn
- <Biểu thức điều kiện> sau từ khóa *if* phải được viết trong cặp dấu ngoặc đơn
- Trong trường hợp <câu lệnh> là khối lệnh gồm nhiều câu lệnh, cần đặt chúng trong cặp mốc nhọn {<khối lệnh>}

Ví dụ 3.3. Khối lệnh gồm nhiều câu lệnh

So sánh ba đoạn chương trình sau đây:

- Đoạn chương trình 1

```
int a = 0, x = 1, y = 1;  
if (a > 0) {  
    x++; y++;  
}  
printf("x = %d; y = %d\n", x, y); //Xuat "x = 1, y = 1"
```

- Đoạn chương trình 2

```
int a = 0, x = 1, y = 1;  
if (a > 0) x++;  
y++; //x++ nam trong nhanh a>0, y++ nam ngoai.  
printf("x = %d; y = %d\n", x, y); //Xuat "x = 1, y = 2"
```

- Đoạn chương trình 3

```
int a = 0, x = 1, y = 1;  
if (a > 0) x++;  
y++; //cau truc if da ket thuc  
else x += 2; // loi bien dich tai else  
printf("x = %d; y = %d\n", x, y);
```

Nhận xét:

- Đoạn chương trình 1: hai câu lệnh `x++` và `y++` nằm trong cấu trúc `if`, được thực hiện khi $a > 0$.
- Đoạn chương trình 2: câu lệnh `x++` nằm trong cấu trúc `if`, được thực hiện khi $a > 0$; câu lệnh `y++` nằm ngoài cấu trúc `if`, được thực hiện sau đó.
- Đoạn chương trình 3: câu lệnh `x++` nằm trong cấu trúc `if`, được thực hiện khi $a > 0$; câu lệnh `y++` nằm ngoài cấu trúc `if`, được thực hiện sau đó; cấu trúc `if` đã kết thúc (*if thiếu*). Sự xuất hiện của `else` sau đó gây ra lỗi biên dịch.

Chú ý:

- Câu lệnh `if` có thể lồng vào nhau và `else` sẽ tương ứng với `if` gần nó nhất; giữa một cặp `if ... else` chỉ có duy nhất một khối lệnh.
- Nên dùng `else` để loại trừ trường hợp.

Ví dụ 3.4. Xếp loại học lực theo điểm tổng kết (DTK)

So sánh hai đoạn chương trình:

- Sử dụng các câu lệnh `if` thiếu một cách độc lập (1):

```
if (DTK < 5) printf("Hoc luc yeu\n");
if (DTK >= 5 && DTK < 7) printf("Hoc luc trung binh\n");
if (DTK >= 7 && DTK < 8) printf("Hoc luc kha\n");
if (DTK >= 8) printf("Hoc luc yeu\n");
```

- Sử dụng câu lệnh `if` đầy đủ lồng vào nhau (2):

```
if (DTK < 5) printf("Hoc luc yeu\n");
else if (DTK < 7) printf("Hoc luc trung binh\n");
else if (DTK < 8) printf("Hoc luc kha\n");
else printf("Hoc luc yeu\n");
```

Nhận xét:

- Đoạn chương trình (1) yêu cầu thực hiện tất cả 4 câu lệnh `if` thiếu với 4 biểu thức điều kiện (tổng cộng có 6 toán tử quan hệ và 2 toán tử logic)
- Đoạn chương trình (2) yêu cầu thực hiện duy nhất 1 câu lệnh `if` đầy đủ lồng vào nhau, gồm 4 nhánh nhưng câu lệnh sẽ kết thúc khi có một trong số các biểu thức điều kiện có kết quả đúng (true hoặc khác 0). Trường hợp xấu nhất, nhánh `else` cuối

cùng được thực hiện. Như vậy, câu lệnh này thực hiện tối đa 3 toán tử quan hệ và không yêu cầu toán tử logic.

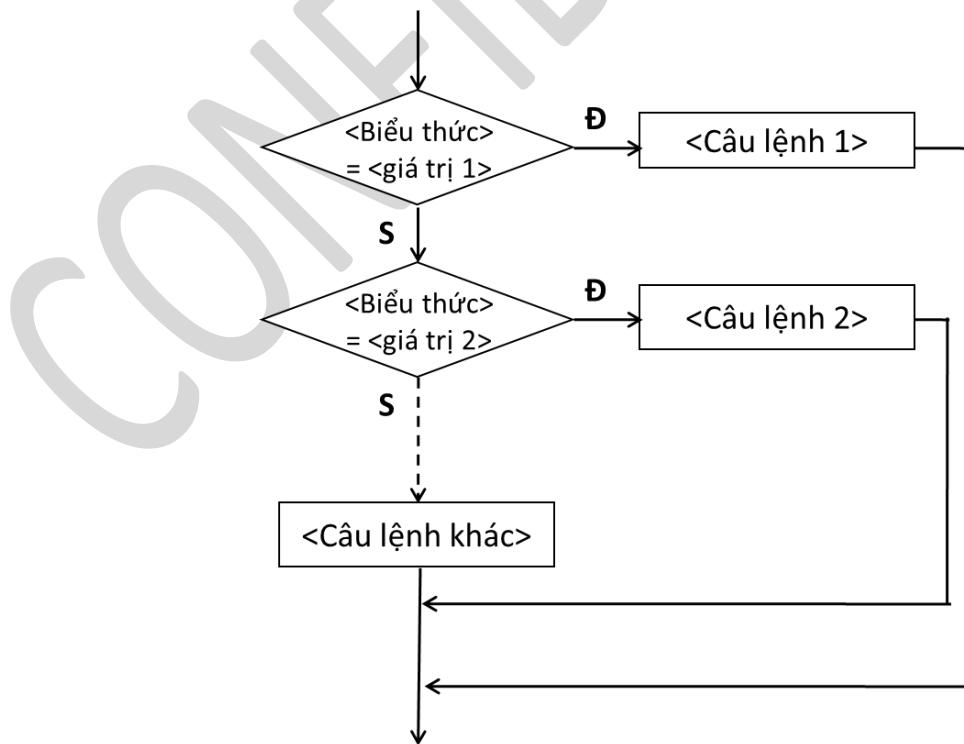
3.2.2 Câu lệnh switch

Một trường hợp đặc biệt của câu lệnh if đầy đủ lồng nhau, khi biểu thức điều kiện của mỗi nhánh là một phép so sánh giữa cùng một biểu thức với các giá trị thuộc kiểu rác như số nguyên, ký tự. Khi đó, câu lệnh switch được sử dụng với cú pháp sau đây:

```
switch (<biểu thức>)
{
    case <giá trị 1>; <câu lệnh 1>; break;
    case <giá trị 2>; <câu lệnh 2>; break;
    ...
    case <giá trị n>; <câu lệnh n>; break;
    default: <câu lệnh khác>; break;
}
```

Với câu lệnh **switch** nói trên, tùy theo giá trị của <biểu thức> trùng khớp với <giá trị k> của nhánh **case** nào thì <câu lệnh k> tương ứng được thực hiện cho đến khi gặp lệnh **break**; thì thoát khỏi câu lệnh **switch**.

Lưu đồ câu lệnh **switch** như hình sau:



Trình tự thực hiện câu lệnh **switch**:

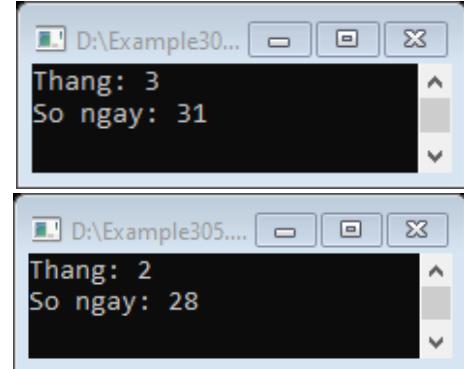
Bước 1: Tính giá trị <biểu thức>

Bước 2: Lần lượt so sánh giá trị của <biểu thức> vừa tính được với các giá trị <giá trị 1>, <giá trị 2>, ... trong mỗi nhánh **case**. Nếu nhánh nào có sự trùng khớp thì <câu lệnh> tương ứng được thực hiện, cho đến khi gặp lệnh **break**. Nếu không có nhánh nào có giá trị trùng khớp với giá trị của <biểu thức> thì <câu lệnh khác> ở nhánh **default** được thực hiện.

Bước 3: Thoát khỏi câu lệnh **switch**, và chuyển quyền điều khiển sang câu lệnh kế tiếp.

Ví dụ 3.5. Số ngày của tháng (năm không nhuận)

```
#include <stdio.h>
int main()
{
    int th; //thang
    printf("Thang: ");
    scanf("%d", &th);
    switch (th) {
        case 2: printf("So ngay: 28\n"); break;
        case 4:
        case 6:
        case 9:
        case 11: printf("So ngay: 30\n"); break;
        default: printf("So ngay: 31\n"); break;
    }
    getchar();
    return 0;
}
```



3.3 Cấu trúc lặp

Cấu trúc lặp cho phép lặp lại nhiều lần một câu lệnh hay một khối lệnh cho đến khi biểu thức điều kiện còn thỏa.

Các loại cấu trúc lặp trong C:

- Cấu trúc lặp *for*
- Cấu trúc lặp *while*
- Cấu trúc lặp *do ... while*

3.3.1 Cấu trúc lặp for

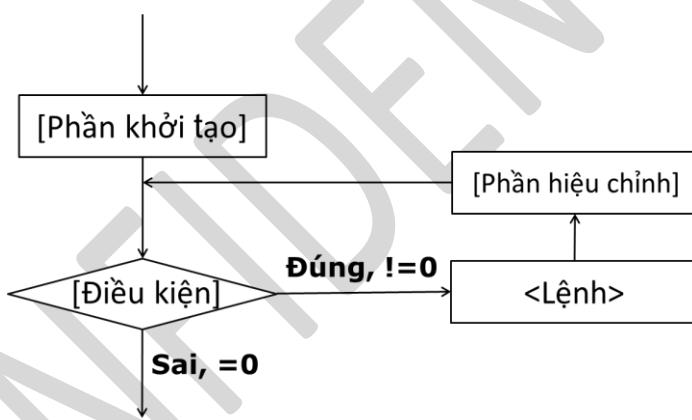
Cấu trúc lặp for cho phép lặp lại một câu lệnh hay khối lệnh với số lần lặp được xác định trước với cú pháp sau đây:

for ([<phản khởi tạo>] ; [<điều kiện>] ; [<phản hiệu chỉnh>]) <lệnh>;

trong đó:

- **<phản khởi tạo>**: một hay nhiều biểu thức gán (được phân cách bởi dấu phẩy) có nhiệm vụ khởi tạo giá trị ban đầu cho các biến đếm
- **<điều kiện>**: biểu thức logic, cho kết quả đúng (true, khác 0), sai (false, bằng 0)
- **<phản hiệu chỉnh>**: một hay nhiều biểu thức gán (được phân cách bởi dấu) có nhiệm vụ thay đổi giá trị của các biến đếm
- **<lệnh>**: câu lệnh đơn, khối lệnh, hoặc câu lệnh điều khiển

Lưu đồ câu lệnh **for** như hình sau:



Trình tự thực hiện câu lệnh **for**:

Bước 1: Thực hiện **<phản khởi tạo>** (nếu có)

Bước 2: Tính giá trị của **<điều kiện>**

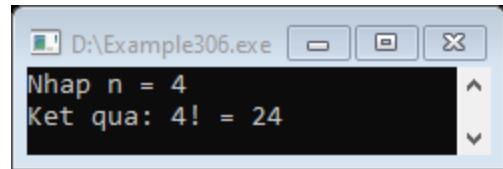
Bước 3: Nếu **<điều kiện>** đúng (true, khác 0) thì thực hiện **<lệnh>**, sau đó thực hiện **<phản hiệu chỉnh>** (nếu có) và quay về **Bước 2**, ngược lại, nếu **<điều kiện>** sai (false, bằng 0) thì chuyển sang **Bước 4**

Bước 4: Thoát khỏi cấu trúc lặp for, và chuyển quyền điều khiển sang câu lệnh kế tiếp sau lệnh for

Ví dụ 3.6. Tính giai thừa của n nhập từ bàn phím

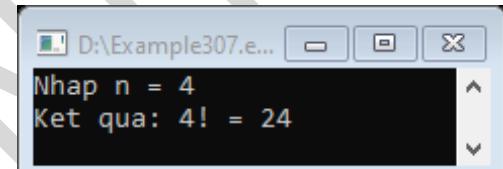
/*su dung cau truc for day du*/

```
#include <stdio.h>
int main(){
    int n;
    printf("Nhập n = "); scanf("%d", &n);
    long gai_thua = 1;
    for (int i=2; i<=n; i++) gai_thua *= i;
    printf("Kết quả: %d! = %ld\n", n, gai_thua);
    return 0;
}
```



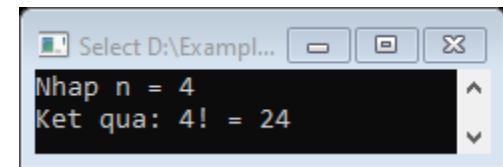
Ví dụ 3.7. Tính giai thừa của số n nhập từ bàn phím

```
/*sử dụng cấu trúc for thiếu phần khởi tạo*/
#include <stdio.h>
int main(){
    int n, i=2;
    printf("Nhập n = "); scanf("%d", &n);
    long gai_thua = 1;
    for (; i<=n; i++) gai_thua *= i;
    printf("Kết quả: %d! = %ld\n", n, gai_thua);
    return 0;
}
```



Ví dụ 3.8. Tính giai thừa của n nhập từ bàn phím

```
/*sử dụng cấu trúc for thiếu phần điều kiện*/
#include <stdio.h>
int main(){
    int n; long gai_thua = 1;
    printf("Nhập n = "); scanf("%d", &n);
    for (int i=2; ; i++) {
        if (i > n) break;
        gai_thua *= i;
    }
    printf("Kết quả: %d! = %ld\n", n, gai_thua);
    return 0;
}
```



3.3.2 Cấu trúc lặp while

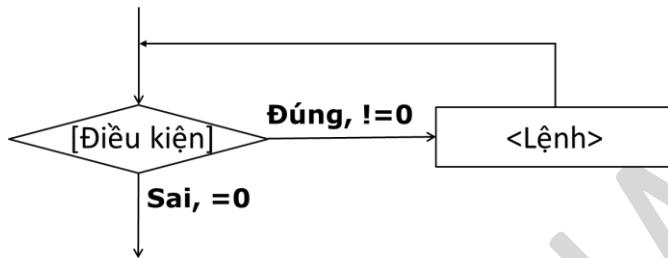
Cấu trúc lặp **while** cho phép xem xét điều kiện lặp, chỉ khi điều kiện này đúng mới thực hiện hành động lặp (câu lệnh hay khối lệnh). Cú pháp của cấu trúc lặp while:

while (<điều kiện>) <lệnh>;

trong đó:

- **<điều kiện>**: biểu thức logic cho kết quả đúng (true, khác 0), sai (false, bằng 0)
- **<lệnh>**: câu lệnh đơn, khối lệnh, hoặc câu lệnh điều khiển

Lưu đồ câu lệnh **while** như hình sau:



Trình tự thực hiện câu lệnh **while**:

Bước 1: Tính giá trị của điều kiện (đúng hoặc sai)

Bước 2: Nếu giá trị của **<điều kiện>** là đúng (true, khác 0), thực hiện **<lệnh>**, sau đó quay về **Bước 1**, ngược lại, chuyển sang **Bước 3**.

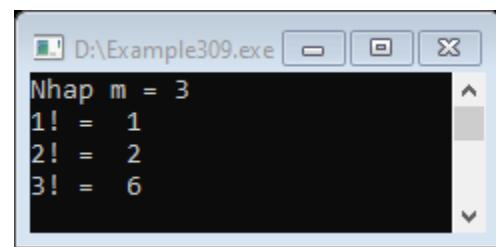
Bước 3: Thoát khỏi vòng lặp, và chuyển quyền điều khiển sang câu lệnh kế tiếp sau lệnh while.

Chú ý:

- Do **<điều kiện>** được kiểm tra trước, nên xảy ra trường hợp nội dung lặp (**<lệnh>**) của cấu trúc **while** không được thực hiện dù chỉ một lần.
- Để bảo đảm cho cấu trúc lặp **while** kết thúc sau một số lần lặp (không xác định), trong nội dung lặp phải có ít nhất một câu lệnh làm thay đổi giá trị **<điều kiện>**.

Ví dụ 3.9. Xuất m giá trị giai thừa đầu tiên với m nhập từ bàn phím

```
#include <stdio.h>
int main(){
    int m, i = 1; long giai_thua = 1;
    printf("Nhập m = "); scanf("%d", &m);
    while (giai_thua <= m) {
        giai_thua *= i;
        printf("%d! = %ld\n", i, giai_thua);
        i++;
    }
    return 0;
}
```



3.3.3 Cấu trúc do ... while

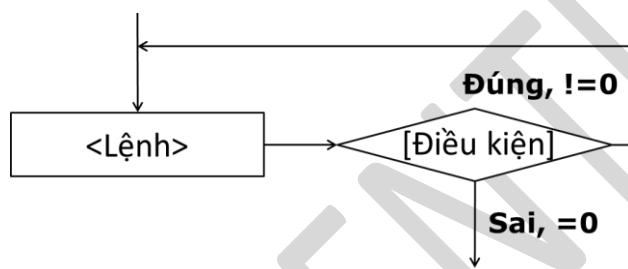
Cấu trúc lặp do ... while cho phép lặp lại một hành động (câu lệnh hay khối lệnh) trong khi điều kiện lặp đúng với cú pháp sau đây:

do <lệnh>; while (<điều kiện>);

trong đó:

- <điều kiện>: biểu thức logic cho kết quả đúng (true, khác 0), sai (false, bằng 0)
- <lệnh>: câu lệnh đơn, khối lệnh, hoặc câu lệnh điều khiển

Lưu đồ câu lệnh **do ... while** như hình sau:



Trình tự thực hiện câu lệnh **do ... while**:

Bước 1: Thực hiện hành động lặp <lệnh>

Bước 2: Tính giá trị của <điều kiện>, nếu là đúng (true, khác 0), quay về **Bước 1**, ngược lại, chuyển sang **Bước 3**.

Bước 3: Thoát khỏi vòng lặp, và chuyển quyền điều khiển sang câu lệnh kế tiếp sau lệnh do ... while.

Chú ý:

- Do <lệnh> được thực hiện trước rồi mới đánh giá <điều kiện>, nên nội dung lặp (<lệnh>) của cấu trúc **do ... while** được thực hiện ít nhất một lần.
- Để bảo đảm cho cấu trúc lặp **do ... while** kết thúc sau một số lần lặp (không xác định trước), trong nội dung lặp phải có ít nhất một câu lệnh làm thay đổi giá trị <điều kiện>.

Ví dụ 3.10. Nhập một số nguyên dương (nhập lại khi nhập số không dương)

```
#include <stdio.h>
int main() {
    int n;
```

```

do {
    printf("Hay nhap mot so >0: ");
    scanf("%d", &n);
} while (n<=0);
printf("So hop le n = %d\n", n);
return 0;
}

```

```

D:\Example310...
Hay nhap mot so >0: -1
Hay nhap mot so >0: -2
Hay nhap mot so >0: 0
Hay nhap mot so >0: 3
So hop le n = 3

```

3.4 Câu lệnh break và continue, goto

3.4.1 Câu lệnh break

Câu lệnh ***break*** cho phép thoát khỏi cấu trúc rẽ nhánh ***switch*** hoặc cấu trúc lặp trực tiếp chứa nó. Nói cách khác, trình thực thi sẽ bỏ qua các câu lệnh sau lệnh ***break***. Cú pháp của câu lệnh ***break***:

break;

Ví dụ 3.11. Xuất m giá trị giai thừa đầu tiên với m nhập từ bàn phím

```

/*su dung lenh break de thoat khoi cau truc lap while*/
#include <stdio.h>
int main(){
    int m, i = 1; long giai_thua = 1;
    printf("Nhập m = "); scanf("%d", &m);
    while (1) {
        giai_thua *= i;
        printf("%d! = %ld\n", i, giai_thua);
        i++;
        if (giai_thua > m) break;
    }
    return 0;
}

```

```

D:\Ex...
Nhập m = 3
1! = 1
2! = 2
3! = 6

```

3.4.2 Câu lệnh continue

Câu lệnh ***continue*** cho phép bỏ qua các lệnh còn lại trong nội dung lặp lần này để bắt đầu một lần lặp mới. Cú pháp của câu lệnh ***continue***:

continue;

Ví dụ 3.12. Tính tổng của các số chẵn nhỏ hơn m nhập từ bàn phím

```

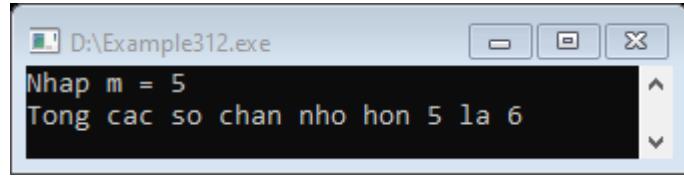
/*su dung lenh continue de bo qua cac cau lenh khong duoc thuc hien*/
#include <stdio.h>
int main(){

```

```

int m, i = 1; long tong = 0;
printf("Nhập m = ");
scanf("%d", &m);
for (i = 1; i < m; i++){
    if (i % 2 != 0) continue;
    tong += i;
}
printf("Tổng các số chẵn nhỏ hơn %d là %d\n", m, tong);
return 0;
}

```



3.4.3 Câu lệnh goto

Câu lệnh **continue** cho phép chuyển quyền điều khiển đến câu lệnh đặt tại vị trí <nhãn> và thực hiện lệnh này. Cú pháp của câu lệnh **goto**:

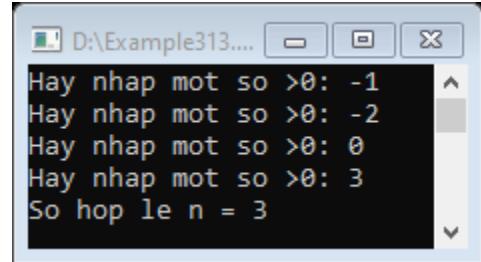
goto <nhãn>;

Ví dụ 3.13. Nhập một số nguyên dương (nhập lại khi nhập số không dương)

```

/*sử dụng lệnh goto thay thế cho câu trúc lặp*/
#include <stdio.h>
int main() {
    int n;
    Loop:
    printf("Hay nhập một số >0: ");
    scanf("%d", &n);
    if (n <= 0) goto Loop;
    printf("So hop le n = %d\n", n);
    return 0;
}

```



Chú ý:

- Câu lệnh **goto** làm cho chương trình khó theo dõi, khó kiểm soát, làm mất tính cấu trúc của chương trình.
- Câu lệnh **goto** ít được dùng và có thể được thay thế bằng các cấu trúc điều khiển.

3.5 Câu hỏi ôn tập

- 1) Trình bày cú pháp, ý nghĩa, lưu đồ của các cấu trúc điều khiển trong ngôn ngữ C.
- 2) Hãy rút gọn đoạn chương trình sau:

```
if ( a > b ) {
```

```
        b = x; a = 1;  
    }  
else {  
    b = x; a = 0;  
}
```

3) Hãy rút gọn đoạn chương trình sau:

```
if ( a > b ) {  
    a -= b; c = x;  
}  
else {  
    a += b; c = x;  
}
```

4) Hãy rút gọn đoạn chương trình sau:

```
if ( a > b ) {  
    c = a;  
}  
else {  
    c = a; a = 0;  
}
```

5) Hãy rút gọn đoạn chương trình sau:

```
if ( a > b ) {  
    a = 1; c = x;  
}  
else {  
    b = 2; c = x;  
}
```

6) Hãy chuyển đổi câu lệnh for thành câu lệnh while:

```
for ( int i = 0; i < 100; i++ ) printf("%d\n", i);
```

7) Hãy chuyển đổi câu lệnh for thành câu lệnh while:

```
for ( int i = 2000; i > 0; i /= 2 ) printf("%d\n", i);
```

8) Hãy chuyển đổi câu lệnh for thành câu lệnh while:

```
for ( int i = 0; i < 100; i++ )  
    for ( int j = 0; j < 200; j++ ) printf("%d\n", i * j);
```

9) Hãy chuyển đổi câu lệnh while thành câu lệnh for:

```
int i = 0;
```

```
while ( i < 100 ) {  
    printf("%d\n", i); i++;  
}
```

10) Hãy chuyển đổi câu lệnh while thành câu lệnh for:

```
int count = 0;  
while ( count < 100 ) {  
    count++; printf("%d\n", count );  
}
```

11) Hãy chuyển đổi câu lệnh while thành câu lệnh for:

```
int x = 10, i = x, y = 0, j = 0;  
while ( i > j ) {  
    y += i; i--;  
}
```

12) Hãy chuyển đổi câu lệnh while thành câu lệnh do ... while:

```
int x = 10, i = x, y = 0, j = 0;  
while ( i > j ) {  
    y += i; i--;  
}
```

13) Hãy cho biết kết quả xuất ra màn hình của các biến x, y. Giải thích.

```
int x = 0, y = 1;  
for ( int i = 0; i < 5; i++ )  
{  
    for ( int j = 0; j < 5; j++ ) if ( ( i + j ) % 2 == 0 ) x++;  
    y += x;  
}
```

14) Hãy cho biết kết quả xuất ra màn hình của các biến x, y. Giải thích.

```
int x = 0, y = 1;  
for ( int i = 0; i < 5; i++ )  
    for ( int j = 0; j < 5; j++ )  
    {  
        if ( ( i + j ) % 2 == 0 ) x++;  
        y += x;  
    }
```

15) Hãy cho biết kết quả xuất ra màn hình của các biến x, y. Giải thích.

```
int x = 0, y = 1;  
for ( int i = 0; i < 5; i++ ) {  
    for ( int j = 0; j < 5; j++ )
```

```

        if ( (x + y) % 2 == 0 ) x++;
        y += x;
    }

```

16) Hãy cho biết kết quả xuất ra màn hình của các biến x, y. Giải thích.

```

int x = 0, y = 1;
for ( int i = 0; i < 5; i++ )
    for ( int j = 0; j < 5; j++ )
        if ( ( i + j ) % 2 == 0 ) x++;
        else y += x;

```

17) Hãy cho biết kết quả xuất ra màn hình. Giải thích.

```

int i = 0, x = 0;
do {
    if ( x % 5 == 0 ){
        x++;
        printf("%d\n", x);
    }
    i++;
} while ( i < 20 );

```

18) Hãy cho biết kết quả xuất ra màn hình. Giải thích.

```

int i = 0, x = 0;
while ( i < 20 ) {
    if ( x % 5 == 0 ){
        x += i;
        printf("%d\n", x);
    }
    i++;
}

```

19) Hãy cho biết kết quả xuất ra màn hình. Giải thích.

```

int i, j, k, x = 0;
for ( i = 1; i < 5; i++ )
    for ( j = 0; j < i; j++ ){
        switch ( i + j - 1 ){
            case -1:
            case 0: x += 1; break;
            case 1:
            case 2:
            case 3: x += 2; break;
            default: x += 3;
        }
    }

```

```
        printf("%d\n", x);
    }
```

3.6 Bài tập thực hành

3.6.1 Cấu trúc rẽ nhánh if ... else

- 1) Nhập một số nguyên bất kỳ, cho biết đó là số 0, số âm hay số dương, số chẵn hay số lẻ.

```
#include <stdio.h>
int main()
{
    int so;
    printf("Nhập số: "); scanf("%d", &so);
    if (so==0) printf("So 0");
    else {
        if (so<0) printf("So âm");
        else printf("So dương");
        if (so%2==0) printf(", chẵn");
        else printf(", lẻ");
    }
    getchar();
    return 0;
}
```

- 2) Giải phương trình bậc hai $ax^2 + bx + c = 0$.

```
#include <stdio.h>
#include <math.h>
int main(){
    float a,b, c, delta;
    printf("Nhập các số a, b, c:"); scanf("%f %f %f", &a, &b, &c);
    if (a==0) // giải pt bậc nhất bx + c = 0
        if (b==0)
            if (c==0) printf("Phương trình vô định\n");
            else printf("Phương trình vô nghiệm\n");
        else printf("Một nghiệm x = %.2f\n", -c/b);
    else { // Phương trình bậc 2
        delta = b*b - 4*a*c;
        printf("delta=% .2f\n", delta);
        if (delta<0) printf("Phương trình vô nghiệm");
        else if (delta==0) printf("Nghiệm kép x = %.2f\n", -b/2/a);
        else printf("Hai nghiệm: x1 = %.2f, x2 = %.2f\n",
                   (-b+sqrt(delta))/2/a, (-b-sqrt(delta))/2/a );
    }
    getchar();
}
```

```
        return 0;  
    }  
}
```

- 3) Nhập 4 số nguyên a, b, c và d. Tìm số nhỏ nhất, số lớn nhất trong 4 số trên.

```
#include <stdio.h>  
int main(){  
    int a,b, c, d, max, min;  
    printf("Nhập 4 số a, b, c, d: ");  
    scanf("%d %d %d %d", &a, &b, &c, &d);  
    if (a>b) { max = a; min = b;} else { max = b; min = a;}  
    if (c>max) max = c; else if (c<min) min = c;  
    if (d>max) max = d; else if (d<min) min = d;  
    printf("Max = %d, min = %d\n", max, min );  
    getchar();  
    return 0;  
}
```

- 4) Tính tiền đi taxi từ số km nhập vào. Biết rằng:

1 km đầu giá 15000đ

Từ km thứ 2 đến km thứ 5 giá 13500đ

Từ km thứ 6 trở đi giá 11000đ

Nếu trên 120 km được giảm 10% tổng tiền.

```
#include <stdio.h>  
#define d1 1  
#define d2 5  
#define d3 120  
int main(){  
    int d; long st;  
    printf("Nhập số km:"); scanf("%d", &d);  
    if (d<=d1) st = 15000;  
    else if (d<=d2) st = 15000 + (d - d1) * 13500;  
    else st = 15000 + (d2 - d1) * 13500 + (d - d2) * 10000;  
    if (d>=d3) st = st * 0.9;  
    printf("So tien: %d dong\n", st);  
    getchar();  
    return 0;  
}
```

3.6.2 Cấu trúc rẽ nhánh switch ... case

- 1) Cho biết số ngày của một tháng trong năm bất kỳ (nhuận hoặc không nhuận với năm nhuận là năm chia hết cho 400 hoặc chia hết cho 4 nhưng không chia hết cho 100).

```
#include <stdio.h>
int main(){
    int th, n, sn;
    printf("Nhập tháng: "); scanf("%d", &th);
    switch (th) {
        case 2:
            printf("Nhập năm: "); scanf("%d", &n);
            if (n%400==0 || n%4 ==0 && n%100 !=0) sn = 29;
            else sn = 28;
            break;
        case 4:
        case 6:
        case 9:
        case 11: sn = 30; break;
        default: sn = 31;
    }
    printf("Số ngày: %d\n", sn);
    getch();
}
```

- 2) Cho trước 2 số nguyên và các kí hiệu phép toán sau: +, -, *, /, %. Viết chương trình tính và xuất kết quả phép tính ra màn hình (có lưu ý đến lỗi chia cho giá trị 0).

```
#include <stdio.h>
int main() {
    int a, b, kq, valid = 2;
    char c;
    printf("Nhập 2 số a, b: "); scanf("%d %d", &a, &b);
    printf("Phép toán: "); fflush(stdin); c = getchar();
    switch (c) {
        case '+': kq = a+b; break;
        case '-': kq = a-b; break;
        case '*': kq = a*b; break;
        case '/': if (b==0) valid=1; else kq = a/b; break;
        case '%': if (b==0) valid=1; else kq = a%b; break;
        default: valid = 0;
    }
    if (valid==0) printf("Phép toán không hợp lệ\n");
    else if (valid==1) printf("Lỗi phép chia cho 0\n");
}
```

```

        else printf("Ket qua: %d %c %d = %d\n", a, c, b, kq);
        getchar();
        return 0;
    }

```

3.6.3 Cấu trúc lặp

- 1) Cho 2 số thực x , y , số nguyên $n < 1000$. Viết chương trình tính và in giá trị biểu thức:

$$F = 1 + \frac{x}{y} + \frac{x^2}{y^2} + \cdots + \frac{x^n}{y^n}$$

```

#include <stdio.h>
int main() {
    float x, y, F=1, t = 1;
    int n;
    printf("Nhập 2 số x, y: "); scanf("%f %f", &x, &y);
    printf("Nhập số nguyên n: "); scanf("%d", &n);
    for (int i=1; i<=n; i++) {
        t = t*x/y; F+=t;
    }
    printf("Kết quả: F = %f\n", F);
    getchar();
    return 0;
}

```

- 2) Cho số thực x , số nguyên $n < 1000$. Viết chương trình tính và in giá trị biểu thức

$$F = 1 + \frac{x}{1!} - \frac{x^2}{2!} - \cdots - (-1)^n \cdot \frac{x^n}{n!}$$

```

#include <stdio.h>
int main() {
    float x, F=1, t = -1;
    int n;
    printf("Nhập số thực x:"); scanf("%f", &x);
    printf("Nhập số nguyên n: "); scanf("%d", &n);
    for (int i=1; i<=n; i++) {
        t = -t*x/i; F+=t;
    }
    printf("Kết quả: F = %f\n", F);
    getchar();
    return 0;
}

```

- 3) Viết chương trình nhập vào 2 số nguyên m, n ($m \leq n \leq 5000$), hiển thị ra màn hình tất cả các số trong khoảng $[m..n]$ thỏa tính chất “ $Tổng\ các\ chữ\ số = Tích\ các\ chữ\ số$ ” (Ví dụ: 22, 123, ...)

```
#include <stdio.h>
int main() {
    int m, n, tong, tich, so, k;
    printf("Nhập 2 số nguyên m, n (m<=n): ");
    scanf("%d %d", &m, &n);
    for (int i=m; i<=n; i++) {
        tong = 0, tich = 1, so = i;
        while (so>0) {
            k = so%10; so = so/10;
            tong += k; tich*=k;
        }
        if (tong == tich) printf("%d\n", i);
    }
    getchar();
    return 0;
}
```

- 4) Viết chương trình nhập vào số nguyên $n > 0$ và hiển thị ra màn hình các số nguyên tố nhỏ hơn n .

```
#include <stdio.h>
#include <math.h>
int main()
{
    int n, snt; float q;
    printf("Nhập số nguyên n : ");
    scanf("%d", &n);
    for (int i=2; i<=n; i++) { //xet so nguyen i
        snt = 1; q = sqrt(i);
        for (int j=2; j<=q; j++) //tim j là uoc so cua so nguyen i
            if (i%j ==0) { snt = 0; break; }
        if (snt==1) printf("%d\n", i); //so i la nguyen to khi khong co uoc so
    }
    getchar();
    return 0;
}
```

- 5) Viết chương trình kiểm tra xem số nguyên n nhập vào có phải là số Fibonacci?

```
#include <stdio.h>
#include <math.h>
int main()
```

```

{
    int n, m, a, b=1, c=1;
    printf("Nhập số nguyên dương n : "); scanf("%d", &n);
    if (n<2) printf("Không phải số Fibonacy\n");
    else {
        while (c<n) {
            m = b+c; a=b; b=c; c=m;
        }
        if (n==c) printf("Số Fibonacy\n");
        else printf("Không phải số Fibonacy\n");
    }
    getchar();
    return 0;
}

```

- 6) Viết chương trình phân tích số nguyên dương n thành tích các thừa số nguyên tố. Ví dụ: $19 = 1 * 19$; $20 = 1 * 2 * 2 * 2 * 3 * 5$

```

#include <stdio.h>
#include <math.h>
int main()
{
    int n, k = 2;
    printf("Nhập số nguyên dương n : "); scanf("%d", &n);
    printf("%d = 1 ", n);
    while (n>1) {
        while (n % k==0) {
            printf(" * %d", k); n = n/k;
        }
        k++;
    }
    printf("\n");
    getchar();
    return 0;
}

```

3.7 Bài tập đề nghị

- 1) Cho chương trình sau đây:

```

#include <stdio.h>
int main() {
    int n, s;
    printf("Nhập số n: "); scanf("%d", &n);

```

```

while (1) {
    s = 0;
    while (n>0) {
        s+=n%10; n/=10;
    }
    if (s<10) break;
    n = s;
}
printf("%d\n", s);
return 0;
}

```

Cho biết chức năng chương trình và kết quả xuất ra màn hình khi thực hiện chương trình và nhập $n = 123456$.

2) Cho chương trình sau đây:

```

#include <stdio.h>
int main() {
    int n, m = 1, s = 0, k = 0, d = 1;
    printf("Nhập số n: "); scanf("%d", &n);
    for (int i = 1; i<=n; i++) {
        printf("%d ", m*d);
        s+=d*m; d = -d; k++; m+=k;
    }
    printf("\n%d\n", s);
    return 0;
}

```

Cho biết chức năng chương trình và kết quả xuất ra màn hình khi thực hiện chương trình khi nhập $n = 6$.

- 3) Viết chương trình nhập một ký tự bất kỳ, cho biết đó là chữ cái, chữ số hay các dấu, nếu là chữ thường thì đổi sang chữ hoa và xuất ra màn hình.
- 4) Viết chương trình giải phương trình trùng phương $a^4 + b^2 + c = 0$.
- 5) Viết chương trình nhập 3 số nguyên dương đại diện cho 3 cạnh của 1 tam giác. Hiển thị ra màn hình cho biết đó là tam giác đều, tam giác cân, tam giác vuông, tam giác vuông cân, tam giác thường, hay không.
- 6) Cho 2 số thực x, y , số nguyên $n < 1000$. Viết chương trình tính và in giá trị biểu thức:

$$F = 1 + \frac{x}{y} - \frac{x^2}{y^2} + \cdots - (-1)^n \cdot \frac{x^n}{y^n}$$

7) Cho số nguyên $n < 1000$, viết chương trình tính và in giá trị biểu thức

$$F = -1 + \frac{2^2}{2!} - \frac{3^3}{3!} + \cdots + (-1)^n \cdot \frac{n^n}{n!}$$

8) Cho 2 số nguyên $m, n < 1000$, viết chương trình tính và in giá trị biểu thức

$$F = \sum_{i=1}^m \sum_{j=1}^n i * j$$

9) Cho số nguyên $n < 1000$, viết chương trình tính và in giá trị biểu thức

$$F = \sum_{i=1}^n \sum_{j=i}^n i * j$$

10) Viết chương trình in ra các ký tự và mã của chúng theo bảng mã ASCII (Chú ý chỉ in ra các ký tự có mã ASCII > 32).

11) Viết chương trình đếm số chữ số của số tự nhiên n nhập vào từ bàn phím.

12) Viết chương trình in các số Armstrong (là số có tổng lập phương của các chữ số bằng chính nó).

13) Cho số tự nhiên n ($n < 2000$). Viết chương trình cho hiện ra màn hình các cách phân tích số thành dạng tổng lập phương của 2 số tự nhiên x, y ($n = x^3 + y^3$).

14) Số hoàn hảo (*perfect number*) là số tự nhiên có tổng các ước số (kể cả 1) bằng chính nó. VD: số tự nhiên 28 là số hoàn hảo. Viết chương trình hiển thị ra màn hình tất cả các số hoàn hảo trong khoảng $[m, n]$ với $1 \leq m \leq n \leq 1000$.

15) Viết chương trình tìm số tự nhiên bé nhất có ít nhất 2 cách biểu diễn nó thành tổng của 4 số chính phương đôi một khác nhau (Ví dụ, $78 = 9 + 16 + 9 + 4 = 36 + 25 + 16 + 1$).

16) Cho số nguyên n . Viết chương trình tính tổng của các chữ số của số n .

17) Cho số nguyên n . Viết chương trình tính và in trị đảo ngược của số n .

18) Cho số nguyên n . Hãy cho biết:

n có bao nhiêu chữ số ?

Tính tổng các chữ số của n .

Tính tích các chữ số của n .

Tìm chữ số đầu tiên của n .

- 19) Viết chương trình nhập hai số tự nhiên a, b và ghi ra màn hình ước số chung lớn nhất, bội số chung nhỏ nhất của chúng.
- 20) Viết chương trình in ra màn hình bảng cửu chương theo chiều dọc và theo chiều ngang.
- 21) Hai số nguyên được gọi là *nguyên tố cùng nhau* nếu không cùng chia hết cho 1 số nguyên nào khác 1. Viết chương trình kiểm tra xem 2 số nguyên a, b có nguyên tố cùng nhau hay không.
- 22) Viết chương trình nhập số nguyên $n > 0$ và ghi ra màn hình n số nguyên tố đầu tiên.
- 23) Viết chương trình nhập số nguyên $n > 0$ và ghi ra màn hình các số Fibonacci nhỏ hơn n .
- 24) Viết chương trình nhập số nguyên $n > 0$ và hiển thị ra màn hình n số Fibonacci đầu tiên.
- 25) Viết chương trình lân lượt nhập các số nguyên cho đến khi nhập số 0 thì dừng. Tính:

Số lượng các số đã nhập thỏa tiêu chuẩn “P” (Ví dụ: “là số âm”, “là số lẻ”, “là số nguyên tố”, “là số Fibonacci”, ...)

Tổng các số đã nhập thỏa tiêu chuẩn “P”.

Trung bình tổng các số đã nhập thỏa tiêu chuẩn “P”.

CHƯƠNG 4. HÀM

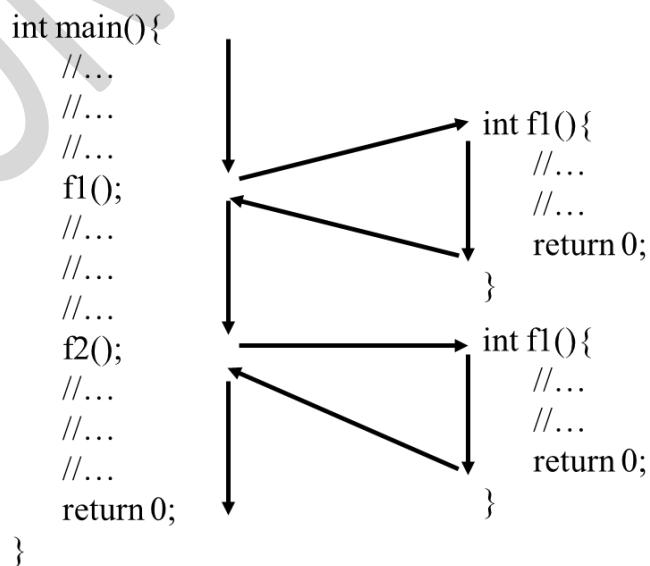
Trong chương này người học được cung cấp các kiến thức, kỹ năng cơ bản về việc sử dụng hàm trong lập trình.

Nội dung:

- Khái niệm
- Cách xây dựng hàm
- Tầm tác dụng của biến
- Truyền tham số cho hàm
- Một số hàm thông dụng

4.1 Giới thiệu hàm

Một hàm là một khối lệnh được đặt tên và có thể được thực thi từ nhiều điểm khác nhau trong chương trình khi được gọi. Hàm nhóm một số cấu trúc điều khiển thành một khối và được đặt tên một cách tường minh. Hàm còn được gọi là chương trình con. Hàm có thể được gọi từ nhiều vị trí khác nhau trong chương trình chính và trong các hàm khác. Hàm có thể có giá trị trả về hoặc không (khi đó nó được gọi là thủ tục). Khi hàm được gọi, khối lệnh tương ứng của hàm được thực thi. Sau khi thực hiện xong, quyền điều khiển được trả về cho chương trình gọi.



Ngôn ngữ lập trình C hỗ trợ hai loại hàm: hàm thư viện và hàm do người dùng định nghĩa. Hàm thư viện là hàm có sẵn, được tích hợp trong các thư viện hàm có sẵn. Việc khai báo thư viện bằng từ khóa #include <tên thư viện> là bắt buộc để có thể sử dụng hàm thư viện. *Ví dụ 4.1* thể hiện hàm do người dùng định nghĩa, trong đó có sử dụng hàm thư viện.

Ví dụ 4.1. Hàm thư viện và hàm người dùng định nghĩa

```
#include <stdio.h>          /*Khai bao thu vien*/
int maximize(int a, int b){ /*Ham nguoi dung dinh nghia*/
    int max = a;
    if (a < b) max = b;
    return max;
}
int main(){                  /*Ham chuong trinh chinh*/
    int x, y, max;
    printf("Nhập hai số tự ban phím: "); /*Gọi ham thu vien*/
    scanf("%d %d", &x, &y);
    max = maximize(x,y);           /*Gọi ham nguoi dung */
    printf("Giá trị lớn nhất của %d và %d là %d",x , y, max);
    getchar();
    return 0;
}
```

4.2 Dạng tổng quát của hàm

Định nghĩa hàm có dạng tổng quát như sau:

```
returnType functionName (parameterList){
    body_of_the_function
}
```

returnType: Kiểu dữ liệu của giá trị trả về bởi hàm, returnType là **void** nếu hàm không có giá trị trả về.

functionName: Tên hàm, quy tắc đặt tên giống như đối với biến.

parameterList: Danh sách tham số, thể hiện theo cặp **kiểu_dữ_liệu tên_tham_số**.

Các tham số cách nhau bởi dấu phẩy (,). Danh sách này rỗng, ký hiệu (), khi hàm không có tham số.

body_of_the_function: Thân hàm được xác định bởi cặp dấu mốc nhọn ({}), chứa khối lệnh gồm các cấu trúc điều khiển thực hiện nhiệm vụ của hàm. Thân hàm phải chứa

lệnh ***return*** sau có thể có giá trị trả về hoặc không có giá trị trả về (ứng với trường hợp kiểu trả về là ***void***).

Ví dụ 4.2. Hàm có giá trị trả về và hàm không có giá trị trả về

```
/*Ham co gia tri tra ve*/
int add(int a, int b){
    return a + b;
}

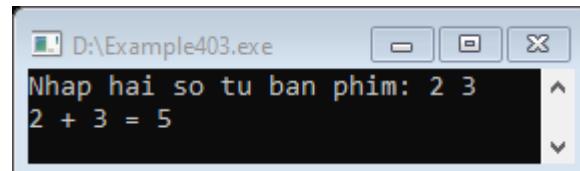
/*Ham khong co gia tri tra ve*/
void spell(int x){
    switch (x){
        case 0: printf("Day la so khong\n"); break;
        case 1: printf("Day la so mot\n"); break;
        case 2: printf("Day la so hai\n"); break;
        //...
        case 9: printf("Day la so chin\n"); break;
        default: printf("%d khong phai la so co mot chu so\n", x); break;
    }
}
```

4.3 Hàm main

Hàm ***main*** là hàm nhận trình điều khiển từ hệ điều hành mỗi khi chương trình thực thi. Nói cách khác, hàm ***main*** là điểm bắt đầu của mọi chương trình C/C++. Khi chương trình thực thi xong, hàm ***main*** trả về cho hệ điều hành một giá trị kiểu số nguyên như mặc định của trình biên dịch C/C++. Thông thường, hàm ***main*** trả về giá trị là 0 khi chương trình thực hiện và kết thúc thành công.

Ví dụ 4.3. Hàm main

```
#include <stdio.h>           /*Khai bao thu vien*/
int main() {                   /*Ham chuong trinh chinh*/
    int x, y, max;
    printf("Nhap hai so tu ban phim: ");
    scanf("%d %d", &x, &y);
    printf("%d + %d = %d\n", x, y, x+y);
    getchar();
    return 0;
}
```



4.4 Quy tắc cài đặt hàm

Một chương trình máy tính bao gồm nhiều hàm, các hàm được cài đặt đáp ứng hai quy tắc sau đây:

- **Quy tắc kết dính:** Mỗi hàm chỉ thực hiện một nhiệm vụ cụ thể, nhiệm vụ có thể là đọc/ghi (dữ liệu), hoặc tính toán (xử lý dữ liệu) nhưng không thể cả hai.
- **Quy tắc độc lập:** Hàm không được truy cập dữ liệu bên ngoài.

Ví dụ 4.4. Lỗi cài đặt hàm

```
/*Kiem tra so nguyen to*/
int isPrime(int n){
    int i;
    printf("Nhập số nguyên n= "); scanf("%d", &n);
    for (i=2; i*i <= n; i++) if (n%i == 0) return 0;
    return 1;
}
/*Tinh trung binh cua ba so*/
int a=1, b=5, c=7;
double average(){
    return (a+b+c)/3.0;
}
```

Nhận xét:

- Hàm isPrime thực hiện nhiệm vụ đọc dữ liệu từ bàn phím và xử lý dữ liệu đó. Các câu lệnh không có liên kết chặt chẽ với nhau. Quy tắc kết dính bị vi phạm.
- Hàm average sử dụng dữ liệu (a, b, c) được khai báo và khởi tạo bên ngoài thân hàm. Quy tắc độc lập bị vi phạm.

Ví dụ 4.5. Cài đặt hàm hợp lệ

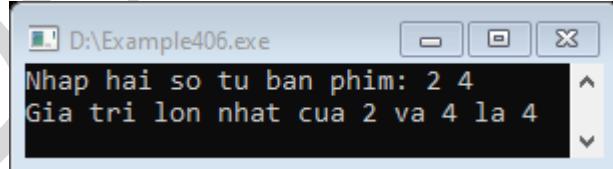
```
/*Kiem tra so nguyen to*/
int isPrime(int n){
    int i;
    for (i=2; i*i <= n; i++) if (n%i == 0) return 0;
    return 1;
}
/*Tinh trung binh cua ba so*/
double average(int a, int b, int c){
    return (a+b+c)/3.0;
}
```

4.5 Quy tắc về phạm vi của hàm

Hàm giống như một hộp đen được biết đến bởi tên, kiểu trả về và danh sách tham số. Các hàm (bao gồm cả hàm **main**) chỉ tương tác với nhau thông qua lời gọi hàm. Nội dung của một hàm (thân hàm) bao gồm các cấu trúc điều khiển, các dữ liệu chỉ có ý nghĩa bên trong hàm đó và không thể tương tác với nội dung của hàm khác. Những biến được khai báo bên trong hàm (kể cả các tham số) là những biến cục bộ (local variable), chỉ có giá trị sử dụng bên trong hàm. Những biến này được tạo ra khi hàm được gọi và biến mất sau khi hàm thực thi xong. Bên cạnh biến cục bộ, chương trình C/C++ còn có biến toàn cục (global variable), câu lệnh khai báo loại biến này không nằm trong bất kỳ hàm nào và được sử dụng trong tất cả các hàm của chương trình.

Ví dụ 4.6. Biến cục bộ, biến toàn cục và tham số của hàm

```
#include <stdio.h>
int MaxN = 100;           /*Bien toan cuc*/
int maximize(int a, int b){ /*Hai tham so cua ham maximize*/
    int max = a;          /*max la bien cuc bo trong ham maximize*/
    if (a < b) max = b;
    return max;
}
int main(){
    int x, y, max; /*x, y, max la bien cuc bo trong ham main*/
    printf("Nhập hai số tự ban phím: ");
    scanf("%d %d", &x, &y);
    max = maximize(x,y);
    printf("Giá trị lớn nhất của %d và %d là %d", x, y, max);
    getchar();
    return 0;
}
```

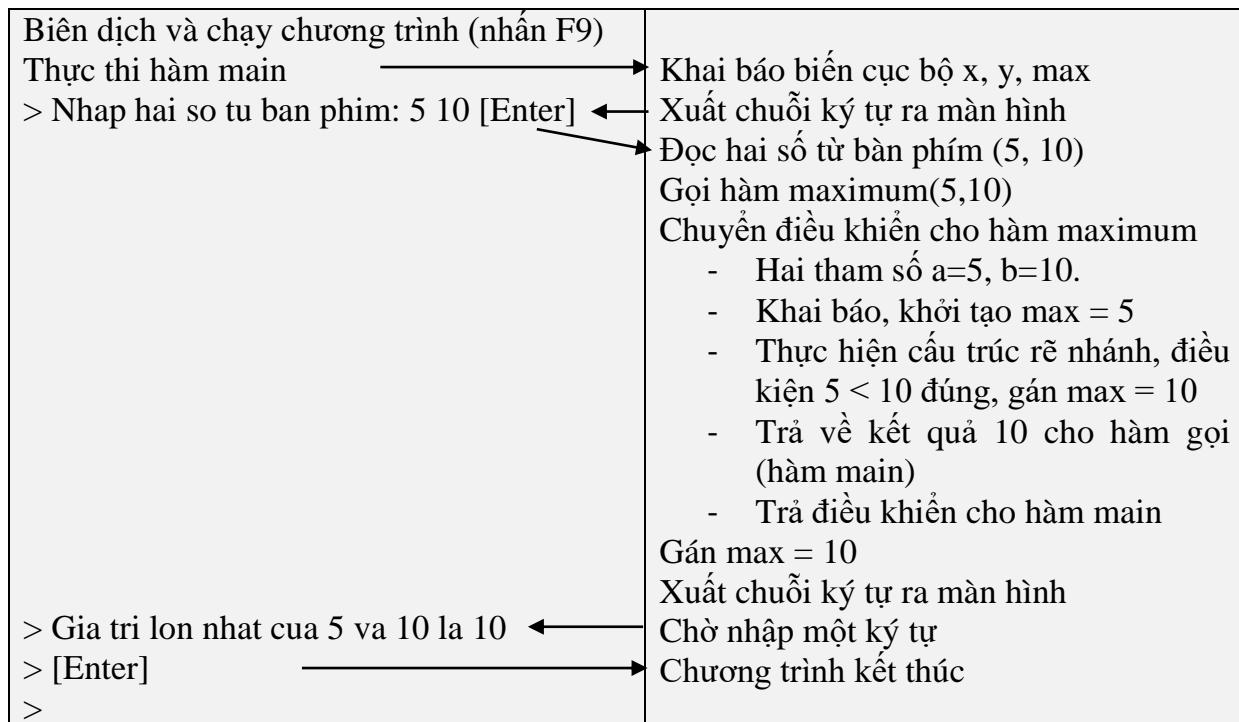


4.6 Tham số và đối số

Trong **Ví dụ 4.6**, hàm **maximum** có hai tham số và mỗi khi hàm này được gọi từ hàm **main** (hàm gọi), nó yêu cầu được cung cấp hai giá trị (đối số - argument) phù hợp với kiểu dữ liệu của hai tham số nói trên. Các giá trị này được sao chép vào các tham số và được sử dụng trong phần thân của hàm **maximum**. Nói cách khác, tham số là tên của dữ

liệu được sử dụng trong cài đặt hàm và đối số là dữ liệu thực tế được truyền cho hàm mỗi khi hàm được gọi.

Ví dụ 4.7. Minh họa thực thi chương trình ở Ví dụ 4.6



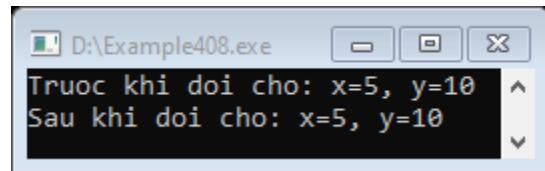
Có hai phương pháp truyền đối số cho tham số bao gồm truyền tham trị và truyền tham chiếu.

4.6.1 Truyền tham trị

Phương pháp truyền tham trị thực hiện sao chép giá trị của đối số cho tham số của hàm. Trong trường hợp này, những thay đổi (nếu có) của tham số không ảnh hưởng đến đối số. Nói cách khác, giá trị của đối số trong hàm gọi không thay đổi trước và sau khi gọi thực thi hàm khác.

Ví dụ 4.8. Truyền tham trị cho hàm

```
#include <stdio.h>
void swap(int a, int b){
    int temp = a;
    a = b; b = temp;
}
int main(){
    int x = 5, y = 10;
```



```

    printf("Truoc khi doi cho: x=%d, y=%d\n", x, y);
    swap(x,y);
    printf("Sau khi doi cho: x=%d, y=%d", x, y);
    getchar();
    return 0;
}

```

Hàm **swap**(int a, int b) với nhiệm vụ hoán vị giá trị của a và b như vậy là không đạt yêu cầu. Mặc dù hàm **swap** hoàn toàn chính xác cả về cú pháp lẫn ý nghĩa nhưng kết quả hiển thị trên màn hình của chương trình không có gì thay đổi trước và sau khi gọi hàm này. Điều này là do chương trình lựa chọn phương pháp truyền tham trị cho hàm **swap**.

Trong trường hợp này, hàm **swap** nhận hai đối số là giá trị của x và y (tức 5 và 10) rồi gán chúng cho hai tham số a và b của hàm. Nói cách khác, khi hàm **swap** nhận trình điều khiển từ hàm main, hai tham số của hàm có giá trị lần lượt là 5 và 10. Kết thúc hàm **swap**, a và b có giá trị lần lượt là 10 và 5 như là kết quả của việc hoán vị hai số. Tuy nhiên, sự thay đổi này chỉ có hiệu lực bên trong hàm **swap** mà thôi. Khi hàm main nhận trình điều khiển do hàm **swap** trả về, giá trị của x và y vẫn là 5 và 10 như trước khi gọi hàm **swap**.

4.6.2 Truyền tham chiếu

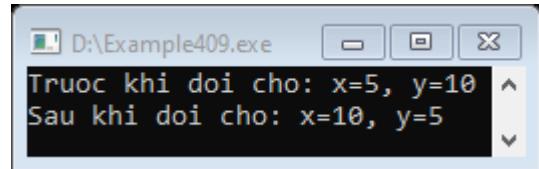
Phương pháp truyền tham chiếu thực hiện sao chép địa chỉ của đối số cho tham số của hàm. Trong trường hợp này, do đối số và tham số có cùng địa chỉ trong bộ nhớ nên mọi thay đổi trên tham số (được lưu trữ tại địa chỉ tương ứng trong bộ nhớ) cũng chính là thay đổi trên đối số (được lưu trữ tại cùng địa chỉ). Nói cách khác, giá trị của đối số trong hàm gọi thay đổi tương ứng với thay đổi của tham số trong hàm được gọi.

Ví dụ 4.9. Truyền tham chiếu cho hàm

```

#include <stdio.h>
void swap2(int &a, int &b){
    int temp = a;
    a = b; b = temp;
}
int main(){
    int x = 5, y = 10;
    printf("Truoc khi doi cho: x=%d, y=%d\n", x, y);

```



```

swap2(x,y);
printf("Sau khi doi cho: x=%d, y=%d", x, y);
getchar();
return 0;
}

```

Hàm ***swap2*** trong Ví dụ 4.9 cũng có hai tham số như hàm ***swap*** trong Ví dụ 4.8. Tuy nhiên, ***swap2*** nhận địa chỉ của hai đối số thay vì nhận giá trị như ***swap***. Kết quả của việc truyền địa chỉ thông qua tham chiếu của các đối số làm cho hai tham số a và b của hàm được gọi (***swap2***) có cùng địa chỉ lưu trữ trong bộ nhớ với hai đối số x và y của hàm gọi (***main***). Do đó, sự thay đổi giá trị của a (từ 5 thành 10) và b (từ 10 thành 5) trong hàm ***swap2*** cũng được ghi nhận đổi với giá trị của x và y trong hàm ***main***. Nói cách khác, khi hàm ***main*** nhận trình điều khiển từ hàm ***swap2***, x và y có giá trị lần lượt là 10 và 5 đúng như yêu cầu đặt ra ban đầu của chương trình.

4.7 Nguyên mẫu hàm

Trong C/C++, tất cả các hàm phải được khai báo trước khi chúng được sử dụng. Việc này thực hiện bằng cách khai báo nguyên mẫu của hàm. Nguyên mẫu hàm cho phép C/C++ cung cấp chức năng kiểm tra sự hợp lệ của tham số khi định nghĩa cũng như khi gọi hàm, cụ thể trình biên dịch sẽ kiểm tra:

- Sự tương thích giữa kiểu dữ liệu của các đối số khi gọi hàm và kiểu dữ liệu của các tham số trong định nghĩa hàm được gọi.
- Sự phù hợp về số lượng đối số cung cấp khi gọi hàm và số lượng tham số trong định nghĩa hàm được gọi.

Dạng tổng quát của một nguyên mẫu hàm:

returnType functionName (parameterList);

returnType: Kiểu dữ liệu của giá trị trả về bởi hàm, returnType là ***void*** nếu hàm không có giá trị trả về (khi đó hàm được gọi là thủ tục).

functionName: Tên hàm, quy tắc đặt tên giống như đối với biến.

parameterList: Danh sách tham số, thể hiện theo cặp *kiểu_dữ_liệu tên_tham_số*. Các tham số cách nhau bởi dấu phẩy (,). Danh sách này rỗng, ký hiệu (), khi hàm không có tham số.

Chú ý:

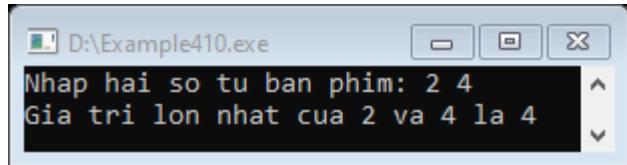
- Khai báo nguyên mẫu hàm kết thúc bởi dấu chấm phẩy (;).
- Hàm phải được cài đặt bên dưới khai báo nguyên mẫu hàm.

Ví dụ 4.10. Nguyên mẫu hàm và cài đặt hàm

```
#include <stdio.h>
int maximize(int a, int b); /*Khai bao nguyen mau ham*/

int main(){
    int x, y, max;
    printf("Nhập hai số từ bàn phím: ");
    scanf("%d %d", &x, &y);
    max = maximize(x,y);
    printf("Giá trị lớn nhất của %d và %d là %d",x , y, max);
    getchar();
    return 0;
}

int maximize(int a, int b){ /*Cài đặt hàm*/
    int max = a;
    if (a < b) max = b;
    return max;
}
```



4.8 Xây dựng chương trình với các hàm

Một chương trình máy tính có thể thực hiện nhiều chức năng khác nhau và để việc cài đặt được thuận lợi, chương trình được phân rã thành nhiều hàm. Việc phân tích nội dung yêu cầu về chức năng của chương trình giúp xác định dữ liệu (xác định bởi các danh từ) cũng như thuật giải (xác định bởi các động từ) phù hợp.

Thuật giải của chương trình là một chuỗi các hành động kế tiếp nhau, trong đó có hành động đơn giản (được giải quyết bởi một biểu thức, một câu lệnh hoặc một hàm thư

viên), hành động phức tạp (được giải quyết bởi một nhóm các biểu thức, câu lệnh, hàm thư viện).

Mỗi hành động phức tạp nói trên được cài đặt thành một hàm với tham số và giá trị trả về phù hợp.

Ví dụ 4.11. Xây dựng chương trình với các hàm

Yêu cầu: Xây dựng chương trình in ra màn hình n số nguyên tố đầu tiên với n là số nguyên nhập từ bàn phím (số nguyên tố là số nguyên lớn hơn 1 và chỉ chia hết cho 1 và chính nó).

Phân tích:

- Danh từ: số nguyên n
- Động từ:
 - + Bắt đầu
 - + Nhập n -> đơn giản
 - + In n số nguyên tố đầu tiên -> phức tạp
 - + Kết thúc

Phân tích:

Hàm: void printPrimes(int n){
 int count = 0; //số lượng số đã in
 int current = 2; //số nhỏ nhất có thể
 while (count < n){
 if (current là số nguyên tố){-> phức tạp
 in current ra màn hình;-> đơn giản
 count++;//tăng số đếm-> đơn giản
 }
 current++;//kiểm tra số kế tiếp-> đơn giản
 }
}

Hàm: int isPrime(int current){
 int res=1, i=0;
 for (i=2; i*i <= current && res == 1; i++)
 if (current%i == 0) res = 0; -> đơn giản
 return res;
}

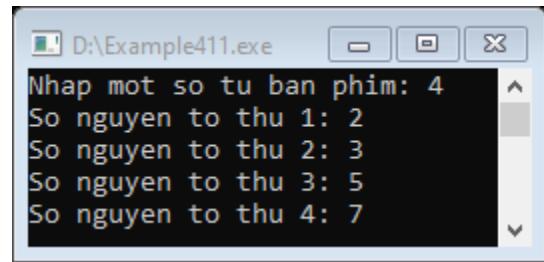
Cài đặt:

```
#include <stdio.h>
int isPrime(int current);
void printPrimes(int n);
int main(){
    int n;
    printf("Nhập một số tự bàn phím: "); scanf("%d", &n);
    printPrimes(n);
    getchar();
    return 0;
}
```

```

int isPrime(int current){
    int res=1, i=0;
    for (i=2; i*i <= current && res == 1; i++)
        if (current%i == 0) res = 0;
    return res;
}

```



```

void printPrimes(int n){
    int count = 0, current = 2;
    while (count < n){
        if (isPrime(current)==1){
            printf("So nguyen to thu %d: %d\n", (count+1), current);
            count++;
        }
        current++;
    }
}

```

4.9 Hàm main có tham số

Hàm **main()**, giống như các hàm khác trong ngôn ngữ lập trình C, có thể nhận tham số. Tuy nhiên, tham số của hàm main tương đối đặc biệt, bao gồm hai thành phần: (1) số lượng tham số và (2) mảng các tham số tương ứng.

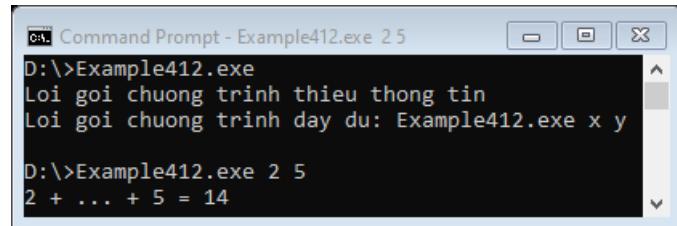
```
int main(int argc, char *argv[]){}
```

Ví dụ 4.12. Xây dựng chương trình nhận tham số khi thực thi

```

/* Chuong trinh tong cac so nam giua hai so truyen vao tu loi goi chuong trinh */
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]){
    int x, y, s=0, i;
    if (argc!=3) {
        printf("Loi goi chuong trinh thieu thong tin\n");
        printf("Loi goi chuong trinh day du: Example412.exe x y\n");
        exit(1);
    }
    x = atoi(argv[1]);
    y = atoi(argv[2]);
    for (i=x; i<=y; i++)
        s+=i;
}

```



```
    printf("%d + ... + %d = %d",x,y,s);
    getchar();
    return 0;
}
```

4.10 Câu hỏi ôn tập

- 1) Trình bày khái niệm về hàm, quy tắc cài đặt hàm, lời gọi hàm
- 2) Trình bày quy tắc về phạm vi của hàm, phân biệt biến và tham số, tham số và đối số
- 3) Trình bày hai phương truyền đối số cho tham số
- 4) Nguyên mẫu hàm là gì? Tại sao cần sử dụng nguyên mẫu hàm trong chương trình?
- 5) Sự khác biệt giữa hàm main có tham số và hàm main không có tham số?
- 6) Hãy cho biết kết quả xuất ra màn hình khi thực hiện lời gọi làm *funct6(10)*. Giải thích.

```
void funct6(int n) {
    int i;
    for (i=1; i<= n/2; i++)
        if (n%i==0) printf("%d, ", i);
}
```

- 7) Hãy cho biết kết quả trả về khi thực hiện lời gọi làm *funct7(10)*. Giải thích.

```
int funct7(int n) {
    int S=0, i;
    for ( i=1; i<=n/2; i++)
        if (n%i==0) S +=i;
    return S;
}
```

- 8) Hãy cho biết kết quả trả về khi thực hiện lời gọi làm *funct8(5,1,2)*. Giải thích.

```
int funct8 (int n, int start, int step) {
    int s=start;
    for (int i=1; i<n; i++) s += step;
    return f;
}
```

- 9) Hãy cho biết kết quả trả về khi thực hiện lời gọi làm *funct9(5,1,2)*. Giải thích.

```
int funct9 (int n, int start, int step) {
    int s=start;
    for (int i=1; i<n; i++) s *= step;
    return s;
}
```

10) Hãy cho biết kết quả trả về khi thực hiện lời gọi làm *funct10(1234)*. Giải thích.

```
int funct10 (int n) {  
    int c=0;  
    do {  
        int r = n%10;  
        n = n/10;  
        c+=(r%2==0?1:0);  
    } while (n>0);  
    return c;  
}
```

4.11 Bài tập thực hành

- Sử dụng hàm để xây dựng chương trình nhập một số nguyên dương n từ bàn phím ($n > 1$), sau đó xuất ra màn hình các số nguyên tố giữa 2 và n.

Phân tích:

- Danh từ: số nguyên n
- Động từ:
 - + Bắt đầu
 - + Nhập n -> đơn giản
 - + In số nguyên tố từ 2..n
 - > phức tạp
 - + Kết thúc

Phân tích:

Hàm: printPrimes(int n){
 int current; //số cần kiểm tra
 for (current=2; current<=n; current++){
 if (current là số nguyên tố){ -> phức tạp
 in current ra màn hình;-> đơn giản
 }
 current++; //số kế tiếp-> đơn giản
 }
Hàm: isPrime(int current){
 int res=1, i=0;
 for (i=2; i*i <= current && res == 1; i++)
 if (current%i == 0) res = 0;-> đơn giản
 return res;
}

Cài đặt:

```
#include <stdio.h>  
int isPrime(int current);  
void printPrimes(int n);  
int main(){  
    int n;  
    printf("Nhập một số từ bàn phím: "); scanf("%d", &n); printPrimes(n);  
    getchar();  
    return 0;  
}
```

```

int isPrime(int current){
    int res=1, i=0;
    for (i=2; i*i <= current && res == 1; i++)
        if (current%i == 0) res = 0;
    return res;
}

void printPrimes(int n){
    int current, count=0;
    for (current=2; current<n; current++){
        if (isPrime(current)==1)
            printf("So nguyen to thu %d: %d\n", (++count), current);
    }
}

```

- 2) Sử dụng hàm để xây dựng chương trình kiểm tra ba số nguyên dương nhập từ bàn phím d, m, y có phải là một ngày hợp lệ.

Phân tích:

- Danh từ: số nguyên d, m, y
- Động từ:
 - + Bắt đầu
 - + Nhập d, m, y -> đơn giản
 - + Kiểm tra ngày tháng
-> phức tạp
 - + Kết thúc

Phân tích:

```

Hàm: int validDate(int d, int m, int y){
    int maxDay;
    if (1<d || d>31 || m<1 || m>12) return 0;
    //Xác định số ngày tối đa cho tháng
    if (m==4 || m==6 || m==9 || m==11)
        maxDay=30;
    else if (m==2) {
        if (y%4 == 0) maxDay=29; //năm nhuận
        else maxDay=28
    }
    else maxDay=31;
    return d < maxDay;
}

```

Cài đặt:

```

#include <stdio.h>
int validDate(int d, int m, int y);
int main(){
    int d, m, y;
    printf("Nhập ba số tu ban phim (d/m/y): ");
    scanf("%d/%d/%d", &d, &m, &y);
    if (validDate(d,m,y)==1)
        printf("%d/%d/%d la mot ngay thang hop le.\n", d, m, y);
    else

```

```

        printf("%d/%d/%d khong la mot ngay thang hop le.\n", d, m, y);
getchar();
return 0;
}
int validDate(int d, int m, int y){
    int maxDay;
    if (1>d || d>31 || m<1 || m>12) return 0;
    //Xac dinh so ngaytoi da cho moi thang
    if (m==4 || m==6 || m==9 || m==11) maxDay=30;
    else if (m==2)
        if (y%4 == 0) maxDay=29; //Nam nhuan
        else maxDay=28
    else maxDay=31;
    return d < maxDay;
}

```

- 3) Sử dụng hàm để xây dựng chương trình kiểm tra một điểm với hai tọa độ x, y có nằm trên đường tròn tâm (0,0), bán kính r hay không, với x, y, r ($r>0$) là ba số thực nhập từ bàn phím.

Phân tích:

- Danh từ: số thực, x, y, r
- Động từ:
 - + Bắt đầu
 - + Nhập x, y, r -> đơn giản
 - + Kiểm tra vị trí
-> phức tạp
 - + Kết thúc

Phân tích:

Hàm: int getPosition(double x, double y, double r){
 int res;
 double d2 = x*x + y*y;
 double r2 = r*r;
 if (d2 < r2) res =1; //nằm trong
 else if (d2 == r2) res 0; //nằm trên
 else res =-1; //nằm ngoài
 return res;

Cài đặt:

```

#include <stdio.h>
int getPosition(double x, double y, double r);
int main(){
    double x, y, r; int res;
    printf("Nhập tọa độ (x, y): ");
    scanf("(%lf, %lf)", &x, &y);
    do {
        printf("Nhập bán kính (>0): ");
        scanf("%lf", &r);
    } while (r<=0);

```

```

res = getPosition(x,y,r);
if (res==1) printf("(%.f,%.f) nam trong duong tron ban kinh %.f", x, y, r);
else if (res==0) printf("(%.f,%.f) nam tren duong tron ban kinh %.f", x, y, r);
else printf("(%.f,%.f) nam ngoai duong tron ban kinh %.f", x, y, r);
getchar();
return 0;
}
int getPosition(double x, double y, double r){
    int res;
    double d2 = x*x + y*y;
    double r2 = r*r;
    if (d2 < r2) res =1;           //nam trong
    else if (d2 == r2) res =0;     //nam tren
    else res =-1;                 //nam ngoai
    return res;
}

```

4) Sử dụng hàm để xây dựng chương trình nhập một số nguyên dương n và xuất n!

Phân tích:

- Danh từ: số nguyên n
- Động từ:
 - + Bắt đầu
 - + Nhập n -> đơn giản
 - + Tính n!-> phức tạp
 - + Kết thúc

Cài đặt:

```

#include <stdio.h>
double calcFactorial(int n);
int main(){
    int n;
    do {
        printf("Nhập số nguyên dương n: "); scanf("%d", &n);
    } while (n<=0);
    printf("%d! = %f", n, calcFactorial(n));
    getchar();
    return 0;
}
double calcFactorial (int n){
    double res = 1;
    for (int i = 2; i<=n; i++) res *= i;
    return res;
}

```

Phân tích:

```

double calcFactorial (int n){
    double res = 1;
    int i;
    for (i = 2; i<=n; i++) res *= i;
    return res;
}

```

}

- 5) Sử dụng hàm để xây dựng chương trình nhập một số nguyên dương n và xuất ra số Fibonacci thứ n.

(Dãy số Fibonacci: 1 1 2 3 5 8 13 21 34 ..., trong đó $F_1=F_2=1$, $F_n=F_{n-2}+F_{n-1}$, $n \geq 3$)

Phân tích:

- Danh từ: số nguyên n
- Động từ:
 - + Bắt đầu
 - + Nhập n -> đơn giản
 - + Tính F_n -> phức tạp
 - + Kết thúc

Phân tích:

```
int calcFibonacci (int n){  
    int f1=1, f2=1, f=1, i;  
    for (i = 3; i<=n; i++){  
        f = f1 + f2; f1 = f2; f2 = f;  
    }  
    return f;  
}
```

Cài đặt:

```
#include <stdio.h>  
int calcFibonacci(int n);  
int main(){  
    int n;  
    do {  
        printf("Nhập số nguyên dương n: ");  
        scanf("%d", &n);  
    } while (n<1);  
    printf("F(%d) = %d", n, calcFibonacci (n));  
    getchar();  
    return 0;  
}  
  
int calcFibonacci(int n){  
    int f1=1, f2=1, f=1, i;  
    for (i = 3; i<=n; i++){  
        f = f1 + f2; f1 = f2; f2 = f;  
    }  
    return f;  
}
```

- 6) Sử dụng hàm để xây dựng chương trình yêu cầu nhập lần lượt từng số nguyên dương và xuất ra tổng các chữ số của số đã nhập; chương trình sẽ kết thúc khi người dùng nhập vào số 0 hoặc số âm.

Ví dụ:

> Nhập một số nguyên dương: 12345

> Tong cac chu so cua 12345 la: 15
> Nhap mot so nguyen duong: 324
> Tong cac chu so cua 324 la: 9
> Nhap mot so nguyen duong: 0
> Chuong trinh ket thuc.

Phân tích:

- Danh từ: số nguyên n
- Động từ:
 - + Bắt đầu
 - + Nhập n -> đơn giản
 - + Tính tổng các chữ số -> phức tạp
 - + Kết thúc

Phân tích:

```
int sumDigits(int n){  
    int sum=0; /* initialize sum of digits */  
    int digit; /*current digit*/  
    do {  
        digit = n%10 ;  
        n = n/10;  
        sum += digit;  
    } while (n>0);  
    return sum;  
}
```

Cài đặt:

```
#include <stdio.h>  
int sumDigits(int n);  
  
int main(){  
    int n;  
    do {  
        printf("Nhập số nguyên dương n: ");  
        scanf("%d", &n);  
    } while (n<0);  
    printf("Tổng các chữ số của %d là: %d", n, sumDigits (n));  
    getchar();  
    return 0;  
}  
  
int sumDigits(int n){  
    int sum=0, digit;  
    do {  
        digit = n%10;  
        n = n/10;  
        sum += digit;  
    } while (n>0);  
    return sum;
```

}

- 7) Sử dụng hàm để xây dựng chương trình nhập hai số nguyên tương ứng với phần nguyên và phần thập phân (>0) của một số thực và xuất ra số thực đó.

Ví dụ:

Input: 32 25 -51 139

Output: 32.25 -51.139

Phân tích:

- Danh từ: số nguyên n, k
- Động từ:
 - + Bắt đầu
 - + Nhập n, k \rightarrow đơn giản
 - + Ghép thành số thực \rightarrow phức tạp
 - + Kết thúc

Phân tích:

```
double makeDouble(int n, int k){  
    /*tạo phần thập phân <1 */  
    while (k >=1) k = k/10;  
    /*đổi dấu phần thập phân (nếu cần)*/  
    if (n<0) k = -k;  
    return n + k;  
}
```

Cài đặt:

```
#include <stdio.h>  
double makeDouble(int n, int k);  
  
int main(){  
    int n, k;  
    do {  
        printf("Nhập hai số nguyên n, k (k>=0): ");  
        scanf("%d, %d", &n, &k);  
    } while (k<0);  
    printf("Số thực tương ứng là %f: ", makeDouble (n,k));  
    getchar();  
    return 0;  
}  
  
double makeDouble(int n, int k){  
    double d = (double) k;  
    while (d >=1)  
        d = d/10;  
    if (n<0) d = -d;  
    return n + d;  
}
```

- 8) Sử dụng hàm để xây dựng chương trình nhập hai số nguyên và xuất ước số chung lớn nhất (GCD – greatest common divisor) và bội số chung nhỏ nhất (LCM – least common multiple) của chúng.

Phân tích:

- Danh từ: số nguyên n, k
- Động từ:
 - + Bắt đầu
 - + Nhập n, k -> đơn giản
 - + Tính USCLN -> phức tạp
 - + Tính BSCNN -> phức tạp
 - + Kết thúc

Phân tích:

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

int calcLCM(int a, int b){
    return a * b / calcGCD(a, b);
}
```

Cài đặt:

```
#include <stdio.h>
int calcGCD(int a, int b);
int calcLCM(int a, int b);
int main(){
    int n, k;
    do {
        printf("Nhập hai số nguyên dương n k: ");
        scanf("%d %d", &n, &k);
    } while (n <= 0 || k <= 0);
    printf("GCD(%d,%d) = %d; LCM(%d,%d) = %d", n, k, calcGCD(n,k), n, k, calcLCM(n,k));
    getchar();
    return 0;
}
int calcGCD(int a, int b){
    while (a != b)
        if (a > b) a -= b;
        else b -= a;
    return a;
}
int calcLCM(int a, int b){
    return a * b / calcGCD(a, b);
}
```

- 9) Sử dụng hàm để xây dựng chương trình nhập một số nguyên dương và xuất chữ số lớn nhất, chữ số nhỏ nhất của số đó.

Phân tích:

- Danh từ: số nguyên n

Phân tích:

- Động từ:
 - + Bắt đầu
 - + Nhập n -> đơn giản
 - + Tìm chữ số lớn nhất -> phức tạp
 - + Tìm chữ số nhỏ nhất -> phức tạp
 - + Kết thúc

```

int minDigit(int n){
    int min, digit;
    min=digit=n%10;
    n=n/10;
    while (n>0)
        digit=n%10; n=n/10;
        if (min > digit) min=digit;
    return min;
}

int maxDigit(int n){
    int max, digit;
    max =digit=n%10;
    n=n/10;
    while (n>0)
        digit=n%10; n=n/10;
        if (max < digit) max =digit;
    return max;
}

```

Cài đặt:

```

#include <stdio.h>
int minDigit(int n);
int maxDigit(int n);
int main(){
    int n;
    do {
        printf("Nhập số nguyên dương n: ");
        scanf("%d", &n);
    } while (n<=0);
    printf("Chữ số nhỏ nhất, lớn nhất của %d là %d, %d", n, minDigit(n), maxDigit(n));
    getchar();
    return 0;
}

int minDigit(int n){
    int min, digit;
    min=digit=n%10; n=n/10;
    while (n>0){
        digit=n%10; n=n/10;
        if (min > digit) min=digit;
    }
    return min;
}

```

```

    }

int maxDigit(int n){
    int max, digit;
    max = digit = n % 10; n = n / 10;
    while (n > 0){
        digit = n % 10; n = n / 10;
        if (max < digit) max = digit;
    }
    return max;
}

```

10) Sử dụng hàm để xây dựng chương trình kiểm tra một số nguyên dương nhập từ bàn phím có phải là số chính phương hay không.

Phân tích:

- Danh từ: số nguyên n
- Động từ:
 - + Bắt đầu
 - + Nhập n -> đơn giản
 - + Kiểm tra số chính phương -> phức tạp
 - + Kết thúc

Phân tích:

```

int checkSquare(int n){
    int m; double s;
    s = sqrt(n); //Lấy căn bậc hai của n
    m = (int) s; //Lấy phần nguyên của s
    return m == s; //Lấy kết quả 1 (chính
                    phương)
}

```

Cài đặt:

```

#include <stdio.h>
#include <math.h> /* Thu vien ham chua ham sqrt()*/
int checkSquare(int n);
int main(){
    int n;
    do {
        printf("Nhập số nguyên dương n: ");
        scanf("%d", &n);
    } while (n <= 0);
    if (checkSquare(n) == 1) printf("%d là số chính phương.\n", n);
    else printf("%d không là số chính phương.\n", n);
    getchar();
    return 0;
}

int checkSquare(int n){
    int m; double s;
    s = sqrt(n);
    m = (int) s;
}

```

```
        return m==s;  
    }  
}
```

4.12 Bài tập đề nghị

- 1) Sử dụng hàm để xây dựng chương trình giải phương trình bậc hai $ax^2 + bx + c = 0$ với a, b, c là ba số nguyên nhập từ bàn phím.
- 2) Sử dụng hàm để xây dựng chương trình xuất ra màn hình n số chính phương đầu tiên, với n nguyên dương nhập từ bàn phím.
- 3) Sử dụng hàm để xây dựng chương trình nhập một số nguyên dương n và kiểm tra n có phải là số Fibonacci hay không.
- 4) Sử dụng hàm để xây dựng chương trình xuất ra màn hình n số Fibonacci đầu tiên, với n nguyên dương nhập vào từ bàn phím.
- 5) Sử dụng hàm để xây dựng chương trình xuất ra màn hình tam giác Pascal chiều cao n nguyên dương nhập từ bàn phím.
- 6) Sử dụng hàm để xây dựng chương trình nhập số tiền vay P (nguyên dương), số năm vay n (nguyên dương) và lãi suất r (số dương, thực, nhỏ hơn 1) và xuất ra số tiền phải trả F với công thức sau đây:

$$F = P * (1 + r)^n$$

- 7) Sử dụng hàm để xây dựng chương trình xuất ra các chữ số chẵn, các chữ số lẻ của số nguyên dương nhập từ bàn phím.
- 8) Sử dụng hàm để xây dựng chương trình nhập vào số nguyên dương n (tối đa 10000), số nguyên dương k có một chữ số và xuất ra số lần xuất hiện của k trong n.

Ví dụ:

- > Nhập hai số nguyên dương (n, k): 8902 1
- > Số lần xuất hiện của 1 trong 8902 là: 0
- > Nhập hai số nguyên dương (n, k): 8949 9
- > Số lần xuất hiện của 9 trong 8949 là: 2

- 9) Sử dụng hàm để xây dựng chương trình kiểm tra một số nguyên dương có 6 chữ số nhập từ bàn phím có phải là một số tài khoản hay không (Chữ số cuối cùng là kết quả phép chia lấy dư cho 6 của tổng bình phương các chữ số còn lại)

Ví dụ:

- Số 123456 không phải là số tài khoản vì $(1^2+2^2+3^2+4^2+5^2) = 55\%6 = 1 \neq 6$
- Số 123451 là số tài khoản vì $(1^2+2^2+3^2+4^2+5^2) = 55\%6 = 1 = 1$

10) Sử dụng hàm để xây dựng chương trình kiểm tra số nguyên dương nhập từ bàn phím có phải là số đối xứng hay không.

CONFIDENTIAL

CHƯƠNG 5. MẢNG

Trong chương này người học được cung cấp các kiến thức, kỹ năng cơ bản về việc sử dụng mảng trong lập trình.

Nội dung:

- Phần tử của mảng và các chỉ số mảng
 - Khai báo một mảng
 - Quản lý mảng trong C
 - Khởi tạo mảng
 - Mảng chuỗi/ ký tự
 - Mảng hai chiều
 - Khởi tạo mảng nhiều chiều.

5.1 Giới thiệu mảng

Mảng là một tập hợp các biến có cùng kiểu dữ liệu nằm liên tiếp nhau trong bộ nhớ và được tham chiếu bởi cùng một tên (tên mảng). Mỗi phần tử của mảng được tham chiếu thông qua chỉ mục (index). Nếu mảng có n phần tử thì phần tử đầu tiên có chỉ mục là 0 và phần tử cuối có chỉ mục là $n-1$. Để tham chiếu đến một phần tử ta dùng tên mảng và chỉ mục của phần tử được đặt trong cặp móc vuông ([]).

Số lượng phần tử trong mảng được gọi là kích thước của mảng. Kích thước của mảng là cố định và phải được xác định trước; nó không thể thay đổi trong suốt quá trình thực hiện chương trình. Có 2 loại mảng thông dụng là mảng 1 chiều và mảng nhiều chiều. Mảng cần được khai báo và khởi tạo trước khi sử dụng. Các thao tác phổ biến trên mảng bao gồm thêm mới phần tử, sửa giá trị phần tử, xóa phần tử và duyệt mảng.

Ví dụ 5.1: Mảng một chiều và mảng hai chiều

a) Mảng một chiều a có 10 phần tử, mỗi phần tử có kiểu số nguyên.

Chỉ số

Biến a (mảng)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 4 | 3 | 6 | 9 | 0 | 1 | 7 | 2 | 8 |
|---|---|---|---|---|---|---|---|---|---|

$a[i], i = 0 \dots 9$

b) Mảng hai chiều b có 12 phần tử (3×4), mỗi phần tử có kiểu số nguyên.

| Biến b (mảng) | Chỉ số cột (Columns) | | | | Chỉ số hàng (rows) |
|------------------|-------------------------|---|---|---|-----------------------|
| | 0 | 1 | 2 | 3 | |
| | 5 | 3 | 4 | 2 | 0 |
| | 6 | 7 | 1 | 9 | 1 |
| | 8 | 5 | 7 | 4 | 2 |

$b[1][2] = 1$

5.2 Mảng một chiều

5.2.1 Khai báo mảng một chiều

Dạng tổng quát để khai báo một mảng một chiều là:

type arrayName[elements];

type: kiểu dữ liệu của mỗi phần tử mảng

elements: số phần tử có trong mảng

arrayName: tên mảng

Giống như những biến khác, mảng phải được khai báo tường minh để cho trình biên dịch có thể cấp phát bộ nhớ cho nó. Kích thước (tính bằng byte) của mảng được tính theo công thức:

$$\text{TotalSize} = \text{sizeof(type)} * \text{elements}$$

Ví dụ 5.2: Khai báo mảng một chiều

Câu lệnh sau đây dùng để khai báo mảng a có 100 phần tử số nguyên:

int a[100];

Với kích thước của 1 số nguyên là 2 byte, mảng a có kích thước là $2 * 100 = 200$ byte. Sau khi được khai báo, mỗi phần tử trong mảng có thể được xem như một biến thông thường:

`a[0] = 2; //phan tu dau tien cua mang a duoc gan gia tri la 2`

`a[1] = a[0] + 3 //a[1] co gia tri la 5`

```
a[2] = a[0] + a[1]; //a[2] co gia tri la 7  
printf("a[%d] = %d",2,a[2]); //Ghi ra man hinh dong chu a[2]=7
```

5.2.2 Khai báo và khởi tạo mảng một chiều

Ngoài ra, mảng một chiều có thể được khai báo và khởi tạo đồng thời với cú pháp:

```
type arrayName[] = {value1, value2, ..., valueN};
```

Chú ý:

- Không khai báo kích thước mảng.
- Số lượng phần tử trong mảng là số giá trị được cung cấp trong cặp dấu ngoặc {}.
- Mỗi giá trị phân cách nhau dùng dấu phẩy.

Ví dụ 5.3: Khai báo và khởi tạo mảng một chiều

Câu lệnh sau đây dùng để khai báo và khởi tạo mảng có năm phần tử là số nguyên:

```
int soChan[] = {2,4,6,8,10};
```

Các phần tử trong mảng soChan lần lượt là:

soChan[0] có giá trị là 2

soChan[1] có giá trị là 4

...

soChan[4] có giá trị là 10

5.2.3 Truy xuất các phần tử trong mảng

Khi mảng **a** gồm n phần tử được khởi tạo, bộ nhớ cấp cho mảng **a** một vùng nhớ đủ để lưu trữ n phần tử này. Địa chỉ trong bộ nhớ của mảng **a** chính là địa chỉ trong bộ nhớ của phần tử đầu tiên trong mảng **a** (tức là $a[0]$). Toán tử & được dùng để lấy địa chỉ trong bộ nhớ của một biến ($\&a[0]$).

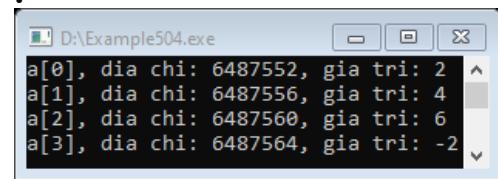
Mặt khác, các phần tử của mảng **a** được lưu trữ kế tiếp nhau trong bộ nhớ, do đó, thông qua địa chỉ của **a** hay $a[0]$, tất cả các phần tử thứ i của mảng **a** đều được xác định bởi địa chỉ $a + i$. Chính vì vậy, mảng **a** có tính chất truy xuất ngẫu nhiên dựa vào chỉ số của phần tử trong mảng.

Giả sử **type** là kiểu dữ liệu của phần tử trong mảng, d_0 là địa chỉ của phần tử $a[0]$, địa chỉ d_i của phần tử $a[i]$ trong mảng **a** được xác định bởi công thức sau đây:

$$d_i = d_0 + i * sizeof(type)$$

Ví dụ 5.4: Khai báo, khởi tạo và truy xuất mảng một chiều

```
#include <stdio.h>
int main(){
    int i, a[] = {2, 4, 6, -2};
    for (i = 0; i < 4; i++)
        printf("a[%d], dia chi: %u, gia tri: %d\n", i, a+i, a[i]);
    getchar();
    return 0;
}
```

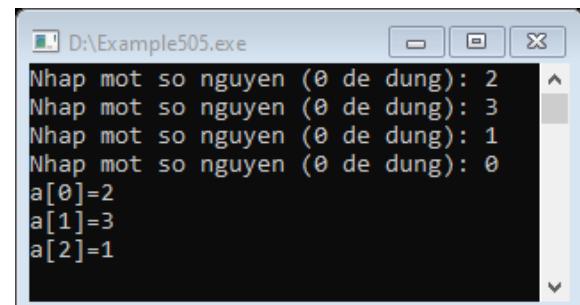


5.2.4 Sử dụng mảng một chiều như là tham số của hàm

Mảng có thể được sử dụng như tham số cho chương trình con (hàm, thủ tục) trong trường hợp chương trình con nhận nhiều tham số có cùng kiểu dữ liệu với số lượng không xác định. Khi đó, một bộ hai tham số cần được khai báo cho chương trình con bao gồm khai báo mảng và số lượng phần tử thực tế trong mảng. Mảng có thể xuất hiện trong chương trình con như là tham số đầu vào và tham số đầu ra.

Ví dụ 5.5: Sử dụng mảng như là tham số của chương trình con

```
#include <stdio.h>
/*Doc cac so nguyen tu ban phim va luu vao mang (cho den khi doc so 0 thi dung)*/
void GET(int a[], int &n){
    int x;
    do {
        printf("Nhập một số nguyên (0 để dừng): ");
        scanf("%d", &x);
        if (x!=0) a[n++]=x; //tương đương a[n]=x; n++;
    }
    while (x != 0);
}
/*In mang da nhap*/
void PUT(int a[], int n){
    for (int i = 0; i < n; i++) printf("a[%d]=%d\n", i, a[i]);
}
int main(){
    int a[100], n=0;
    GET(a, n);
    PUT(a,n);
    getchar();
    return 0;
}
```



5.2.5 Một số thao tác trên mảng một chiều

5.2.5.1 Tìm kiếm phần tử trong mảng (duyệt mảng)

Yêu cầu: Duyệt qua các phần tử của mảng, thực hiện thao tác đối với các phần tử thỏa mãn điều kiện cho trước

Ý tưởng: Sử dụng cấu trúc lặp for kết hợp cấu trúc rẽ nhánh if (hoặc switch)

Giải pháp: Duyệt tới (forward traversal) và duyệt lui (back traversal)

- Duyệt tới: cho biến (i) chạy từ chỉ số đầu tiên đến chỉ số cuối cùng của mảng
- Duyệt lui: cho chạy (i) chạy từ chỉ số cuối cùng đến chỉ số đầu tiên của mảng

Ví dụ 5.6: Tìm vị trí (chỉ số) của phần tử trong mảng a, kích thước n có giá trị là x

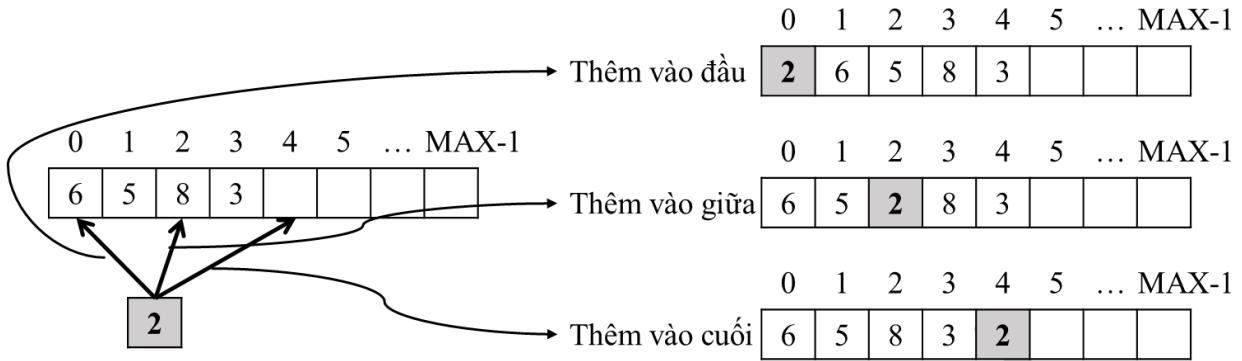
```
int findIndex(int a[], int n, int x){  
    for (int i = 0; i < n; i++)  
        if (x == a[i]) break; //chi so i la vi tri cua phan tu co gia tri x  
    return i; //tra ve ket qua tim thay (i<n), khong tim thay (i=n)  
}
```

Ví dụ 5.7: Tìm phần tử có giá trị lớn nhất trong mảng a, kích thước n

```
int findMaxValue(int a[], int n){  
    int i, Max = a[0];           //bat dau voi Max = phan tu dau tien cua mang  
    for (i = 1; i < n; i++)  
        if (Max < a[i])        //neu co phan tu lon hon Max, Max la phan tu do  
            Max = a[i];  
    return Max;  
}
```

5.2.5.2 Thêm phần tử vào mảng

Yêu cầu: Thêm một phần tử vào mảng trong trường hợp mảng chưa đầy (kích thước hiện hữu nhỏ hơn kích thước tối đa của mảng). Phần tử mới có thể được thêm vào đầu, cuối hoặc vào một vị trí (chỉ số) bất kỳ sao cho vẫn đảm bảo tính liên tục của chỉ số của mảng như hình vẽ dưới đây:



Ý tưởng: Sử dụng cấu trúc lặp for để di chuyển các phần tử hiện có trong mảng nhằm tạo vị trí “trống” phù hợp với yêu cầu thêm phần tử mới vào mảng.

Giải pháp:

- Kiểm tra mảng đã đầy hay chưa, thực hiện các bước tiếp theo nếu mảng chưa đầy
- Đẩy các phần tử từ vị trí t (vị trí của phần tử mới) sang bên phải một đơn vị
- “Thêm” phần tử mới vào vị trí t
- Tăng kích thước hiện hữu lên 1

Ví dụ 5.8: Thêm phần tử vào mảng a (kích thước tối đa MAX)

```

int addFirst(int a[], int &n, int x){ /*Them phan tu x vao dau mang a, kich thuoc n*/
    int result = 0, i;
    if (n < MAX) {
        /*Day cac phan tu trong mang sang phai mot don vi*/
        for (i = n-1; i >= 0; i--) a[i+1] = a[i];
        /*Them phan tu moi vao dau mang, tang kich thuoc mang len 1*/
        a[0] = x; n++; result = 1;
    }
    return result;
}

int addLast(int a[], int &n, int x){ /*Them phan tu x vao cuoi mang a, kich thuoc n*/
    int result = 0, i;
    if (n < MAX) {
        /*Them phan tu vao cuoi mang, tang kich thuoc mang len 1 don vi*/
        a[n] = x; n++; result = 1;
    }
    return result;
}

/*Them phan tu x vao vi tri t cua mang a, kich thuoc n*/
int addMiddle(int a[], int &n, int x, int t){

```

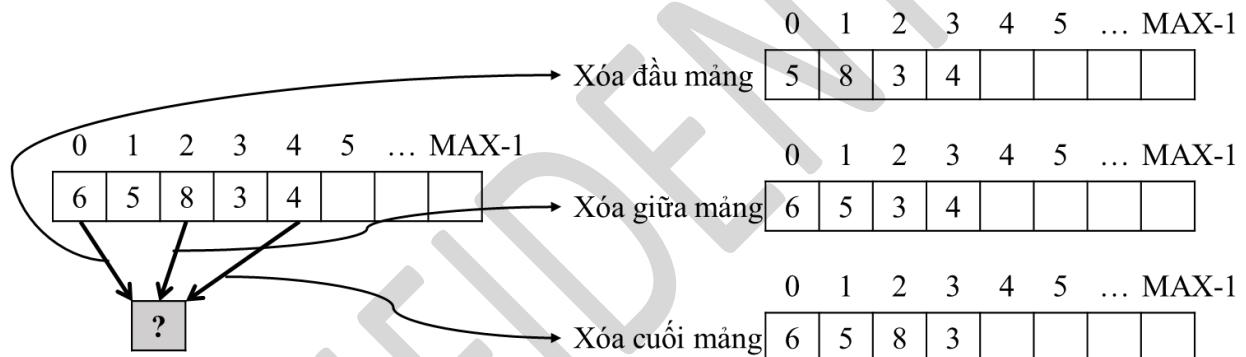
```

int result = 0, i;
if (n < MAX && t >= 0 && t <= n) {
    /*Day cac phan tu trong mang tu vi tri t sang phai mot don vi*/
    for (i = n-1; i >= t; i--) a[i+1] = a[i];
    /*Them phan tu moi vao vi tri t, tang kich thuoc mang len 1*/
    a[t] = x; n++; result = 1;
}
return result;
}

```

5.2.5.3 Xóa phần tử ra khỏi mảng

Yêu cầu: Xóa một phần tử ra khỏi mảng trong trường hợp mảng không rỗng (kích thước hiện hữu lớn hơn 0). Phần tử bị xóa có thể ở đầu, cuối hoặc ở một vị trí (chỉ số) bất kỳ sao cho vẫn đảm bảo tính liên tục của chỉ số của mảng như hình vẽ dưới đây:



Ý tưởng: Sử dụng cấu trúc lặp for để di chuyển các phần tử hiện có trong mảng nhằm lấp đầy vị trí “trống” khi phần tử bị xóa ra khỏi mảng.

Giải pháp:

- Kiểm tra mảng không trống và vị trí cần xóa nằm trong miền giới hạn của chỉ số mảng, thực hiện các bước tiếp theo nếu cả hai điều kiện này thỏa mãn
- Đẩy các phần tử từ vị trí t (vị trí của phần tử bị xóa) sang bên trái một đơn vị
- Giảm kích thước hiện hữu đi 1

Ví dụ 5.9: Xóa phần tử ra khỏi mảng

```

int delFirst(int a[], int &n){ /*Xoa phan tu dau tien cua mang a, kich thuoc n*/
    int result = 0, i;
    if (n > 0) {
        /*Day cac phan tu sang trai mot don vi*/
        for (i = 0; i < n-1; i++) a[i] = a[i+1];
    }
}

```

```

        /*Giam kich thuoc cua mang di 1*/
        n--; result = 1;
    }
    return result;
}

int delLast(int a[], int &n){ /*Xoa phan tu cuoi cung cua mang a, kich thuoc n*/
    int result = 0, i;
    if (n > 0) {
        /*Giam kich thuoc cua mang di 1*/
        n--; result = 1;
    }
    return result;
}

int DelAtIndex(int a[], int &n, int t){ /*Xoa phan tu tai vi tri t cua mang a, kich thuoc n*/
    int result = 0, i;
    if (n > 0 && t >= 0 && t < n) {
        /*Day cac phan tu sau t sang trai mot don vi*/
        for (i = t; i < n - 1; i++)
            a[i] = a[i+1];
        /*Giam kich thuoc cua mang di 1*/
        n--; result = 1;
    }
    return result;
}

```

5.3 Mảng nhiều chiều

Ngôn ngữ lập trình C/C++ hỗ trợ mảng nhiều chiều. Dạng đơn giản nhất của mảng nhiều chiều là mảng hai chiều. Mảng hai chiều thực chất là mảng của những mảng một chiều. Mảng hai chiều có thể được xem là một ma trận gồm các hàng và các cột.

5.3.1 Khai báo mảng hai chiều

Dạng tổng quát để khai báo một mảng hai chiều là:

type arrayName[rows][columns];

type: kiểu dữ liệu của mỗi phần tử mảng

rows: số hàng trong mảng

columns: số cột trong mảng

arrayName: tên mảng

Giống như những biến khác, mảng hai chiều phải được khai báo tường minh để cho trình biên dịch có thể cấp phát bộ nhớ cho nó. Kích thước (tính bằng byte) của mảng hai chiều được tính theo công thức:

$$\text{totalSize} = \text{sizeof(type)} * \text{rows} * \text{columns}$$

Ví dụ 5.10: Khai báo mảng hai chiều

Câu lệnh sau đây dùng để khai báo mảng b có 10 hàng, 10 cột:

```
int b[10][10];
```

Với kích thước của 1 số nguyên là 2 byte, mảng b có kích thước là $2*10*10=200$ byte. Sau khi được khai báo, mỗi phần tử trong mảng được xem như một biến thông thường.

5.3.2 Khai báo và khởi tạo mảng hai chiều

Ngoài ra, ta còn có thể vừa khai báo vừa khởi tạo các phần tử của mảng hai chiều. Dạng tổng quát như sau:

```
type arrayName[][][columns] = { {value1, value2, ..., valueN},  
                                {value1, value2, ..., valueN},  
                                {...}  
                                {value1, value2, ..., valueN}};
```

Chú ý:

- Số phần tử của mỗi hàng phải bằng số cột (columns)
- Số hàng (rows) của khai báo mảng hai chiều để trống
- Số hàng của mảng được xác định dựa vào số hàng trong phần khởi tạo
- Giá trị các phần tử trong mỗi hàng được đặt trong cặp {}, các hàng phân cách nhau bằng một dấu phẩy

Ví dụ 5.11: Khai báo và khởi tạo mảng hai chiều

Câu lệnh sau đây dùng để khai báo và khởi tạo mảng hai chiều b với 3 hàng, 4 cột với 12 số nguyên

```
int b[][4] = {{1,2,3,4},{3,4,5,6},{6,7,8,9}};
```

Mảng b có 12 phần tử, chia làm 3 hàng, mỗi hàng có 4 phần tử (ứng với 4 cột) như sau:

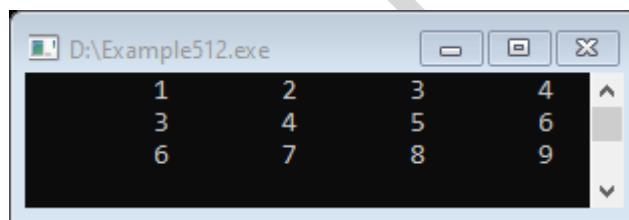
Hàng 1 có 4 phần tử $b[0][0] = 1, b[0][1] = 2, b[0][2] = 3, b[0][3] = 4$

Hàng 2 có 4 phần tử $b[1][0] = 3, b[1][1] = 4, b[1][2] = 5, b[1][3] = 6$

Hàng 3 có 4 phần tử $b[2][0] = 6$, $b[2][1] = 7$, $b[2][2] = 8$, $b[2][3] = 9$

Ví dụ 5.12: Khai báo, khởi tạo và truy xuất mảng hai chiều

```
#include <stdio.h>
int main(){
    int i, j, b[][4] = {{1,2,3,4},{3,4,5,6},{6,7,8,9}};
    for (i = 0; i < 3; i++){
        for (j = 0; j < 4; j++)
            printf("\t%d",b[i][j]);
        printf("\n");
    }
    getchar();
    return 0;
}
```



5.4 Chương trình minh họa sử dụng mảng

Yêu cầu: Phát triển chương trình C giúp quản lý mảng một chiều gồm tối đa 100 số nguyên sử dụng thực đơn sau đây:

- 1- Thêm một phần tử vào cuối mảng
- 2- Xóa phần tử tại một vị trí trong mảng
- 3- Tìm kiếm phần tử trong mảng
- 4- In tất cả phần tử trong mảng
- 5- Thoát khỏi chương trình

```
#include <stdio.h>
#define MAXSIZE 1000
int addLast(int a[], int &n, int x);
int findIndex(int a[], int n, int x);
int delAdIndex(int a[], int &n, int t);
void printArray(int a[], int n);
int main(){
    int a[MAXSIZE], n=0, choice, t, x, pos;
    printf("Chao mung den voi chuong trinh minh hoa MANG trong C\n");
    do
    {
        printf("-----\n");
        printf("1-Them so nguyen vao cuoi mang\n");
        printf("2-Xoa so nguyen trong mang\n");
        printf("3-Tim so nguyen trong mang\n");
        printf("4-In mang\n");
        printf("5-Thoat\n");
        printf("-----\n");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1:
                printf("Nhap so nguyen can them: ");
                scanf("%d", &x);
                addLast(a, &n, x);
                break;
            case 2:
                printf("Nhap vi tri can xoa: ");
                scanf("%d", &t);
                delAdIndex(a, &n, t);
                break;
            case 3:
                printf("Nhap so can tim: ");
                scanf("%d", &x);
                pos = findIndex(a, n, x);
                if (pos == -1)
                    printf("Khong tim thay so %d\n", x);
                else
                    printf("So %d co vi tri %d\n", x, pos);
                break;
            case 4:
                printArray(a, n);
                break;
            case 5:
                exit(0);
        }
    } while (choice != 5);
}
```

```

printf("4-In mang\n");
printf("5-Thoat\n");
printf("\n-----\n");
printf("Vui long chon: "); scanf("%d", &choice);
switch (choice){
    case 1:
        printf("Nhap mot so nguyen: "); scanf("%d", &x);
        if (addLast(a, n, x)==1)
            printf("Them %d vao mang thanh cong", x);
        else
            printf("Khong them %d vao mang duoc", x);
        break;
    case 2:
        printf("Nhap vi tri muon xoa: "); scanf("%d", &t);
        if (delAdIndex(a, n, t)==1)
            printf("Xoa duoc vi tri %d trong mang", t);
        else
            printf("Khong xoa duoc vi tri %d trong mang", t);
        break;
    case 3:
        printf("Nhap so nguyen muon tim: "); scanf("%d", &x);
        if (findIndex(a,n,x) != -1)
            printf("Tim thay %d trong mang", x);
        else
            printf("Khong tim thay %d trong mang", x);
        break;
    case 4:
        printf("Cac phan tu trong mang: ");
        printArray(a,n);
        break;
    case 5:
        printf("Cam on da su dung\n");
        break;
    default:
        printf("Vui long nhap cac so tu 1 den 5\n");
        break;
}
}

while (choice != 5);
getchar();
return 0;
}
/*

```

Cài đặt các hàm (xem ở trên)
*/

5.5 Câu hỏi ôn tập

- 1) Trình bày khái niệm mảng một chiều
- 2) Khai báo, khởi tạo và truy xuất phần tử trong mảng một chiều
- 3) Trình bày cách thức sử dụng mảng một chiều như tham số của hàm
- 4) Trình bày thao tác thêm (xóa) phần tử vào (ra khỏi) mảng một chiều
- 5) Trình bày khái niệm mảng hai chiều, cách khai báo, khởi tạo và truy xuất phần tử trong mảng hai chiều
- 6) Hãy cho biết kết quả xuất ra màn hình khi chạy đoạn chương trình sau đây. Giải thích.

```
int a[] = {1,2,3,4,5,6,7,8,9,10}, s=0;  
for (int i=0; i<10; i++){  
    s += (a[i]>2!=0?1:0);  
}  
printf("s=%d", s);
```

- 7) Hãy cho biết kết quả xuất ra màn hình khi chạy đoạn chương trình sau đây. Giải thích.

```
int a[] = {1,2,3,4,5,6,7,8,9,10}, s=0;  
for (int i=1; i<=10; i++){  
    if (a[i-1]>2==0) s+=i*a[i-1];  
}  
printf("s=%d", s);
```

- 8) Hãy cho biết kết quả xuất ra màn hình khi chạy đoạn chương trình sau đây. Giải thích.

```
int a[] = {1,2,3,4,5,6,7,8,9,10}, p=1, i=0;  
while (i<10){  
    if (p>10) break;  
    else p*=a[i];  
    i++;  
}  
printf("p=%d", p);
```

- 9) Hãy cho biết kết quả xuất ra màn hình khi chạy đoạn chương trình sau đây. Giải thích.

```
int a[] = {1,2,3,4,5,6,7,8,9,10}, p=1, i=0;  
do {  
    p *= a[i++];  
} while (i<10 && p<10);  
printf("p=%d", p);
```

10) Hãy cho biết kết quả xuất ra màn hình khi chạy đoạn chương trình sau đây. Giải thích.

```
int a[][3] = {{1,2,3},{4,5,6},{7,8,9}}, p=1, i=0, j=0;
do {
    p *= a[i++][j++];
} while (i<3 && j<3);
printf("p=%d", p);
```

5.6 Bài tập thực hành

- 1) Viết chương trình nhập vào hai số nguyên dương n và m sau đó nhập tiếp ma trận các số nguyên gồm n hàng, m cột. Xuất ma trận đó ra màn hình.

```
#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE 100
void scanMatrix(int a[][MAXSIZE], int n, int m);
void printMatrix(int a[][MAXSIZE], int n, int m);
int main(){
    int a[MAXSIZE][MAXSIZE], n, m;
    do {
        printf("Nhập kích thước của ma trận (n m): ");
        scanf("%d %d", &n, &m);
    } while (n<=0 || m<=0);
    scanMatrix(a,n,m);
    printMatrix(a,n,m);
    getchar();
    return 0;
}
void scanMatrix(int a[][MAXSIZE], int n, int m){
    int i, j;
    printf("\nNhập ma trận ======\n");
    for (i=0; i<n; i++){
        for (j=0; j<m; j++){
            printf("a[%d][%d] = ", i, j); scanf("%d", &a[i][j]);
        }
    }
}
void printMatrix(int a[][MAXSIZE], int n, int m){
    int i, j;
    printf("\nXuất ma trận ======\n");
    for (i=0; i<n; i++){
        for (j=0; j<m; j++){
            printf("%d\t", a[i][j]);
        }
    }
}
```

```
        printf("\n");
    }
}
```

- 2) Viết chương trình nhập vào một mảng số nguyên và kiểm tra tính đối xứng, tính có thứ tự của nó.

```
#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE 1000
void scanArray(int a[], int n);
int testSymmetry(int a[], int n);
int testAsc(int a[], int n);
int testDesc(int a[], int n);
int main(){
    int a[MAXSIZE], n;
    do {
        printf("Nhập kích thước của mảng: "); scanf("%d", &n);
    } while (n<=0);
    scanArray(a,n);
    if (testSymmetry(a,n)==1) printf("\nMảng đối xứng");
    else printf("\nMảng không đối xứng");
    if (testAsc(a,n)==1 || testDesc(a,n)==1)
        printf("\nMảng có thu tu");
    else printf("\nMảng không có thu tu");
    getchar();
    return 0;
}
void scanArray(int a[], int n){ //Ham nhap mang a, kich thuoc n tu ban phim
    int i;
    printf("\nNhập mang ======\n");
    for (i=0; i<n; i++){
        printf("a[%d] = ", i); scanf("%d", &a[i]);
    }
}
int testSymmetry(int a[], int n){ //Ham kiem tra tinh doi xung cua mang a, kich thuoc n
    int res=1, i;
    for (i=0; i<n/2 && res==1; i++) if (a[i] != a[n-i-1]) res=0;
    return res;
}
int testAsc(int a[], int n){ //Ham kiem tra tinh tang dan cua mang a, kich thuoc n
    int i, res=1;
    for (i=0; i<n-1 && res==1; i++) if (a[i] > a[i+1]) res = 0;
```

```

        return res;
    }
int testDesc(int a[], int n){ //Ham kiem tra tinh giam dan cua mang a, kich thuoc n
    int i, res=1;
    for (i=0; i<n-1 && res==1; i++) if (a[i] < a[i+1]) res = 0;
    return res;
}

```

- 3) Viết chương trình nhập vào hai ma trận vuông gồm m hàng, m cột. Xuất ra màn hình hai ma trận đã nhập và ma trận tổng, tích của hai ma trận đó.

```

#include <stdio.h>
#define MAXSIZE 100
void scanMatrix(int a[][MAXSIZE], int n);
void printMatrix(int a[][MAXSIZE], int n);
void printSum(int a[][MAXSIZE], int b[][MAXSIZE], int n);
void printProduct(int a[][MAXSIZE], int b[][MAXSIZE], int n);
int main(){
    int a[MAXSIZE][MAXSIZE], b[MAXSIZE][MAXSIZE], n;
    do {
        printf("Nhập kích thước của ma trận vuông (n): "); scanf("%d", &n);
    } while (n<=0);
    scanMatrix(a,n);
    scanMatrix(b,n);
    printSum(a,b,n);
    printProduct(a,b,n);
    getchar();
    return 0;
}
void scanMatrix(int a[][MAXSIZE], int n){ //Ham nhap ma tran vuong a, kich thuoc n
    int i, j;
    printf("\nNhập ma tran ======\n");
    for (i=0; i<n; i++){
        for (j=0; j<n; j++){
            printf("a[%d][%d] = ", i, j); scanf("%d", &a[i][j]);
        }
    }
}
void printMatrix(int a[][MAXSIZE], int n){ //Ham xuat ma tran vuong a, kich thuoc n
    int i, j;
    printf("\nXuat ma tran ======\n");
    for (i=0; i<n; i++){
        for (j=0; j<n; j++){
            printf("%d\t", a[i][j]);
        }
    }
}

```

```

        }
        printf("\n");
    }
}

void printSum(int a[][MAXSIZE], int b[][MAXSIZE], int n){ //Ham xuat tong hai ma tran
    int i, j;
    for (i=0; i<n; i++){
        for (j=0; j<n; j++)
            printf("%d\t", a[i][j] + b[i][j]);
        printf("\n");
    }
}

void printProduct(int a[][MAXSIZE], int b[][MAXSIZE], int n){ //Ham xuat tich hai ma tran
    int c, k, i, j;
    for (i=0; i<n; i++){
        for (j=0; j<n; j++){
            c = 0;
            for (k=0; k<n; k++)
                c += a[i][k] * b[k][j];
            printf("%d\t", c);
        }
        printf("\n");
    }
}

```

4) Viết chương trình nhập vào số nguyên dương n và xuất ra màn hình n số nguyên tố.

```

#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE 1000
void createArray(int a[], int n);
int isPrime(int current);
void printArray(int a[], int n);
int main(){
    int a[MAXSIZE], n;
    do {
        printf("Nhập kích thước của mảng: "); scanf("%d", &n);
    } while (n<=0);
    createArray(a,n);
    printArray(a,n);
    getchar();
    return 0;
}

```

```

int isPrime(int current){ //Ham kiem tra current la so nguyen to
    int res=1, i=0;
    for (i=2; i*i <= current && res == 1; i++)
        if (current%i == 0) res = 0;
    return res;
}
void createArray(int a[], int n){ //Ham tao mang a, chua n so nguyen to dau tien
    int i, current=2;
    for (i=0; i<n; current++){
        if (isPrime(current)==1) { a[i] = current; i++; }
    }
}
void printArray(int a[], int n){ //Ham xuat mang a, kich thuoc n
    int i;
    printf("\nXuat mang ======\n");
    for (i=0; i<n; i++){
        printf("%d\t", a[i]);
    }
}

```

5) Viết chương trình nhập vào mảng gồm các số nguyên dương và xuất ra màn hình mảng các số đã nhập sau khi đã sắp xếp theo trật tự tăng dần.

```

#include <stdio.h>
#define MAXSIZE 1000
void scanArray(int a[], int n);
void sortArray(int a[], int n);
void printArray(int a[], int n);
int main(){
    int a[MAXSIZE], n;
    do {
        printf("Nhập kích thước của mảng: "); scanf("%d", &n);
    } while (n<=0);
    scanArray(a,n);
    sortArray(a,n);
    printArray(a,n);
    getchar();
    return 0;
}
void scanArray(int a[], int n){ //Ham nhap mang a, kich thuoc n tu ban phim
    int i;
    printf("\nNhập mang ======\n");
    for (i=0; i<n; i++) {

```

```

        printf("a[%d] = ", i); scanf("%d", &a[i]);
    }
}
void sortArray(int a[], int n){      //Ham sap xep mang a, kich thuoc n
    int i, j, minIndex, minTemp;
    for (i=0; i<n-1; i++){
        minIndex = i;
        for (j=i+1; j<n; j++) if (a[minIndex] > a[j]) minIndex = j;
        if (minIndex > i) {
            minTemp = a[i];
            a[i] = a[minIndex];
            a[minIndex] = minTemp;
        }
    }
}
void printArray(int a[], int n){      //Ham xuat mang a, kich thuoc n
    int i;
    printf("\nXuat mang =====\n");
    for (i=0; i<n; i++){
        printf("%d\t", a[i]);
    }
}

```

5.7 Bài tập đề nghị

- 1) Viết chương trình nhập vào một mảng số nguyên và xuất ra màn hình:
 - Phần tử lớn nhất của mảng
 - Phần tử nhỏ nhất của mảng
 - Tính tổng của các phần tử trong mảng
 - Tính tích của các phần tử trong mảng
 - 2) Viết chương trình nhập vào một mảng số nguyên theo thứ tự tăng dần, nếu nhập sai quy cách thì yêu cầu nhập lại. Xuất ra màn hình mảng sau khi đã nhập xong.
- > Give a number to array (0 to quit): 4
- > Give a number to array (0 to quit): 3
- > Ops! Your number broke the rule! Try again!
- > Give a number to array (0 to quit): 4
- > Give a number to array (0 to quit): 6
- > Give a number to array (0 to quit): 0

> Your inputted array: 4 4 6

> Bye!

- 3) Viết chương trình nhập vào một ma trận gồm m hàng, n cột. Xuất ra màn hình ma trận đã nhập và ma trận chuyển vị của nó.
- 4) Viết chương trình nhập vào một mảng số nguyên và cho biết số lần xuất hiện của mỗi giá trị phân biệt trong mảng.

Ví dụ:

> Input an array of integers (0 to stop): 1 2 5 7 6 2 3 2 3 1 0

> Distinct values and its frequencies:

> 1 2

> 2 3

> 5 1

> 7 1

> 6 1

> 3 2

- 5) Mã số xuất bản ISBN có đúng 10 chữ số. Việc xác định tính hợp lệ của mã số ISBN được thực hiện như sau:

- Nhân mỗi chữ số từ chữ số đầu tiên tới chữ số thứ chín với trọng số giảm dần từ 10 xuống 2 tương ứng.
- Tổng của các phép nhân nói trên cộng với chữ số thứ mười phải chia hết cho 11; nếu phép chia này có dư, số ban đầu không phải là mã số ISBN hợp lệ.

Viết chương trình nhập vào một mảng có 10 chữ số và cho biết mảng này có phải là một mã số ISBN hợp lệ hay không.

Ví dụ:

> ISBN validator =====

> ISBN (0 to quit): 0003194876

> This is a valid ISBN

> ISBN (0 to quit): 0003194875

> This is not a valid ISBN

- > ISBN (0 to quit): 0
- > Have a nice day
- 6) Viết chương trình nhập vào một ma trận vuông gồm m hàng, m cột. Kiểm tra tính đối xứng của nó.
- 7) Viết chương trình nhập vào hai mảng số nguyên a và b. Kiểm tra mảng b có phải là mảng con của mảng a hay không.
- > Input the first array (0 to quit): 1 3 8 7 6 4 0
- > Input the second array (0 to quit): 1 3 8 0
- > Yeah! The second array is a part of the first array.
- > Good job.
- > Do it again (y/n)? y
- > Input the first array (0 to quit): 1 3 8 7 6 4 0
- > Input the second array (0 to quit): 1 8 3 0
- > Ops! The second array is not a part of the first array.
- > Do it again (y/n)? n
- > Bye!
- 8) Viết chương trình nhập vào một mảng số nguyên sao cho mỗi số nguyên chỉ xuất hiện một lần trong mảng.
- > Give a number to array (0 to stop): 3
- > Give a number to array (0 to stop): 2
- > Give a number to array (0 to stop): 2
- > Ops! 2 has been already existed in your array. Try again.
- > Give a number to array (0 to stop): 6
- > Give a number to array (0 to stop): 0
- > Your inputted array: 3 2 6
- > Have a nice day!
- 9) Viết chương trình nhập vào hai mảng số nguyên và nối hai mảng này thành một mảng duy nhất. Xuất ra màn hình mảng kết quả.
- > Input the first array (0 to quit): 1 2 5 4 0

> Input the second array (0 to quit): 5 3 2 0

> Concatenate them: 1 2 5 4 5 3 2

> Good job!

10) Viết chương trình nhập vào một mảng số nguyên. Xuất ra màn hình các số nguyên và số lần xuất hiện của chúng trong mảng đã nhập.

CONFIDENTIAL

CHƯƠNG 6. CON TRỎ

Trong chương này người học được cung cấp các kiến thức, kỹ năng cơ bản về việc sử dụng con trỏ trong lập trình.

Nội dung:

- Con trỏ và địa chỉ
- Khai báo con trỏ
- Con trỏ và mảng một chiều
- Con trỏ và mảng nhiều chiều
- Mảng các con trỏ
- Con trỏ hàm
- Cấp phát bộ nhớ động

6.1 Khái niệm con trỏ

Một **con trỏ** là một biến, nó chứa địa chỉ vùng nhớ của một biến khác chứ không lưu trữ giá trị của biến đó. Khi một biến chứa địa chỉ của một biến khác, biến này được gọi là **con trỏ** đến biến kia. Một con trỏ cung cấp phương thức gián tiếp để truy xuất giá trị của các phần tử dữ liệu (Ví dụ 6.1).

Ví dụ 6.1. Biến giá trị và biến con trỏ

Xét hai biến var1 và var2, var1 có giá trị 5 và được lưu tại địa chỉ 1000 trong bộ nhớ. Nếu var2 được khai báo như là một con trỏ tới biến var1, sự biểu diễn sẽ như sau:

| Vị trí bộ nhớ | Giá trị lưu trữ | Tên biến |
|---------------|-----------------|----------|
| ... | | |
| 1000 | 5 | var1 |
| 1001 | | |
| ... | | |
| 1108 | 1000 | var2 |
| ... | | |

Các con trỏ có thể trỏ đến các biến của các kiểu dữ liệu cơ sở như **int**, **char**, hay **double** hoặc dữ liệu có cấu trúc như **mảng**. Con trỏ có thể được sử dụng trong một số trường hợp sau:

- Để trả về nhiều hơn một giá trị từ một hàm.
- Truyền các mảng từ một hàm đến một hàm khác.
- Để cấp phát bộ nhớ động và truy xuất vào vùng nhớ được cấp phát này.

6.2 Biến con trỏ

Nếu một biến được sử dụng như một con trỏ, nó phải được khai báo trước. Câu lệnh khai báo con trỏ bao gồm một kiểu dữ liệu cơ bản, một dấu *, và một tên biến. Cú pháp tổng quát để khai báo một biến con trỏ như sau:

type *pointerVariable;

Trong đó:

type: là một kiểu dữ liệu hợp lệ bất kỳ

pointerVariable: là tên của biến con trỏ, biến này lưu địa chỉ của một biến có kiểu dữ liệu **type**

Ví dụ 6.2. Khai báo biến con trỏ

Trong *Ví dụ 6.1*, vì **var2** là một biến con trỏ lưu trữ địa chỉ của biến **var1** có kiểu **int**, nó sẽ được khai báo như sau:

int *var2;

Bây giờ, **var2** có thể được sử dụng trong một chương trình để trực tiếp truy xuất giá trị của **var1**.

Chú ý: **var2** không phải là một biến có kiểu dữ liệu **int** mà là một biến con trỏ trỏ đến một biến có kiểu dữ liệu **int**.

Kiểu dữ liệu cơ sở của con trỏ xác định kiểu của biến mà con trỏ trỏ đến. Về mặt kỹ thuật, một con trỏ có thể trỏ đến bất kỳ vị trí nào trong bộ nhớ. Tuy nhiên, tất cả các phép toán số học trên con trỏ đều có liên quan đến kiểu cơ sở của nó, vì vậy khai báo kiểu dữ liệu của con trỏ một cách rõ ràng là điều rất quan trọng.

6.3 Các toán tử con trỏ

Hai toán tử đặc biệt được dùng với con trỏ, đó là toán tử **&** và *****. Toán tử **&** là một toán tử một ngôi và nó trả về địa chỉ của toán hạng. Hai toán tử này có cùng độ ưu tiên như toán tử âm (-) và cao hơn các toán tử số học còn lại.

Ví dụ 6.3. Toán tử &

Xét hai câu lệnh:

(1) int var1 = 5, *var2;

(2) var2 = &var1;

- Câu lệnh (1) khai báo biến var1 (số nguyên) có giá trị khởi tạo là 5, biến con trỏ var2 trỏ đến biến số nguyên.
- Câu lệnh (2) lấy địa chỉ bộ nhớ của biến var1 gán cho biến con trỏ var2.
- Toán tử **&** có thể hiểu là trả về “địa chỉ của”. Vì vậy, phép gán (2) có nghĩa là “var2 nhận địa chỉ của var1”. Giả sử địa chỉ trong bộ nhớ của biến var1 là **1000**. Sau phép gán (2), **var2** sẽ có giá trị **1000**.

Toán tử ***** được dùng với con trỏ là phần bổ sung của toán tử **&**. Nó là một toán tử một ngôi và trả về giá trị chứa trong địa chỉ của biến được trỏ tới bởi biến con trỏ.

Ví dụ 6.4. Toán tử *

Xét ba câu lệnh:

(1) int var1 = 5, *var2;

(2) var2 = &var1;

(3) int var3 = *var2;

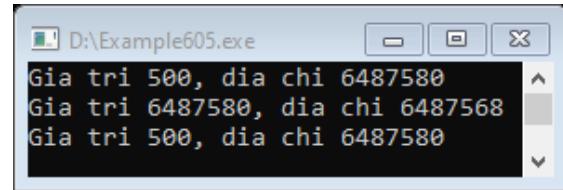
- Câu lệnh (1) khai báo biến var1 (số nguyên) có giá trị khởi tạo là 5, biến con trỏ var2 trỏ đến biến số nguyên.
- Câu lệnh (2) lấy địa chỉ bộ nhớ của biến var1 gán cho biến con trỏ var2.
- Câu lệnh (3) khai báo biến giá trị var3 (số nguyên) và khởi tạo bằng cách gán giá trị được lưu trữ tại địa chỉ của biến được trỏ tới bởi biến con trỏ var2.

Toán tử ***** có thể hiểu là “tại địa chỉ”. Vì vậy, phép gán (3) có nghĩa là “var3 nhận giá trị được trỏ tới bởi var2”. Giả sử var2 có giá trị là 1000, var3 sẽ nhận giá trị được lưu trữ tại

địa chỉ 1000 trong bộ nhớ và giá trị này hiện tại đang là 5. Sau phép gán (3), **var3** sẽ có giá trị **5**.

Ví dụ 6.5. Giá trị và địa chỉ của biến con trỏ

```
#include <stdio.h>
int main(){
    int var = 500, *ptr_var;
    ptr_var = &var;
    /*Xuat gia tri va dia chi cua bien var*/
    printf("Gia tri %d, dia chi %u\n", var, &var);
    /*Xuat gia tri va dia chi cua bien ptr_var*/
    printf("Gia tri %u, dia chi %u\n", ptr_var, &ptr_var);
    /*Xuat gia tri va dia chi cua bien duoc tro toi boi bien con trỏ ptr_var*/
    printf("Gia tri %d, dia chi %u\n", *ptr_var, ptr_var);
    getchar();
    return 0;
}
```



6.4 Các phép toán liên quan đến con trỏ

6.4.1 Gán giá trị cho biến con trỏ

Giá trị của biến con trỏ chính là địa chỉ của biến được nó trỏ tới, do đó phép gán giá trị cho biến con trỏ có thể thực hiện theo nhiều cách, bao gồm gán địa chỉ của biến giá trị, gán giá trị của biến con trỏ khác và gán giá trị NULL (Ví dụ 6.6).

Ví dụ 6.6. Gán giá trị cho biến con trỏ

Xét các câu lệnh:

```
int var = 5, *ptr1, *ptr2, *ptr3; //Khai báo biến giá trị và các biến con trỏ
ptr1 = &var; //Địa chỉ của var được lưu trong biến ptr1
ptr2 = ptr1; //Giá trị của biến ptr1 được gán cho biến ptr2
ptr3 = 0; //Giá trị của biến ptr3 là NULL
```

Nhìn chung, các biểu thức có chứa con trỏ tuân theo quy tắc như các biểu thức khác trong ngôn ngữ C/C++. Điều quan trọng cần chú ý là phải gán giá trị cho biến con trỏ trước khi sử dụng chúng; nếu không chúng có thể trỏ đến một giá trị không xác định nào đó.

6.4.2 Phép toán số học con trỏ

Biến con trỏ là một loại biến đặc biệt với giá trị là địa chỉ trong bộ nhớ (thuộc kiểu số nguyên không dấu – không âm). Do đó, các phép toán số học có thể áp dụng với biến con trỏ là phép cộng và trừ các số không âm mà thôi.

Bảng 6.1. Các phép toán số học con trỏ (ptr là biến con trỏ trả về biến số nguyên var)

| | |
|---|---------------------------------------|
| <code>++ptr_var or ptr_var++</code> | TrỎ ĐẾN SỐ NGUYÊN ĐỨNG NGAY SAU VAR |
| <code>--ptr_var or ptr_var--</code> | TRỎ ĐẾN SỐ NGUYÊN ĐỨNG NGAY TRƯỚC VAR |
| <code>ptr_var + i</code> | TRỎ ĐẾN SỐ NGUYÊN THỨ i SAU VAR |
| <code>ptr_var - i</code> | TRỎ ĐẾN SỐ NGUYÊN THỨ i TRƯỚC VAR |
| <code>++*ptr_var or (*ptr_var)++</code> | TƯƠNG ĐƯƠNG VỚI VAR++ |

Ví dụ 6.7. Phép toán số học con trỏ

Xét ba câu lệnh:

```
int var, *ptr; //Giá trị var, ptr lưu tại địa chỉ 1000 và 1100  
ptr = &var; //Giá trị của ptr là 1000  
ptr++; //Giá trị mới của ptr là 1002
```

Giá trị sau cùng của biến con trỏ ptr là 1002 chứ không phải 1001 là do kích thước của một số nguyên là 2 byte. Điều này có nghĩa là ptr trỏ đến một số nguyên được lưu tại địa chỉ 1002. Mỗi khi ptr được tăng lên, nó sẽ trỏ đến số nguyên kế tiếp và bởi vì các số nguyên là 2 bytes, ptr sẽ được tăng trị là 2. Điều này cũng tương tự với phép toán giảm trị.

6.4.3 So sánh con trỏ

Hai con trỏ có thể được so sánh trong một biểu thức quan hệ. Tuy nhiên, điều này chỉ có thể nếu cả hai biến này đều trỏ đến các biến có cùng kiểu dữ liệu. Trong trường `ptr_a` và `ptr_b` là hai biến con trỏ trỏ đến các phần tử dữ liệu `a` và `b` cùng kiểu, Bảng 6.2 liệt kê một số phép so sánh có thể thực hiện:

Bảng 6.2. Các phép so sánh giữa hai biến con trỏ

| | |
|-------------------------------|---|
| <code>ptr_a < ptr_b</code> | TRẢ VỀ GIÁ TRỊ TRUE NẾU a ĐƯỢC LƯU TRỮ Ở VỊ TRÍ TRƯỚC b |
|-------------------------------|---|

| | |
|--------------------------------|--|
| <code>ptr_a > ptr_b</code> | Trả về giá trị true nếu a được lưu trữ ở vị trí sau b |
| <code>ptr_a <= ptr_b</code> | Trả về giá trị true nếu a được lưu trữ ở vị trí trước b hoặc <code>ptr_a</code> và <code>ptr_b</code> trỏ đến cùng một vị trí |
| <code>ptr_a >= ptr_b</code> | Trả về giá trị true nếu a được lưu trữ ở vị trí sau b hoặc <code>ptr_a</code> và <code>ptr_b</code> trỏ đến cùng một vị trí |
| <code>ptr_a == ptr_b</code> | Trả về giá trị true nếu cả hai con trỏ <code>ptr_a</code> và <code>ptr_b</code> trỏ đến cùng một phần tử dữ liệu |
| <code>ptr_a != ptr_b</code> | Trả về giá trị true nếu cả hai con trỏ <code>ptr_a</code> và <code>ptr_b</code> trỏ đến các phần tử dữ liệu khác nhau nhưng có cùng kiểu dữ liệu |
| <code>ptr_a == NULL</code> | Trả về giá trị true nếu <code>ptr_a</code> được gán giá trị <code>NULL</code> (0) |

6.4.4 Con trỏ và mảng một chiều

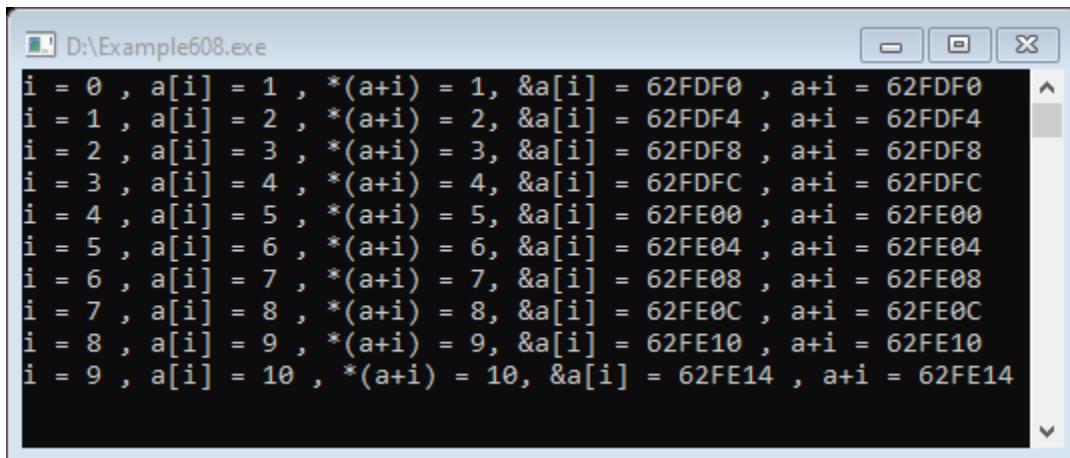
Tên của một mảng thật ra là một con trỏ trỏ đến phần tử đầu tiên của mảng đó. Vì vậy, nếu **a** là một mảng một chiều, thì địa chỉ của phần tử đầu tiên trong mảng (**a[0]**) có thể được biểu diễn là **&a[0]** hoặc đơn giản chỉ là **a**. Tương tự, địa chỉ của phần tử mảng thứ hai có thể được viết như **&a[1]** hoặc **a + 1**, ... Tổng quát, địa chỉ của phần tử mảng thứ (*i* + 1) có thể được biểu diễn là **&a[i]** hay **(a + i)**. Ví dụ 6.8 xuất giá trị và địa chỉ của các phần tử trong mảng một chiều.

Ví dụ 6.8. Con trỏ và mảng một chiều

```
#include<stdio.h>
int main(){
    int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int i;
    for (i = 0; i < 10; i++){
        printf("i = %d , a[i] = %d , *(a+i) = %d , ", i, a[i], *(a+i));
        printf("&a[i] = %X , a+i = %X\n", &a[i], a+i);
    }
    getchar();
    return 0;
}
```

Chương trình trên định nghĩa mảng một chiều **a**, có 10 phần tử kiểu số nguyên, các phần tử mảng được gán giá trị tương ứng là 1, 2, ..10. Vòng lặp **for** được dùng để hiển thị giá trị và địa chỉ tương ứng của mỗi phần tử mảng. Chú ý rằng, giá trị của mỗi phần tử được xác định theo hai cách khác nhau, **a[i]** và ***(a+i)**, nhằm minh họa sự tương đương của

chúng. Tương tự, địa chỉ của mỗi phần tử mảng cũng được hiển thị theo hai cách. Kết quả của chương trình trên như sau:



```
D:\Example608.exe
i = 0 , a[i] = 1 , *(a+i) = 1, &a[i] = 62FDF0 , a+i = 62FDF0
i = 1 , a[i] = 2 , *(a+i) = 2, &a[i] = 62FDF4 , a+i = 62FDF4
i = 2 , a[i] = 3 , *(a+i) = 3, &a[i] = 62FDF8 , a+i = 62FDF8
i = 3 , a[i] = 4 , *(a+i) = 4, &a[i] = 62FDFA , a+i = 62FDFA
i = 4 , a[i] = 5 , *(a+i) = 5, &a[i] = 62FE00 , a+i = 62FE00
i = 5 , a[i] = 6 , *(a+i) = 6, &a[i] = 62FE04 , a+i = 62FE04
i = 6 , a[i] = 7 , *(a+i) = 7, &a[i] = 62FE08 , a+i = 62FE08
i = 7 , a[i] = 8 , *(a+i) = 8, &a[i] = 62FE0C , a+i = 62FE0C
i = 8 , a[i] = 9 , *(a+i) = 9, &a[i] = 62FE10 , a+i = 62FE10
i = 9 , a[i] = 10 , *(a+i) = 10, &a[i] = 62FE14 , a+i = 62FE14
```

Khi gán một giá trị cho một phần tử mảng như **a[i]**, vé trái của lệnh gán có thể được viết là **a[i]** hoặc ***(a+i)**. Vì vậy, một giá trị có thể được gán trực tiếp đến một phần tử mảng hoặc nó có thể được gán đến vùng nhớ mà địa chỉ của nó là phần tử mảng. Đôi khi cần thiết phải gán một địa chỉ đến một định danh. Trong những trường hợp như vậy, một con trỏ phải xuất hiện trong vé trái của câu lệnh gán. Không thể gán một địa chỉ tùy ý cho một tên mảng hoặc một phần tử của mảng. Vì vậy, các biểu thức như **a**, **(a+i)** và **&a[i]** không thể xuất hiện trong vé trái của một câu lệnh gán. Hơn thế nữa, địa chỉ của một mảng không thể thay đổi một cách tùy ý, vì thế các biểu thức như **a++** là không được phép. Lý do là vì: **a** là địa chỉ của mảng **a** và khi mảng được khai báo, địa chỉ mảng đã được xác định và không thay đổi; việc cố gắng tăng/giảm địa chỉ mảng là điều vô nghĩa.

6.5 Con trỏ và mảng nhiều chiều

Một mảng nhiều chiều cũng có thể được biểu diễn dưới dạng con trỏ của mảng một chiều (tên của mảng) với một giá trị chỉ số. Điều này được thực hiện trên cơ sở mảng nhiều chiều là một tập hợp của các mảng một chiều. Chẳng hạn, một mảng hai chiều có thể được định nghĩa như là một con trỏ trỏ đến mảng, trong đó mỗi phần tử là một mảng một chiều. Cú pháp khai báo mảng hai chiều có thể viết như sau:

```
type (*ptr_a)[expr_2];
```

thay vì

```
type arr_a[expr_1][expr_2];
```

Khai báo này có thể được tổng quát hóa cho các mảng nhiều chiều:

```
type (*ptr_a)[exp_2]...[exp_N];
```

thay vì

```
type arr_a[exp_1][exp_2]...[exp_N];
```

trong đó,

type: kiểu dữ liệu của mảng

ptr_a: tên của biến con trỏ,

arr_a: tên mảng

exp_1, exp_2, ..., exp_N: số lượng phần tử tối đa trong mỗi chiều của mảng

Ví dụ 6.9. Con trỏ và mảng nhiều chiều

Một số khai báo và truy xuất mảng nhiều chiều:

```
int (*ptr)[20]; //ptr là mảng hai chiều có nhiều dòng, mỗi dòng có tối đa 20 phần tử  
int arr[10][20]; //arr là mảng hai chiều có 10 dòng, mỗi dòng có tối đa 20 phần tử  
arr[3][8] = 5; //gán 5 cho phần tử tại dòng 4, cột 9 của mảng arr  
*(ptr+3)+8) = 5; //gán 5 cho phần tử tại dòng 4, cột 9 của mảng ptr
```

6.6 Cấp phát bộ nhớ

6.6.1 Hàm thư viện malloc()

Giống như các loại biến khác, khi biến con trỏ được khai báo, bộ nhớ sẽ cấp cho nó vùng nhớ để lưu trữ giá trị là địa chỉ của biến được nó trỏ tới. Trong trường hợp biến con trỏ được dùng để quản lý mảng, một vùng nhớ đủ để lưu trữ các phần tử trong mảng là cần thiết và phải được chuẩn bị trước. Sự cấp phát bộ nhớ như vậy được gọi là cấp phát bộ nhớ động và được thực hiện bằng hàm thư viện **malloc()** như sau:

```
int *a; //khai báo biến con trỏ a, bộ nhớ cấp vùng nhớ kích thước 2 byte  
a = malloc(20 *sizeof(int)); //khởi tạo vùng nhớ đủ để lưu trữ liên tiếp 20 số nguyên
```

Ví dụ 6.10. Chương trình minh họa cấp phát bộ nhớ động

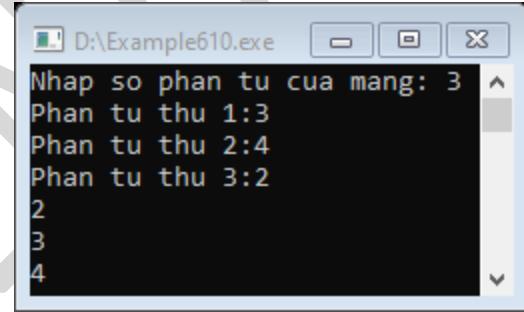
```
/* Yêu cầu: Nhập mảng số nguyên (sử dụng con trỏ) và sắp xếp tăng dần */
```

```
#include <stdio.h>  
#include <malloc.h>
```

```

int main(){
    int *p, n, i, j, temp;
    printf("Nhập số phần tử của mảng: ");
    scanf("%d", &n);
    p = (int*) malloc(n*sizeof(int));
    /*Nhập n số*/
    for (i = 0; i < n; i++) {
        printf("Phần tử thứ %d:", i+1);
        scanf("%d", p + i);
    }
    /*Sắp xếp n số theo thứ tự tăng dần*/
    for(i = 0; i < n - 1; i++)
        for (j = i + 1; j < n; j++)
            if(*(p + i) > *(p + j)){
                temp = *(p + i);
                *(p + i) = *(p + j);
                *(p + j) = temp;
            }
    /*Xuất n số sau khi sắp xếp */
    for(i = 0; i < n; i++) printf("%d\n", *(p + i));
    getchar();
    return 0;
}

```



6.6.2 Hàm thư viện free()

Một điều quan trọng cần chú ý khi sử dụng con trỏ là một khi đã cấp vùng nhớ được con trỏ trả tới, vùng nhớ đó sẽ bị chiếm đóng ngay cả khi chương trình đã kết thúc thành công. Vùng nhớ này chỉ được giải phóng bởi hàm thư viện free() của ngôn ngữ C. Dạng tổng quát của hàm free():

```
void free(void *ptr);
```

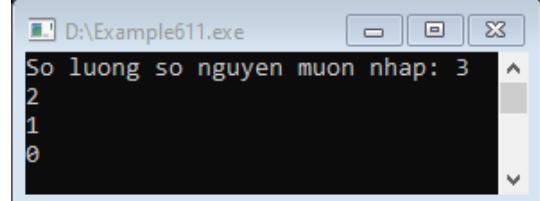
Hàm free() giải phóng vùng nhớ đã cấp cho con trỏ *ptr* trước đó bằng cách gọi hàm malloc(), calloc(), realloc(). Vùng nhớ được giải phóng này có thể sử dụng trong tương lai.

Ví dụ 6.11. Cấp phát và giải phóng vùng nhớ đối với biến con trỏ

```

#include <stdio.h>
#include <stdlib.h>
int main(){
    int n, i, *ptr;
    printf("So luong so nguyen muon nhap: ");

```



```

scanf("%d", &n);
ptr = (int *) malloc (n*sizeof(int)); /*cap vung nho n so nguyen*/
if(ptr != NULL){ /*vung nho cap phat thanh cong*/
    for(i = 0; i < n; i++) *(ptr+i) = i;
    for(i=n; i>0; i--)
        printf("%d\n", *(ptr+(i-1))); /*xuat n so*/
    free(ptr); /*giai phong vung nho da cap*/
}
else printf("Viec cap phat vung nho that bai. Khong du bo nho.\n");
getchar();
return 0;
}

```

6.6.3 Một số hàm thư viện cấp phát bộ nhớ động

Hàm thư viện calloc() tương tự như malloc(), nhưng giá trị mặc định là 0 được trong vùng nhớ đã cấp. Trong khi đó, hàm malloc() cấp phát vùng nhớ với giá trị mặc định tùy ý. Hàm calloc đòi hỏi hai đối số, đối số thứ nhất là số các phần tử được lưu trữ trong vùng nhớ, đối số thứ hai là kích thước của mỗi phần tử đó. Cú pháp chung của hàm calloc():

```
void *calloc( size_t num, size_t size );
```

Giống như malloc, calloc sẽ trả về một con trỏ rỗng (void) nếu sự cấp phát bộ nhớ là thành công, ngược lại nó sẽ trả về một con trỏ NULL.

Ví dụ 6.12. Sử dụng hàm thư viện calloc()

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    float *calloc1, *calloc2;
    int i;
    calloc1 = (float *) calloc(3, sizeof(float));
    calloc2 = (float *) calloc(3, sizeof(float));
    if(calloc1 != NULL && calloc2 != NULL){
        for(i = 0; i < 3; i++) {
            printf("\ncalloc1[%d] = %05.5f ", i, calloc1[i]);
            printf("\ncalloc2[%d] = %05.5f", i, *(calloc2 + i));
        }
        free(calloc1);
        free(calloc2);
    }
    return 0;
}

```

```

    }
else {
    printf("Not enough memory\n");
    return 1;
}
}

```

Hạn chế của hàm malloc() cũng như hàm calloc() là khi con trỏ đã được cấp vùng nhớ, kích thước của nó không thể thay đổi. Nói cách khác, không thể cấp thêm vùng nhớ cho một con trỏ đã được cấp vùng nhớ. Trong trường hợp này, hàm thư viện realloc() được sử dụng để cấp phát thêm vùng nhớ bổ sung cho con trỏ mà không ảnh hưởng tới dữ liệu hiện có. Hàm realloc() nhận hai đối số, đối số thứ nhất là một con trỏ trỏ đến vùng nhớ đã cấp trước đó, đối số thứ hai là kích thước vùng nhớ được cấp phát thêm. Hàm realloc() trả về một con trỏ rỗng nếu thành công, ngược lại con trỏ NULL được trả về. Cú pháp chung của hàm realloc():

```
void *realloc( void *ptr, size_t size );
```

Ví dụ 6.13. Sử dụng hàm thư viện realloc()

```
/* Yêu cầu: Sử dụng con trỏ để lưu trữ mang có 5 số, sau đó mở rộng thêm 2 số nữa. */
```

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

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

```
int main()
```

```
    int *ptr;
```

```
    int i;
```

```
    ptr = (int *) calloc(5, sizeof(int *));
```

```
    if(ptr!=NULL)
```

```
{
```

```
        *ptr = 1;
```

```
        *(ptr + 1) = 2;
```

```
        ptr[2] = 4; ptr[3] = 8; ptr[4] = 16;
```

```
        ptr = (int *)realloc(ptr, 7 * sizeof(int));
```

```
        if(ptr!=NULL)
```

```
{
```

```
            printf("Cap them vung nho cho con tro ... \n");
```

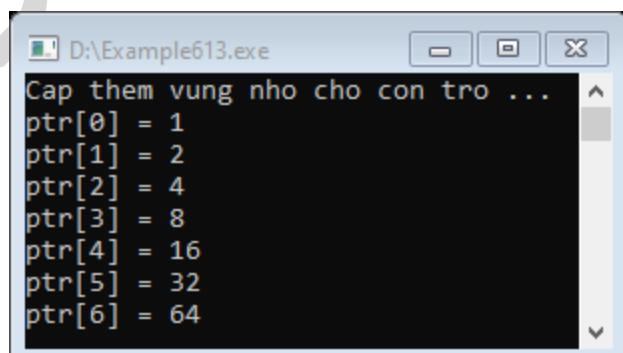
```
            ptr[5] = 32; /* Hop le*/
```

```
            ptr[6] = 64;
```

```
            for(i = 0; i < 7; i++) printf("ptr[%d] = %d\n", i, ptr[i]);
```

```
            return 0;
```

```
}
```



```
D:\Example613.exe
Cap them vung nho cho con tro ...
ptr[0] = 1
ptr[1] = 2
ptr[2] = 4
ptr[3] = 8
ptr[4] = 16
ptr[5] = 32
ptr[6] = 64
```

```

        else
    {
        printf("Khong du bo nho. Viec cap phat them that bai\n");
        return 1;
    }
}
else
{
    printf("Khong du bo nho. Viec cap phat vung nho that bai.\n");
    return 1;
}
}

```

6.7 Câu hỏi ôn tập

- 1) Trình bày khái niệm biến con trỏ, các toán tử con trỏ
- 2) Trình bày các phép toán liên quan đến con trỏ
- 3) Trình bày cách sử dụng con trỏ thay thế cho mảng một chiều
- 4) Trình bày cách sử dụng con trỏ thay thế cho mảng hai chiều
- 5) Trình bày cách cấp phát bộ nhớ động cho biến con trỏ
- 6) Hãy cho biết kết quả xuất ra màn hình khi chạy đoạn chương trình sau đây. Giải thích.

```

int n=7, m=8;
int *p1=&n, *p2=&m;
*p1 += 12 - m + (*p2);
*p2 = m + n - 2 * (*p1);
printf("%d", m+n);

```

- 7) Hãy cho biết kết quả xuất ra màn hình khi chạy đoạn chương trình sau đây. Giải thích.

```

int n=7, m=8;
int *p1= &n, *p2=&m;
*p1 += 5 + 3 * (*p2) - n;
*p2 = 5 * (*p1) - 4*m + 2*n;
printf("m=%d, n=%d", m, n);

```

- 8) Hãy cho biết kết quả xuất ra màn hình khi chạy đoạn chương trình sau đây. Giải thích.

```

int n=6, m=9;
int *p1= &n, *p2=&m;
*p1 += 5 + m * (*p2) - n;
*p2 -= 5 * (*p1) - 4*m + 2*n;
m = n + (*p1) + (*p2);

```

```
printf("m=%d, n=%d", m, n);
```

- 9) Hãy cho biết kết quả xuất ra màn hình khi chạy chương trình sau đây. Giải thích.

```
#include <stdio.h>
int main(){
    int n2=10, n1=6, n0=5;
    printf("n2=%d, n1=%d, n0=%d\n", n2, n1, n0);
    int *p = &n1;
    *p=9; p++; *p=15;
    p--; p--; *p=-3;
    printf("n2=%d, n1=%d, n0=%d\n", n2, n1, n0);
    getchar();
    return 0;
}
```

- 10) Hãy cho biết kết quả xuất ra màn hình khi chạy chương trình sau đây. Giải thích.

```
#include<stdio.h>
int main(){
    int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int *p=&a[0];
    for (int i = 0; i < 10; i++)
        (*p) += a[i];
    printf("a[0]=%d", a[0]);
    return 0;
}
```

6.8 Bài tập thực hành

- 1) Viết chương trình nhập vào một mảng số nguyên (sử dụng con trỏ) và kiểm tra tính đối xứng, tính có thứ tự của nó.

```
#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE 1000
void scanArray(int *a, int n);
int testSymmetry(int *a, int n);
int testAsc(int *a, int n);
int testDesc(int *a, int n);
int main(){
    int *a, n;
    do {
        printf("Nhập kích thước của mảng: "); scanf("%d", &n);
    } while (n<=0);
```

```

a = (int*) malloc(n*sizeof(int));
scanArray(a,n);
if (testSymmetry(a,n)==1) printf("Mang doi xung\n");
else printf("Mang khong doi xung\n");
if (testAsc(a,n)==1 || testDesc(a,n)==1) printf("Mang co thu tu");
else printf("Mang khong co thu tu");
getchar();
return 0;
}
void scanArray(int *a, int n){
    int i;
    printf("\nNhập mảng ======\n");
    for (i=0; i<n; i++){
        printf("a[%d] = ", i); scanf("%d", a+i);
    }
}
int testSymmetry(int *a, int n){
    int res=1, i;
    for (i=0; i<n/2 && res==1; i++) if (*(a+i) != *(a+n-i-1)) res=0;
    return res;
}
int testAsc(int *a, int n){
    int i, res=1;
    for (i=0; i<n-1 && res==1; i++) if (*(a+i) > *(a+i+1)) res = 0;
    return res;
}
int testDesc(int *a, int n){
    int i, res=1;
    for (i=0; i<n-1 && res==1; i++) if (*(a+i) < *(a+i+1)) res = 0;
    return res;
}

```

- 2) Viết chương trình nhập vào số nguyên dương n và xuất ra màn hình mảng gồm n số nguyên tố (sử dụng con trỏ).

```

#include <stdio.h>
#include <stdlib.h>
void createArray(int *a, int n);
int isPrime(int current);
void printArray(int *a, int n);
int main(){
    int *a, n;
    do {

```

```

        printf("Nhập kích thước của mảng: "); scanf("%d", &n);
    } while (n<=0);
    a = (int*) malloc(n*sizeof(int));
    createArray(a,n);
    printArray(a,n);
    getchar();
    return 0;
}
void createArray(int *a, int n){
    int i, current=2;
    for (i=0; i<n; current++){
        if (isPrime(current)==1) { *(a+i) = current; i++; }
    }
}
int isPrime(int current){
    int res=1, i=0;
    for (i=2; i*i <= current && res == 1; i++)
        if (current%i == 0) res = 0;
    return res;
}
void printArray(int *a, int n){
    int i;
    printf("\nXuất mảng ======\n");
    for (i=0; i<n; i++)
        printf("%d\t", *(a+i));
}

```

- 3) Viết chương trình nhập vào mảng gồm các số nguyên dương và xuất ra màn hình mảng các số đã nhập sau khi đã sắp xếp theo trật tự tăng dần.

```

#include <stdio.h>
#include <stdlib.h>
void scanArray(int *a, int n);
void sortArray(int *a, int n);
void printArray(int *a, int n);
int main(){
    int *a, n;
    do {
        printf("Nhập kích thước của mảng: "); scanf("%d", &n);
    } while (n<=0);
    a = (int*) malloc(n*sizeof(int));
    scanArray(a,n);
    sortArray(a,n);
}

```

```

printArray(a,n);
getchar();
return 0;
}
void scanArray(int *a, int n){
    int i;
    printf("\nNhap mang ======\n");
    for (i=0; i<n; i++){
        printf("a[%d] = ", i); scanf("%d", a+i);
    }
}
void sortArray(int *a, int n){
    int i, j, minIndex, minTemp;
    for (i=0; i<n-1; i++){
        minIndex = i;
        for (j=i+1;j<n;j++) if (*(a+minIndex) > *(a+j)) minIndex = j;
        if (minIndex > i) {
            minTemp=*(a+i);
            *(a+i)=*(a+minIndex);
            *(a+minIndex) = minTemp;
        }
    }
}
void printArray(int a[], int n){
    int i;
    printf("\nXuat mang ======\n");
    for (i=0; i<n; i++){
        printf("%d\t", a[i]);
    }
}

```

- 4) Viết chương trình nhập vào hai số nguyên dương n và m sau đó nhập tiếp ma trận các số nguyên gồm n hàng, m cột. Xuất ma trận đó ra màn hình.

```

#include <stdio.h>
#include <stdlib.h>
void scanMatrix(int **a, int n, int m);
void printMatrix(int **a, int n, int m);
int main(){
    int **a, n, m;
    do {
        printf("Nhap kich thuoc cua ma tran (n m): ");
        scanf("%d %d", &n, &m);

```

```

} while (n<=0 || m<=0);
a=(int **)malloc(sizeof(int*) * n);
for (int i=0; i<n; i++) a[i]=(int *)malloc(sizeof(int*) * m);
scanMatrix(a,n,m);
printMatrix(a,n,m);
getchar();
return 0;
}
void scanMatrix(int **a, int n, int m){
    int i, j;
    printf("\nNhập ma trận ======\n");
    for (i=0; i<n; i++){
        for (j=0; j<m; j++){
            printf("a[%d][%d] = ", i, j); scanf("%d", &a[i][j]);
        }
    }
}
void printMatrix(int **a, int n, int m){
    int i, j;
    printf("\nXuất ma trận ======\n");
    for (i=0; i<n; i++){
        for (j=0; j<m; j++){
            printf("%d\t", a[i][j]);
        }
        printf("\n");
    }
}

```

- 5) Viết chương trình nhập vào hai ma trận vuông gồm m hàng, m cột. Xuất ra màn hình hai ma trận đã nhập và ma trận tổng, tích của hai ma trận đó.

```

#include <stdio.h>
#include <stdlib.h>
void scanMatrix(int **a, int n);
void printMatrix(int **a, int n);
int ** sumMatrix(int **a, int **b, int n);
int ** productMatrix(int **a, int **b, int n);
int main(){
    int **a, **b, **c, **d, n;
    do {
        printf("Nhập kích thước của ma trận vuông (n): "); scanf("%d", &n);
    } while (n<=0);
    a=(int **)malloc(sizeof(int*) * n);
    b=(int **)malloc(sizeof(int*) * n);

```

```

for (int i=0; i<n; i++) {
    a[i]=(int *)malloc(sizeof(int*) * n);
    b[i]=(int *)malloc(sizeof(int*) * n);
}
scanMatrix(a,n);
scanMatrix(b,n);
c=sumMatrix (a,b,n);
printMatrix(c,n);
d=productMatrix (a,b,n);
printMatrix(d,n);
getchar();
return 0;
}
void scanMatrix(int **a, int n){
    int i, j;
    printf("\nNhap ma tran ======\n");
    for (i=0; i<n; i++)
        for (j=0; j<n; j++){
            printf("a[%d][%d] = ", i, j); scanf("%d", &a[i][j]);
        }
}
void printMatrix(int **a, int n){
    int i, j;
    printf("\nXuat ma tran ======\n");
    for (i=0; i<n; i++){
        for (j=0; j<n; j++){
            printf("%d\t", a[i][j]);
        }
        printf("\n");
    }
}
int ** sumMatrix (int **a, int **b, int n){
    int **c, i, j;
    c=(int **)malloc(sizeof(int*) * n);
    for (i=0; i<n; i++) c[i]=(int *)malloc(sizeof(int*) * n);
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            c[i][j]=a[i][j] + b[i][j];
    return c;
}
int ** productMatrix (int **a, int **b, int n){
    int **c, k, i, j;
    c=(int **)malloc(sizeof(int*) * n);

```

```

for (i=0; i<n; i++) c[i]= (int *)malloc(sizeof(int*) * n);
for (i=0; i<n; i++){
    for (j=0; j<n; j++){
        c[i][j] = 0;
        for (k=0; k<n; k++) c[i][j] += a[i][k] + b[k][j];
    }
}
return c;
}

```

6.9 Bài tập đề nghị

Yêu cầu chung: Thực hiện các yêu cầu dưới đây, trong đó, sử dụng con trỏ để hiện thực mảng (một chiều, hai chiều) gồm các số nguyên.

- 1) Viết chương trình nhập vào một mảng số nguyên và xuất ra màn hình:
 - Phần tử lớn nhất của mảng
 - Phần tử nhỏ nhất của mảng
 - Tính tổng của các phần tử trong mảng
 - Tính tích của các phần tử trong mảng
- 2) Viết chương trình nhập vào một mảng số nguyên theo thứ tự tăng dần, nếu nhập sai quy cách thì yêu cầu nhập lại. Xuất ra màn hình mảng sau khi đã nhập xong.

> Give a number to array (0 to quit): 4
 > Give a number to array (0 to quit): 3
 > Ops! Your number broke the rule! Try again!
 > Give a number to array (0 to quit): 4
 > Give a number to array (0 to quit): 6
 > Give a number to array (0 to quit): 0
 > Your inputted array: 4 4 6
 > Bye!

- 3) Viết chương trình nhập vào một ma trận gồm m hàng, n cột. Xuất ra màn hình ma trận đã nhập và ma trận chuyển vị của nó.
- 4) Viết chương trình nhập vào một mảng số nguyên và cho biết số lần xuất hiện của mỗi giá trị phân biệt trong mảng.

Ví dụ:

> Input an array of integers (0 to stop): 1 2 7 2 3 2 3 1 0

> Distinct values and its frequencies:

> 1 2

> 2 3

> 7 1

> 3 2

5) Mã số xuất bản ISBN có đúng 10 chữ số. Việc xác định tính hợp lệ của mã số ISBN được thực hiện như sau:

- Nhân mỗi chữ số từ chữ số đầu tiên tới chữ số thứ chín với trọng số giảm dần từ 10 xuống 2 tương ứng.
- Tổng của các phép nhân nói trên cộng với chữ số thứ mười phải chia hết cho 11; nếu phép chia này có dư, số ban đầu không phải là mã số ISBN hợp lệ.

Viết chương trình nhập vào một mảng có 10 chữ số và cho biết mảng này có phải là một mã số ISBN hợp lệ hay không.

Ví dụ:

> ISBN validator =====

> ISBN (0 to quit): 0003194876

> This is a valid ISBN

> ISBN (0 to quit): 0003194875

> This is not a valid ISBN

> ISBN (0 to quit): 0

> Have a nice day

6) Viết chương trình nhập vào một ma trận vuông gồm m hàng, m cột. Kiểm tra tính đối xứng của nó.

7) Viết chương trình nhập vào hai mảng số nguyên a và b. Kiểm tra mảng b có phải là mảng con của mảng a hay không.

> Input the first array (0 to quit): 1 3 8 7 6 4 0

> Input the second array (0 to quit): 1 3 8 0

> Yeah! The second array is a part of the first array.
> Good job.
> Do it again (y/n)? y
> Input the first array (0 to quit): 1 3 8 7 6 4 0
> Input the second array (0 to quit): 1 8 3 0
> Ops! The second array is not a part of the first array.
> Do it again (y/n)? n
> Bye!

8) Viết chương trình nhập vào một mảng số nguyên sao cho mỗi số nguyên chỉ xuất hiện một lần trong mảng.

> Give a number to array (0 to stop): 3
> Give a number to array (0 to stop): 2
> Give a number to array (0 to stop): 2
> Ops! 2 has been already existed in your array. Try again.
> Give a number to array (0 to stop): 6
> Give a number to array (0 to stop): 0
> Your inputted array: 3 2 6
> Have a nice day!

9) Viết chương trình nhập vào hai mảng số nguyên và nối hai mảng này thành một mảng duy nhất. Xuất ra màn hình mảng kết quả.

> Input the first array (0 to quit): 1 2 5 4 0
> Input the second array (0 to quit): 5 3 2 0
> Concatenate them: 1 2 5 4 5 3 2
> Good job!

10) Viết chương trình nhập vào một mảng số nguyên. Xuất ra màn hình các số nguyên và số lần xuất hiện của chúng trong mảng đã nhập.

CHƯƠNG 7. KIẾU CHUỖI KÝ TỰ

Trong chương này, người học sẽ làm quen với chuỗi ký tự: cách khai báo, các thao tác với chuỗi ký tự và sử dụng thư viện hàm trong ngôn ngữ lập trình C liên quan đến chuỗi ký tự.

Nội dung:

- Khai báo chuỗi ký tự
- Các thao tác với chuỗi ký tự
- Các hàm xử lý chuỗi ký tự

7.1 Khai báo chuỗi ký tự

Chuỗi ký tự trong ngôn ngữ lập trình C là mảng một chiều gồm các ký tự, được kết thúc bằng ký tự ‘\0’ (ký tự NULL trong bảng mã ASCII). Cú pháp khai báo chuỗi ký tự như sau:

```
char HoTen[30];
```

trong đó:

HoTen là chuỗi ký tự có độ dài tối đa là 29 ký tự, vị trí cuối cùng để chứa ký tự kết thúc chuỗi ‘\0’.

Ví dụ 7.1. Khai báo và khởi tạo chuỗi ký tự

Ba câu lệnh khai báo và khởi tạo chuỗi ký tự như là biến mảng với các phần tử là ký tự

- (1) char Greeting[] = {'H', 'E', 'L', 'L', 'O', '\0'};
- (2) char Greeting[6] = "HELLO";
- (3) char Greeting[] = "HELLO";

Trong đó:

- Câu lệnh (1) xác định tường minh phần tử cuối cùng trong mảng là ký tự NULL
- Câu lệnh (2) cung cấp kích thước của mảng (nhiều hơn 1 so với chuỗi hằng được dùng để khởi tạo)

- Câu lệnh (3) không cung cấp kích thước cụ thể, biến mảng được tạo ra có kích thước tối đa là số lượng ký tự trong chuỗi hằng được dùng để khởi tạo cộng thêm 1 (ứng với ký tự NULL).

7.2 Các thao tác đọc, ghi chuỗi ký tự

Ngôn ngữ lập trình C cung cấp các hàm **scanf()** / **printf()** và **fgets()** / **fputs()** để đọc (ghi) chuỗi ký tự từ bàn phím (ra màn hình). Cú pháp các hàm này được trình bày trong **Bảng 7.1** dưới đây.

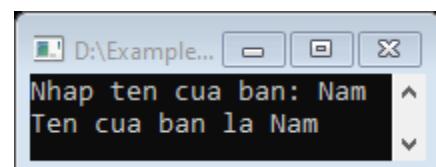
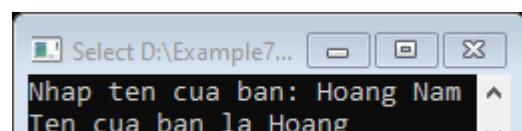
Bảng 7.1. Các thao tác đọc, ghi chuỗi ký tự

| Hàm định nghĩa sẵn | Mô tả |
|---|---|
| scanf(“%s”, char *str); | Đọc vào biến chuỗi ký tự str một chuỗi từ bàn phím cho đến khi nhận được ký tự khoảng trắng (dấu cách, xuống dòng, tab...) |
| char *fgets(char *str, int n, FILE *input) | Đọc chuỗi ký tự có chiều dài tối đa n - 1 từ luồng input cho đến khi nhận dấu xuống dòng và lưu vào biến chuỗi ký tự str . Luồng stdin đại diện cho bàn phím. |
| printf(“%s”, const char *str); | Ghi chuỗi ký tự lưu trữ tại biến str ra màn hình |
| int fputs(const char *str, FILE *output) | Ghi vào luồng output một chuỗi ký tự được lưu trữ tại biến str . Luồng stdout đại diện cho màn hình. |

Chú ý: Hàm **gets()** có thể được sử dụng để nhập chuỗi ký tự, tuy nhiên nó đã được loại bỏ khỏi C chuẩn. Nguyên nhân là do **gets()** cho phép nhập chuỗi có độ dài bất kỳ nên có thể dẫn đến tính trạng tràn bộ đệm.

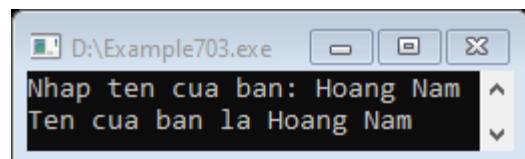
Ví dụ 7.2. Sử dụng hàm **scanf()**, **printf()** để đọc, ghi chuỗi ký tự

```
#include <stdio.h>
int main(){
    char ten[20];
    printf ("Nhập tên của bạn: ");
    scanf ("%s", ten);
    printf ("Tên của bạn là %s", ten);
    getchar();
    return 0;
}
```



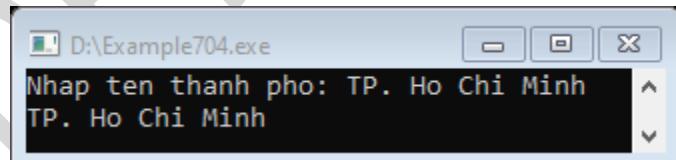
Ví dụ 7.3. Sử dụng hàm fgets(), printf() để đọc, ghi chuỗi ký tự

```
#include <stdio.h>
#include <conio.h>
int main() {
    char name[30];
    printf ("Nhập tên của bạn: ");
    fgets (name, sizeof (name), stdin);
    printf ("Tên của bạn là %s", name);
    getchar();
    return 0;
}
```



Ví dụ 7.4. Sử dụng hàm printf(), fgets() và fputs() để đọc, ghi chuỗi ký tự

```
#include <stdio.h>
#include <conio.h>
int main() {
    char city[40];
    printf ("Nhập tên thành phố: ");
    fgets (city, sizeof (city), stdin);
    fputs (city, stdout);
    getchar();
    return 0;
}
```



7.3 Các hàm xử lý chuỗi ký tự

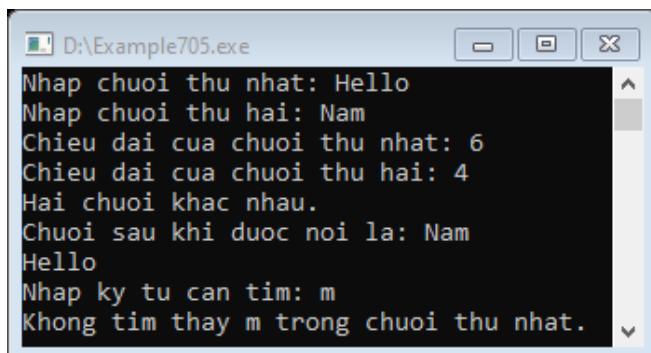
| Hàm và cú pháp | Mô tả |
|--|--|
| size_t strlen (const char *str) | Tính độ dài của chuỗi ký tự lưu trữ trong biến str |
| int strcmp (const char *str1, const char *str2) | So sánh hai chuỗi str1 và str2, trả về 0 nếu hai chuỗi giống nhau, trả về -1 nếu str1<str2 hay str1 là chuỗi con của str2, trả về 1 nếu str1>str2 hay str2 là chuỗi con của str1 |
| int stricmp (const char *str1, const char *str2) | So sánh hai chuỗi, không phân biệt chữ hoa, chữ thường |
| int strncmp (const char *str1, const char *str2, size_t n) | So sánh n ký tự đầu tiên của hai chuỗi |
| int strnicmp (const char *str1, const char *str2, size_t n) | So sánh n ký tự đầu tiên của hai chuỗi, không phân biệt chữ hoa, chữ thường |

| | |
|--|---|
| char * strcat (char *str1, char *str2) | Nối hai chuỗi và trả về chuỗi được nối |
| char * strncat (char *str1, char *str2, int n) | Nối n ký tự đầu tiên của chuỗi str2 với chuỗi str1 |
| char * strcpy (char *str1, const char *str2) | Chuỗi str2 được sao chép vào chuỗi str1 |
| char * strchr (char *str, int ch) | Trả về giá trị NULL nếu không tìm thấy ký tự ch trong chuỗi str, ngược lại trả về con trỏ trỏ đến vị trí xuất hiện đầu tiên của ký tự ch trong chuỗi str. |
| char * strrchr (char *str, int ch) | Trả về giá trị NULL nếu không tìm thấy ký tự ch trong chuỗi str, ngược lại trả về con trỏ trỏ đến vị trí xuất hiện cuối cùng của ký tự ch trong chuỗi str. |
| char * strstr (const char *str1, const char *str2) | Trả về giá trị NULL nếu không tìm thấy chuỗi str2 trong chuỗi str1, ngược lại trả về con trỏ trỏ đến vị trí xuất hiện đầu tiên của chuỗi str2 trong chuỗi str1. |

Chú ý: Hàm *strlen* trả về chiều dài của chuỗi được lưu trữ trong mảng trong khi hàm *sizeof* trả về kích thước mảng được cấp để lưu trữ chuỗi.

Ví dụ 7.5. Sử dụng các hàm xử lý chuỗi

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main() {
    char str1[40], str2[40], ch;
    printf ("Nhập chuỗi thứ nhất: ");
    fgets (str1, sizeof(str1), stdin);
    printf ("Nhập chuỗi thứ hai: ");
    fgets (str2, sizeof(str2), stdin);
    /*Xuất ra màn hình chiều dài chuỗi*/
    printf("Chiều dài của chuỗi thứ nhất: %d\n",strlen(str1));
    printf("Chiều dài của chuỗi thứ hai: %d\n",strlen(str2));
    /*So sánh hai chuỗi*/
    if (strcmp(str1,str2)==0) printf("Hai chuỗi giống nhau.\n");
    else printf("Hai chuỗi khác nhau.\n");
    /*Nối hai chuỗi*/
    strcat (str2,str1);
```



```

printf ("Chuoi sau khi duoc noi la: %s", str2);
/*Tim kiem ky tu trong chuoi*/
fflush(stdin); /*Xoa luong doc du lieu tu ban phim*/
printf("Nhap ky tu can tim: "); scanf("%c",&ch);
if (strchr(str1,ch)==NULL)
    printf("Khong tim thay %c trong chuoi thu nhat.\n",ch);
    else printf("Tim thay %c trong chuoi thu nhat.\n",ch);
getch();
return 0;
}

```

7.4 Một số hàm xử lý chuỗi ký tự khác

| Hàm và cú pháp | Mô tả |
|--|---|
| Thư viện string.h | |
| char * strlwr (char *str) | Chuyển đổi chuỗi str sang chữ thường |
| char * strupr (char *str) | Chuyển đổi chuỗi str sang chữ hoa |
| char * strrev (char *str) | Đảo ngược chuỗi str |
| char * strdup (char *str) | Nhân bản chuỗi str |
| char * strset (const char *str, char ch) | Thiết lập tất cả ký tự trong chuỗi str về ký tự ch |
| char * strset (const char *str, char ch, int n) | Thiết lập n ký tự đầu tiên trong chuỗi str về ký tự ch |
| Lưu ý: Hàm strcpy và hàm strdup đều tạo ra chuỗi giống với chuỗi gốc, nhưng đối với strcpy , người dùng phải tự cấp phát bộ nhớ cho chuỗi mới được tạo ra; ngược lại strdup tự cấp phát vùng nhớ cho chuỗi mới. | |
| Thư viện ctype.h | |
| int toupper (int ch) | Chuyển ch thành chữ hoa |
| int tolower (int ch) | Chuyển ch thành chữ thường |
| int isalnum (int ch) | Kiểm tra ch có phải là chữ cái hoặc số không |
| int isalpha (int ch) | Kiểm tra ch có phải là chữ cái không |
| int iscntrl (int ch) | Kiểm tra ch có phải là ký tự điều khiển không |
| int isdigit (int ch) | Kiểm tra ch có phải là số thập phân không |
| int islower (int ch) | Kiểm tra ch có phải là chữ thường |
| int isupper (int ch) | Kiểm tra ch có phải là chữ hoa |
| int isspace (int ch) | Kiểm tra ch có phải là ký tự khoảng trắng (ký tự cách, tab, xuống dòng...) |

| Thư viện stdlib.h | |
|--|--|
| int atoi (const char *str) | Chuyển đổi chuỗi str thành số nguyên |
| long int atol (const char *str) | Chuyển chuỗi str thành số nguyên kiểu long |
| double atof (const char *str) | Chuyển chuỗi str thành số thực dấu chấm động (kiểu float) |

7.5 Câu hỏi ôn tập

- 1) Chuỗi kí tự là gì? Nêu một số cách khai báo chuỗi kí tự?
- 2) Liệt kê một số hàm đọc, ghi chuỗi kí tự.
- 3) Nêu sự khác biệt giữa các hàm so sánh chuỗi sau: strcmp(), stricmp(), strncmp(), strnicmp().
- 4) Nêu sự khác biệt giữa các hàm tìm kiếm chuỗi sau: strchr(), strrchr(), strstr().
- 5) Nêu sự khác biệt giữa hàm strcpy() và hàm strdup().
- 6) Cho biết chương trình sau thực hiện chức năng gì?

```
#include <stdio.h>
int main() {
    int marks[10], i, n, sum = 0, average;
    printf("Enter number of elements: "); scanf("%d", &n);
    for(i=0; i<n; ++i) {
        printf("Enter number%d: ", i+1);
        scanf("%d", &marks[i]);
        sum += marks[i];
    }
    average = sum/n;
    printf("Average = %d", average);
    return 0;
}
```

- 7) Cho biết chương trình sau thực hiện chức năng gì?

```
#include <stdio.h>
#include <string.h>
char check(char s1[100]){
    int i;
    for (i = 0; i < strlen(s1) / 2; i++){
        if (s1[i] != s1[strlen(s1) - 1 - i]) return 0;
    }
    return 1;
}
```

```
int main(){
    char s1[100];
    printf("Enter the string:\n"); gets(s1);
    if (check(s1) == 0){
        printf("This is a satisfactory string.\n");
    }
    if (check(s1) == 1){
        printf("This is not a satisfactory string.\n");
    }
}
```

8) Cho biết chương trình sau thực hiện chức năng gì?

```
#include <stdio.h>
#include <string.h>
int main(){
    char arr[1000][50];
    int n;
    do{
        printf("\nEnter a number: ");
        scanf("%d", &n);
    } while(n < 1);
    for (int i = 0; i < n; i++){
        printf("Name of the student number %d: ", i+1);
        fflush(stdin);
        gets(arr[i]);
    }
    for (int i = 0; i < n; i++)
        printf("\nName of the student number %d: %s", i+1, arr[i]);
}
```

9) Cho biết chương trình sau thực hiện chức năng gì?

```
#include <stdio.h>
int main() {
    char line[150];
    printf("Enter a string: ");
    fgets(line, sizeof(line), stdin);
    for (int i = 0, j; line[i] != '\0'; ++i){
        while (!(line[i] >= 'a' && line[i] <= 'z')
            && !(line[i] >= 'A' && line[i] <= 'Z') && !(line[i] == '\0')) {
            for (j = i; line[j] != '\0'; ++j) line[j] = line[j + 1];
            line[j] = '\0';
        }
    }
}
```

```

        printf("Output String: ");
        puts(line);
        return 0;
    }

10) Cho biết chương trình sau thực hiện chức năng gì?

#include <stdio.h>
#include <string.h>
int main() {
    char str[5][50], temp[50];
    printf("Enter 5 words: ");
    for (int i = 0; i < 5; ++i) fgets(str[i], sizeof(str[i]), stdin);
    for (int i = 0; i < 5; ++i) {
        for (int j = i + 1; j < 5; ++j) {
            if (strcmp(str[i], str[j]) > 0) {
                strcpy(temp, str[i]);
                strcpy(str[i], str[j]);
                strcpy(str[j], temp);
            }
        }
    }
    printf("\nThe right order is: \n");
    for (int i = 0; i < 5; ++i) fputs(str[i], stdout);
    return 0;
}

```

7.6 Bài tập có lời giải

- 1) Viết hàm tính độ dài của một chuỗi mà không sử dụng hàm strlen.

```

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int main() {
    char str[100]; /* Khai báo chuỗi có độ dài 100*/
    int l= 0;
    printf ("\nTINH DO DAI CUA CHUOI\n");
    printf ("-----\n");
    printf ("Moi nhap chuoi vao: ");
    fgets (str, sizeof str, stdin);
    while (str[l]!='\0') l++;
    printf("Do dai cua chuoi vua nhap la : %d\n\n", l-1);
    getchar();
    return 0;
}

```

2) Viết chương trình tách tất cả ký tự của một chuỗi được nhập vào.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int main(){
    char str[100];
    int l = 0;
    printf("\nTACH TAT CA KI TU CUA MOT CHUOI\n");
    printf("-----\n");
    printf("Moi nhap chuoi vao : ");
    fgets(str, sizeof str, stdin);
    printf("Cac ki tu cua chuoi vua nhap la: \n");
    while (str[l]!='\0') {
        printf ("%c ", str[l]);
        l++;
    }
    printf ("\n");
    getchar();
    return 0;
}
```

3) Viết chương trình in chuỗi nhập vào theo chiều ngược lại.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
int main() {
    char str[100];
    int len,i;
    printf("\nIN CHUOI THEO CHIEU DAO NGUOC:\n");
    printf("-----\n");
    printf("Moi nhap chuoi vao: ");
    fgets(str, sizeof str, stdin);
    len=strlen (str);
    printf ("Cac ki tu cua chuoi theo chieu dao nguoc la: \n");
    for (i=len;i>=0;i--){
        printf ("%c ", str[i]);
    }
    printf ("\n");
    getchar();
    return 0;
}
```

4) Viết chương trình chuyển một chuỗi từ chữ hoa thành chữ thường và ngược lại.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <conio.h>
int main() {
    char str[100];
    int ctr, ch, i;
    printf("\nCHUYEN DOI CHU HOA - CHU THUONG\n");
    printf("-----\n");
    printf("Xin moi nhap chuoi ki tu: ");
    fgets(str, sizeof str, stdin);
    i=strlen(str);
    ctr = i;
    printf("Sau khi chuyen, chuoi moi co dang nhu sau: ");
    for(i=0; i < ctr; i++) {
        ch = islower(str[i]) ? toupper(str[i]) : tolower(str[i]);
        putchar(ch);
    }
    printf("\n\n");
    getchar();
    return 0;
}
```

5) Viết chương trình đếm số từ của một chuỗi nhập vào.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
#define str_size 100
int main(){
    char str[str_size];
    int i, wrd;
    printf("\n DEM SO TU CUA MOT CHUOI\n");
    printf("-----\n");
    printf("Moi nhap chuoi vao: ");
    fgets(str, sizeof str, stdin);
    i = 0;
    wrd = 1;
    while (str[i]!='\0'){
        if (str[i]==' ' || str[i]=='\n' || str[i]=='\t') wrd++;
        i++;
    }
    printf("So tu cua chuoi la: %d", wrd);
}
```

```

        i++;
    }
    printf("So tu cua chuoi vua nhap la: %d\n", wrd-1);
    getchar();
    return 0;
}

```

6) Viết chương trình so sánh 2 chuỗi nhập vào (không sử dụng hàm có sẵn)

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
#define str_size 100
int main() {
    char str1[str_size], str2[str_size];
    int flg=0;
    printf("\n SO SANH 2 CHUOI\n");
    printf("-----\n");
    printf("Nhập chuỗi thu 1: ");
    fgets(str1, sizeof str1, stdin);
    printf("Nhập chuỗi thu 2: ");
    fgets(str2, sizeof str2, stdin);
    int i=0;
    while(str1[i] == str2[i]) {
        if(str1[i] == '\0' || str2[i] == '\0') break;
        i++;
    }
    if(str1[i-1] == '\0' && str2[i-1]=='\0') flg=0;
    else if(str1[i] > str2[i]) flg=1;
    else if(str1[i] < str2[i]) flg=-1;
    if(flg == 0) printf("\n Hai chuỗi giống nhau\n");
    else if (flg == -1)
        printf("\n Chieu dài chuỗi thu 1 nhỏ hơn chuỗi thu 2.\n");
    else
        printf("\n Chieu dài chuỗi thu 1 lớn hơn chuỗi thu 2.\n");
    getchar();
    return 0;
}

```

7) Viết chương trình cho biết số ký tự xuất hiện nhiều nhất trong một chuỗi.

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

```

```

#include <conio.h>
#define str_size 100
#define chr_no 255
int main()
{
    char str[str_size];
    int ch_fre[chr_no];
    int i = 0, max;
    int ascii;
    printf("\nTIM KI TU XUAT HIEN NHIEU NHAT TRONG CHUOI\n");
    printf("-----\n");
    printf("Moi nhap chuoi vao: ");
    fgets(str, sizeof str, stdin);
    for (i=0; i<chr_no; i++) ch_fre[i] = 0;
    i=0;
    while (str[i] != '\0') {
        ascii = (int)str[i];
        ch_fre[ascii] += 1;
        i++;
    }
    max = 0;
    for (i=0; i<chr_no; i++){
        if(i!=32)
            if(ch_fre[i] > ch_fre[max]) max = i;
    }
    printf ("Ki tu xuat hien nhieu nhat la '%c' voi so lan xuat hien la: %d \n\n", max,
           ch_fre[max]);
    getchar();
    return 0;
}

```

8) Viết chương trình sắp xếp các ký tự của một chuỗi nhập vào theo thứ tự tăng dần.

```

#include <stdio.h>
#include <string.h>
#include <conio.h>
int main() {
    char str[100],ch;
    int i,j,l;
    printf("\nSAP XEP CAC KI TU TRONG CHUOI TANG DAN\n");
    printf("-----\n");
    printf("Moi nhap chuoi vao: ");
    fgets(str, sizeof str, stdin);
    l=strlen(str);

```

```

/* Qua trinh sap xep */
for(i=1;i<l;i++)
    for(j=0;j<l-i;j++)
        if(str[j]>str[j+1]) {
            ch=str[j];
            str[j] = str[j+1];
            str[j+1]=ch;
        }
printf("Chuoi sau khi sap xep co dang: \n");
printf("%s\n\n",str);
getchar();
return 0;
}

```

9) Viết chương trình tìm từ dài nhất và từ ngắn nhất của một chuỗi.

```

#include <stdio.h>
#include <string.h>
#include <conio.h>
int main() {
    int i=0,j=0,k=0,a,minIndex=0,maxIndex=0,max=0,min=0;
    char str1[100]={0}, substr[100][100]={0}, c;
    printf("Xin moi nhap chuoi ki tu\n");
    fgets(str1, sizeof str1, stdin);
    while(str1[k]!='\0') {
        j=0;
        while(str1[k]!=' '&&str1[k]!='\0') {
            substr[i][j]=str1[k];
            k++; j++;
        }
        substr[i][j]='\0';
        i++;
        if (str1[k]!='\0') k++;
    }
    int len=i;
    max=strlen(substr[0]);
    min=strlen(substr[0]);
    for(i=0; i<len; i++) {
        a=strlen(substr[i]);
        if (a>max) {
            max=a; maxIndex=i;
        }
        if (a<min) {
            min=a; minIndex=i;
        }
    }
}

```

```
    }  
}  
printf("Tu dai nhat la: %s \nTu ngan nhat la:  
%s\n",substr[maxIndex],substr[minIndex]);  
}
```

7.7 Bài tập đề nghị

- 1) Viết chương trình đếm số ký tự là chữ cái, chữ số và ký tự đặc biệt khác của một chuỗi nhập vào.
- 2) Viết chương trình đếm số nguyên âm, phụ âm của một chuỗi nhập vào.
- 3) Viết chương trình loại bỏ các ký tự không phải là chữ cái ra khỏi chuỗi.
- 4) Viết chương trình đếm số lần xuất hiện của các ký tự có trong chuỗi.
- 5) Viết chương trình kiểm tra xem một chuỗi có đối xứng hay không.
- 6) Viết chương trình đếm số lần xuất hiện của một từ trong chuỗi.
- 7) Viết chương trình loại bỏ các khoảng trắng trước và sau chuỗi ký tự, với các khoảng trắng giữa các từ thì chỉ giữ lại một khoảng trắng.

CHƯƠNG 8. KIỂU CẤU TRÚC VÀ KIỂU HỢP

Trong chương này, chúng ta sẽ làm quen với kiểu cấu trúc và kiểu hợp, bao gồm các khái niệm, thao tác và các vấn đề khác liên quan đến kiểu cấu trúc.

Nội dung:

- Kiểu cấu trúc
- Thao tác với kiểu cấu trúc
- Mảng cấu trúc
- Con trỏ cấu trúc
- Cấu trúc và hàm
- Hợp

8.1 Kiểu cấu trúc

8.1.1 Khái niệm

Kiểu cấu trúc (struct) là tập hợp một hoặc nhiều thành phần có kiểu khác nhau được nhóm lại để tiện cho việc quản lý. Kiểu cấu trúc còn được gọi là kiểu bản ghi (record) trong một số ngôn ngữ lập trình khác. Kiểu cấu trúc cho phép tổ chức các dữ liệu phức tạp bởi nó cho phép một nhóm các trường có liên quan được quản lý như một đơn vị duy nhất. Kiểu cấu trúc được khai báo theo các cách sau đây:

Cách 1: Khai báo kiểu cấu trúc kết hợp khai báo các biến thuộc kiểu cấu trúc đó trong cùng một câu lệnh.

```
struct <tên của kiểu cấu trúc> {  
    <kiểu 1> <trường 1>;  
    <kiểu 2> <trường 2>;  
    ...  
    <kiểu n> <trường n>;  
} <tên biến 1>, <tên biến 2>, ... <tên biến n>;
```

Cách 2: Khai báo kiểu cấu trúc, sau đó khai báo các biến thuộc kiểu cấu trúc đó

```
struct <định danh của kiểu cấu trúc> {  
    <kiểu 1> <trường 1>;  
    <kiểu 2> <trường 2>;  
    ...  
    <kiểu n> <trường n>;  
};
```

struct <định danh của kiểu cấu trúc> <tên biến 1>, <tên biến 2>, ... <tên biến n>;

Cách 3: Khai báo kiểu cấu trúc, sử dụng từ khóa typedef

```
typedef struct {  
    <kiểu 1> <trường 1>;  
    <kiểu 2> <trường 2>;  
    ...  
    <kiểu n> <trường n>;  
} <định danh của kiểu cấu trúc>;
```

<định danh của kiểu cấu trúc> <tên biến 1>, <tên biến 2>, ... <tên biến n>;

Ví dụ 8.1. Sử dụng câu lệnh khai báo cấu trúc

a) Cấu trúc point (điểm gồm tọa độ x, y)

```
struct point  
{  
    int x;  
    int y;  
};  
struct point pt; //pt là biến thuộc kiểu cấu trúc point
```

b) Cấu trúc date (gồm day, month, year)

```
struct date  
{  
    unsigned char day;  
    unsigned char month;  
    unsigned int year;  
} date1, date2; //date1, date2 là hai biến thuộc kiểu cấu trúc date
```

Ví dụ 8.2. Khai báo cấu trúc sử dụng từ khóa typedef

```
typedef struct {  
    int x;  
    int y;  
} point;  
point p1;
```

Lưu ý: các thành phần của kiểu cấu trúc và tên của kiểu cấu trúc có thể giống nhau mà không gây ra bất kì xung đột nào do chúng sẽ được phân biệt theo ngữ cảnh. Các thành phần của kiểu cấu trúc cũng có thể xuất hiện trong nhiều kiểu cấu trúc khác nhau.

Ví dụ 8.3. Khai báo cấu trúc sử dụng cấu trúc khác như một thành phần

```
/*Hình chu nhat duoc xac dinh boi diem tren cung ben trai va diem duoi cung ben phai*/  
struct rect {  
    struct point pt1;  
    struct point pt2;  
};  
struct rect screen;
```

8.1.2 Thao tác với kiểu cấu trúc

8.1.2.1 Khởi tạo biến kiểu cấu trúc

Biến cấu trúc có thể được khởi tạo giá trị trong lúc khai báo, các trường của cấu trúc được đặt giữa cặp dấu mốc nhọn { ... }, các trường cách nhau bởi dấu phẩy (,) theo cú pháp sau đây:

struct <tên kiểu cấu trúc> <tên biến cấu trúc> = {giá trị 1, ..., giá trị n};

Ví dụ 8.4. Khai báo và khởi tạo biến cấu trúc

```
struct point  
{  
    int x;  
    int y;  
};  
struct point pt = {320, 200}; //pt la diem co x=320 va y=200
```

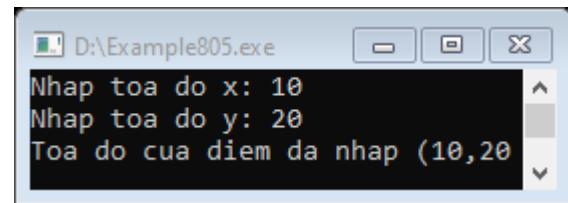
8.1.2.2 Truy cập vào các thành phần của cấu trúc

Thành phần của biến cấu trúc được truy xuất thông qua toán tử chấm(.) dưới dạng:

<tên biến cấu trúc>.<tên thành phần>

Ví dụ 8.5. Đọc và ghi tọa độ của điểm

```
#include <stdio.h>
struct point {
    int x;
    int y;
};
int main(){
    struct point pt;
    printf("Nhập tọa độ x: ");
    scanf("%d",&pt.x);
    printf("Nhập tọa độ y: ");
    scanf("%d",&pt.y);
    printf("Tọa độ của điểm đã nhập (%d,%d)",pt.x,pt.y);
    getchar();
    return 0;
}
```



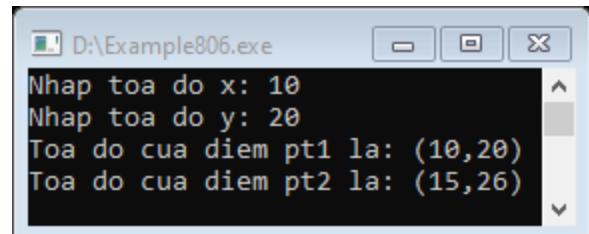
8.1.2.3 Gán dữ liệu kiểu cấu trúc

Phép gán được thực hiện theo hai cách:

- Gán trực tiếp giữa hai biến có cùng kiểu cấu trúc
- Gán gián tiếp lần lượt từng thành phần của biến cấu trúc với các giá trị tương ứng

Ví dụ 8.6. Phép gán đối với biến kiểu cấu trúc

```
#include <stdio.h>
struct point {
    int x;
    int y;
};
int main(){
    struct point pt, pt1, pt2;
    printf("Nhập tọa độ x: "); scanf("%d",&pt.x);
    printf("Nhập tọa độ y: "); scanf("%d",&pt.y);
    pt1 = pt;
    pt2.x = pt.x+5; pt2.y = pt.y+6;
    printf("Tọa độ của điểm pt1 là: (%d,%d)",pt1.x,pt1.y);
    printf("\nTọa độ của điểm pt2 là: (%d,%d)",pt2.x,pt2.y);
    getchar();
    return 0;
}
```



8.2 Mảng cấu trúc

Sau khi được khai báo, kiểu cấu trúc có thể được sử dụng như là kiểu cơ sở của mảng, trong đó mỗi phần tử trong mảng thuộc cùng kiểu cấu trúc. Các thao tác đối với mảng cấu trúc cũng tương tự như đối với mảng thông thường (mảng số nguyên, mảng chuỗi ký tự, ...).

Ví dụ 8.7. Sử dụng mảng cấu trúc để quản lý danh sách sinh viên

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
/* Thông tin sinh viên bao gồm: mã số sinh viên, tên sinh viên */
struct Student {
    int id;
    char name[10];
};

int main(){
    int i;
    struct Student st[3]; //Khai báo mảng gồm 3 phần tử thuộc kiểu Student
    printf("NHAP THONG TIN CUA 3 SINH VIEN\n");
    for (i=0;i<3;i++){
        printf("\nNhập Mã số sinh viên: ");
        scanf("%d",&st[i].id);
        printf("\nNhập Tên sinh viên: ");
        scanf("%s",&st[i].name);
    }
    printf ("\nDanh sách sinh viên được in ra như sau: ");
    for (i=0;i<3;i++)
        printf ("\nMã sinh viên: %d, Tên sinh viên: %s", st[i].id, st[i].name);
    getchar();
    return 0;
}
```

8.3 Con trỏ cấu trúc

Trong trường hợp kiểu cấu trúc có gán kèm thì truyền tham số bằng con trỏ sẽ hiệu quả hơn truyền toàn bộ cấu trúc. Con trỏ kiểu cấu trúc cũng tương tự như con trỏ của các kiểu khác. Con trỏ cấu trúc được khai báo như sau:

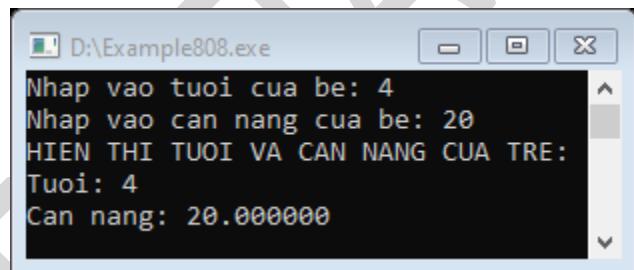
```
struct <tên kiểu cấu trúc> *pp;
```

Giả sử *pp* là một con trỏ tới kiểu cấu trúc *point* với hai thành phần kiểu số nguyên là *x* và *y*. Nếu *pp* chỉ đến vị trí lưu trữ trong bộ nhớ của biến kiểu cấu trúc *point*, **pp* là bản thân giá trị của biến đó và do đó *(*pp).x* và *(*pp).y* là các thành phần của nó. Dấu ngoặc () là cần thiết trong *(*pp).x* vì độ ưu tiên của toán tử chấm (.) cao hơn toán tử hoa thị (*). Biểu thức **pp.x* hay **(pp.x)* là một biểu thức không hợp lệ do *pp.x* không phải là một con trỏ.

Ví dụ 8.8. Sử dụng con trỏ cấu trúc để quản lý thông tin trẻ em

```
#include <stdio.h>
/* Thông tin của trẻ bao gồm tuổi và cân nặng */
struct Kid {
    int age;
    float weight;
};

int main() {
    struct Kid *ptKid, kid1;
    ptKid = &kid1;
    printf ("Nhập vào tuổi của bé: ");
    scanf ("%d", &ptKid->age);
    printf ("Nhập vào cân nặng của bé: ");
    scanf ("%f", &ptKid->weight);
    printf ("HIỂN THỊ TUỔI VÀ CAN NANG CỦA TRẺ:\n");
    printf ("Tuổi: %d\n", ptKid->age);
    printf ("Cân nặng: %f", (*ptKid).weight);
    getchar();
    return 0;
}
```

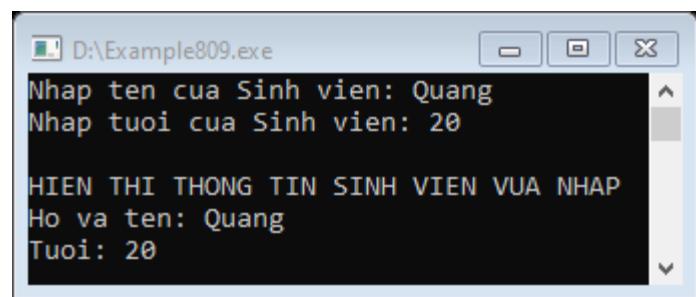


8.4 Cấu trúc và hàm

Kiểu cấu trúc có thể xuất hiện với vai trò là kiểu trả về và kiểu tham số với hai phương án truyền tham trị và truyền tham chiếu.

Ví dụ 8.9. Truyền tham trị kiểu cấu trúc cho hàm

```
#include <stdio.h>
#include <conio.h>
struct Student {
    char name[50];
    int age;
};
void displayInfo(struct Student st);
int main()
```



```

    struct Student st;
    printf ("Nhap ten cua Sinh vien: ");
    scanf ("%[^n]*c", st.name);
    printf ("Nhap tuoi cua Sinh vien: ");
    scanf ("%d", &st.age);
    displayInfo(st); // Truyen tham tri kieu cau truc cho ham
    return 0;
}
void displayInfo (struct Student st) {
    printf ("\nHien thi thong tin sinh vien vua nhap\n");
    printf ("Ho va ten: %s", st.name);
    printf ("\nTuoi: %d", st.age);
}

```

Ở ví dụ trên biến *st* có kiểu cấu trúc *Student* được tạo. Biến được truyền vào tham số của hàm *displayInfo* thông qua câu lệnh *displayInfo(st)*.

Ví dụ 8.10. Giá trị trả về thuộc kiểu cấu trúc

```

#include <stdio.h>
struct Student {
    char name[50];
    int age;
};
struct Student setupInfo();
void displayInfo(struct Student st);
int main() {
    struct Student st;
    st = setupInfo();
    displayInfo(st);
    getchar();
    return 0;
}
struct Student setupInfo() {
    struct Student st;
    printf ("Nhap ho va ten cua sinh vien: ");
    scanf ("%[^n]*c", st.name);
    printf ("Nhap tuoi cua sinh vien: ");
    scanf ("%d", &st.age);
    return st;
}
void displayInfo (struct Student st) {
    printf ("\nHien thi thong tin sinh vien vua nhap\n");

```

```
    printf ("Ho va ten: %s", st.name);
    printf ("\nTuoi: %d", st.age);
}
```

Ở đây, hàm *setupInfo()* được gọi thông qua câu lệnh *st=setupInfo()*. Hàm trả về kiểu cấu trúc *Student*. Cấu trúc trả về được hiển thị trong hàm *main()*.

Ví dụ 8.11. Truyền tham chiếu kiểu cấu trúc cho hàm

```
#include <stdio.h>
typedef struct complexNumber
{
    float real;
    float imaginary;
} complexNumber;
void complexNumberSum (complexNumber n1, complexNumber n2, complexNumber
*result);

int main()
{
    complexNumber n1, n2, result;
    printf ("NHAP SO PHUC THU 1:\n");
    printf ("Nhập phần thực: ");
    scanf ("%f", &n1.real);
    printf ("Nhập phần ảo: ");
    scanf ("%f", &n1.imaginary);
    printf ("NHAP SO PHUC THU 2:\n");
    printf ("Nhập phần thực: ");
    scanf ("%f", &n2.real);
    printf ("Nhập phần ảo: ");
    scanf ("%f", &n2.imaginary);
    complexNumberSum (n1, n2, &result);
    printf ("\nKET QUA:");
    printf ("\nTổng phần thực = %.1f\n", result.real);
    printf ("Tổng phần ảo = %.1f", result.imaginary);
    return 0;
}
void complexNumberSum (complexNumber n1, complexNumber n2, complexNumber
*result)
{
    result ->real = n1.real + n2.real;
    result ->imaginary = n1. imaginary + n2. imaginary;
}
```

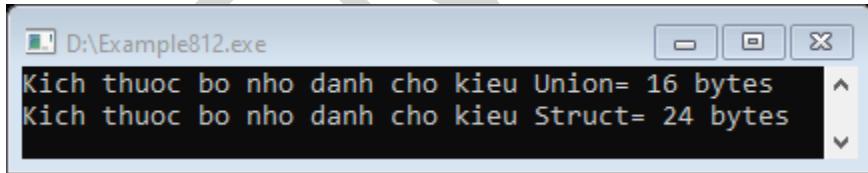
Ở ví dụ trên, 3 cấu trúc *n1*, *n2* và địa chỉ của *result* được truyền vào hàm *complexNumberSum*. Ở đây, *result* được truyền bằng tham chiếu. Khi biến *result* trong hàm *complexNumberSum* bị thay đổi, biến *result* trong hàm *main()* cũng thay đổi theo.

8.5 Hợp

Hợp (*union*) được khai báo và sử dụng tương tự như *struct*. Điểm khác biệt duy nhất của *union* và *struct* là: các thành phần của *struct* sử dụng bộ nhớ riêng, còn các thành phần của *union* chia sẻ chung bộ nhớ.

Ví dụ 8.12. So sánh union và struct

```
#include <stdio.h>
union un {
    char a[15];
    double b;
} u_var;
struct st {
    char a[15];
    double b;
} s_var;
int main()
{
    printf("Kich thuoc bo nho danh cho kieu Union= %d bytes", sizeof(u_var));
    printf("\nKich thuoc bo nho danh cho kieu Struct= %d bytes", sizeof(s_var));
    getchar();
    return 0;
}
```



Với trường hợp của struct, bộ nhớ được cấp phát cho biến sẽ là tổng bộ nhớ được cấp phát cho các thành phần. Đối với union, bộ nhớ được cấp phát cho biến là bộ nhớ được cấp phát cho thành phần cần nhiều bộ nhớ nhất. Do đó, các thành phần của union không thể được xử lý cùng lúc.

Ví dụ 8.13. Sử dụng union – các thành phần được xử lý cùng lúc

```
#include <stdio.h>
#include <conio.h>
union un {
    int a;
    int b;
```

```

} u_var;

int main() {
    u_var.a = 1;
    printf("Gia tri cua bien b = %d", u_var.b);      //Xuat "Gia tri cua bien b = 1"
    u_var.b = 2;
    printf("\nGia tri cua bien a = %d", u_var.a);    //Xuat "Gia tri cua bien a = 2"
    getchar();
    return 0;
}

```

Ví dụ 8.14. Sử dụng struct – các thành phần được xử lý độc lập

```

#include <stdio.h>
#include <conio.h>
struct st {
    int a;
    int b;
} s_var;
int main() {
    s_var.a = 1;
    printf("Gia tri cua bien b = %d", s_var.b);      //Xuat "Gia tri cua bien b = 0"
    s_var.b = 2;
    printf("\nGia tri cua bien a = %d", s_var.a);    //Xuat "Gia tri cua bien a = 1"
    return 0;
}

```

8.6 Câu hỏi ôn tập

- 1) Kiểu cấu trúc là gì? Nêu một số cách khai báo kiểu cấu trúc.
- 2) Nêu một số cách khởi tạo giá trị cho biến kiểu cấu trúc.
- 3) Có thể thực hiện việc gán giá trị cho biến kiểu cấu trúc thông qua các hình thức nào?
- 4) Nêu ưu điểm của việc sử dụng con trỏ kiểu cấu trúc.
- 5) Nêu sự khác nhau giữa kiểu cấu trúc và kiểu hợp.
- 6) Cho biết các giá trị xuất ra màn hình của đoạn lệnh sau:

```

#include <stdio.h>
struct numbers {
    int num1, num2;
};
int main(){
    struct numbers s1 = {s1.num2 = 22, s1.num1 = 11};

```

```
    struct numbers s2 = {s2.num2 = 30};  
    printf ("num1: %d, num2: %d\n", s1.num1, s1.num2);  
    printf ("num1: %d", s2.num2);  
    return 0;  
}
```

7) Cho biết các giá trị xuất ra màn hình của đoạn lệnh sau:

```
#include <stdio.h>  
#include <string.h>  
struct student {  
    int roll_no;  
    char name[30];  
    int phone_number;  
};  
int main() {  
    struct student p1 = { 1,"Brown",123443 };  
    struct student p2;  
    p2 = p1;  
    printf("roll_no : %d\n", p2.roll_no);  
    printf("name : %s\n", p2.name);  
    printf("phone_number : %d\n", p2.phone_number);  
    return 0;  
}
```

8) Cho biết các giá trị xuất ra màn hình của đoạn lệnh sau:

```
#include <stdio.h>  
struct student {  
    char name[30];  
    int roll_no;  
};  
int main() {  
    struct student stud = {"Alex",2};  
    struct student *ptr;  
    ptr = &stud;  
    printf("%s %d\n",stud.name,stud.roll_no);  
    printf("%s %d\n",ptr->name,ptr->roll_no);  
    return 0;  
}
```

9) Cho biết các giá trị xuất ra màn hình của đoạn lệnh sau:

```
#include <stdio.h>  
#include <string.h>  
struct student {
```

```

int roll_no;
char name[30];
int phone_number;
};

int main() {
    struct student p1 = { 1,"Elsa",113112443 };
    struct student p2, p3;
    p2.roll_no = 2;
    strcpy(p2.name,"Hans");
    p2.phone_number = 113112556;
    p3.roll_no = 3;
    strcpy(p3.name,"Olaf");
    p3.phone_number = 113112754;
    printf("First Student\n");
    printf("roll_no : %d\n", p1.roll_no);
    printf("name : %s\n", p1.name);
    printf("phone_number : %d\n", p1.phone_number);
    printf("Second Student\n");
    printf("roll_no : %d\n", p2.roll_no);
    printf("name : %s\n", p2.name);
    printf("phone_number : %d\n", p2.phone_number);
    printf("Third Student\n");
    printf("roll_no : %d\n", p3.roll_no);
    printf("name : %s\n", p3.name);
    printf("phone_number : %d\n", p3.phone_number);
    return 0;
}

```

10) Cho biết các giá trị xuất ra màn hình của đoạn lệnh sau:

```

#include <stdio.h>
#include <string.h>
struct student {
    int roll_no;
    char name[30];
    int phone_number;
};
void display(struct student *st) {
    printf("Roll no : %d\n", st -> roll_no);
    printf("Name : %s\n", st -> name);
    printf("Phone no : %d\n", st -> phone_number);
}
int main() {
    struct student s;

```

```
s.roll_no = 4;  
strcpy(s.name,"Christof");  
s.phone_number = 98765432;  
display(&s);  
return 0;  
}
```

8.7 Bài tập có lời giải

- 1) Viết chương trình nhập vào hai phân số sau đó hiển thị dạng rút gọn của 2 phân số, tổng, hiệu, tích và thương của 2 phân số đó.

```
#include <stdio.h>  
#include <string.h>  
#include <math.h>  
#include <conio.h>  
int USCLN (int a, int b) {  
    a = abs(a);  
    b = abs(b);  
    while (a * b != 0) {  
        if (a > b) a %= b;  
        else b %= a;  
    }  
    return a + b;  
}  
  
int BSCNN (int a, int b) {  
    return a * b / USCLN(a, b);  
}  
  
typedef struct PhanSo {  
    int tuso, mauso;  
} PS;  
  
PS rutGon(PS a) {  
    PS c;  
    c.tuso = a.tuso / USCLN (a.tuso, a.mauso);  
    c.mauso = a.mauso / USCLN (a.tuso, a.mauso);  
    return c;  
}  
  
PS cong(PS a, PS b) {  
    PS c;  
    c.tuso = a.tuso * b.mauso + a.mauso * b.tuso;
```

```
c.mauso = a.mauso * b.mauso;
c = rutGon(c);
return c;
}

PS tru (PS a, PS b) {
    PS c;
    c.tuso = a.tuso * b.mauso - a.mauso * b.tuso;
    c.mauso = a.mauso * b.mauso;
    c = rutGon(c);
    return c;
}

PS nhan (PS a, PS b) {
    PS c;
    c.tuso = a.tuso * b.tuso;
    c.mauso = a.mauso * b.mauso;
    c = rutGon(c);
    return c;
}

PS chia (PS a, PS b) {
    PS c;
    c.tuso = a.tuso * b.mauso;
    c.mauso = a.mauso * b.tuso;
    c = rutGon(c);
    return c;
}

void print(PS a) {
    printf("%d/%d", a.tuso, a.mauso);
}

int main() {
    PS a, b, c;
    printf("\nNhap phan so a: ");
    scanf("%d %d", &a.tuso, &a.mauso);
    printf("\nNhap phan so b: ");
    scanf("%d %d", &b.tuso, &b.mauso);
    printf("\nToi gian a: "); a = rutGon(a); print(a);
    printf("\nToi gian b: "); b = rutGon(b); print(b);
    printf("\nTong cua hai phan so = "); c = cong(a, b); print(c);
    printf("\nHieu cua hai phan so = "); c = tru(a, b); print(c);
```

```

        printf("\nTich cua hai phan so = "); c = nhan(a, b); print(c);
        printf("\nThuong cua hai phan so = "); c = chia(a, b); print (c);
        getch();
        return 0;
    }

```

- 2) Viết chương trình cho phép nhập vào số lượng môn học, tên môn học và số tín chỉ tương ứng của môn học, sau đó in ra màn hình thông tin tất cả các môn học, bao gồm tên môn học và số tín chỉ tương ứng.

```

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
struct MonHoc {
    int sotc;           //so tin chi
    char tenmh[30];    //ten mon hoc
};

int main() {
    struct MonHoc *ptr;
    int i, slmh;        //so luong mon hoc
    printf("Nhap vao so luong mon hoc: ");
    scanf("%d", &slmh);
    ptr = (struct MonHoc *)malloc(slmh * sizeof(struct MonHoc));
    for (i = 0; i < slmh; ++i) {
        printf("Nhap vao ten mon hoc va so tin chi tuong ung:\n");
        scanf("%s %d", &(ptr + i)->tenmh, &(ptr + i)->sotc);
    }
    printf("\nHIEN THI THONG TIN:\n");
    for (i = 0; i < slmh; ++i)
        printf("%s\t%d\n", (ptr + i)->tenmh, (ptr + i)->sotc);
    getchar();
    return 0;
}

```

- 3) Viết chương trình tính khoảng cách giữa hai điểm thời gian. Mỗi điểm thời gian là một cấu trúc gồm các thông tin: giờ, phút, giây.

```

#include <stdio.h>
#include <conio.h>
struct mTime {
    int sec;
    int min;

```

```

        int hour;
    };
void subtract(struct mTime t1, struct mTime t2, struct mTime *d);
int main() {
    struct mTime t1, t2, d;
    printf("THOI DIEM KET THUC \n"); printf("Nhap vao gio, phut, giay: ");
    scanf("%d %d %d", &t1.hour, &t1.min, &t1.sec);
    printf("THOI DIEM BAT DAU\n"); printf("Nhap vao gio, phut, giay: ");
    scanf("%d %d %d", &t2.hour, &t2.min, &t2.sec);
    subtract(t1, t2, &d);
    printf("\nKhoang thoi gian la: %d:%d:%d - ", t1.hour, t1.min, t1.sec);
    printf("%d:%d:%d ", t2.hour, t2.min, t2.sec);
    printf("= %d:%d:%d\n", d.hour, d.min, d.sec);
    return 0;
}
void subtract(struct mTime t1, struct mTime t2, struct mTime *d) {
    while (t2.sec > t1.sec) {
        --t1.min;
        t1.sec += 60;
    }
    d->sec = t1.sec - t2.sec;
    while (t2.min > t1.min) {
        --t1.hour;
        t1.min += 60;
    }
    d->min = t1.min - t2.min;
    d->hour = t1.hour - t2.hour;
}

```

8.8 Bài tập đề nghị

- 1) Viết chương trình nhập vào mảng gồm n phân số, sau đó xuất ra màn hình các phân số sau khi đã được tối giản, tìm phân số nhỏ nhất/lớn nhất trong mảng, sắp xếp mảng theo thứ tự tăng dần/giảm dần.
- 2) Viết chương trình nhập vào tọa độ 2 điểm A, B sau đó tính khoảng cách giữa 2 điểm.
- 3) Viết chương trình nhập vào tọa độ một điểm sau đó xuất ra màn hình tọa độ các điểm đối xứng với điểm vừa nhập qua trực tung và trực hoành.
- 4) Viết chương trình nhập vào tọa độ 3 điểm của tam giác sau đó xuất ra màn hình chu vi, diện tích của tam giác đó.

- 5) Viết chương trình nhập vào danh sách các học sinh, sau đó xuất ra màn hình danh sách theo thứ tự giảm dần của điểm tổng kết.

CONFIDENTIAL

CHƯƠNG 9. KIỂU TẬP TIN

Trong chương này, chúng ta sẽ làm quen với kiểu tập tin trong ngôn ngữ C, bao gồm các khái niệm, các thao tác trên tập tin, hàm có tập tin là tham số và một số hàm xử lý tập tin thông dụng.

Nội dung:

- Một số khái niệm
- Thao tác trên tập tin
- Hàm có tập tin là tham số
- Một số hàm xử lý tập tin thông dụng

9.1 Một số khái niệm

Việc lưu trữ dữ liệu lâu dài là cần thiết do việc quản lý, điều khiển một lượng dữ liệu lớn bằng các chương trình là khó khăn. Hơn nữa, sau khi thực thi chương trình xong, tất cả dữ liệu được nhập sẽ bị mất do dữ liệu lưu trữ thông qua các biến có tính chất tạm thời. Ngôn ngữ C cung cấp khái niệm tập tin để giải quyết vấn đề này.

9.1.1 Tập tin

Một tập tin là một chuỗi các byte dữ liệu có liên quan với nhau được lưu trữ trên đĩa. Tập tin được tạo để lưu trữ dữ liệu bền vững. Ngôn ngữ C cho phép truy xuất để điều khiển các tập tin trên thiết bị lưu trữ thông qua các hàm chức năng ở mức cao và các lời gọi ở mức thấp (mức hệ điều hành).

9.1.2 Phân loại tập tin

Phụ thuộc vào tiêu chí phân loại, có thể có các loại tập tin sau đây:

➤ Phân loại theo cách truy cập:

- Tập tin truy cập tuần tự: Trong kiểu tập tin này, dữ liệu được lưu trữ một cách tuần tự; do đó, để đọc được một phần tử bất kỳ trong tập tin thì phải đọc qua tất cả các phần tử trước nó.

- Tập tin truy cập ngẫu nhiên: Đối với loại tập tin này, dữ liệu có thể đọc và chỉnh sửa một cách ngẫu nhiên. Ta có thể đọc trực tiếp một phần tử bất kỳ một cách nhanh chóng, không tốn nhiều thời gian như tập tin tuần tự.

➤ *Phân loại theo bản chất dữ liệu*

- Tập tin văn bản (phần mở rộng .txt):
 - ✓ Với loại tập tin này, con người có thể đọc được vì mọi thứ được lưu trữ dưới dạng văn bản.
 - ✓ Dữ liệu được lưu thành các dòng, mỗi dòng được kết thúc bằng ký tự xuống dòng là CR (Carriage Return – về đầu dòng, mã 10) và LF (Line Feed – xuống dòng, mã 13). Tập tin văn bản kết thúc bằng ký tự EOF (End Of File) có mã 26 (Ctrl + Z)
- Tập tin nhị phân (phần mở rộng .com, .exe):
 - ✓ Với loại tập tin này, con người không thể đọc được vì mọi thứ được viết dưới dạng các ký tự nhị phân 0 và 1.

9.1.3 Biến tập tin và con trỏ tập tin

➤ Biến tập tin

- Biến thuộc kiểu tập tin dùng đại diện cho một tập tin.
- Dữ liệu chứa trong tập tin được truy xuất thông qua thao tác với biến tập tin.

➤ Con trỏ tập tin

- Dùng để xác định vị trí của phần tử hiện tại để đọc hoặc ghi trên tập tin.
- Khi tập tin được mở để đọc hoặc ghi, con trỏ tập tin luôn ở vị trí đầu tập tin.
- Mỗi khi đọc hoặc ghi trên tập tin, con trỏ tập tin tự động tăng lên một khoảng theo đúng số byte vừa đọc hoặc ghi trên tập tin.

9.2 Thao tác trên tập tin

9.2.1 Khai báo biến tập tin

Cú pháp khai báo tập tin sử dụng con trỏ tập tin:

FILE *fp;

9.2.2 Mở và đóng tập tin

Mở tập tin: Tập tin cần được mở trước đọc hoặc ghi. Hàm *fopen()* được sử dụng để mở tập tin. Hàm trả về con trỏ biến tập tin, trỏ đến vị trí đầu tiên của tập tin nếu mở tập tin thành công, trường hợp có lỗi hàm trả về NULL. Cú pháp hàm *fopen()* như sau:

FILE *fopen(const char *filename, const char *mode);

trong đó:

filename: Tên tập tin

mode: Chế độ mở tập tin (Bảng 9.1)

Bảng 9.1. Chế độ mở tập tin

| Giá trị mode | Ý nghĩa |
|--------------|---|
| r | Mở tập tin văn bản để đọc. Nếu tập tin không tồn tại, hàm trả về giá trị NULL. |
| w | Mở tập tin văn bản để ghi, ghi đè lên tập tin đã có. Nếu tập tin chưa tồn tại thì nó sẽ được tạo mới. |
| a | Mở tập tin văn bản và ghi nối vào cuối tập tin. Nếu tập tin chưa có nó sẽ được tạo mới. |
| r+ | Mở tập tin văn bản để đọc và ghi. Nếu tập tin không tồn tại, hàm trả về giá trị NULL. |
| w+ | Mở tập tin văn bản để ghi và đọc, ghi đè lên tập tin đã có. Nếu tập tin chưa tồn tại, nó sẽ được tạo mới. |
| a+ | Mở tập tin văn bản hoặc tạo mới để đọc và ghi nối vào cuối. Nếu tập tin chưa tồn tại thì nó sẽ được tạo mới. |
| rb | Mở tập tin nhị phân để đọc. Nếu tập tin không tồn tại, hàm trả về giá trị NULL. |
| wb | Mở tập tin nhị phân để ghi, ghi đè lên tập tin đã có. Nếu tập tin chưa tồn tại, nó sẽ được tạo mới. |
| ab | Ghi nối vào cuối tập tin nhị phân. Nếu tập tin chưa tồn tại nó sẽ được tạo mới. |
| rb+ | Mở ra tập tin nhị phân để đọc và ghi. Nếu tập tin không tồn tại, hàm trả về giá trị NULL. |
| wb+ | Tạo ra tập tin nhị phân để đọc/ghi. Nếu tập tin tồn tại thì nội dung được ghi đè, nếu chưa tồn tại thì sẽ được tạo mới. |
| ab+ | Mở tập tin nhị phân để đọc và nối vào cuối tập tin. Nếu tập tin chưa tồn tại thì nó sẽ được tạo mới. |

Ví dụ 9.1. Các chế độ mở tập tin trong ngôn ngữ lập trình C

```
fopen ("D:\\CSLT\\openfile_1.txt","w"); //Mo tap tin van ban de ghi  
fopen ("D:\\CSLT\\openfile_2.bin","rb"); //Mo tap tin nhi phan de doc
```

Đóng tập tin: Tập tin cần được đóng sau khi sử dụng hoặc trước khi kết thúc chương trình để giải phóng bộ nhớ. Hàm *fclose()* được sử dụng để đóng tập tin *fp*. Hàm trả về 0 nếu đóng tập tin thành công, trường hợp có lỗi hàm trả về EOF. Cú pháp của hàm *fclose()* như sau:

```
fclose(FILE *fp);
```

9.2.3 Đọc/ghi dữ liệu

9.2.3.1 Đọc và ghi dữ liệu từ/lên tập tin văn bản

➤ Đọc dữ liệu từ tập tin văn bản

- Hàm **getc()** được dùng để đọc một ký tự từ tập tin *fp*, giá trị trả về là ký tự đọc được hoặc EOF trong trường hợp bị lỗi. Cú pháp của hàm như sau:

```
int getc(FILE *fp);
```

- Hàm **fgets()** được dùng để đọc và trả về một dòng (tối đa **maxlen** ký tự) từ tập tin *fp*. Cú pháp của hàm như sau:

```
char[] fgets(char line[], int maxlen, FILE *fp);
```

- Hàm **fscanf()** được sử dụng để đọc các thành phần có định dạng **format** từ tập tin *fp*. Hàm hoạt động tương tự như hàm **scanf()** nhưng thay vì đọc dữ liệu từ thiết bị nhập thì nó sẽ đọc dữ liệu từ tập tin. Các tham số của hàm **fscanf()** cũng tương tự như hàm **scanf()**. Nếu thành công, hàm trả về số lượng các giá trị đọc được; ngược lại hàm trả về EOF hoặc -1. Cú pháp của hàm như sau:

```
int fscanf(FILE *fp, const char *format [, argument, ...]);
```

➤ Ghi dữ liệu lên tập tin văn bản

- Hàm **putc()** được sử dụng để ghi ký tự *ch* lên tập tin *fp* hoặc trả về EOF nếu ghi không thành công. Cú pháp của hàm như sau:

```
int putc (int ch, FILE *fp)
```

- Hàm **fputs()** được sử dụng để ghi một chuỗi ký tự *str* lên tập tin *fp*, không bao gồm ký tự NULL. Nếu thành công hàm trả về một giá trị không âm, ngược lại hàm trả về EOF. Cú pháp của hàm như sau:

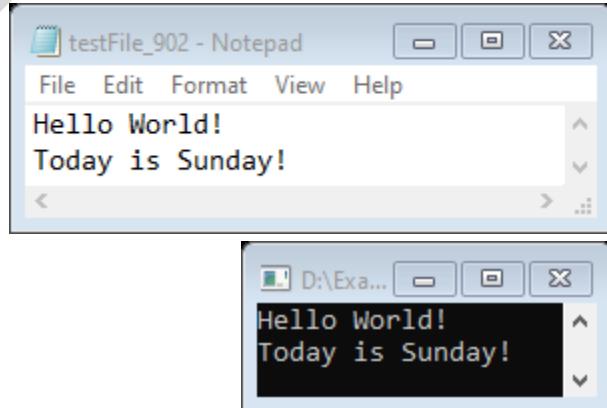
*int fputs (const char *str, FILE *fp)*

- Hàm **fprintf()** được sử dụng để ghi vào tập tin *fp* các thành phần có định dạng *format*. Hàm hoạt động tương tự như hàm printf() nhưng thay vì ghi dữ liệu ra màn hình thì nó sẽ ghi dữ liệu vào tập tin. Các tham số của hàm fprintf() cũng tương tự như hàm printf(). Nếu thành công, hàm trả về số lượng các giá trị ghi được; ngược lại hàm trả về EOF hoặc -1. Cú pháp của hàm như sau:

*int fprintf (FILE *fp, const char *format, ...)*

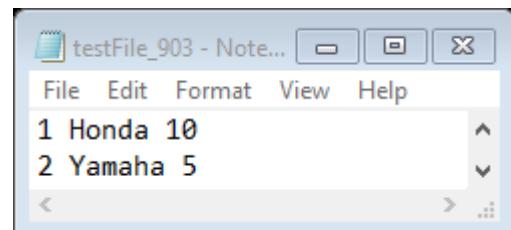
Ví dụ 9.2. Hiển thị nội dung tập tin văn bản ra màn hình (sử dụng getc)

```
#include <stdio.h>
int main() {
    FILE *fp = fopen("testFile_902.txt", "r");
    int ch;
    if (fp == NULL)
        printf("\nKhong the mo tap tin hoac tap tin khong ton tai.");
    else {
        ch = getc(fp);
        while (ch != EOF) {
            putchar(ch);
            ch = getc(fp);
        }
    }
    fclose(fp);
    getchar();
    return 0;
}
```



Ví dụ 9.3. Hiển thị nội dung tập tin văn bản ra màn hình (sử dụng fscanf)

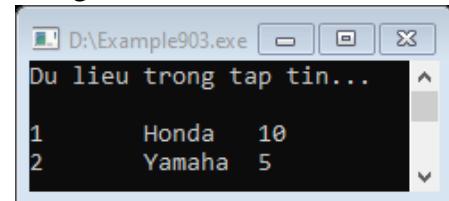
```
#include <stdio.h>
int main() {
    FILE *fp;
    char ch;
    int stt, soluong;
    char tenhang[10];
```



```

fp = fopen("testFile_903.txt","r");
if(fp == NULL) {
    printf("\nKhong the mo tap tin hoac tap tin khong ton tai.");
} else {
    printf("\nDu lieu trong tap tin...\n");
    while((fscanf(fp,"%d\t%s\t%d",&sdt,tenhang,&soluong))!=EOF)
        printf("\n%d\t%s\t%d",sdt,tenhang,soluong);
    fclose(fp);
}
getchar();
return 0;
}

```



9.2.3.2 Đọc và ghi dữ liệu từ/lên tập tin nhị phân

➤ *Ghi dữ liệu từ tập tin nhị phân*

Hàm **fwrite()** được sử dụng để ghi dữ liệu lên tập tin nhị phân với cú pháp sau:

size_t fwrite (const void *ptr, size_t size, size_t nmemb, FILE *stream)

trong đó:

- ptr: địa chỉ của khối dữ liệu cần ghi
- size: kích thước của mỗi phần tử được ghi
- nmemb: số phần tử được ghi
- stream: con trỏ chỉ tới tập tin được ghi

Hàm trả về số phần tử được ghi thành công lên tập tin, hoặc trả về một giá trị nhỏ hơn *nmemb* nếu bị lỗi. Chú ý các tham số *size* và *nmemb* và giá trị trả về của hàm có kiểu *size_t*, là kiểu unsigned int.

➤ *Đọc dữ liệu lên tập tin nhị phân*

Hàm **fread()** được sử dụng để đọc dữ liệu từ tập tin nhị phân với cú pháp sau:

size_t fread (void *ptr, size_t size, size_t nmemb, FILE *stream)

trong đó:

- ptr: con trỏ tới vùng dữ liệu được đọc
- size: kích thước của mỗi phần tử được đọc
- nmemb: số lượng các phần tử được đọc
- stream: con trỏ tới tập tin được đọc

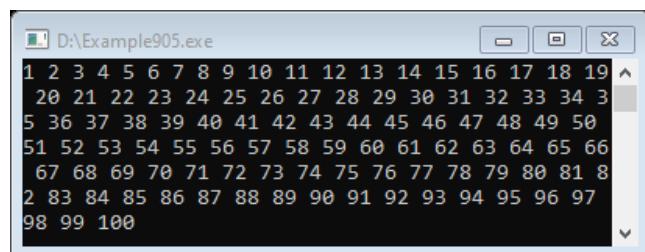
Hàm trả về số lượng phần tử được đọc thành công hoặc trả về số khác *nmembr* khi xảy ra lỗi trong quá trình đọc hoặc EOF.

Ví dụ 9.4. Ghi dữ liệu vào tập tin nhị phân

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    FILE *fp=fopen("testFile_904.txt","wb");
    if (fp == NULL) printf("\nKhong the mo tap tin hoac tap tin khong ton tai.");
    else {
        for (int i=1;i<=100;i++) fwrite(&i,sizeof(int),1,fp);
        fclose(fp);
    }
    getchar();
    return 0;
}
```

Ví dụ 9.5. Đọc dữ liệu từ tập tin nhị phân

```
/*Doc du lieu tu tap tin o Vi du 9.4*/
#include <stdio.h>
#include <stdlib.h>
int main() {
    FILE *fp=fopen("testFile_904.txt","rb");
    int i;
    do {
        fread(&i,sizeof(int),1,fp);
        if (!feof(fp)) printf("%d ",i);
    }
    while (!feof(fp));
    fclose(fp);
    getchar();
    return 0;
}
```



9.2.4 Di chuyển con trỏ trong tập tin

Ngôn ngữ lập trình C cung cấp hàm **fseek()** cho phép thay đổi vị trí con trỏ trong tập tin với cú pháp sau:

int fseek (FILE *stream, long int offset, int origin)

trong đó:

- stream:* con trỏ tới tập tin
- offset:* số nguyên chỉ định số byte con trỏ di chuyển tới kể từ vị trí được xác định bởi tham số *origin*
- origin:* số nguyên chỉ định vị trí ban đầu của con trỏ, được chỉ định thông qua 3 hằng số sau:
- **SEEK_SET:** vị trí bắt đầu của tập tin
 - **SEEK_CUR:** vị trí hiện tại của con trỏ tập tin
 - **SEEK_END:** vị trí cuối tập tin

Hàm trả về giá trị 0 nếu thành công, ngược lại hàm trả về giá trị khác 0.

Ví dụ 9.6. Di chuyển con trỏ trong tập tin

```
#include <stdio.h>
#include <stdlib.h>
int main () {
    FILE *fp= fopen("testFile_906.txt","r");
    char ch;
    if (fp==NULL) {
        printf("\nKhong the mo tap tin.");
        exit(1);
    }
    fseek (fp,15,SEEK_SET); //Nhay den vi tri 15 (tinh tu chi so 0)
    while (!feof(fp)) {
        ch=fgetc(fp);
        printf("%c",ch);
    }
    fclose(fp);
    getchar();
    return 0;
}
```

9.3 Một số hàm xử lý tập tin thông dụng

| Hàm và cú pháp | Mô tả |
|-------------------------------------|--|
| int ferror (FILE *fp); | Kiểm tra có lỗi trong quá trình đọc/ghi đối với tập tin <i>fp</i> . Nếu không có lỗi, hàm trả về 0, ngược lại, hàm trả về khác 0 |
| int remove (const char *fp); | Xóa tập tin có tên <i>fp</i> ; nếu thành công, hàm trả về 0, ngược lại, hàm trả về -1 |

| | |
|---|---|
| int rename (const char *old_filename, const char *new_filename) | Đổi tên tập tin từ tên cũ (<i>old_filename</i>) thành tên mới (<i>new_filename</i>); nếu thành công, hàm trả về 0, ngược lại, hàm trả về -1 |
| long int ftell (FILE *fp) | Trả về kích thước tập tin <i>fp</i> |

Ví dụ 9.7. Kiểm tra lỗi đối với thao tác trên tập tin

```
#include <stdio.h>
#include <conio.h>
int main () {
    FILE *fp;
    char c;
    fp = fopen("testFile_9.7.txt", "w"); //Mo tap tin de ghi
    c = fgetc(fp);                  //Doc tap tin -> Lỗi phát sinh
    if(ferror(fp))
        printf("Co loi trong qua trinh doc file.\n");
    else
        printf("\nDoc file thanh cong");
    clearerr(fp);
    fclose(fp);
    getchar();
    return(0);
}
```

Hãy thử thay đổi kí tự “w” được in đậm ở trên thành kí tự “r” để thấy sự khác biệt.

Ví dụ 9.8. Đổi tên tập tin

```
#include <stdio.h>
#include <conio.h>
int main() {
    int kq;
    char oldname[] = "D:\\CSLT\\testFile_9.8.txt";
    char newname[] = "D:\\CSLT\\newTestFile_9.8.txt";
    kq = rename (oldname, newname);
    if (kq == 0)
        printf ("Doi ten tap tin thanh cong.");
    else
        printf ("Doi ten tap tin khong thanh cong.");
    getchar();
    return(0);
}
```

Ví dụ 9.9. Xác định kích thước của tập tin

```
#include <stdio.h>
#include <conio.h>
int main () {
    FILE *fp;
    int len;
    fp = fopen("D:\\CSLT\\testFile_9.9.txt", "r");
    if(fp == NULL) {
        perror("Loi xay ra khi mo tap tin.");
        return(-1);
    }
    fseek(fp, 0, SEEK_END);
    len = ftell(fp);
    fclose(fp);
    printf("Kich thuoc cua tap tin la: %d bytes", len);
    getchar();
    return 0;
}
```

9.4 Hàm có tập tin là tham số

Ví dụ 9.10. Truyền tham số là tập tin cho hàm

```
#include <stdio.h>
#include <conio.h>
FILE* fileopen();
void read_line(FILE *fh);
int main() {
    FILE *fh= fileopen();
    read_line(fh);
    return 0;
}
void read_line(FILE *fh){
    char s[50];
    while(fgets(s,49,fh)!=NULL)
        printf("%s", s);
    fclose(fh);
}
FILE* fileopen(){
    FILE *file = fopen("D:\\CSLT\\testFile_9.10.txt", "r");
    return file;
}
```

9.5 Câu hỏi ôn tập

- 1) Tập tin là gì? Nêu một số tiêu chí dùng để phân loại tập tin.
- 2) Phân biệt các hàm đọc dữ liệu từ tập tin văn bản sau: getc(), fgets(), fscanf().
- 3) Phân biệt các hàm ghi dữ liệu lên tập tin văn bản sau: putc(), fputs(), fprintf().
- 4) Nêu các hàm đọc ghi dữ liệu lên tập tin nhị phân.
- 5) Nêu công dụng của các hàm thao tác tập tin sau: perror(), remove(), rename(), ftell().
- 6) Chương trình sau đây thực hiện chức năng gì?

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    char sentence[1000];
    FILE *fptr;
    fptr = fopen("D://program.txt", "w");
    if(fptr == NULL) {
        printf("Error!");
        exit(1);
    }
    printf("Enter a sentence:\n");
    gets(sentence);
    fprintf(fptr,"%s", sentence);
    fclose(fptr);
    return 0;
}
```

- 7) Chương trình sau đây thực hiện chức năng gì?

```
#include <stdio.h>
int main() {
    char name[50];
    int marks, i, num;
    printf("Enter number of students: ");
    scanf("%d", &num);
    FILE *fptr;
    fptr = (fopen("D:\\student.txt", "w"));
    if(fptr == NULL) {
        printf("Error!");
        exit(1);
    }
    for(i = 0; i < num; ++i) {
        printf("For student%d\\nEnter name: ", i+1);
```

```

        scanf("%s", name);
        printf("Enter marks: ");
        scanf("%d", &marks);
        fprintf(fp, "\nName: %s \nMarks=%d \n", name, marks);
    }
    fclose(fp);
    return 0;
}

```

8) Chương trình sau đây thực hiện chức năng gì?

```

#include <stdio.h>
void sort_numbersAscending(int number[], int count) {
    int temp, i, j, k;
    for (j = 0; j < count; ++j) {
        for (k = j + 1; k < count; ++k) {
            if (number[j] > number[k]) {
                temp = number[j];
                number[j] = number[k];
                number[k] = temp;
            }
        }
    }
    FILE *fp;
    fp = (fopen("D:\\data.txt", "w"));
    for (i = 0; i < count; ++i)
        fprintf(fp, "%d\n", number[i]);
}
int main() {
    int i, count, number[20];
    printf("How many numbers you are gonna enter:");
    scanf("%d", &count);
    printf("\nEnter the numbers one by one:");
    for (i = 0; i < count; ++i)
        scanf("%d", &number[i]);
    sort_numbersAscending(number, count);
}

```

9) Chương trình sau đây thực hiện chức năng gì?

```

#include<stdio.h>
#include<conio.h>
void main() {
    FILE *fp;
    char ch;

```

```

int size = 0;
fp = fopen("D:\\data.txt", "r");
if (fp == NULL) printf("\nFile unable to open...");  

else {
    printf("\nFile opened...");  

    fseek(fp, 0, 2);
    size = ftell(fp);
    printf("The size of given file is: %d\n", size);
    fclose(fp);
}
}

```

10) Chương trình sau đây thực hiện chức năng gì?

```

#include<stdio.h>
#include<dirent.h>
int main() {
    DIR *d;
    struct dirent *dir;
    d = opendir(".");
    if (d) {
        while ((dir = readdir(d)) != NULL) {
            printf("%s\n", dir->d_name);
        }
        closedir(d);
    }
    return(0);
}

```

9.6 Bài tập có lời giải

1) Viết chương trình đếm số dòng trong một tập tin văn bản.

```

#include <stdio.h>
#include <conio.h>
int main() {
    FILE *fp;
    int rows = 0;
    char c;
    printf("\nCHUONG TRINH DEM SO DONG CUA TAP TIN\n");
    printf("*****\n");
    fp = fopen("D:\\CSLT\\Bai9.1.txt", "r");
    if (fp == NULL) {
        printf("Co loi trong qua trinh mo tap tin.");
        return -1;
    }
}

```

```

    }
    for (c = getc(fp); c != EOF; c = getc(fp))
        if (c == '\n')
            rows = rows + 1;
    fclose(fp);
    printf("So dong trong tap tin la: %d \n", rows +1);
    getchar();
    return 0;
}

```

- 2) Viết chương trình đếm số từ và số ký tự trong một tập tin. Tên tập tin được truyền như tham số khi chạy chương trình.

```

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int main(int argc, char *argv[]) {
    FILE *fp;
    char ch;
    int nWords=1, nChars=1;
    printf("\nCHUONG TRINH DEM SO TU VA SO KI TU CUA TAP TIN.\n");
    printf("*****\n");
    fp=fopen(argv[1],"r");
    if(fp==NULL)
        printf("Khong the mo tap tin.");
    else {
        ch=fgetc(fp);
        printf("NOI DUNG CUA TAP TIN %s LA:\n\n",argv[1]);
        while(ch!=EOF) {
            printf("%c",ch);
            if(ch==' '||ch=='\n')
                nWords++;
            else
                nChars++;
            ch=fgetc(fp);
        }
        printf("\n\nSo tu trong tap tin %s la : %d\n",argv[1],nWords -1);
        printf("So ki tu trong tap tin %s la : %d\n\n",argv[1],nChars -1);
    }
    fclose(fp);
    getchar();
    return 0;
}

```

- 3) Viết chương trình xóa một dòng trong tập tin. Tên của tập tin và dòng cụ thể được chỉ định khi chạy chương trình.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#define MAX 256
int main(int argc, char *argv[]) {
    int row, count = 0;
    char ch;
    FILE *fp1, *fp2;
    char str[MAX], temp[] = "C:\\DATA\\NCKH\\tam.txt";
    printf("\n\nCHUONG TRINH XOA MOT DONG TRONG TAP TIN\n");
    printf("*****\n");

    fp1 = fopen(argv[1], "r");
    if (!fp1) {
        printf("Co loi xay ra trong qua trinh mo tap tin.\n");
        return -1;
    }
    fp2 = fopen(temp, "w");
    if (!fp2) {
        printf("Khong the mo tap tin tam de ghi.\n");
        fclose(fp1);
        return -1;
    }
    row = atoi(argv[2]);
    // Sao chep tat ca noi dung vao tap tin tam, ngoai tru dong duoc chi dinh xoa.
    while (!feof(fp1)) {
        strcpy(str, "\0");
        fgets(str, MAX, fp1);
        if (!feof(fp1)) {
            count++;
            // Bo qua dong duoc chi dinh xoa
            if (count != row)
                fprintf(fp2, "%s", str);
        }
    }
    fclose(fp1);
    fclose(fp2);
    remove(argv[1]);           // Xoa tap tin doc
    rename(temp, argv[1]);     // Doi ten tap tin tam thanh ten cua tap tin goc
```

```

//Hien thi noi dung cua tap tin sau khi xoa
fp1=fopen(argv[1],"r");
ch=fgetc(fp1);
printf("NOI DUNG CUA TAP TIN SAU KHI XOA LA:\n");
while(ch!=EOF) {
    printf("%c",ch);
    ch=fgetc(fp1);
}
fclose(fp1);
getchar();
return 0;
}

```

- 4) Viết chương trình in nội dung đảo ngược của một tập tin.

```

#include <stdio.h>
#include <string.h>
#include <conio.h>
int main(int argc, char *argv[]) {
    FILE *fp1;
    int count = 0;
    int i = 0;
    if (argc < 2) {
        printf("Chua co ten tap tin.\n");
        return -1;
    }
    fp1 = fopen(argv[1], "r");
    if (fp1==NULL) {
        printf("\nKhong the mo tap tin %s.\n", argv[1]);
        return -1;
    }
    printf("\nNOI DUNG CUA TAP TIN SAU KHI DAO NGUOC LA:");
    printf("\n*****");
    //Di chuyen con tro den cuoi tap tin
    fseek(fp1, 0, SEEK_END);
    //Lay vi tri cua con tro tap tin
    count = ftell(fp1);
    while (i < count) {
        i++;
        fseek(fp1,-i,SEEK_END);
        printf("%c", fgetc(fp1));
    }
    printf("\n");
}

```

```
fclose(fp1);
getchar();
return 0;
}
```

9.7 Bài tập đề nghị

- 1) Viết chương trình nối nội dung của 2 tập tin thành một tập tin mới.
- 2) Viết chương trình in ra ký tự dài nhất và dòng dài nhất của một tập tin.
- 3) Cho 2 tập tin file1.txt và file2.txt gồm các số nguyên. Viết chương trình tạo tập tin file3.txt chứa tất cả các số của 2 tập tin file1.txt và file2.txt được sắp xếp theo thứ tự tăng dần.
- 4) Viết chương trình thay thế nội dung của một dòng trong tập tin.
- 5) Viết chương trình tìm tất cả các từ trong tập tin và thay thế bằng từ mới.