



BÀI GIẢNG LẬP TRÌNH C

Biên soạn: TS. Nguyễn Tuấn Linh

THÁI NGUYÊN – 02/2023

MỤC LỤC

MỤC LỤC	2
Chương 1: NGÔN NGỮ LẬP TRÌNH VÀ PHƯƠNG PHÁP LẬP TRÌNH	7
1.1. Ngôn ngữ lập trình (Programming Language)	7
1.1.1. Thuật giải (Algorithm)	7
1.1.2. Chương trình (Program)	7
1.1.3. Ngôn ngữ lập trình (Programming language)	7
1.2. Các bước lập trình	7
1.3. Kỹ thuật lập trình	7
1.3.1. Quy trình nhập - xử lý - xuất	7
1.3.2. Sử dụng lưu đồ (Flowchart)	8
Bài tập Chương 1	14
Chương 2: LÀM QUEN LẬP TRÌNH C QUA CÁC VÍ DỤ ĐƠN GIẢN	15
2.1. Cài đặt và sử dụng Dev-C++	15
2.1.1. Giới thiệu về Dev-C++	15
2.1.2. Thao tác cơ bản với Dev-C++	15
2.1.2.1. Cấu hình nhanh Dev-C++ (chỉ thực hiện lần đầu)	15
2.1.2.2. Tạo chương trình mới hoặc dự án mới	15
2.2. Các ví dụ làm quen với C	20
2.2.1. Ví dụ 1	20
2.2.2. Ví dụ 2	22
2.2.3. Ví dụ 3	23
2.2.4. Ví dụ 4	25
Câu hỏi và bài tập chương 2	26
Chương 3: CÁC THÀNH PHẦN TRONG NGÔN NGỮ C	27
3.1. Từ khóa	27
3.2. Tên	27
3.3. Kiểu dữ liệu	28
3.4. Ghi chú	28

3.5. Biến.....	29
3.5.1. Tên biến.....	29
3.5.2. Khai báo biến	29
3.5.3. Vừa khai báo vừa khởi gán	29
3.6. Lệnh và khối lệnh.....	29
3.6.1. Lệnh	29
3.6.2. Khối lệnh	30
3.7. Thư viện trong C	30
3.7.1. Khai báo thư viện.....	30
3.7.2. Một số thư viện chuẩn trong C	30
Câu hỏi chương 3	32
Chương 4: HÀM NHẬP - XUẤT DỮ LIỆU	33
4.1. Hàm printf.....	33
4.2. Hàm scanf	37
Bài tập Chương 4	38
Chương 5: CẤU TRÚC Rẽ NHÁNH CÓ ĐIỀU KIỆN.....	39
5.1. Lệnh if.....	39
5.1.1. Dạng 1 (if thiếu)	39
5.1.2. Dạng 2 (if đủ)	41
5.1.3. Cấu trúc else if.....	42
5.1.4. Cấu trúc if lồng nhau	44
5.2. Lệnh switch.....	45
5.2.1. Cấu trúc switch...case (switch thiếu).....	45
5.2.2. Cấu trúc switch...case...default (switch đủ).....	46
5.2.3. Cấu trúc switch lồng.....	47
Bài tập Chương 5	50
Chương 6 : CẤU TRÚC VÒNG LẶP.....	53
6.1. Lệnh for.....	53
6.2. Lệnh break.....	54

6.3. Lệnh continue.....	54
6.4. Lệnh while.....	55
6.5. Lệnh do...while.....	55
6.6. Vòng lặp lồng nhau	Error! Bookmark not defined.
6.7. So sánh sự khác nhau của các vòng lặp	Error! Bookmark not defined.
Bài tập Chương 6	56
Chương 7: HÀM.....	58
7.1. Giới thiệu về hàm.....	58
7.2. Các ví dụ về hàm.....	59
7.3. Tham số dạng tham biến và tham trị.....	60
7.4. Phạm vi của biến	61
7.5. Dùng dẫn hướng #define.....	62
Bài tập chương 7	63
Chương 8: MẢNG VÀ CHUỖI	64
8.1. Mảng	64
8.2. Mảng một chiều	64
8.1.1. Cách khai báo mảng một chiều.....	64
8.1.2. Tham chiếu đến từng phần tử mảng một chiều	64
8.1.4. Đọc dữ liệu từ mảng một chiều	65
8.1.5. Kỹ thuật nhập các phần tử mảng một chiều	65
8.1.6. Khởi tạo mảng một chiều	65
8.1.7. Khởi tạo mảng một chiều không bao hàm kích thước.....	65
8.3. Mảng hai chiều.....	65
8.3.1. Khai báo mảng hai chiều	65
8.3.2. Tham chiếu đến từng phần tử mảng 2 chiều.....	66
8.3.3. Nhập dữ liệu cho mảng 2 chiều	66
8.3.4. Đọc (in) dữ liệu từ mảng 2 chiều.....	67
8.3.5. Khởi tạo mảng 2 chiều.....	67
8.4. Chuỗi.....	67

8.4.1. Cách khai báo chuỗi.....	67
8.4.2. Hàm nhập (gets), xuất (puts) chuỗi.....	68
8.4.3. Khởi tạo chuỗi	68
8.4.4. Mảng chuỗi	68
Bài tập Chương 8	69
Chương 9. CON TRỎ TRONG C	70
9.1. Giới thiệu	70
9.2. Biến con trỏ	70
9.2.1. Khai báo biến con trỏ.....	71
9.2.2. Lấy địa chỉ của một biến.....	71
9.2.3. Cách truy xuất.....	72
9.2.4. Một số phép toán trên biến con trỏ	72
9.3. Cấp phát và thu hồi vùng nhớ	74
9.3.1 Cấp phát.....	74
9.3.2. Thu hồi và kiểm tra vùng nhớ còn lại.....	75
9.4. Hàm có đối là con trỏ	75
9.5. Mối quan hệ giữa con trỏ và mảng, chuỗi ký tự	76
9.5.1. Con trỏ và mảng 1 chiều.....	76
9.5.2. Nhập và xuất mảng trong hàm.....	78
9.5.3. Con trỏ và xâu ký tự	79
9.5.4. Con trỏ và mảng 2 chiều, mảng các con trỏ, con trỏ đa cấp.....	79
9.5.4.1. Con trỏ và mảng 2 chiều	79
9.5.4.2. Mảng các con trỏ	81
Bài tập Chương 9	84
Chương 10: CÁC KIỂU DỮ LIỆU TỰ ĐỊNH NGHĨA.....	85
10.1. Kiểu structure (kiểu cấu trúc).....	85
10.1.1. Khai báo kiểu structure.....	85
10.1.2. Cách khai báo biến có kiểu structure.....	85
10.1.3. Tham chiếu các phần tử trong structure	85

10.1.4. Khởi tạo structure	87
10.1.5. Structure lồng nhau.....	88
10.2. Kiểu Enum (kiểu liệt kê).....	89
10.2.1. Định nghĩa kiểu enum.....	89
10.2.2. Cách khai báo biến có kiểu enum.....	90
10.2.3 Sử dụng enum trong chương trình.....	91
Bài tập chương 10	93
Chương 11: MỘT SỐ BÀI TẬP VẬN DỤNG CÓ LỜI GIẢI.....	94

Chương 1: NGÔN NGỮ LẬP TRÌNH VÀ PHƯƠNG PHÁP LẬP TRÌNH

1.1. Ngôn ngữ lập trình (Programming Language)

1.1.1. Thuật giải (Algorithm)

Là một dãy các thao tác xác định trên một đối tượng, sao cho sau khi thực hiện một số hữu hạn các bước thì đạt được mục tiêu.

1.1.2. Chương trình (Program)

Là một tập hợp các mô tả, các phát biểu, nằm trong một hệ thống qui ước nhằm điều khiển máy tính làm việc. Chương trình có thể hiểu đơn giản như sau:

Chương trình = Thuật giải + Cấu trúc dữ liệu

Các thuật giải và chương trình đều có cấu trúc dựa trên **3 cấu trúc điều khiển cơ bản**:

* **Tuần tự** (Sequential)

* **Chọn lọc** (Selection)

* **Lặp lại** (Repetition)

Các bước để học lập trình chuyên nghiệp:

Tìm, xây dựng thuật giải (trên giấy) → viết chương trình (trên máy)
→ dịch chương trình → chạy và thử chương trình

1.1.3. Ngôn ngữ lập trình (Programming language)

Ngôn ngữ lập trình là hệ thống các ký hiệu tuân theo các qui ước về ngữ pháp và ngữ nghĩa, dùng để xây dựng thành các chương trình cho máy tính.

1.2. Các bước lập trình

Bước 1: Phân tích vấn đề và xác định các đặc điểm. (xác định **Input-Process-Output (I-P-O)**)

Bước 2: Lập ra giải pháp (đưa ra thuật giải)

Bước 3: Cài đặt (viết chương trình)

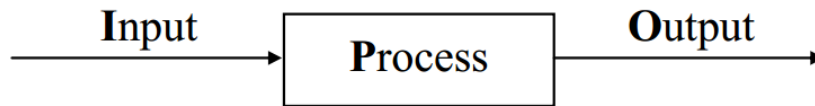
Bước 4: Chạy thử chương trình (dịch chương trình)

Bước 5: Kiểm chứng và hoàn thiện chương trình (thử nghiệm bằng nhiều số liệu và đánh giá).

1.3. Kỹ thuật lập trình

1.3.1. Quy trình nhập - xử lý - xuất

Còn gọi là I-P-O Cycle (Input-Process-Output Cycle), đây là quy trình xử lý cơ bản của máy tính, được mô tả bởi sơ đồ sau:



Ví dụ 1: Xác định Input, Process, Output của việc làm 1 ly nước chanh nóng

Input: Ly, đường, chanh, nước nóng, muỗng.

Process: - Cho hỗn hợp đường, chanh, nước nóng vào ly.
- Dùng muỗng khuấy đều.

Output: Ly chanh nóng đã sẵn sàng để dùng.

Ví dụ 2: Xác định Input, Process, Output của chương trình tính tiền lương công nhân tháng 10/2015 biết rằng lương = lương cơ bản * hệ số lương

Input: Lương căn bản, hệ số lương

Process: Nhân lương cơ bản với hệ số lương

Output: Lương

Ví dụ 3: Xác định Input, Process, Output của chương trình giải phương trình bậc nhất $ax + b = 0$

Input: Hệ số a, b

Process: Chia -b cho a

Output: Nghiệm x

Ví dụ 4: Xác định Input, Process, Output của chương trình tìm số lớn nhất của 2 số a và b.




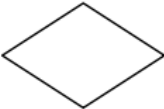
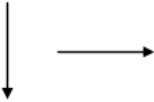


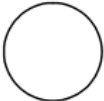
Input: a, b

Process: Nếu $a > b$ thì *Output* = a lớn nhất

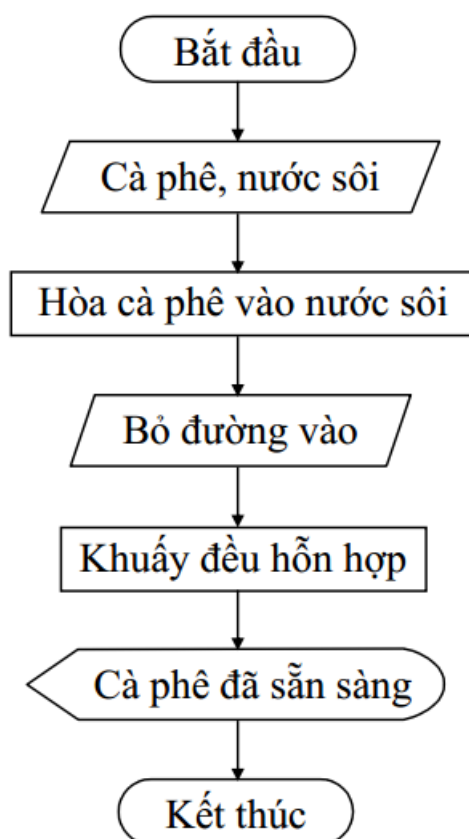
Ngược lại *Output* = b lớn nhất

1.3.2. Sử dụng lưu đồ (Flowchart)

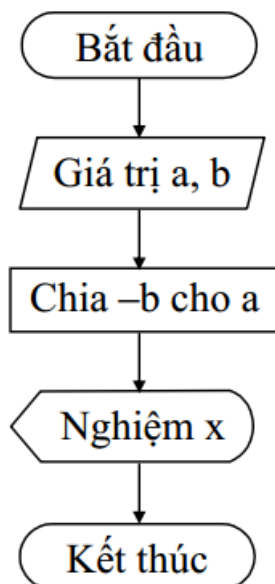
Để dễ hình dung hơn về quy trình xử lý, các nhà lập trình đưa ra dạng lưu đồ để minh họa từng bước quá trình xử lý một vấn đề (bài toán).

Hình dạng (symbol)	Hành động (Activity)
	Dữ liệu vào (Input)
	Xử lý (Process)
	Dữ liệu ra (Output)
	Quyết định (Decision), sử dụng điều kiện
	Luồng xử lý (Flow lines)
	Gọi CT con, hàm... (Procedure, Function...)
	Bắt đầu, kết thúc (Begin, End)
	Điểm ghép nối (Connector)

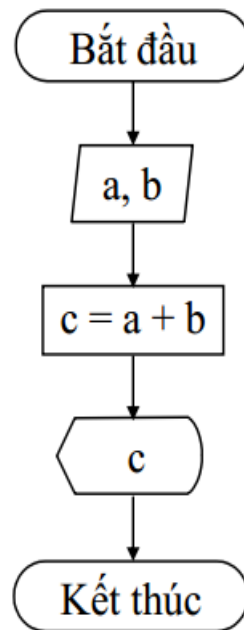
Ví dụ 1: Lưu đồ mô tả quá trình pha cà phê



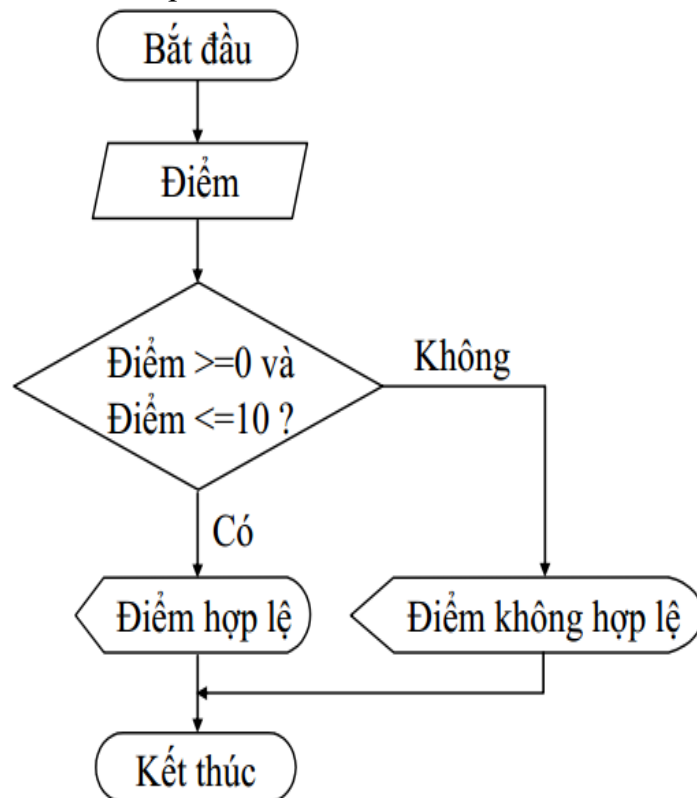
Ví dụ 2: Lưu đồ mô tả các bước giải phương trình bậc nhất $ax + b = 0$



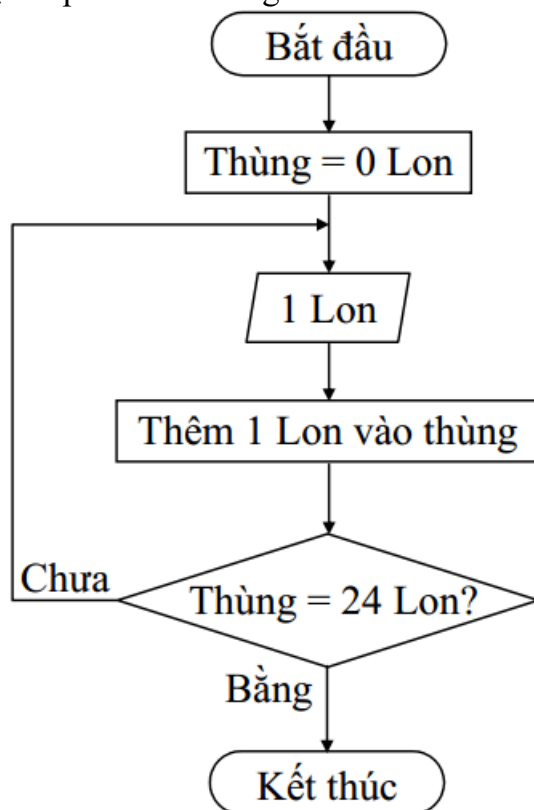
Ví dụ 3: Lưu đồ mô tả các việc cộng 2 số



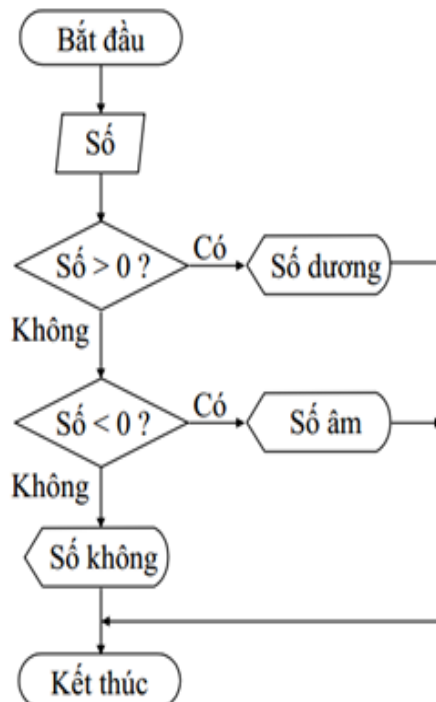
Ví dụ 4: Lưu đồ kiểm tra tính hợp lệ của điểm



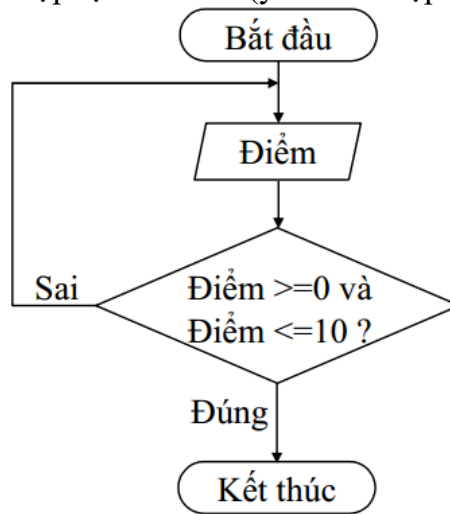
Ví dụ 5: Lưu đồ mô tả việc xếp lon vào thùng



Ví dụ 6: Lưu đồ kiểm tra một số là số âm hay số dương



Ví dụ 7: Lưu đồ kiểm tra tính hợp lệ của điểm (yêu cầu nhập lại)



Bài tập Chương 1

1. Xác định Input, Process, Output của các chương trình sau:

- a. Đổi từ tiền VND sang tiền USD.
- b. Tính điểm trung bình của học sinh gồm các môn Toán, Lý, Hóa.
- c. Giải phương trình bậc 2: $ax^2 + bx + c = 0$
- d. Đổi từ độ sang radian và đổi từ radian sang độ
(công thức $\alpha/\pi = a/180$, với α : radian, a: độ)
- e. Kiểm tra 2 số a, b bằng nhau hay khác nhau.

2. Vẽ lưu đồ cho các chương trình sau:

- a. Đổi từ tiền VND sang tiền USD.
- b. Tính điểm trung bình của học sinh gồm các môn Toán, Lý, Hóa.
- c. Giải phương trình bậc 2: $ax^2 + bx + c = 0$
- d. Đổi từ độ sang radian và đổi từ radian sang độ
(công thức $\alpha/\pi = a/180$, với α : radian, a: độ)
- e. Kiểm tra 2 số a, b bằng nhau hay khác nhau.

Chương 2: LÀM QUEN LẬP TRÌNH C QUA CÁC VÍ DỤ ĐƠN GIẢN

2.1. Cài đặt và sử dụng Dev-C++

2.1.1. Giới thiệu về Dev-C++

Dev-C++ (hay còn gọi Dev-Cpp) là một bộ công cụ phát triển tích hợp (IDE Integrated Development Environment) các ứng dụng C/C++ thuộc dạng mã nguồn mở và có thể download (gõ download Dev-C++ vào google). DevCpp sử dụng cho cả hệ thống Windows và Linux. Hiện nay DevCpp là công cụ phát triển các ứng dụng C/C++ được sử dụng rộng rãi để dạy về lập trình cũng như để phát triển các ứng dụng mã nguồn mở.

2.1.2. Thao tác cơ bản với Dev-C++

Có 2 phiên bản của Dev-C++. Bản cài đặt (Setup) và bản dùng ngay không cần cài đặt (portable).

2.1.2.1. Cấu hình nhanh Dev-C++ (chỉ thực hiện lần đầu)

- Chạy file cài đặt DEV-C++ (file devcpp.exe).
- Vào menu "Tools" chọn "Compiler Options".
- Vào tab "Settings" tab, nhấn vào "Linker" ở khung bên trái và thay đổi thông số "Generate debugging information" (Hiển thị thông tin gỡ lỗi khi dịch chương trình) sang "Yes".
- Nhấn OK

2.1.2.2. Tạo chương trình mới hoặc dự án mới

Các bước sử dụng Dev-CPP để tạo ra chương trình C/C++ đơn giản như sau:

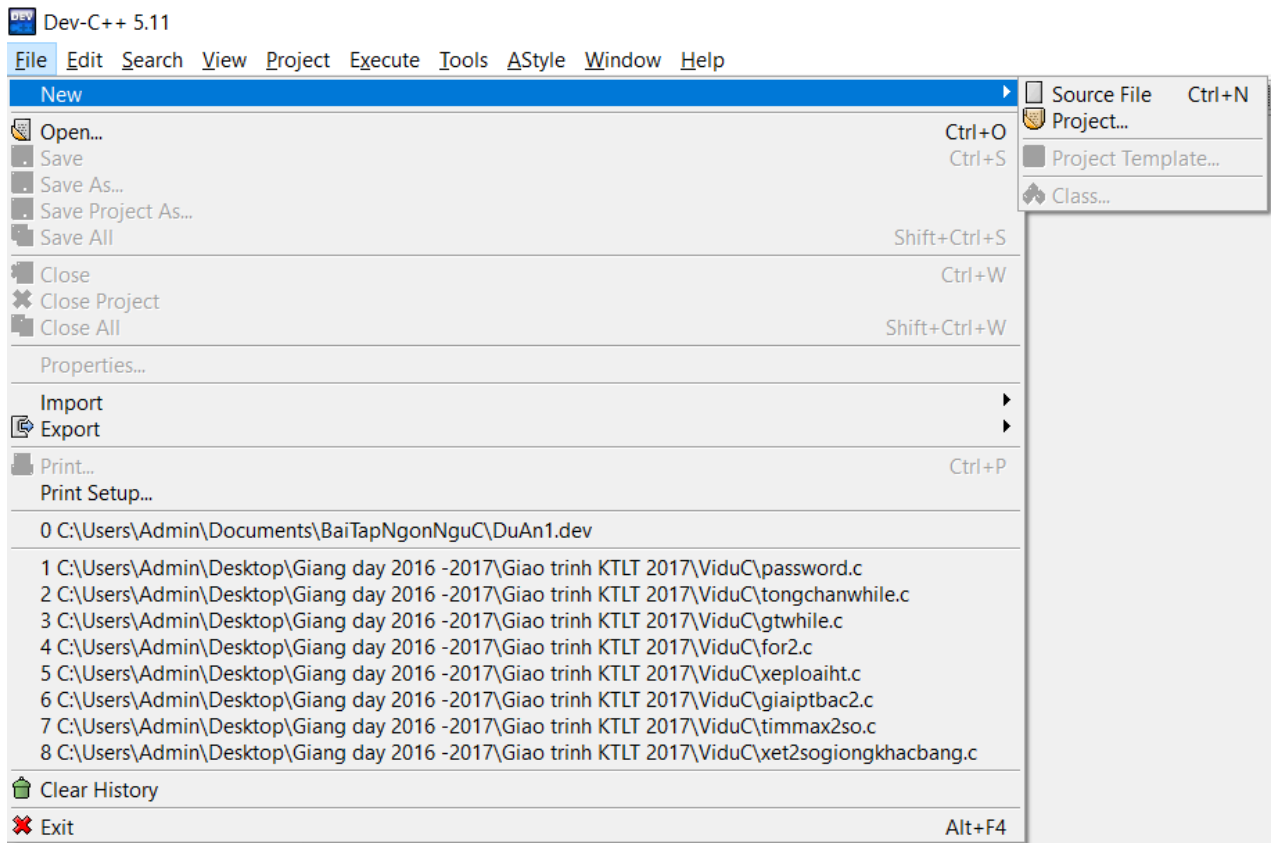
- Bước 1. Tạo file/project
- Bước 2. Soạn thảo mã nguồn
- Bước 3. Biên dịch, sửa lỗi (nếu có)
- Bước 4. Chạy thử và kiểm tra kết quả
- Bước 5. Debug (nếu cần)

Chi tiết các bước cụ thể như sau:

Bước 1: Tạo file hoặc project (dự án) mới

a. Trường hợp chương trình đơn giản:

Trong trường hợp chương trình đơn giản, chúng ta chỉ cần 1 file để lưu mã nguồn, khi đó chúng ta chọn menu File | New | Source File (Ctrl-N) và nhập mã nguồn trực tiếp vào file.



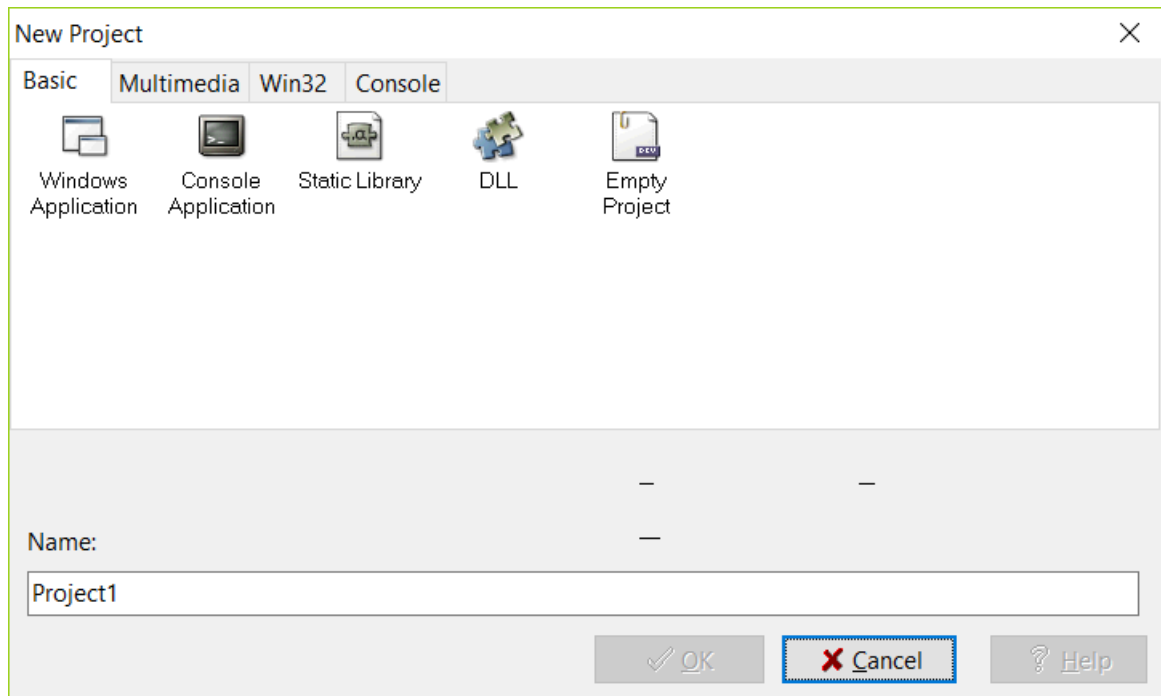
b. Trường hợp chương trình phức tạp:

Đối với chương trình C/C++ phức tạp, 1 chương trình cần nhiều file để thực hiện thì Dev-C++ có khái niệm dự án (project). Project là nơi cất giữ tất cả những thông tin liên quan đến mã nguồn, thư viện, tài nguyên,... để biên dịch thành chương trình.

Để tạo Project mới, vào menu "File" chọn "New", "Project...".

Chọn "Console Application" và chọn "C project" nếu là lập trình C, Chọn "C++ Project" nếu là C++. Nhập tên project, tên project cũng là tên của file thực thi (.exe) sẽ tạo ra khi biên dịch.

Nhấn "OK". Dev-C++ sẽ hỏi nơi lưu trữ project. Nhấn "OK".



Bước 2: Soạn thảo mã nguồn

- Nếu là bước 1.a thì chúng ta bắt tay ngay vào việc nhập mã nguồn chương trình
- Nếu là bước 1.b, chúng ta có 2 cách thêm mã nguồn vào dự án.

Cách 1: Vào menu "File" chọn "New Source File" (CTRL+N).

Cách 2: Vào menu "Project" chọn "New File".

Dev-C++ sẽ không hỏi tên file cho đến khi chúng ta thực hiện:

Compile

Save the project

Save the source file

Exit Dev-C++

- Các quy tắc soạn thảo gần giống với các chương trình soạn thảo văn bản (như Word...). Lưu ý sử dụng một vài tính năng hỗ trợ soạn thảo nhanh như: Undo, Redo, Copy, Paste, Cut, Select All... Tham khảo thêm thực đơn Edit để biết chi tiết và các phím tắt trong khi soạn thảo.

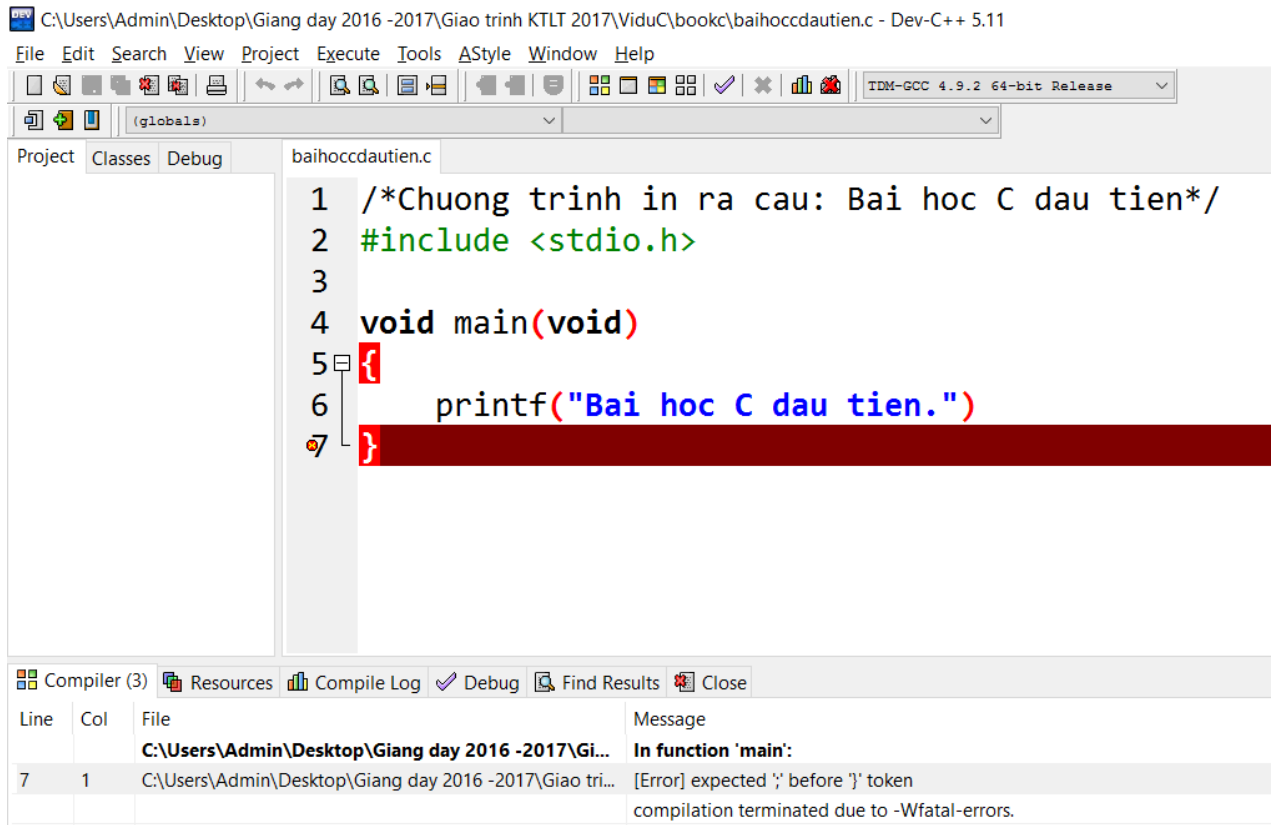
Bước 3: Biên dịch (Compile)

Sau khi nhập file nguồn xong, để biên dịch chúng ta thực hiện:

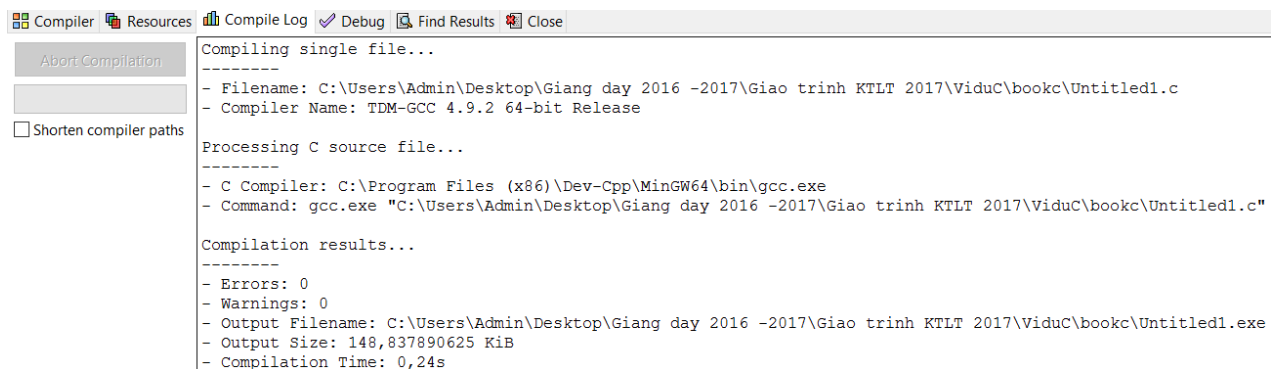
- Vào menu "Execute" chọn "Compile" (CTRL+F9).

Thông thường, khi biên dịch lần đầu sẽ dễ có lỗi về Syntax. Lỗi Syntax sẽ hiển thị ở tab biên dịch ở phía dưới màn hình. Để biết lỗi xảy ra ở vị trí nào của mã nguồn, double-click vào lỗi đó, lập tức con trỏ sẽ trở về vị trí có lỗi khi biên dịch.

Hình dưới minh họa lỗi syntax (thiếu dấu ;) khi biên dịch.



Trong trường hợp biên dịch thành công, mục Compiler log hiển thị thông báo:



Bước 4: Thực thi chương trình.

Sau khi sửa lỗi, chúng ta có thể chạy chương trình

- Vào menu "Execute" chọn "Run".

- Chú ý 1: Bước 4 & 5 có thể thực hiện đồng thời bằng menu Execute, chọn Compile & Run (phím tắt là F11).

- Chú ý 2: Màn hình kết quả không xuất hiện:

Nếu khi chạy chương trình, chúng ta thấy màn hình console DOS màu đen xuất hiện, chớp sáng 1 hoặc vài lần và trở lại màn hình soạn thảo thì tức là chương trình của chúng ta đã chạy xong.

Điều này có nghĩa là chương trình thực thi xong và tự đóng cửa sổ console sau khi chương trình thoát. Để giải quyết trường hợp này chúng ta có thể sử dụng lệnh “getch();” để tạm dừng màn hình, và có thể quan sát kết quả chương trình. Ta thêm đoạn code trên trước dấu “}” đóng hàm main, để lệnh đó sử dụng được ta thêm thư viện: #include <conio.h> ở đầu chương trình.

Chương trình hoàn thiện sẽ như sau:

```
Untitled1.c
1 /*Chương trình in ra câu: Bai hoc C dau tien*/
2 #include <stdio.h>
3 #include <conio.h>
4 void main(void)
5 {
6     printf("Bai hoc C dau tien.");
7     getch();
8 }
```

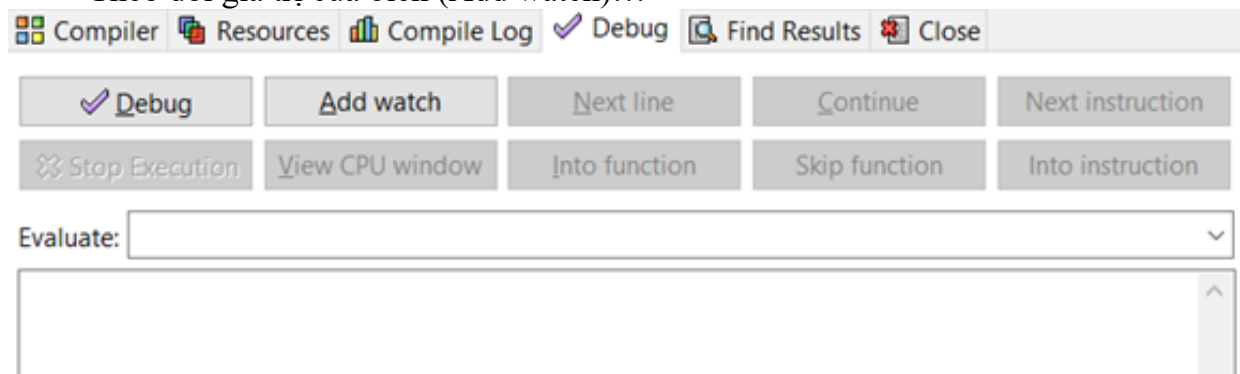
Bước 5: Debug.

Trong trường hợp chương trình chạy không như ý của chúng ta, sử dụng công cụ Debug để xác định lỗi cụ thể. Một vài tính năng chính của debug như:

Chuyển dòng kế tiếp (Next line)

Bỏ qua hàm (Skip function)

Theo dõi giá trị của biến (Add watch)...



2.2. Các ví dụ làm quen với C

2.2.1. Ví dụ 1

[*] Untitled1.c

```
1 /*Chương trình in ra câu: Bai hoc C dau tien*/
2 #include <stdio.h>
3
4 void main(void)
5 {
6     printf("Bai hoc C dau tien.");
7 }
```

Kết quả in ra màn hình

Bai hoc C dau tien. _

Dòng thứ 1: Bắt đầu bằng `/*` và kết thúc bằng `*/` cho biết dòng này là dòng diễn giải (chú thích). Khi dịch và chạy chương trình, dòng này không được dịch và cũng không thi hành lệnh gì cả. Mục đích của việc ghi chú này giúp chương trình rõ ràng hơn. Sau này chúng ta đọc lại chương trình biết chương trình làm gì.

Dòng thứ 2: Chứa phát biểu tiền xử lý `#include <stdio.h>`. Vì trong chương trình này ta sử dụng hàm thư viện của C là `printf`, do đó chúng ta cần phải có khai báo của hàm thư viện này để báo cho trình biên dịch C biết. Nếu không khai báo chương trình sẽ báo lỗi.

Dòng thứ 3: Dòng trống tạo ra với ý đồ làm cho bảng chương trình thoáng, dễ đọc.

Dòng thứ 4: `void main(void)` là thành phần chính của mọi chương trình C (chúng ta có thể viết `main()` hoặc `void main()` hoặc `main(void)`). Tuy nhiên, chúng ta nên viết theo dạng `void main(void)` để chương trình rõ ràng hơn. Mọi chương trình C đều bắt đầu thi hành từ hàm `main`. Cặp dấu ngoặc `()` cho biết đây là khối hàm (function). Hàm `void main(void)` có từ khóa `void` đầu tiên cho biết hàm này không trả về giá trị, từ khóa `void` trong ngoặc đơn cho biết hàm này không nhận vào đối số.

Dòng thứ 5 và 7: Cặp dấu ngoặc móc `{}` giới hạn thân của hàm. Thân hàm bắt đầu bằng dấu `{` và kết thúc bằng dấu `}`.

Dòng thứ 6: `printf("Bai hoc C dau tien.");`, chỉ thị cho máy in ra chuỗi ký tự nằm trong nháy kép (`"`). Dòng này được gọi là một câu lệnh, kết thúc một câu lệnh trong C phải là dấu chấm phẩy (`;`).

Chú ý:

- Các từ `include`, `stdio.h`, `void`, `main`, `printf` phải viết bằng chữ thường.
- Chuỗi trong nháy kép cần in ra "Chúng ta có thể viết chữ HOA, thường tùy, ý".

- Kết thúc câu lệnh phải có dấu chấm phẩy.
- Kết thúc tên hàm không có dấu chấm phẩy hoặc bất cứ dấu gì.
- Ghi chú phải đặt trong cặp /* */.
- Thân hàm phải được bao bởi cặp { }.
- Các câu lệnh trong thân hàm nên viết thụt vào và căn lề trái

Bây giờ chúng ta sửa lại dòng thứ 6 bằng câu lệnh `printf("Bai hoc C dau tien.\n");`
Sau đó dịch và chạy lại chương trình, quan sát kết quả.

Kết quả in ra màn hình

Bai hoc C dau tien.

Ở dòng chúng ta vừa sửa có thêm `\n`, `\n` là ký hiệu xuống dòng sử dụng trong lệnh `printf`. Sau đây là một số ký hiệu khác.

+ Các ký tự điều khiển:

`\n` : Nhảy xuống dòng kế tiếp canh về cột đầu tiên.

`\t` : Canh cột tab ngang.

`\r` : Nhảy về đầu hàng, không xuống hàng.

`\a` : Tiếng kêu bip.

+ Các ký tự đặc biệt:

`\\` : In ra dấu `\`

`\"` : In ra dấu `"`

`\'` : In ra dấu `'`

Bây giờ chúng ta sửa lại dòng thứ 6 bằng câu lệnh `printf("\tBai hoc C dau tien.\a\n");`, sau đó dịch và chạy lại chương trình, quan sát kết quả.

Kết quả in ra màn hình

Bai hoc C dau tien.

Khi chạy chương trình chúng ta nghe tiếng bip phát ra từ loa.

Để màn hình kết quả dừng lại chờ nhận một ký tự bất kỳ từ bàn phím sau đó mới thoát, ta sửa chương trình lại như sau:

```

Untitled1.c [*] Untitled1.c
1  /*Chương trình in ra cau: Bai hoc C dau tien*/
2  #include <stdio.h>
3  #include <conio.h>
4
5  void main(void)
6  {
7      printf("Bai hoc C dau tien.\a\n");
8      getch();
9  }

```

Dòng thứ 3: chứa phát biểu tiền xử lý `#include <conio.h>`. Vì trong chương trình này ta sử dụng hàm thư viện của C là ***getch***, do đó chúng ta cần phải có khai báo của hàm thư viện này để báo cho trình biên dịch C biết. **Nếu không khai báo chương trình sẽ báo lỗi.**

Dòng thứ 8: ***getch()***; chờ nhận 1 ký tự bất kỳ từ bàn phím, nhưng không in ra màn hình. Vì thế ta sử dụng hàm này để khi chạy chương trình xong sẽ dừng lại ở màn hình kết quả, sau đó ta ấn phím bất kỳ sẽ quay lại màn hình soạn thảo.

2.2.2. Ví dụ 2

```

Untitled1.c [*] Untitled1.c vidu2bai2.c
1  /*Chương trình nhập và in ra màn hình giá trị biến*/
2  #include <stdio.h>
3  #include <conio.h>
4
5  void main(void)
6  {
7      int i;
8      printf("Nhap vao mot so:");
9      scanf("%d",&i);
10     printf("So ban vua nhap la: %d.\n",i);
11     getch();
12 }

```

Kết quả in ra màn hình

```

Nhap vao mot so: 15
So ban vua nhap la: 15.
—

```

Dòng thứ 7: **int i;** là lệnh khai báo, mẫu tự i gọi là tên biến. Biến là một vị trí trong bộ nhớ dùng lưu trữ giá trị nào đó mà chương trình sẽ lấy để sử dụng. Mỗi biến phải thuộc một kiểu dữ liệu. Trong trường hợp này ta sử dụng biến i kiểu số nguyên (integer) viết tắt là int.

Dòng thứ 9: **scanf("%d", &i).** Sử dụng hàm scanf để nhận từ người sử dụng một trị nào đó. Hàm scanf trên có 2 đối mục. Đối mục **"%d"** được gọi là chuỗi định dạng, cho biết loại dữ kiện mà người sử dụng sẽ nhập vào. Chẳng hạn, ở đây phải nhập vào là số nguyên. Đối mục thứ 2 **&i** có dấu & đi đầu gọi là toán tử địa chỉ (address operator), dấu & phối hợp với tên biến cho hàm scanf biến đem trị gõ từ bàn phím lưu vào biến i.

Dòng thứ 10: **printf("So ban vua nhap la: %d.\n", i);**. Hàm này có 2 đối mục. Đối mục thứ nhất là một chuỗi định dạng có chứa chuỗi văn bản **So ban vua nhap la:** và **%d** (ký hiệu khai báo chuyển đổi dạng thức) cho biết số nguyên sẽ được in ra. Đối mục thứ 2 là i cho biết giá trị lấy từ biến i để in ra màn hình.

2.2.3. Ví dụ 3

```
Untitled1.c  [*] Untitled1.c  vidu2bai2.c  [*] vidu3bai2.c
1  /*Chương trình nhap vao 2 so va tinh tong 2 so vua nhap*/
2  #include <stdio.h>
3  #include <conio.h>
4
5  void main(void)
6  {
7      int a,b;
8      printf("Nhap vao so a:");
9      scanf("%d",&a);
10     printf("Nhap vao so b:");
11     scanf("%d",&b);
12     printf("Tong cua 2 so %d va %d la %d.\n",a,b,a+b);
13     getch();
14 }
```

Kết quả in ra màn hình

```
Nhap vao so a: 4
Nhap vao so b: 14
Tong cua 2 so 4 va 14 la 18.
—
```

Dòng thứ 12:

printf("Tong cua 2 so %d va %d la %d.\n", a, b, a+b);

The diagram illustrates the argument passing mechanism for the `printf` function. It shows three arguments: `a`, `b`, and `a+b`. Arrows indicate the flow of data from these arguments to the corresponding placeholders in the format string: `%d` for `a`, `%d` for `b`, and `%d` for `a+b`. The arrows show that the value of `a` is passed to the first `%d`, the value of `b` is passed to the second `%d`, and the value of `a+b` is passed to the third `%d`.

2.2.4. Ví dụ 4

```
Untitled1.c  [*] Untitled1.c  vidu2bai2.c  [*] vidu3bai2.c  vidu4bai2.c
1  /*Chương trình nhập bán kính hình tròn. Tính diện tích*/
2  #include <stdio.h>
3  #include <conio.h>
4
5  #define PI 3.14
6
7  void main(void)
8  {
9      float fR;
10     printf("Nhập vào bán kính hình tròn:");
11     scanf("%f",&fR);
12     printf("Diện tích hình tròn: %.2f.\n",2*PI*fR);
13     getch();
14 }
```

Kết quả in ra màn hình

```
Nhập vào bán kính hình tròn: 1
Diện tích hình tròn: 6.28
—
```

Dòng thứ 5: **#define PI 3.14**, dùng chỉ thị define để định nghĩa hằng số PI có giá trị 3.14. Trước define phải có dấu # và cuối dòng không có dấu chấm phẩy. Dòng thứ 12: **printf("Diện tích hình tròn: %.2f.\n", 2*PI*fR);**. Hàm này có 2 đối mục. Đối mục thứ nhất là một chuỗi định dạng có chứa chuỗi văn bản **Diện tích hình tròn:** và **%.2f** (ký hiệu khai báo chuyển đổi dạng thức) cho biết dạng số chấm động sẽ được in ra, trong đó **.2** nghĩa là in ra với 2 số lẻ. Đối mục thứ 2 là biểu thức hằng **2*PI*fR**;

Câu hỏi và bài tập chương 2

1. Thực hành các bước cài đặt Dev-C
2. Thực hành các thao tác sử dụng Dev-C cơ bản?
3. Ý nghĩa của việc dùng ghi chú trong lập trình?
4. Vì sao phải khai báo thư viện trong C.
5. Ý nghĩa của việc trình bày các dòng lệnh theo khối?
6. Ý nghĩa của các ký tự điều khiển phổ biến?
7. Ý nghĩa của lệnh getch?
8. Ý nghĩa của lệnh define?

Chương 3: CÁC THÀNH PHẦN TRONG NGÔN NGỮ C

3.1. Từ khóa

Từ khóa là từ có ý nghĩa xác định dùng để khai báo dữ liệu, viết câu lệnh... Trong C có các từ khóa phổ biến sau:

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

Lưu ý các từ khóa phải viết bằng *chữ thường*

3.2. Tên

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

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

Ví dụ 1:

Các tên đúng: delta, a_1, Num_ODD, Case

Các tên sai:

3a_1 (ký tự đầu là số)

num-odd (sử dụng dấu gạch ngang)

int (đặt tên trùng với từ khóa)

del ta (có khoảng trắng)

f(x) (có dấu ngoặc tròn)

Lưu ý: Trong C, tên phân biệt chữ hoa, chữ thường

Ví dụ 2:

number khác Number

case khác Case (case là từ khóa, do đó chúng ta đặt tên là Case vẫn đúng)

3.3. Kiểu dữ liệu

Có 4 kiểu dữ liệu cơ bản trong C là: char, int, float, double. Tuy nhiên từ 4 kiểu dữ liệu cơ bản, C có cung cấp một số kiểu mở rộng khác chi tiết trong bảng.

TT	Kiểu dữ liệu (Type)	Kích thước (Length)	Miền giá trị (Range)
1	unsigned char	1 byte	0 đến 255
2	char	1 byte	– 128 đến 127
3	enum	2 bytes	– 32,768 đến 32,767
4	unsigned int	2 bytes	0 đến 65,535
5	short int	2 bytes	– 32,768 đến 32,767
6	int	2 bytes	– 32,768 đến 32,767
7	unsigned long	4 bytes	0 đến 4,294,967,295
8	long	4 bytes	– 2,147,483,648 đến 2,147,483,647
9	float	4 bytes	$3.4 * 10^{-38}$ đến $3.4 * 10^{38}$
10	double	8 bytes	$1.7 * 10^{-308}$ đến $1.7 * 10^{308}$
11	long double	10 bytes	$3.4 * 10^{-4932}$ đến $1.1 * 10^{4932}$

3.4. Ghi chú

Trong khi lập trình cần phải ghi chú để giải thích các biến, hằng, thao tác xử lý giúp cho chương trình rõ ràng dễ hiểu, dễ nhớ, dễ sửa chữa và để người khác xem dễ hiểu.

Trong C có các ghi chú sau: // Nội dung ghi chú (trên một dòng)

hoặc /* nội dung ghi chú (trên nhiều dòng)*/

Ví dụ:

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

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

Tóm lại, đối với ghi chú dạng // dùng để ghi chú một dòng và dạng /* */ có thể ghi chú một dòng hoặc nhiều dòng.

3.5. Biến

3.5.1. Tên biến

Tên biến hợp lệ là một chuỗi ký tự liên tục gồm: **Ký tự chữ, số và dấu gạch dưới**. Ký tự đầu của tên phải là **chữ hoặc dấu gạch dưới**. Khi đặt tên không được đặt trùng với các từ khóa (cách đặt tên biến tuân theo quy tắc như mục **Tên** ở trên).

3.5.2. Khai báo biến

Cú pháp

Kiểu dữ liệu *Danh sách tên biến*;

- Kiểu dữ liệu: Là một trong các kiểu ở mục **Kiểu dữ liệu**
- Danh sách tên biến: gồm các tên biến có cùng kiểu dữ liệu, mỗi tên biến cách nhau dấu phẩy (,), cuối cùng là dấu chấm phẩy (;).

Lưu ý: Có nhiều cách đặt tên biến

3.5.3. Vừa khai báo vừa khởi gán

Có thể kết hợp việc khai báo với toán tử gán để biến nhận ngay giá trị cùng lúc với khai báo.

Ví dụ:

Khai báo trước, gán giá trị sau:

```
void main()
```

```
{
```

```
    int a, b, c;
```

```
    a = 1;
```

```
    b = 2;
```

```
    c = 5;
```

```
    ...
```

```
}
```

Vừa khai báo vừa gán giá trị:

```
void main()
```

```
{
```

```
    int a = 1, b = 2, c = 5;
```

```
    ...
```

```
}
```

3.6. Lệnh và khối lệnh

3.6.1. Lệnh

Là một tác vụ, biểu thức hoặc công thức tính toán...

Ví dụ:

```
x = x + 2;  
printf("Day la mot lenh\n");
```

3.6.2. Khối lệnh

Là một dãy các câu lệnh được bọc bởi cặp dấu { }, các lệnh trong khối lệnh phải viết thụt vào một tab so với cặp dấu { }

Ví dụ:

```
{//dau khoi  
    //viet thut vao 1 tab so voi cap {}  
    a=5;  
    b=6;  
    printf("Tong %d+%d=%d",a,b,a+b);  
}//cuoi khoi
```

Lưu ý đừng quên dùng cặp dấu { } bao bọc khi sử dụng khối lệnh, hoặc mở dấu { và quên đóng dấu }

3.7. Thư viện trong C

Thư viện trong C là một tập hợp các hàm, hằng và header file như <stdio.h>, <stdlib.h>, <math.h>, <string.h> ... đã được xây dựng sẵn. Để sử dụng các hàm, hằng hay các macro đã được xây dựng sẵn trong C, chúng ta cần khai báo các header file này ở phần đầu chương trình.

3.7.1. Khai báo thư viện

Cách 1: #include <tên_thư_viện.h>

Cách 2: #include "tên_thư_viện.h"

#include có tác dụng gọi thư viện (chèn nội dung file được khai báo vào nội dung file đang dịch ở vị trí xuất hiện của nó)

3.7.2. Một số thư viện chuẩn trong C

stdio.h: Thư viện chứa các hàm vào/ ra chuẩn (standard input/output). Gồm các hàm printf(), scanf(), fopen(), fclose(), fread(), fwrite(),...

math.h: Thư viện chứa các hàm tính toán gồm các hàm abs(), sqrt(), log(), log10(), sin(), cos(), tan(), acos(), asin(), atan(), pow()...

conio.h : Thư viện chứa các hàm vào ra trong chế độ DOS (DOS console). Gồm các hàm clrscr(), getch(), ...

alloc.h: Thư viện chứa các hàm liên quan đến việc quản lý bộ nhớ. Gồm các hàm calloc(), realloc(), malloc(), free(), farmalloc(), farcalloc(), farfree(),...

graphics.h: Thư viện chứa các hàm liên quan đến đồ họa. Gồm initgraph(), line(), circle(), putpixel(), getpixel(), setcolor(),...

stdint.h : Dùng trong việc định nghĩa các kiểu nguyên khác nhau.

Ngoài ra còn nhiều thư viện khác như io.h, stdlib.h, ...

Câu hỏi chương 3

1. Từ khóa trong lập trình là gì? Lưu ý khi sử dụng từ khóa trong lập trình C?
2. Thế nào là một tên hợp lệ trong lập trình C?
3. Trong C có những kiểu dữ liệu gì? Kích thước của các kiểu dữ liệu đó?
4. Có mấy cách ghi chú trong lập trình C? Mô tả cách sử dụng và lấy ví dụ minh họa?
5. Cú pháp khai báo biến trong C? Mô tả quy tắc Hungarian Notation?
6. Cách thức khai báo thư viện trong C? Trình bày một số thư viện chuẩn trong C?

Chương 4: HÀM NHẬP - XUẤT DỮ LIỆU

4.1. Hàm printf

Công dụng: Hàm printf dùng để in kết quả ra các thiết bị xuất (hay kết xuất dữ liệu được định dạng).

Cú pháp

printf("chuỗi định dạng", đối mục 1, đối mục 2, ...);

Lưu ý: Khi sử dụng hàm phải khai báo tiền thư viện **#include <stdio.h>**

- printf: tên hàm, **phải viết bằng chữ thường**.

- đối mục 1, đối mục 2...: là các mục dữ kiện cần in ra màn hình. Các đối mục này có thể là biến, hằng hoặc biểu thức phải được định trị trước khi in ra.

- chuỗi định dạng: được đặt trong cặp nháy kép (" "), gồm 3 loại:

+ Đối với chuỗi kí tự ghi như thế nào in ra giống như vậy.

+ Đối với những kí tự chuyển đổi dạng thức cho phép kết xuất giá trị của các đối mục ra màn hình tạm gọi là mã định dạng. Sau đây là các dấu mô tả định dạng:

%c : Kí tự đơn

%s : Chuỗi

%d : Số nguyên hệ thập phân có dấu

%f : Số chấm động (ký hiệu thập phân)

%e : Số chấm động (ký hiệu có số mũ)

%g : Số chấm động (%f hay %g)

%x : Số nguyên hex không dấu

%o : Số nguyên bát phân không dấu

l : Tiền tố dùng kèm với %d, %u, %x, %o để chỉ số nguyên dài (ví dụ %ld)

+ Các ký tự điều khiển và ký tự đặc biệt

\n : Nhảy xuống dòng kế tiếp canh về cột đầu tiên.

\t : Canh cột tab ngang.

**** : In ra dấu \

\" : In ra dấu "

\' : In ra dấu '

%% : In ra dấu %

Ví dụ 1:

```
printf("Bai hoc C dau tien. \n");
```

└──────────┬──────────┘
 | |
 | └─> ký tự điều khiển
 └──────────> chuỗi ký tự

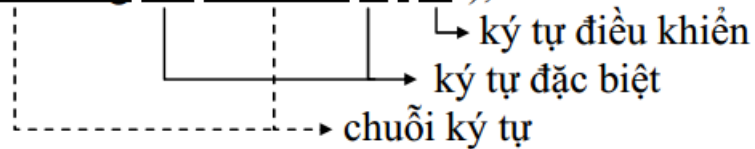
Kết quả in ra màn hình

Bài học C đầu tiên.

—

Ví dụ 2:

```
printf("Ma dinh dang \\" in ra dau \" . \n");
```



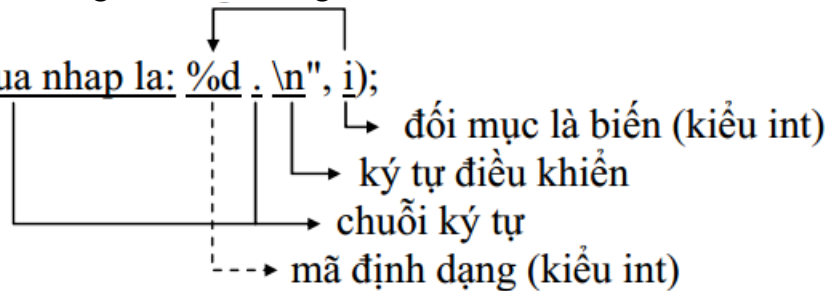
Kết quả in ra màn hình

Ma dinh dang \" in ra dau \".

—

Ví dụ 3: Giả sử biến i có giá trị = 5, xuất giá trị biến i

```
printf("So ban vua nhap la: %d . \n", i);
```



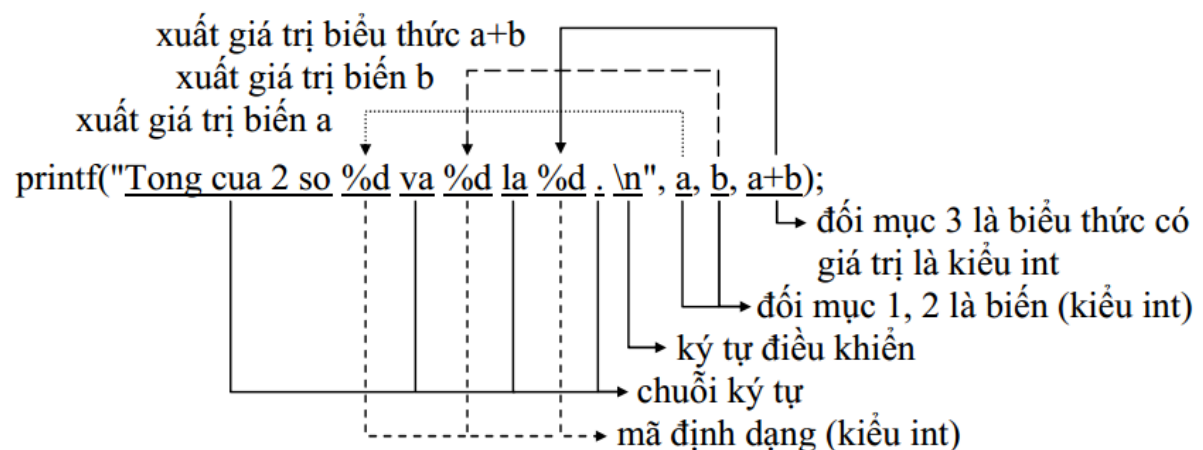
Kết quả in ra màn hình

So ban vua nhap la: 5.

—

Ví dụ 4: Giả sử biến a có giá trị = 7 và b có giá trị = 4

```
printf("Tong cua 2 so %d va %d la %d . \n", a, b, a+b);
```



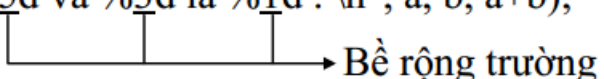
Kết quả in ra màn hình

Tong cua 2 so 7 va 4 la 11.

—

Ví dụ 5: Sửa lại ví dụ 4

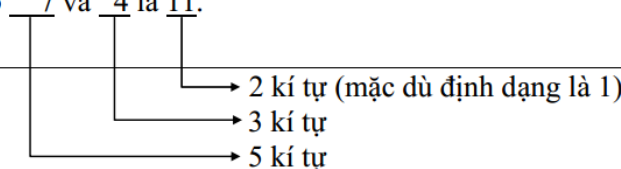
```
printf("Tong cua 2 so %5d va %3d la %1d . \n", a, b, a+b);
```



Bề rộng trường

Kết quả in ra màn hình

Tong cua 2 so 7 va 4 la 11.



—

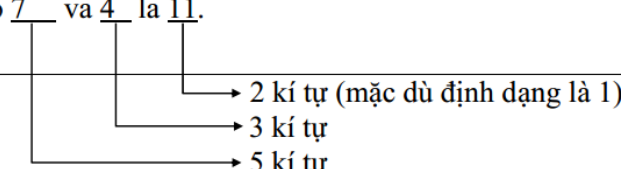
2 kí tự (mặc dù định dạng là 1)
3 kí tự
5 kí tự

Ví dụ 6: Sửa lại ví dụ 5

```
printf("Tong cua 2 so %-5d va %-3d la %-1d . \n", a, b, a+b);
```

Kết quả in ra màn hình

Tong cua 2 so 7 va 4 la 11.



—

2 kí tự (mặc dù định dạng là 1)
3 kí tự
5 kí tự

Dấu trừ trước bề rộng trường sẽ kéo kết quả sang trái

Ví dụ 7: Sửa lại ví dụ 4

```
printf("Tong cua 2 so %02d va %02d la %04d . \n", a, b, a+b);
```

Kết quả in ra màn hình

Tong cua 2 so <u>07</u> va <u>04</u> la <u>0011</u> .	
—	
	thêm 2 số 0 trước -> đủ 4 kí tự
	thêm 1 số 0 trước -> đủ 2 kí tự
	thêm 1 số 0 trước -> đủ 2 kí tự

Ví dụ 8:

Giả sử $int\ a = 6, b = 1234, c = 62$

```
printf("%7d%7d%7d.\n", a, b, c);
```

```
printf("%7d%7d%7d.\n", 165, 2, 965);
```

Kết quả in ra màn hình

6 1234 62 165 2 965	Số canh về bên phải bề rộng trường.
—	

Nếu sửa lại:

```
printf("%-7d%-7d%-7d.\n", a, b, c);
```

```
printf("%-7d%-7d%-7d.\n", 165, 2, 965);
```

Kết quả in ra màn hình

6 1234 62 165 2 965	Số canh về bên trái bề rộng trường.
—	

Ví dụ 9: Giả sử $float\ a = 6.4, b = 1234.56, c = 62.3$

```
printf("%7.2d%7.2d%7.2d.\n", a, b, c);
```

số số lẻ

Kết quả in ra màn hình

6.40 1234.56 62.30	Số canh về bên phải bề rộng trường.
—	
	7 kí tự

Bề rộng trường bao gồm: Phần nguyên, phần lẻ và dấu chấm động

Ví dụ 10:

Giả sử $float\ a = 6.4, b = 1234.55, c = 62.34$

```
printf("%10.1f%10.1f%10.1f.\n", a, b, c);
```

```
printf("%10.1f%10.1f%10.1f\n", 165, 2, 965);
```

Kết quả in ra màn hình

6.4 1234.6 62.3 165.0 2.0 965.0 —	Số canh về bên phải bề rộng trường.
---	-------------------------------------

```
printf("%-10.2f%-10.2f%-10.2f\n", a, b, c);
```

```
printf("%-10.2d%-10.2d%-10.2d\n", 165, 2, 965);
```

Kết quả in ra màn hình

6.40 1234.55 62.34 165.00 2.00 965.00 —	Số canh về bên trái bề rộng trường.
---	-------------------------------------

4.2. Hàm scanf

Định dạng khi nhập liệu.

Cú pháp

```
scanf ("chuỗi định dạng" [, đối mục 1, đối mục 2,...]);
```

Lưu ý: Khi sử dụng hàm phải khai báo thư viện **#include <stdio.h>**

- scanf: Tên hàm **phải viết bằng chữ thường**.

- chuỗi định dạng (hay mã định dạng): Được đặt trong cặp nháy kép (" ") là điều khiển của dạng dữ liệu nhập vào.

- đối mục 1,...: là danh sách các đối mục cách nhau bởi dấu phẩy, mỗi đối mục sẽ tiếp nhận giá trị nhập vào.

Ví dụ 1:

```
scanf("%d", &i);
```

 ↓ ↓
 đổi mục 1 mã định dạng

Nhập vào 12abc, biến i chỉ nhận giá trị 12. Nhập 3.4 chỉ nhận giá trị 3.

Ví dụ 2: scanf("%d%d", &a, &b);

Nhập vào 2 số a, b phải cách nhau bằng **khoảng trắng** hoặc **enter**.

Ví dụ 3: scanf("%d/%d/%d", &ngay, &thang, &nam);

Nhập vào ngày, tháng, năm theo dạng ngày/thang/nam (20/12/2002)

Ví dụ 4: scanf("%d%*c%d%*c%d", &ngay, &thang, &nam);

Nhập vào ngày, tháng, năm với dấu phân cách /, -, ...; ngoại trừ số.

Ví dụ 5: scanf("%2d%2d%4d", &ngay, &thang, &nam);

Nhập vào ngày, tháng, năm theo dạng dd/mm/yyyy

Bài tập Chương 4

1. Viết chương trình đổi một số nguyên hệ 10 sang hệ 16.
2. Viết chương trình đọc và 2 số nguyên và in ra kết quả của phép (+), phép trừ (-), phép nhân (*), phép chia (/). Nhận xét kết quả chia 2 số nguyên.
3. Viết chương trình nhập vào bán kính sau đó in ra diện tích, thể tích của hình cầu đó.

Hướng dẫn: $S = 4\pi R^2$ và $V = (4/3)\pi R^3$.

4. Viết chương trình nhập vào một số a bất kỳ và in ra giá trị bình phương (a^2), lập phương (a^3) của a và giá trị a^4 .
5. Viết chương trình đọc từ bàn phím 3 số nguyên biểu diễn ngày, tháng, năm và xuất ra màn hình dưới dạng "ngay/thang/nam" (chỉ lấy 2 số cuối của năm).
6. Viết chương trình nhập vào số giây từ 0 đến 86399, đổi số giây nhập vào thành dạng "gio:phut:giay", mỗi thành phần là một số nguyên có 2 chữ số.
Ví dụ: 02:11:05

Chương 5: CẤU TRÚC Rẽ NHÁNH CÓ ĐIỀU KIỆN

5.1. Lệnh if

Câu lệnh if cho phép lựa chọn một trong hai nhánh tùy thuộc vào giá trị của biểu thức logic là đúng (true) hay sai (false) hoặc khác không hay bằng không.

5.1.1. Dạng 1 (if thiếu)

Công dụng: Đưa ra quyết định sẽ thực hiện hay không thực hiện một khối lệnh.

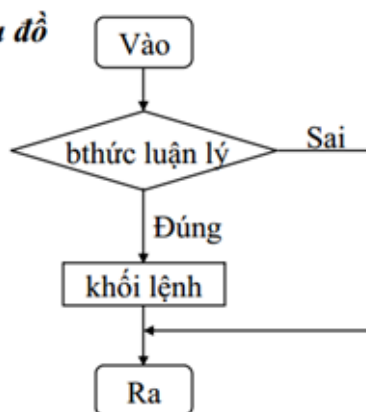
- **Cú pháp lệnh**

if (biểu thức logic)

khối lệnh;

Lưu ý: Từ khóa **if** phải viết bằng chữ thường, kết quả của **biểu thức logic** phải là **đúng ($\neq 0$)** hoặc **sai ($= 0$)**

- **Lưu đồ**



Giải thích lưu đồ: Nếu biểu thức logic đúng thì thực hiện khối lệnh và thoát khỏi if, ngược lại không làm gì cả và thoát khỏi if.

Lưu ý:

Nếu **khối lệnh** bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

+ Nếu khối lệnh là một lệnh ta viết lệnh if như sau: if (biểu thức logic) lệnh;

+ Khối lệnh bao gồm nhiều lệnh: lệnh 1, lệnh 2..., ta viết lệnh if như sau:

if (biểu thức logic)

{

lệnh 1;

lệnh 2;

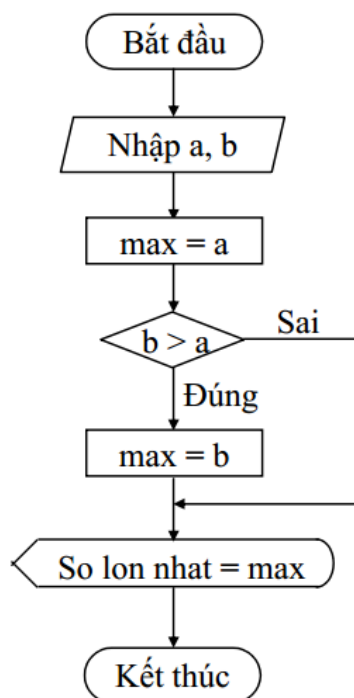
...

}

+ **Không đặt dấu chấm phẩy sau câu lệnh if.**

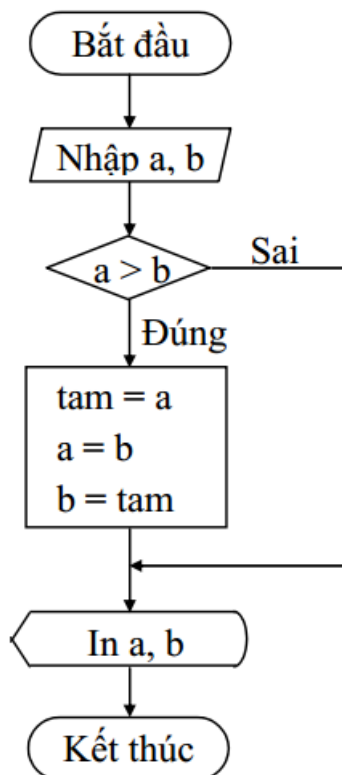
Ví dụ 1: if(biểu thức logic);

Ví dụ 2: Viết chương trình nhập vào 2 số nguyên a, b. Tìm và in ra số lớn nhất.
Mô tả bằng lưu đồ



Ví dụ 3: Viết chương trình nhập vào 2 số nguyên a, b. Nếu a lớn hơn b thì hoán đổi giá trị a và b, ngược lại không hoán đổi. In ra giá trị a, b.

Mô tả bằng lưu đồ



5.1.2. Dạng 2 (if đủ)

Công dụng: Đưa ra quyết định sẽ thực hiện 1 trong 2 khối lệnh cho trước.

Cú pháp lệnh:

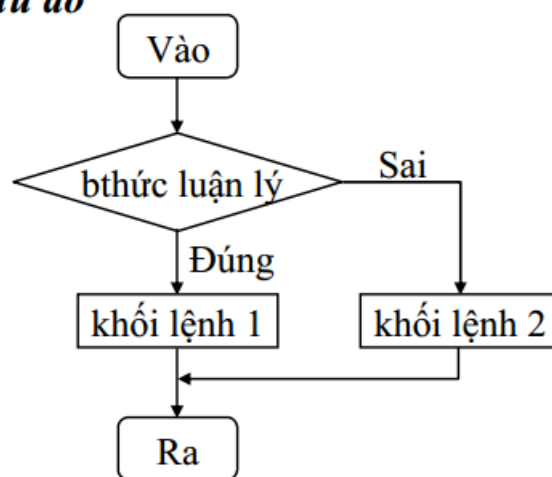
if (biểu thức logic)

 khối lệnh 1;

else

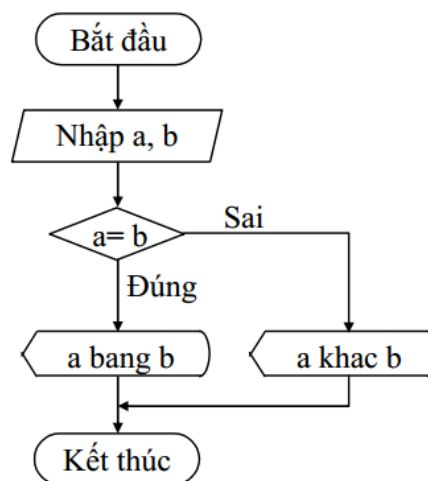
 khối lệnh 2;

Lưu đồ



Ví dụ 1: Viết chương trình nhập vào 2 số nguyên a, b. In ra thông báo "a bằng b" nếu $a = b$, ngược lại in ra thông báo "a khác b".

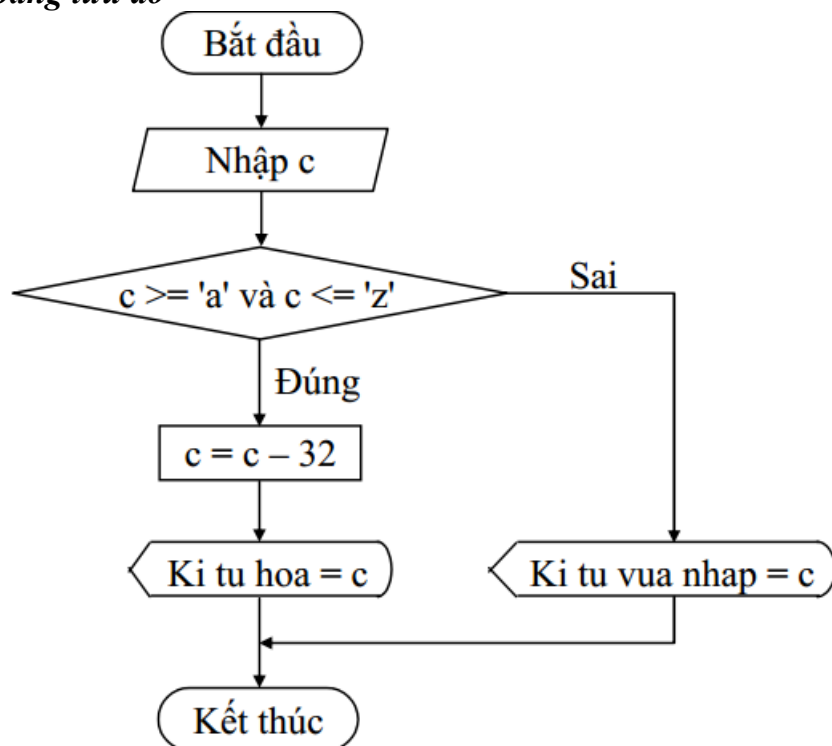
Mô tả bằng lưu đồ



Kết quả in ra màn hình

Ví dụ 3: Viết chương trình nhập vào kí tự c. Kiểm tra xem nếu kí tự nhập vào là kí tự thường trong khoảng từ 'a' đến 'z' thì đổi sang chữ in hoa và in ra, ngược lại in ra thông báo "Kí tự chúng ta vừa nhập là: c".

Mô tả bằng lưu đồ



5.1.3. Cấu trúc else if

Công dụng: Đưa ra quyết định sẽ thực hiện 1 trong n khối lệnh cho trước.

- **Cú pháp lệnh**

if (biểu thức logic 1)

khối lệnh 1;

else if (biểu thức logic 2)

khối lệnh 2;

...

else if (biểu thức logic n-1)

khối lệnh n-1;

else

khối lệnh n;

Giải thích cách thực hiện dựa trên cú pháp lệnh:

Nếu **biểu thức logic 1** đúng thì thực hiện khối lệnh 1 và thoát khỏi cấu trúc if

Ngược lại nếu **biểu thức logic 2** đúng thì thực hiện khối lệnh 2 và thoát khỏi cấu trúc if

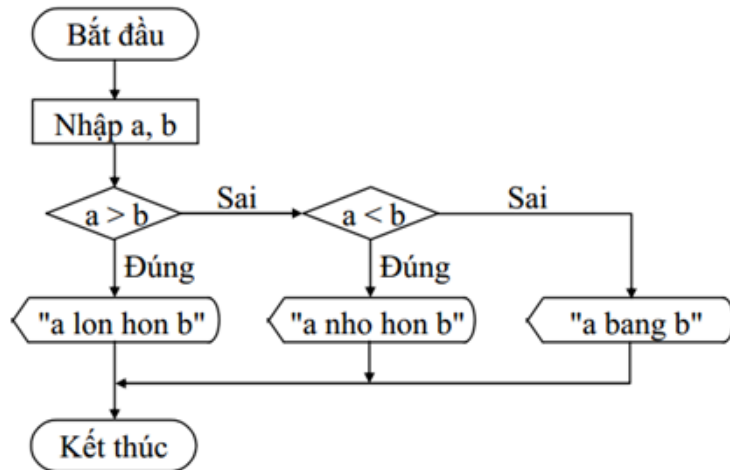
...

Ngược lại nếu **biểu thức logic n-1** đúng thì thực hiện khối lệnh n-1 và thoát khỏi cấu trúc if

Ngược lại thì thực hiện khối lệnh n.

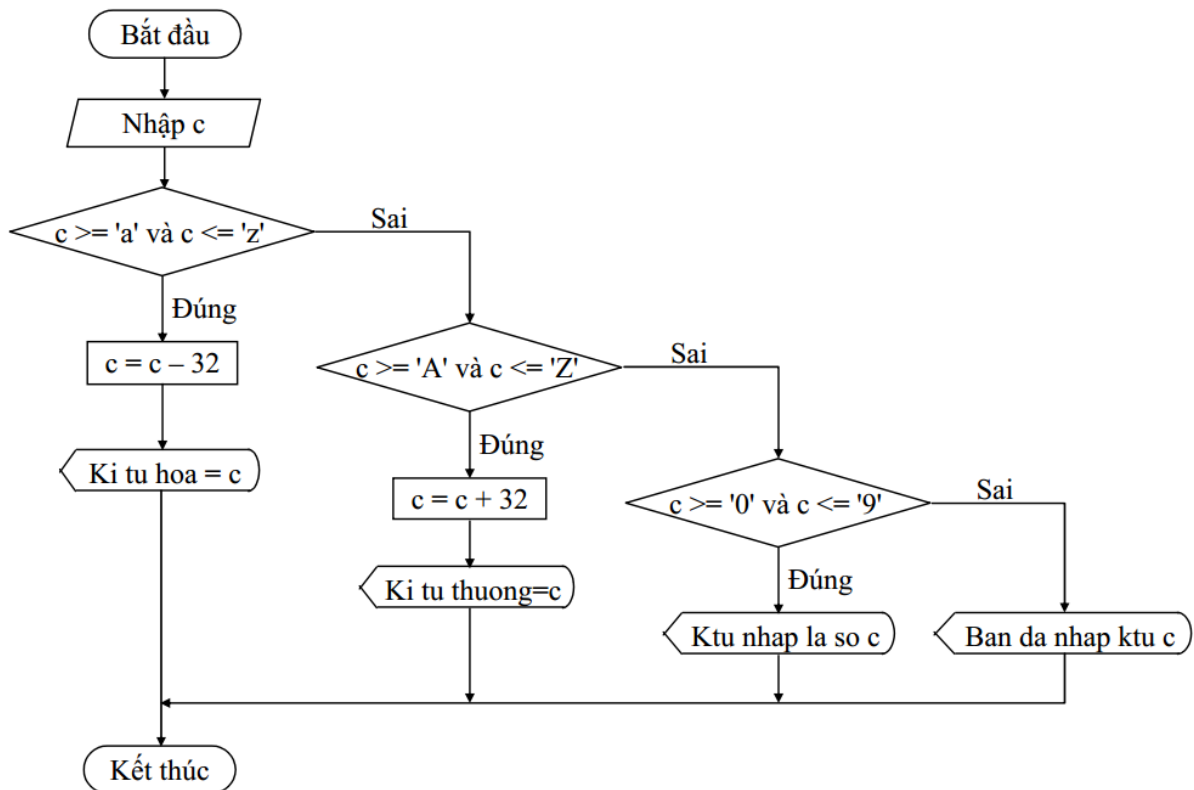
Ví dụ 1: Viết chương trình nhập vào 2 số nguyên a, b. In ra thông báo "a lớn hơn b" nếu $a > b$, in ra thông báo "a nhỏ hơn b" nếu $a < b$, in ra thông báo "a bằng b" nếu $a = b$.

Mô tả bằng lưu đồ



Ví dụ 2: Viết chương trình nhập vào kí tự c. Kiểm tra xem nếu kí tự nhập vào là kí tự thường trong khoảng từ 'a' đến 'z' thì đổi sang chữ in hoa và in ra, nếu kí tự in hoa trong khoảng A đến Z thì đổi sang chữ thường và in ra, nếu kí tự là số từ 0 đến 9 thì in ra câu "Kí tự chúng ta vừa nhập là số ...(in ra kí tự c)", còn lại không phải 3 trường hợp trên in ra thông báo "Chúng ta đã nhập kí tự ...(in ra kí tự c)".

Mô tả bằng lưu đồ



5.1.4. Cấu trúc if lồng nhau

Công dụng: Đưa ra quyết định sẽ thực hiện 1 trong n khối lệnh cho trước.

• Cú pháp lệnh

Là một trong 3 dạng trên, nhưng trong 1 hoặc nhiều khối lệnh bên trong phải chứa ít nhất một trong 3 dạng trên gọi là cấu trúc if lồng nhau. Thường cấu trúc if lồng nhau càng nhiều cấp thì độ phức tạp càng cao, chương trình chạy càng chậm và trong lúc lập trình dễ bị nhầm lẫn.

Lưu ý: Các lệnh **if...else** lồng nhau thì **else** sẽ luôn luôn kết hợp với **if** nào chưa có **else** gần nhất. Vì vậy khi gặp những lệnh if không có **else**, Chúng ta phải đặt chúng trong những **khối lệnh rõ ràng** để tránh bị hiểu sai câu lệnh.

Ví dụ 1: Viết chương trình nhập vào giá vé và tuổi sau đó tính tiền được miễn giảm nếu: Tuổi ≤ 15 tuổi thì miễn giảm như sau:

Dưới 6 tuổi miễn giảm 50% giá vé

Từ 6 đến 10 tuổi miễn giảm 30% giá vé

Từ 10 đến 15 tuổi miễn giảm 15% giá vé

Trên 15 tuổi không được miễn giảm giá vé

Ví dụ 2: Viết chương trình nhập vào điểm của một học sinh. In ra xếp loại học tập của học sinh đó. (Cách xếp loại. Nếu điểm ≥ 9 , Xuất sắc. Nếu điểm từ 8 đến cận 9, Giỏi.

Nếu điểm từ 7 đến cận 8, Khá. Nếu điểm từ 6 đến cận 7, TBKhá. Nếu điểm từ 5 đến cận 6, TBình. Còn lại là Yếu). Kiểm tra tính hợp lệ của điểm (điểm ≥ 0 và điểm ≤ 10)

Ví dụ 3: Viết chương trình nhập vào 3 số nguyên a, b, c. Tìm và in ra số lớn nhất.

5.2. Lệnh switch

Lệnh switch cũng có công dụng giống như else if, nhưng đôi khi nó mềm dẻo và linh động hơn so với sử dụng if. Tuy nhiên, nó cũng có mặt hạn chế là kết quả của biểu thức phải là giá trị hằng nguyên (có giá trị cụ thể). Một bài toán sử dụng lệnh switch thì cũng có thể sử dụng if, nhưng ngược lại còn tùy thuộc vào giải thuật của bài toán.

5.2.1. Cấu trúc switch...case (switch thiếu)

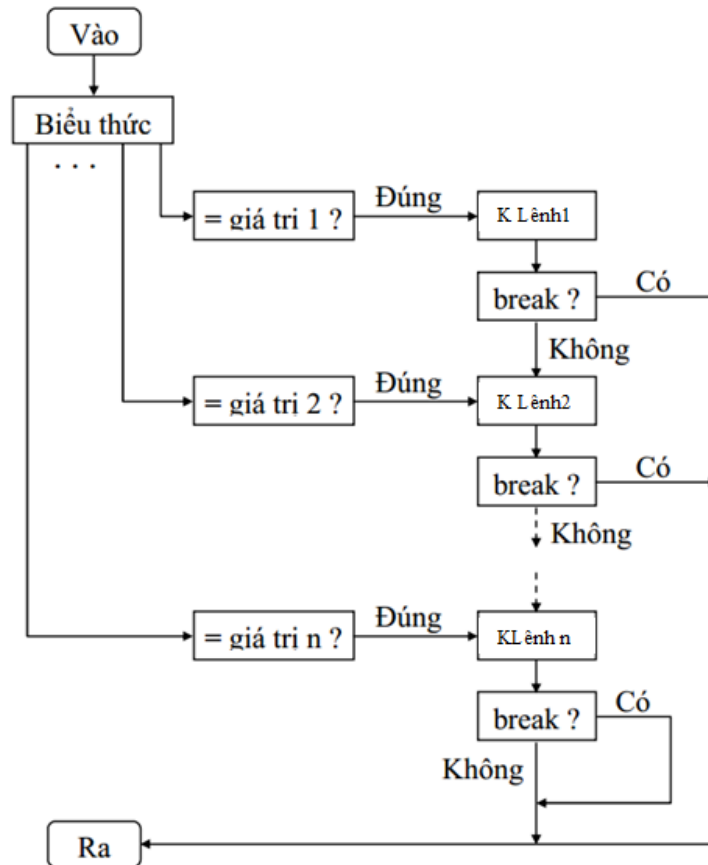
Công dụng: Lựa chọn thực hiện 1 trong n lệnh cho trước.

switch (biểu thức)

```
{  
    case giá trị 1 : khối lệnh 1;  
                  break;  
    case giá trị 2 : khối lệnh 2;  
                  break;  
    ...  
    case giá trị n : khối lệnh n;  
                  [break;]  
}
```

• Lưu đồ

Khi giá trị của biểu thức bằng giá trị i thì lệnh i sẽ được thực hiện. Nếu sau lệnh i không có lệnh break thì sẽ tiếp tục thực hiện lệnh i + 1...Nếu có break thì sẽ thoát khỏi cấu trúc switch.



Ví dụ 1: Viết chương trình nhập vào số 1, 2, 3. In ra tương ứng 1, 2, 3 dấu sao.

b. Lưu ý thêm về chương trình

Không đặt dấu chấm phẩy sau câu lệnh switch.

Ví dụ: Nếu sử dụng `switch(i);`

→ trình biên dịch không báo lỗi nhưng các lệnh trong switch không được thực hiện.

Ví dụ 2: Viết chương trình nhập vào thứ là số và in ra thứ là chữ

Ví dụ 3: Viết chương trình nhập vào tháng và in ra quý. (tháng 1 -> quý 1, tháng 10

-> quý 4)

5.2.2. Cấu trúc switch...case...default (switch đủ)

Công dụng: Lựa chọn thực hiện 1 trong $n + 1$ lệnh cho trước.

• Cú pháp lệnh

switch (biểu thức)

{

case giá trị 1 : lệnh 1;

break;

case giá trị 2 : lệnh 2;

break;

...

```

        case giá trị n    : lệnh n;
                        [break;]
        default          : lệnh;
                        [break;]
    }

```

Lưu ý:

Ví dụ 1: Viết lại chương trình nhập vào số 1, 2, 3. In ra tương ứng 1, 2, 3 dấu sao.

Ví dụ 2: Viết lại chương trình nhập vào tháng và in ra quý. (tháng 1 -> quý 1, tháng 10 -> quý 4). Thông báo cho người dùng biết nếu tháng nhập vào không ≥ 1 và ≤ 12 .

5.2.3. Cấu trúc switch lồng

Công dụng: Lựa chọn thực hiện 1 trong n khối lệnh cho trước.

• **Cú pháp lệnh**

Cú pháp là một trong 2 dạng trên, nhưng trong 1 hoặc nhiều lệnh bên trong phải chứa ít nhất một trong 2 dạng trên gọi là cấu trúc switch lồng nhau. Thường cấu trúc switch lồng nhau càng nhiều cấp độ phức tạp càng cao, chương trình chạy càng chậm và trong lúc lập trình dễ bị nhầm lẫn.

Ví dụ 1: Viết chương trình xây dựng menu 2 cấp

```

/* Chương trình menu 2 cấp */
#include <stdio.h>
#include <conio.h>
void main(void)
{
    int imenu, isubmenu;
    printf("-----\n");
    printf(" MAIN MENU \n");
    printf("-----\n");
    printf("1. File\n");
    printf("2. Edit\n");
    printf("3. Search\n");
    printf("Chon muc tuong ung: ");
    scanf("%d", &imenu);
}

```

```

switch(imenu)
{
    case 1: printf("-----\n");
            printf(" MENU FILE \n");
            printf("-----\n");
            printf("1. New\n");
            printf("2. Open\n");
            printf("Chon muc tuong ung: ");
            scanf("%d", &isubmenu);
            switch(isubmenu)
            {
                case 1: printf("Ban da chon chuc nang New File\n");
                        break;
                case 2: printf("Ban da chon chuc nang Open File\n");
                        break;
            }
            break; //break cua case 1 – switch(imenu)
    case 2: printf("Ban da chon chuc nang Edit\n");
            break;
    case 3: printf("Ban da chon chuc nang Search\n");
            break;
};
getch();
}

```

Kết quả in ra màn hình

```

-----
MAIN MENU
-----
1. File
2. Edit
3. Search
Chon muc tuong ung: 1
-----
MENU FILE
-----
1. New

```


2. Open

Chon muc tuong ung: 2

Ban da chon chuc nang Open File

Bài tập Chương 5

Sử dụng lệnh if

1. Viết lại chương trình ví dụ 3, sử dụng cấu trúc if dạng 2.
2. Viết lại chương trình ví dụ 11, sử dụng cấu trúc if dạng 1.
3. Viết lại chương trình ví dụ 11, sử dụng cấu trúc if dạng 2.
4. Viết chương trình nhập vào số nguyên dương, in ra thông báo số chẵn hay lẻ.

Hướng dẫn: Nhập vào số nguyên dương x. Kiểm tra nếu x chia hết cho hai thì x là số chẵn (hoặc chia cho 2 dư 0) ngược lại là số lẻ.

5. Viết chương trình nhập vào 4 số nguyên. Tìm và in ra số lớn nhất.

Hướng dẫn: Ta có 4 số nguyên a, b, c, d. Tìm 2 số nguyên lớn nhất x, y của 2 cặp (a, b) và (c, d). Sau đó so sánh 2 số nguyên x, y để tìm ra số nguyên lớn nhất.

6. Viết chương trình giải phương trình bậc 2: $ax^2 + bx + c = 0$, với a, b, c nhập vào từ bàn phím.

Hướng dẫn: Nhập vào 3 biến a, b, c.

Tính Delta = $b*b - 4*a*c$

Nếu Delta < 0 thì Phương trình vô nghiệm

Ngược lại

Nếu Delta = 0 thì

$x_1 = x_2 = -b/(2*a)$

Ngược lại

$x_1 = (-b - \sqrt{\text{Delta}})/(2*a)$

$x_2 = (-b + \sqrt{\text{Delta}})/(2*a)$

Hết Nếu

Hết Nếu

7. Viết chương trình nhập vào giờ phút giây (hh:mm:ss). Cộng thêm số giây nhập vào và in ra kết quả dưới dạng hh:mm:ss.

Hướng dẫn: Nhập vào giờ phút giây vào 3 biến gio, phut, giay và nhập và giây cộng thêm vào biến them:

Nếu giay + them < 60 thì

giay = giay + them

Ngược lại

giay = (giay + them) - 60

phut = phut + 1

Nếu phut >= 60 thì

phut = phut - 60

gio = gio + 1

Hết nếu

Hết nếu

Sử dụng lệnh switch

8. Viết chương trình nhập vào tháng, in ra tháng đó có bao nhiêu ngày.

Hướng dẫn: Nhập vào tháng

Nếu là tháng 1, 3, 5, 7, 8, 10, 12 thì có 30 ngày

Nếu là tháng 4, 6, 9, 11 thì có 31 ngày

Nếu là tháng 2 và là năm nhuận thì có 29 ngày ngược lại 28 ngày

(Năm nhuận là năm chia chắn cho 4)

9. Viết chương trình trò chơi One-Two-Three ra cái gì ra cái này theo điều kiện:

- Búa (B) thắng Kéo, thua Giấy.

- Kéo (K) thắng Giấy, thua Búa.

- Giấy (G) thắng Búa, thua Kéo.

Hướng dẫn: Dùng lệnh switch lồng nhau

10. Viết chương trình xác định biến ký tự color rồi in ra thông báo

- RED, nếu color = 'R' hoặc color = 'r'

- GREEN, nếu color = 'G' hoặc color = 'g'

- BLUE, nếu color = 'B' hoặc color = 'b'

- BLACK, nếu color có giá trị khác.

11. Viết chương trình nhập vào 2 số x, y và 1 trong 4 toán tử +, -, *, /. Nếu là + thì in ra kết quả $x + y$, nếu là - thì in ra $x - y$, nếu là * thì in ra $x * y$, nếu là / thì in ra x / y (nếu $y = 0$ thì thông báo không chia được)

Bài tập tổng hợp

12. Viết lại bài tập 8, 9, 10, 11 sử dụng lệnh if.

13. Viết chương trình nhập vào điểm 3 môn thi: Toán, Lý, Hóa của học sinh. Nếu tổng điểm ≥ 15 và không có môn nào dưới 4 thì in kết quả **qua**. Nếu qua mà các môn đều lớn hơn 5 thì in ra lời phê "**Học đều các môn**", ngược lại in ra "Học chưa đều các môn", các trường hợp khác là "Thi trượt".

14. Viết chương trình nhập vào ngày tháng năm (dd:mm:yy), cho biết đó là thứ mấy trong tuần.

15. Viết chương trình nhập số giờ làm và lương giờ rồi tính số tiền lương tổng cộng. Nếu số giờ làm lớn hơn 40 thì những giờ làm đôi ra được tính 1,5 lần.

16. Viết chương trình nhập vào 3 giá trị nguyên dương a, b, c. Kiểm tra xem a, b, c có phải là 3 cạnh của tam giác không? Nếu là 3 cạnh của tam giác thì tính diện tích của tam giác theo công thức sau:

$S = p(p + (a - b)(p - c))$, với p là $1/2$ chu vi của tam giác.

Hướng dẫn: a, b, c là 3 cạnh của tam giác phải thỏa điều kiện sau: $(a + b) > c$ và $(a + c) > b$ và $(b + c) > a$

17. Viết chương trình nhập vào 3 số nguyên rồi in ra màn hình theo thứ tự tăng dần.

18. Viết chương trình tính tiền điện gồm các khoảng sau:

- Tiền thuê bao điện kế: 1000đ/tháng
- Định mức sử dụng điện cho mỗi hộ là: 50 KW với giá 230đ/KW
- Nếu phần vượt định mức $\leq 50KW$ thì tính giá 480đ/KW
- Nếu $50KW < \text{phần vượt định mức} < 100KW$ thì tính giá 700đ/KW
- Nếu phần vượt định mức $\leq 100KW$ thì tính giá 900đ/KW

Chỉ số mới và cũ được nhập vào từ bàn phím

- In ra màn hình chỉ số cũ, chỉ số mới, tiền trả định mức, tiền trả vượt định mức, tổng tiền phải trả.

Chương 6 : CẤU TRÚC VÒNG LẶP

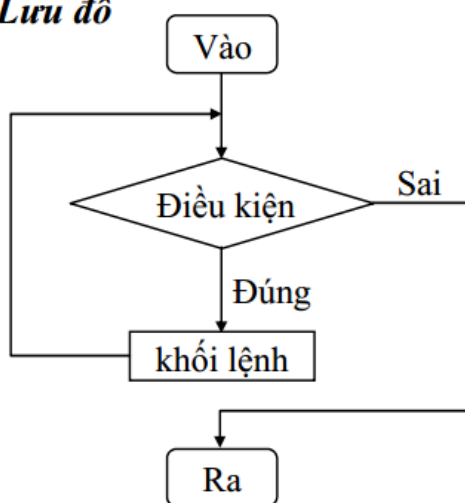
6.1. Lệnh for

Công dụng: Vòng lặp xác định thực hiện lặp lại một số lần xác định của một hay một chuỗi hành động.

- **Cú pháp lệnh**

**for (biểu thức 1; biểu thức 2; biểu thức 3)
khối lệnh;**

- **Lưu đồ**



Làm rõ cú pháp lệnh:

- + Biểu thức 1: Khởi tạo giá trị ban đầu cho biến điều khiển.
- + Biểu thức 2: Là quan hệ logic thể hiện điều kiện tiếp tục vòng lặp.
- + Biểu thức 3: Phép gán dùng thay đổi giá trị biến điều khiển.

Ví dụ 1: Viết chương trình in ra câu "Vi dụ su dung vong lap for" 3 lần.

Nhận xét:

- + Biểu thức 1 bao giờ cũng chỉ được tính toán một lần khi gọi thực hiện for.
- + Biểu thức 2, 3 và thân for có thể thực hiện lặp lại nhiều lần.

Lưu ý:

- + **Biểu thức 1, 2, 3 phải phân cách bằng dấu chấm phẩy (;)**
- + Nếu biểu thức 2 không có, vòng for được xem là luôn luôn **đúng**. Muốn thoát khỏi vòng lặp for phải dùng một trong 3 lệnh **break**, **goto** hoặc **return**.
- + Trong thân for (khối lệnh) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.
- + Khi gặp lệnh **break**, cấu trúc lặp sâu nhất sẽ thoát ra.

+ Trong thân for có thể sử dụng lệnh **continue** để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).

Ví dụ 2: Tính $1 + 2 + 3 + 4 + \dots + n$ với n nhập từ bàn phím

Ví dụ 3: Tính $n!$ với n nhập từ bàn phím

Một vài ví dụ thay đổi biến điều khiển vòng lặp.

- Thay đổi biến điều khiển từ 1 đến 100, mỗi lần tăng 1:

`for(i = 1; i <= 100; i++)`

- Thay đổi biến điều khiển từ 100 đến 1, mỗi lần giảm 1:

`for(i = 100; i >= 1; i--)`

- Thay đổi biến điều khiển từ 7 đến 77, mỗi lần tăng 7:

`for(i = 7; i <= 77; i += 7)`

- Thay đổi biến điều khiển từ 20 đến 2, mỗi lần giảm 2:

`for(i = 20; i >= 2; i -= 2)`

Ví dụ 4: Viết chương trình nhập vào số nguyên n . Tính tổng các giá trị lẻ từ 0 đến n .

Ví dụ 5: Tính tổng $s = 1 + \frac{1}{2} + \frac{1}{4} + \dots + \frac{1}{n}$ với n chẵn

6.2. Lệnh break

Thông thường lệnh break dùng để thoát khỏi vòng lặp không xác định điều kiện dừng hoặc chúng ta muốn dừng vòng lặp theo điều kiện do chúng ta chỉ định. Việc dùng lệnh break để thoát khỏi vòng lặp thường sử dụng phối hợp với lệnh if.

Ví dụ: Viết chương trình cho nhập liên tiếp các ký tự từ bàn phím, chỉ thoát khi bấm phím số 0;

6.3. Lệnh continue

Được dùng trong vòng lặp for, while, do...while. Khi gặp continue chương trình sẽ lộn ngược lên đầu vòng lặp, tất cả những lệnh đi sau trong vòng lặp chứa continue sẽ bị bỏ qua không thi hành.

Ví dụ: Viết chương trình tính tổng các số lẻ sử dụng continue

6.4. Lệnh while

Công dụng: Thực hiện lặp lại trong khi biểu thức còn đúng.

- **Cú pháp lệnh**

while (biểu thức)

khối lệnh;

Ví dụ 1: Viết chương trình in ra câu "Vi dụ su dung vong lap while" 3 lần.

Ví dụ 2: Hãy sử dụng vòng lặp while và để in ra các số từ 1 đến 100.

Ví dụ 3: Viết chương trình tính tổng các số nguyên từ 1 đến n, với n được nhập vào từ bàn phím.

6.5. Lệnh do...while

Công dụng: Thực hiện lặp lại cho đến khi biểu thức sai.

- **Cú pháp lệnh**

do

khối lệnh;

while (biểu thức);

Lưu ý: Sự khác biệt giữa lệnh while và lệnh do... while là ở chỗ:

- Trong lệnh while, biểu thức được kiểm tra trước khi thực hiện khối lệnh.

- Trong lệnh do... while, biểu thức được kiểm tra sau khi khối lệnh được thực hiện, nếu biểu thức còn đúng, khối lệnh tiếp tục được thực hiện. Nếu biểu thức sai thì khối lệnh không được thực hiện lại, như vậy nếu ngay từ đầu, biểu thức đã sai thì khối lệnh trong do... while vẫn được thực hiện ít nhất một lần.

Ví dụ 1: Viết chương trình kiểm tra password.

Ví dụ 2: Viết chương trình nhập vào năm hiện tại, năm sinh. In ra tuổi.

Bài tập Chương 6

1. Viết chương trình in ra bảng mã ASCII
2. Viết chương trình tính tổng bậc 3 của N số nguyên đầu tiên.
3. Viết chương trình nhập vào một số nguyên rồi in ra tất cả các ước số của số đó.
4. Viết chương trình vẽ một tam giác cân bằng các dấu *
5. Viết chương trình tính tổng nghịch đảo của N số nguyên đầu tiên theo công thức $S = 1 + 1/2 + 1/3 + \dots + 1/N$
6. Viết chương trình tính tổng bình phương các số lẻ từ 1 đến N.
7. Viết chương trình nhập vào N số nguyên, tìm số lớn nhất, số nhỏ nhất.
8. Viết chương trình nhập vào N rồi tính giai thừa của N.
9. Viết chương trình tìm USCLN, BSCNN của 2 số.

10. Viết chương trình vẽ một tam giác cân rỗng bằng các dấu *.
11. Viết chương trình vẽ hình chữ nhật rỗng bằng các dấu *.
12. Viết chương trình nhập vào một số và kiểm tra xem số đó có phải là số nguyên tố hay không?
13. Viết chương trình tính số hạng thứ n của dãy Fibonacci.
Dãy Fibonacci là dãy số gồm các số hạng $p(n)$ với:
 $p(n) = p(n-1) + p(n-2)$ với $n > 2$ và $p(1) = p(2) = 1$
Dãy Fibonacci sẽ là: 1 1 2 3 5 8 13 21 34 55 89 144...
14. Viết chương trình tính x^n với x, n được nhập vào từ bàn phím.
15. Viết chương trình nhập vào 1 số từ 0 đến 9. In ra chữ số tương ứng. Ví dụ: nhập vào số 5, in ra "Năm".
16. Viết chương trình lặp lại nhiều lần công việc nhập một ký tự và in ra mã ASCII của ký tự đó, khi nào nhập số 0 thì dừng.
17. Viết chương trình tìm ước số chung lớn nhất và bội số chung nhỏ nhất của 2 số nguyên.
18. Viết chương trình tính dân số của một thành phố sau 10 năm nữa, biết rằng dân số hiện nay là 6.000.000, tỉ lệ tăng dân số hàng năm là 1.8% .
19. Viết chương trình tìm các số nguyên gồm 3 chữ số sao cho tích của 3 chữ số bằng tổng 3 chữ số. Ví dụ: $1*2*3 = 1+2+3$.
20. Viết chương trình tìm các số nguyên tố từ 2 đến N, với N được nhập vào.
21. Viết chương trình lặp đi lặp lại các công việc sau:

- Nhập vào một ký tự trên bàn phím.
- Nếu là chữ thường thì in ra chính nó và chữ HOA tương ứng.
- Nếu là chữ HOA thì in ra chính nó và chữ thường tương ứng.
- Nếu là ký số thì in ra chính nó.
- Nếu là một ký tự điều khiển thì kết thúc chương trình

23. Viết chương trình nhập vào N số nguyên, đếm xem có bao nhiêu số âm, bao nhiêu số dương và bao nhiêu số không.

Chương 7: HÀM

Cung cấp cho người đọc các kiến thức về hàm như: Cách thức khai báo hàm; cách truyền tham số, tham biến, tham trị; sử dụng biến cục bộ, toàn cục trong hàm và sử dụng tiền xử lý #define.

7.1. Giới thiệu về hàm

Hàm trong lập trình là một chương trình con thực hiện một khối công việc được lặp đi lặp lại nhiều lần trong khi chạy chương trình hoặc dùng để tách một khối công việc cụ thể để chương trình đỡ phức tạp.

Quy tắc đặt tên hàm giống tên biến, hằng... Mỗi đối số cách nhau bởi dấu phẩy kèm theo kiểu dữ liệu tương ứng.

7.2. Các ví dụ về hàm

Ví dụ 1:

Untitled9.c [*] in19dausao.c

```
4
5     void line();
6     // ham in 1 dong 19 dau *
7
8     void line()
9     {
10         int i;
11         for(i = 0; i < 19; i++)
12             printf("*");
13         printf("\n");
14     }
15
16     void main(void)
17     {
18         line();
19         printf("* Minh hoa ve ham *\n");
20         line();
21         getch();
22     }
```

Kết quả in ra màn hình

```
*****
* Minh hoa ve ham *
*****
```

—

Ví dụ 2: Xây dựng hàm tính số mũ để tính toán công thức sau:

$$S = 1 + 2^2 + 3^3 + \dots + n^n$$

Kết quả in ra màn hình

```
2 mu 2 = 4.  
2 mu 3 = 8.
```

```
—
```

Giải thích chương trình

Hàm **power** có hai tham số truyền vào là *ix*, *in* có kiểu *int* và kiểu trả về cũng có kiểu *int*.

Dòng 13: *return ip*: trả về giá trị sau khi tính toán.

Dòng 18: đối mục 2 và 3 có kiểu trả về là *int* sau khi thực hiện gọi *power*.

Hai tham số *ix*, *in* của hàm *power* là dạng truyền tham trị.

7.3. Tham số dạng tham trị và tham biến

- Khi truyền tham trị, trình biên dịch sẽ tạo 1 bản sao và truyền bản sao đó vào vị trí muốn truyền, do đó mọi thay đổi sẽ xảy ra trên bản sao này và ko ảnh hưởng đến biến gốc

- Còn truyền tham biến thì nó sẽ truyền ngay địa chỉ hay là chính biến đó vào nơi cần truyền, nên ở trong hàm mà truyền tham biến giá trị của biến sẽ thay đổi sau khi thực hiện hàm.

7.4. Phạm vi của biến

Khi lập trình, chúng ta phải nắm rõ phạm vi của biến. Nếu khai báo và sử dụng không đúng, không rõ ràng sẽ dẫn đến sai sót khó kiểm soát được, vì vậy chúng ta cần phải xác định đúng vị trí, phạm vi sử dụng biến trước khi sử dụng biến.

Khai báo biến ngoài (biến toàn cục): Vị trí biến đặt bên ngoài tất cả các hàm, cấu trúc... Các biến này có ảnh hưởng đến toàn bộ chương trình. Chu trình sống của nó là bắt đầu chạy chương trình đến lúc kết thúc chương trình.

Khai báo biến trong (biến cục bộ): Vị trí biến đặt bên trong hàm, cấu trúc.... Chỉ ảnh hưởng nội bộ bên trong hàm, cấu trúc đó.... Chu trình sống của nó bắt đầu từ lúc hàm, cấu trúc được gọi thực hiện đến lúc thực hiện xong.

7.5. Dùng dẫn hướng #define

Sau đây là một vài ví dụ dùng dẫn hướng #define để định nghĩa hàm đơn giản

```
#define AREA_CIRCLE (frad) (4*PI*frad*frad) //tinh dien tich hinh cau
```

```
#define SUM (x, y) (x + y) //cong 2 so
```

```
#define SQR (x) (x*x) //tinh x binh phuong
```

```
#define MAX(x, y) (x > y) ? x : y //tim so lon nhat giua x va y
```

```
#define ERROR (s) printf("%s.\n", s) //in thong bao voi chuoi s
```

Bài tập chương 7

1. Viết hàm tính $n!$
2. Viết hàm tính tổng $S = 1+2+\dots+n$.
3. Viết hàm kiểm tra số nguyên tố.
4. Viết hàm tính số hạng thứ n trong dãy Fibonacci.
5. Viết hàm tìm số lớn nhất trong 2 số.

Chương 8: MẢNG VÀ CHUỖI

Cung cấp cho người đọc các kiến thức và kỹ năng về: Cách khai báo, nhập, xuất, khởi tạo và một số kỹ thuật thao tác liên quan đến mảng và chuỗi.

8.1. Mảng

Mảng là tập hợp các phần tử có cùng dữ liệu.

Giả sử có tình huống như sau: Chúng ta muốn lưu n số nguyên để tính trung bình, chúng ta không thể khai báo n biến để lưu n giá trị rồi sau đó tính trung bình.

Ví dụ: Chúng ta muốn tính trung bình 10 số nguyên nhập vào từ bàn phím, chúng ta sẽ khai báo 10 biến: $a, b, c, d, e, f, g, h, i, j$ có kiểu `int` và lập thao tác nhập cho 10 biến này như sau:

```
printf("Nhập vào biến a: ");
```

```
scanf("%d", &a);
```

10 biến chúng ta sẽ thực hiện 2 lệnh trên 10 lần, sau đó tính trung bình:

$$(a + b + c + d + e + f + g + h + i + j)/10$$

Điều này chỉ phù hợp với n nhỏ, còn đối với n lớn thì khó có thể thực hiện được. Vì vậy trong trường hợp này mảng được sử dụng

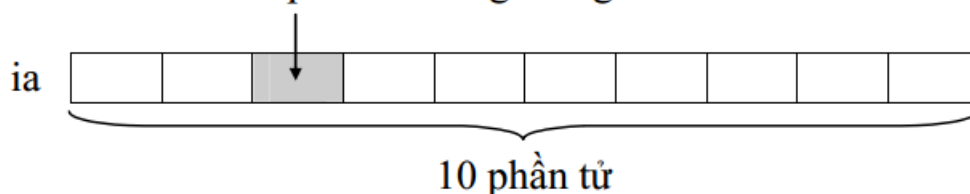
8.2. Mảng một chiều

8.1.1. Cách khai báo mảng một chiều

Ví dụ: `int ia[10];` với `int` là kiểu mảng, `ia` là tên mảng, 10 số phần tử mảng

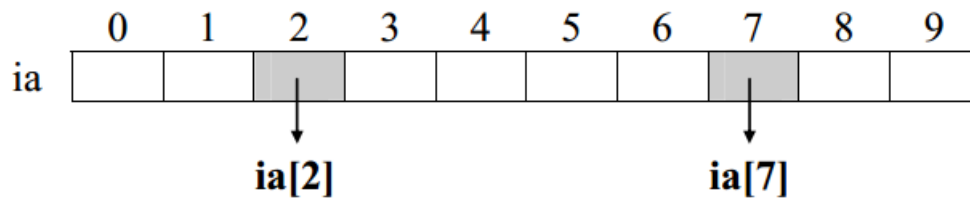
Ý nghĩa: *Khai báo một mảng số nguyên gồm 10 phần tử, mỗi phần tử có kiểu `int`.*

Mỗi phần tử trong mảng có kiểu `int`



8.1.2. Tham chiếu đến từng phần tử mảng một chiều

Sau khi mảng được khai báo, mỗi phần tử trong mảng đều có chỉ số để tham chiếu. Chỉ số bắt đầu từ 0 đến $n-1$ (với n là kích thước mảng). Trong ví dụ trên, ta khai báo mảng 10 phần tử thì chỉ số bắt đầu từ 0 đến 9.



$ia[2]$, $ia[7]$... là phần tử thứ 3, 8... trong mảng xem như là một biến kiểu **int**.

8.1.3. Nhập dữ liệu cho mảng một chiều

8.1.4. Đọc dữ liệu từ mảng một chiều

Ví dụ 1: Viết chương trình nhập vào n số nguyên. Tính và in ra trung bình cộng.

Ví dụ 2: `char cloai[20]; float ftemp[10];` cách tham chiếu, nhập dữ liệu, đọc dữ liệu như trên.

8.1.5. Kỹ thuật nhập các phần tử mảng một chiều

Khi muốn nhập các phần tử mảng, chúng ta có thể sử dụng kỹ thuật Sentinel: Đây là kỹ thuật để nhập liệu giá trị cho các phần tử mảng mà không biết rõ số lượng phần tử sẽ nhập vào là bao nhiêu (không biết số n).

Ví dụ: Viết chương trình nhập vào 1 dãy số dương rồi in tổng các số dương đó.

8.1.6. Khởi tạo mảng một chiều

Ví dụ: Có 8 loại tiền gồm 500000, 200000, 100000, 50000, 20000, 10000, 5000, 2000, 1000 đồng. Hãy viết chương trình nhập vào số tiền sau đó cho biết số số tiền trên gồm mấy loại tiền, mỗi loại bao nhiêu tờ.

8.1.7. Khởi tạo mảng một chiều không bao hàm kích thước

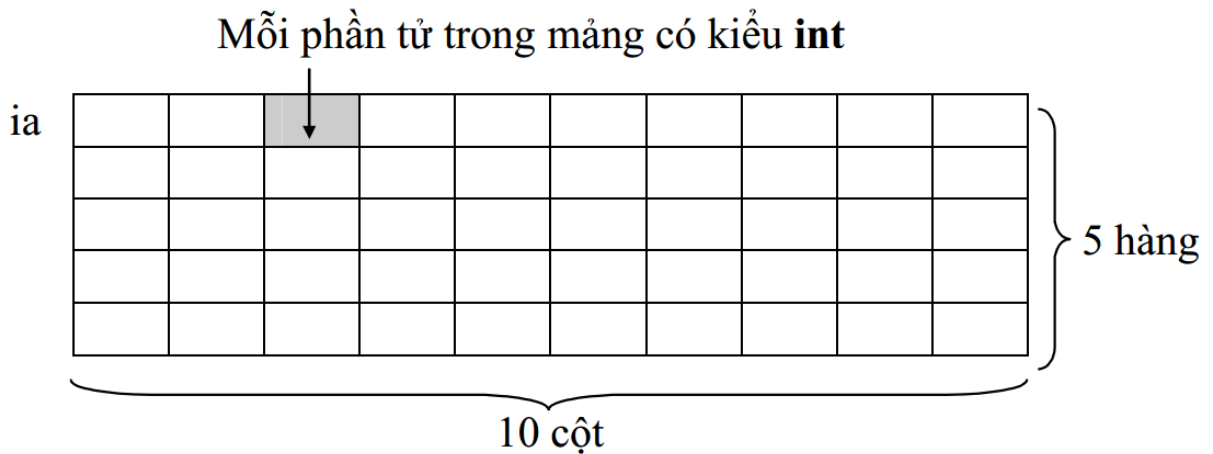
Trong ví dụ trên giả sử ta khai báo `int tien[] = {500000, 200000, 100000, 50000, 20000, 10000, 5000, 2000, 1000}`. Khi đó trình biên dịch sẽ đếm số mục trong danh sách khởi tạo và dùng con số đó làm kích thước mảng.

8.3. Mảng hai chiều

8.3.1. Khai báo mảng hai chiều

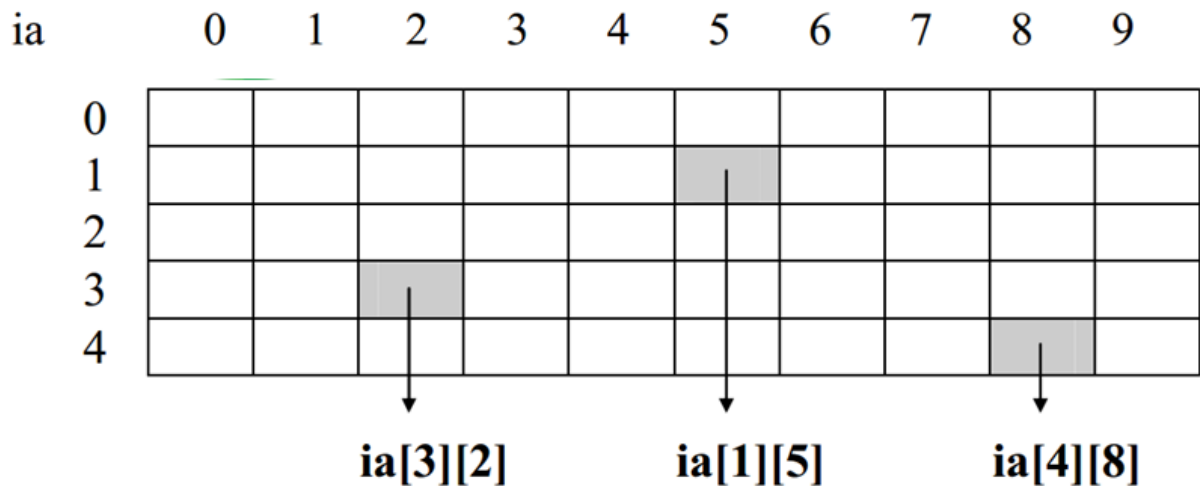
Ví dụ: Khai báo mảng 2 chiều `int ia[5][10]`; với **int** là kiểu mảng, **ia** là tên mảng, số phần tử mảng là 5×10 .

Ý nghĩa: *Khai báo một mảng 2 chiều kiểu nguyên gồm 50 phần tử, mỗi phần tử có kiểu int.*



8.3.2. Tham chiếu đến từng phần tử mảng 2 chiều

Sau khi được khai báo, mỗi phần tử trong mảng 2 chiều đều có 2 chỉ số để tham chiếu, chỉ số hàng và chỉ số cột. Chỉ số hàng bắt đầu từ 0 đến và chỉ số cột bắt đầu từ 0 đến. Cách tham chiếu đến một phần tử trong mảng 2 chiều **ia** là: **ia[chỉ số hàng][chỉ số cột]**



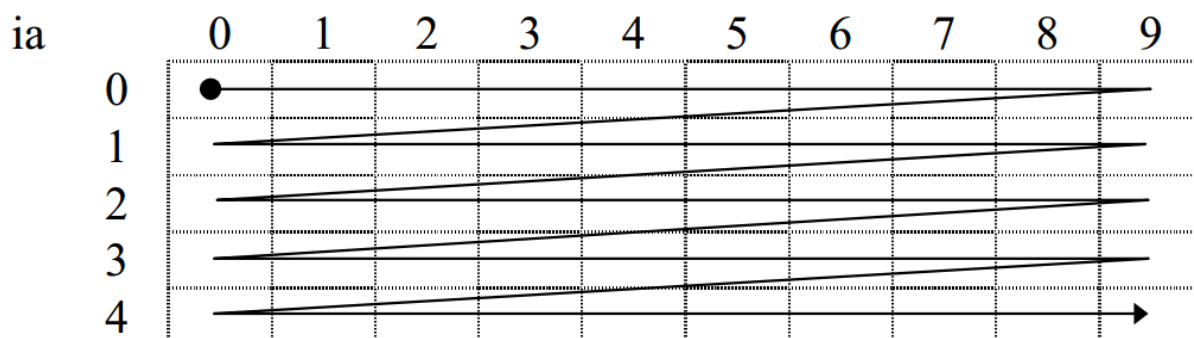
ia[3][2] là phần tử tại hàng 3 cột 2 trong mảng 2 chiều xem như là một biến kiểu **int**.

8.3.3. Nhập dữ liệu cho mảng 2 chiều

Xem xét ví dụ sau:

```
for (i = 0; i < 5; i++) //vòng for có giá trị i chạy từ 0 đến 4 cho hàng
    for (j = 0; j < 10; j++) //vòng for có giá trị ij chạy từ 0 đến 9 cho cột
    {
        printf("Nhap vao phan tu ia[%d][%d]: ", i + 1, j + 1);
        scanf("%d", &ia[i][j]);
    }
```

* Thứ tự nhập dữ liệu vào mảng 2 chiều



8.3.4. Đọc (in) dữ liệu từ mảng 2 chiều

Ví dụ 1: In giá trị các phần tử mảng 2 chiều ra màn hình.

```
for (i = 0; i < 5; i++) //vòng for có giá trị i chạy từ 0 đến 4 cho hàng
{
    for (ij = 0; j < 10; ij++) //vòng for có giá trị ij chạy từ 0 đến 9 cho cột
        printf("%3d ", ia[i][ij]);
    printf("\n"); //xuống dòng để in hàng kế tiếp
}
```

Ví dụ 2: Viết chương trình nhập vào 1 ma trận số nguyên $n \times n$. In ra ma trận vừa nhập vào và ma trận theo thứ tự ngược lại.

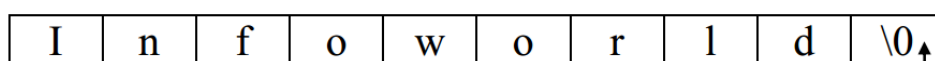
8.3.5. Khởi tạo mảng 2 chiều

Ví dụ: Vẽ chữ H hoa các dấu !.

8.4. Chuỗi

Chuỗi được xem như là một mảng 1 chiều gồm các phần tử có kiểu char như ký tự, con số và bất cứ ký tự đặc biệt như +, -, *, /, \$, #...

Theo quy ước, một chuỗi sẽ được kết thúc bởi ký tự **null** ('\0' : kí tự rỗng). Ví dụ: chuỗi "Infoworld" được lưu trữ như sau:



Kí tự kết thúc chuỗi

8.4.1. Cách khai báo chuỗi

Ví dụ 1: `char cname[30];`

Ý nghĩa: **Khai báo chuỗi cname có chiều dài 30 ký tự.** Do chuỗi kết thúc bằng ký tự null, nên khi chúng ta khai báo chuỗi có chiều dài 30 ký tự chỉ có thể chứa 29 ký tự.

Ví dụ 2: Nhập vào tên và in ra tên

Lưu ý: không cần sử dụng toán tử địa chỉ & trong cname trong lệnh `scanf("%s", cname)`, vì bản thân cname đã là địa chỉ.

8.4.2. Hàm nhập (gets), xuất (puts) chuỗi

Sử dụng hàm gets, puts phải khai báo thư viện `#include <stdio.h>`

Đối với hàm puts kí tự kết thúc chuỗi null (\0) được thay thế bằng kí tự xuống dòng (\n). Hàm gets và puts chỉ có 1 đối số và không sử dụng dạng thức trong nhập liệu cũng như xuất ra màn hình.

8.4.3. Khởi tạo chuỗi

Chiều dài tối đa của chuỗi khởi tạo bằng số kí tự + 1 (kí tự null).

8.4.4. Mảng chuỗi

Bài tập: nhập vào 1 số là các tháng trong năm và in ra tháng đó bằng tiếng Anh.

Bài tập Chương 8

1. Viết hàm tìm số lớn nhất, nhỏ nhất trong một mảng n số nguyên.
2. Viết hàm sắp xếp tăng dần, giảm dần của một dãy số cho trước.
3. Viết hàm tách tên và họ lót từ một chuỗi cho trước.
4. Viết hàm cắt bỏ khoảng trắng thừa ở giữa, hai đầu một chuỗi.
5. Viết hàm chuyển đổi 1 chuỗi sang chữ thường và 1 hàm chuyển đổi sang chữ HOA.
6. Viết hàm chuyển đổi 1 chuỗi sang dạng Title Case (kí tự đầu của mỗi từ là chữ hoa, các kí tự còn lại chữ thường)
7. Viết chương trình nhập vào 1 chuỗi và in ra chuỗi đảo ngược.
Ví dụ: Nhập vào chuỗi "Lap trinh C can ban"
In ra "nab nac C hnirt paL"
8. Viết chương trình nhập vào một chuỗi ký tự rồi đếm xem trong chuỗi đó có bao nhiêu chữ 'th'.
9. Viết chương trình nhập vào 3 số có 3 chữ số. Cho biết dòng chữ mô tả giá trị con số đó. Ví dụ 305 -> ba trăm lẻ năm.
10. Viết chương trình nhập vào một chuỗi sau đó in ra màn hình mỗi dòng là một từ. Ví dụ chuỗi "Lap trinh C". Kết quả in ra
Lap
trinh
C
11. Viết chương trình nhập vào một chuỗi các kí tự, ký số, khoảng trắng và dấu chấm câu. Cho biết chuỗi trên gồm bao nhiêu từ.

Chương 9. CON TRỎ TRONG C

Cung cấp cho người đọc các kiến thức và kỹ năng khi sử dụng con trỏ trong C như: Ý nghĩa của việc sử dụng con trỏ, cách khai báo biến, mảng con trỏ, cách lấy địa chỉ một biến, cách truy xuất, một số phép toán trên con trỏ, cấp phát và thu hồi vùng nhớ, mối quan hệ giữa con trỏ và mảng, chuỗi ký tự.

9.1. Giới thiệu

Các biến chúng ta đã biết và sử dụng trước đây đều là biến có kích thước và kiểu dữ liệu xác định. Người ta gọi các biến kiểu này là biến tĩnh. Khi khai báo biến tĩnh, một lượng ô nhớ cho các biến này sẽ được cấp phát mà không cần biết trong quá trình thực thi chương trình có sử dụng hết lượng ô nhớ này hay không. Mặt khác, các biến tĩnh dạng này sẽ tồn tại trong suốt thời gian thực thi chương trình dù có những biến mà chương trình chỉ sử dụng 1 lần rồi bỏ.

Có một số hạn chế có thể gặp phải khi sử dụng các biến tĩnh:

Cấp phát ô nhớ dư, gây ra lãng phí ô nhớ.

Cấp phát ô nhớ thiếu, chương trình thực thi bị lỗi.

Để tránh những hạn chế trên, ngôn ngữ C cung cấp cho ta một loại biến đặc biệt gọi là biến động với các đặc điểm sau:

Chỉ phát sinh trong quá trình thực hiện chương trình chứ không phát sinh lúc bắt đầu chương trình.

Khi chạy chương trình, kích thước của biến, vùng nhớ và địa chỉ vùng nhớ được cấp phát cho biến có thể thay đổi.

Sau khi sử dụng xong có thể giải phóng để tiết kiệm chỗ trong bộ nhớ. Tuy nhiên các biến động không có địa chỉ nhất định nên ta không thể truy cập đến chúng được. Vì thế, ngôn ngữ C lại cung cấp cho ta một loại biến đặc biệt nữa để khắc phục tình trạng này, đó là biến con trỏ (pointer) với các đặc điểm:

Biến con trỏ không chứa dữ liệu mà chỉ chứa địa chỉ của dữ liệu hay chứa địa chỉ của ô nhớ chứa dữ liệu.

Kích thước của biến con trỏ không phụ thuộc vào kiểu dữ liệu, luôn có kích thước cố định là 2 byte.

9.2. Biến con trỏ

Mỗi biến khi được khai báo đều được cấp phát cho 1 vùng nhớ nhất định ở những nơi (địa chỉ) khác nhau.

Biến con trỏ là biến dùng để lưu trữ địa chỉ của các biến đó.

Ví dụ:

```
#include <stdio.h>

int main() {
    /* khai bao bien x va bien con tro px */
    int x, *px;
    px = &x;
    /* &x : tra ve dia chi cua bien x
    * px = &x : gan dia chi cua bien x cho px hay px tro den x */
    x = 42;
    printf("Vi tri cua bien x la %p \n", &x);
    printf("Noi dung cua bien x la %d \n", x);
    printf("Vi tri cua bien x la %p \n", px);
    printf("Noi dung cua bien x la %d \n", *px);
    *px = 7826;
    printf("\n ----- \n\n");
    printf("Noi dung cua bien x la %d \n", x);
    printf("Noi dung cua bien x la %d \n", *px);
    return 0;
}
```

Kết quả khi chạy chương trình:

Vi tri cua bien x la 0xbff327e4

Noi dung cua bien x la 42

Vi tri cua bien x la 0xbff327e4

Noi dung cua bien x la 42

Noi dung cua bien x la 7826

Noi dung cua bien x la 7826

9.2.1. Khai báo biến con trỏ

Với mỗi kiểu dữ liệu ta có tương ứng một biến con trỏ có kiểu đó.

Cú pháp khai báo: **Kiểu * Tên biến con trỏ;**

Trong ví dụ trên ta khai báo 1 biến con trỏ px thuộc kiểu int.

9.2.2. Lấy địa chỉ của một biến

Ta dùng toán tử & để lấy địa chỉ của 1 biến và sau đó gán địa chỉ đó cho biến con trỏ.

Cú pháp: *Tên con trỏ* = **&biến**;

9.2.3. Cách truy xuất

Với con trỏ px bên trên ta có 2 phép truy xuất là:

px : *Lấy địa chỉ mà nó lưu giữ (trở tới)*

***px** : *Lấy giá trị trong vùng nhớ mà nó trở tới.*

Trong VD trên ta có thể thấy sau phép gán **px = &x** ; thì việc ta viết:

px sẽ tương đương với &x

*px sẽ tương đương với x, ta có thể sử dụng *px trong các phép toán, biểu thức.

9.2.4. Một số phép toán trên biến con trỏ

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

```
int main() {
```

```
    /* khai bao bien x va 2 bien con tro px, qx */
```

```
    int x, *px, *qx;
```

```
    px = &x;
```

```
    printf("Nhap gia tri cho vung nho px tro toi: ");
```

```
    scanf("%d", px);
```

```
    /* px la con tro nen khong viet scanf("%d", &px); */
```

```
    qx = px; /* gan gia tri cua px cho qx, qx cung tro toi x*/
```

```
    printf("Vi tri cua bien x la %p \n", &x);
```

```
    printf("Vi tri cua bien x la %p \n", px);
```

```
    printf("Vi tri cua bien x la %p \n", qx);
```

```
    printf("Noi dung cua bien x la %d \n", x);
```

```
    printf("Noi dung cua bien x la %d \n", *px);
```

```
    printf("Noi dung cua bien x la %d \n", *qx);
```

```
    // tang gia tri cua o nho len, <=> x = x + 7826
```

```
    *px += 7826;
```

```
    printf("Noi dung cua bien x la %d \n", x);
```

```
    px++;
```

```
    /* cong them mot don vi cho px
```

```
    * => px tro toi vung nho tiep theo
```

```
    */
```

```
    printf("Vi tri px tro toi la %p \n", px);
```

```
    return 0;
```


}

Kết quả khi chạy chương trình:

Nhap gia tri cho vung nho px tro toi: 42

Vi tri cua bien x la 0xbfba58a0

Vi tri cua bien x la 0xbfba58a0

Vi tri cua bien x la 0xbfba58a0

Noi dung cua bien x la 42

Noi dung cua bien x la 42

Noi dung cua bien x la 42

Noi dung cua bien x la 7868

Vi tri px tro toi la 0xbfba58a4

Trong ví dụ trên ta thấy có một số phép toán trên con trỏ hay gặp sau: (ngoài ra còn nhiều phép toán khác).

Hai biến con trỏ cùng kiểu có thể được gán cho nhau hoặc thực hiện các phép toán cộng cho một số nguyên, trừ 2 con trỏ cho nhau. Ở ví dụ trên ta thực hiện các phép toán:

Gán: $qx = px$; Trong trường hợp này, qx nhận giá trị của px hiện có là địa chỉ của biến x , tức là qx và px cùng trỏ đến x . Ngoài ra ta có thể gán như sau: $qx = px + 2$; với qx , px là các biến con trỏ cùng kiểu. Phép trừ 2 con trỏ cùng kiểu sẽ trả về 1 giá trị nguyên (int), đây chính là khoảng cách (số phần tử) giữa 2 con trỏ đó.

Tăng: Các phép tăng giảm, cộng trừ được thực hiện trên biến con trỏ tương tự như với các biến số học. Điểm khác biệt duy nhất là nó tăng giảm, cộng trừ theo đơn byte mà kiểu của nó có.

Trong ví dụ trên ta có phép tăng: $px++$; Khi này giả sử px đang trỏ đến địa chỉ: 0xbfba58a0 thì sau phép tăng nó có giá trị là (trỏ đến vị trí) 0xbfba58a4 (tăng lên 4) vì px là con trỏ kiểu int mà mỗi biến kiểu int chiếm 4 byte trong bộ nhớ.

Ngoài ra chúng ta để ý còn phép thay đổi giá trị của biến x bằng phép toán $*px += 3$; Ở phép toán này thực chất là ta đã thay đổi giá trị ở ô nhớ (địa chỉ) mà px trỏ tới, từ đó dẫn đến giá trị của biến x cũng thay đổi theo.

Chú ý:

Tùy theo trình dịch mà dung lượng của các kiểu là khác nhau (trong trình dịch này thì kiểu int chiếm 4 byte nhưng trong trình dịch khác thì nó lại chiếm 2 byte). Để biết dung lượng từng kiểu chúng ta dùng toán tử sizeof().

Mỗi biến con trỏ, dù là con trỏ thuộc kiểu nào (int, float, double,...) cũng chỉ chiếm 2 byte bộ nhớ để lưu trữ địa chỉ của các biến.

9.3. Cấp phát và thu hồi vùng nhớ

9.3.1 Cấp phát

Ví dụ:

```
#include <stdio.h>

int main() {
    int *px;
    *px = 42;
    printf("Vi tri con tro px la %p \n", px);
    printf("Gia tri con tro px tro toi la %d \n", *px);
    return 0;
}
```

Khi biên dịch thì sẽ không có lỗi (có cảnh báo), khi chạy sẽ không hiển thị gì mà chương trình sẽ thoát ra luôn.

Nguyên nhân là khi khai báo biến con trỏ px thì máy mới chỉ cung cấp 2 byte để lưu địa chỉ của biến con trỏ mà chưa cấp phát vùng nhớ để con trỏ px lưu trữ dữ liệu. (tương tự như hợp tác xã cung cấp 2 Kg thóc cho chúng ta để làm giống nhưng lại không cung cấp cho chúng ta ruộng đất để chúng ta gieo mạ vậy).

Lưu ý: Có một số trình dịch sẽ không báo lỗi mà vẫn chạy bình thường nhưng tốt nhất là ta nên cấp phát trước khi sử dụng. Lỗi này sẽ xuất hiện rõ nhất khi chúng ta sử dụng con trỏ với mảng mà ở phần tiếp theo ta sẽ đề cập.

Vậy làm sao để cấp phát vùng nhớ cho con trỏ.

Để cấp phát vùng nhớ cho con trỏ ta dùng các hàm sau trong thư viện stdlib.h.

malloc : tên con trỏ = (kiểu con trỏ *) malloc (sizeof(kiểu con trỏ));

calloc : tên con trỏ = (kiểu con trỏ *) malloc (n, sizeof(kiểu con trỏ));

Trong đó sizeof(kiểu con trỏ) là kích thước của kiểu; n là số lần của sizeof(kiểu con trỏ) được cấp.

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

int main() {
    int *px, *qx;
    px = (int *) malloc(sizeof(int));
    qx = (int *) calloc(1, sizeof(int));
    printf("Vi tri con tro px la %p \n", px);
    printf("Gia tri con tro px tro toi la %d \n", *px);
}
```

```

printf("Vi tri con tro qx la %p \n", qx);
printf("Gia tri con tro qx tro toi la %d \n", *qx);
return 0;
}

```

Ở đây cần chú ý: Sự khác nhau duy nhất giữa malloc và calloc có thể hiểu đơn giản là với malloc thì khi cấp phát máy sẽ cấp phát cho px 1 ô bất kỳ mà không cần biết ô đó có dữ liệu là gì hay không có dữ liệu (do đó *px có giá trị như trên) còn calloc cũng vậy nhưng khác 1 điểm là sau khi cấp phát thì máy sẽ tự động gán luôn giá trị 0 cho ô nhớ mà biến qx trỏ tới, tức qx có giá trị mặc định là 0.

Khi cấp phát cho biến con trỏ một số lượng ô nhớ nào đó mà trong quá trình làm việc ta thiếu và cần cấp phát thêm thì ta sử dụng lệnh **realloc**:

tên con trỏ = (kiểu con trỏ *) realloc (tên con trỏ, số lượng cần cấp phát * sizeof(kiểu con trỏ));

Trong đó: số lượng cần cấp phát = cũ + mới.

Ví dụ: Ban đầu ta cấp phát cho con trỏ px là 10 ô nhớ. Sau đó muốn cấp phát thêm cho nó 5 ô nhớ nữa thì số lượng cấp phát bằng 15.

9.3.2. Thu hồi và kiểm tra vùng nhớ còn lại

Để thu hồi bộ nhớ đã cấp phát ta dùng hàm ***free(tên con trỏ);***

9.4. Hàm có đối là con trỏ

Như trong bài Hàm chúng ta đã biết cách truyền các tham số a,b trong hàm HoanVi là cách truyền bằng tham trị chứ không phải truyền bằng địa chỉ (hay tham biến) nên mặc dù trong hàm thì giá trị các biến đã được thay đổi nhưng sau khi hàm thực hiện xong thì các giá trị vẫn chưa thể thay đổi được. Ta sẽ phải sửa lại bằng cách truyền tham số hình thức là con trỏ a và con trỏ b để khi thực hiện hoán đổi có thể hoán đổi tại địa chỉ của các ô nhớ đó. Khi đó ta mới có được kết quả mong muốn.

```

#include <stdio.h>
void hoanVi(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
int main() {
    int a = 42, b = 7826;
    printf("Truoc khi gọi ham hoan vi: a = %d, b = %d \n", a, b);
}

```

```

    hoanVi(&a, &b);
    printf("Sau khi gọi hàm hoán vị: a = %d, b = %d \n", a, b);
    return 0;
}

```

9.5. Mối quan hệ giữa con trỏ và mảng, chuỗi ký tự

9.5.1. Con trỏ và mảng 1 chiều

Như trong phần 9.1 chúng ta đã biết có thể coi biến mảng như một con trỏ, vì vậy ta có thể sử dụng chính biến mảng đó để truy cập mảng theo cách của con trỏ.

```

#include <stdio.h>

void nhapMang(int a[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("Nhập a[%d] = ", i);
        scanf("%d", &a[i]);
    }
}

void nhapControlo(int a[], int n){
    int i;
    for (i = 0; i < n; i++) {
        printf("Nhập a[%d] = ", i);
        scanf("%d", a + i);
    }
}

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

int main() {
    // khai báo mảng a có n phần tử
    int n = 5;
    int a[n];
    nhapControlo(a, n);
}

```

```

    xuatMang(a, n);
    return 0;
}

```

Ở hàm thứ nhất đã khá quen thuộc, chúng ta không nói tới nữa.

Ở hàm thứ hai, chúng ta chỉ thay &a bằng a+i vì khi khai báo a[20] thì a coi như là một con trỏ và máy sẽ cấp phát cho ta các ô nhớ liên tiếp từ a đến a + 19. Và a + i là địa chỉ của a[i] (tức nó tương đương với &a[i]). a trỏ đến vị trí a[0].

Ngoài ra có thể khai báo 1 mảng sau đó dùng 1 con trỏ trỏ tới đầu mảng thì con trỏ đó cũng trở thành mảng đó.

```

#include <stdio.h>

int main() {
    int n = 5, i;
    int a[n], *pa;
    pa = a; // con tro pa tro toi dau mang a
    for (i = 0; i < n; i++) {
        printf("Nhap a[%d] = ", i);
        scanf("%d", pa + i);
    }
    for (i = 0; i < n; i++) {
        printf("%d\t", *(pa + i));
    }
    return 0;
}

```

Chú ý: Tại sao ta không cần cấp phát ô nhớ cho con trỏ pa mà vẫn sử dụng được bình thường, bởi vì ta đã khai báo mảng a[20] nên máy đã cấp phát ô nhớ để lưu trữ mảng a, khi ta thực hiện trỏ con trỏ pa tới mảng a thì các ô nhớ này đã có rồi nên không cần cấp phát ô nhớ cho pa nữa. Ta xét ví dụ sau:

```

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

int main() {
    int n = 5, i;
    // cap phat bo nho cho pa
    int *pa = (int *) malloc(n * sizeof(int));
    for (i = 0; i < n; i++) {

```

```

    printf("Nhap a[%d] = ", i);
    scanf("%d", pa + i);
}
for (i = 0; i < n; i++) {
    printf("%d \t", *(pa + i));
}
return 0;
}

```

Ở trong ví dụ này ta không khai báo mảng a như trước, do đó cũng không thể trở pa tới vị trí nào, mà muốn thực hiện được ta cần cấp phát các ô nhớ cho pa như trên.

9.5.2. Nhập và xuất mảng trong hàm

Việc nhập mảng không phải lúc nào cũng thuận lợi vì như các ví dụ trước chúng ta cần phải có số lượng phần tử của mảng trước khi cấp phát hoặc nhập, vậy nếu chúng ta chưa biết trước số phần tử mà lại phải dùng hàm để nhập mảng thì sao. Hãy xem ví dụ sau.

```

#include <stdio.h>
#include <stdlib.h>
void nhapContro(int *(*a), int *n) {
    int i;
    printf("Nhap so phan tu cua mang: ");
    scanf("%d", n); // khong phai &n
    *a = (int *) malloc ((*n) * sizeof(int));
    // *a : lay dia chi cua mang a chu khong phai gia tri cua a
    for (i = 0; i < *n; i++) {
        printf("Nhap a[%d] = ", i);
        scanf("%d", (*a + i));
    }
}
void xuatMang(int *a, int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d \t", a[i]);
    }
}

```

```

int main() {
    int *a, n;
    nhapContro(&a, &n); // lay dia chi cua a va n
    xuatMang(a, n);
    return 0;
}

```

Trong ví dụ này ta thực hiện nhập và xuất mảng trong hàm, cấp phát bộ nhớ cũng trong hàm luôn. Có một điểm chú ý là trong hàm nhập mảng a (nhapContro) bằng con trỏ thì có 2 dấu *, một dấu là của mảng a (dấu thứ 2), còn dấu đầu tiên là dùng để truyền địa chỉ làm giá trị của mảng có thể giữ nguyên khi ra khỏi hàm, nó giống như là dấu * trong hàm **HoanVi(int *a,int *b)** vậy.

9.5.3. Con trỏ và chuỗi ký tự

Do chuỗi ký tự bản chất cũng là mảng các ký tự nên phần này nó cũng tương tự như mảng 1 chiều, hãy xem qua một chút bằng ví dụ đơn giản sau.

```

#include <stdio.h>
#include <stdlib.h>
int main() {
    char *name;
    name = (char *) malloc (100*sizeof(char));
    printf("What your name? ");
    gets(name);
    printf("Oh, Hello %s", name);
    return 0;
}

```

9.5.4. Con trỏ và mảng 2 chiều, mảng các con trỏ, con trỏ đa cấp

9.5.4.1. Con trỏ và mảng 2 chiều

Phần trên chúng ta đã tìm hiểu về con trỏ và mảng 1 chiều, vậy con trỏ và mảng 2 chiều cũng có nhiều điểm tương tự.

Như ta đã biết thực chất trong máy tính thì bộ nhớ lưu mảng 2 chiều giống như mảng 1 chiều. Vì vậy ta hoàn toàn có thể biểu diễn mảng 2 chiều bằng con trỏ giống như mảng 1 chiều.

```

#include <stdio.h>
#include <stdlib.h>
int main() {

```

```

double a[10][10], *pa;
int n, m, i;
pa = (double *) a;
printf("Nhap so hang va so cot: ");
scanf("%d %d", &n, &m);
for (i = 0 ; i < n * m; i++) {
    printf("Nhap a[%d][%d] = ", i / m, i % m);
    scanf("%lf", pa + i);
}
for (i = 0 ; i < n * m; i++) {
    if (i % m == 0) printf("\n"); // xuống dòng
    printf("%-5.2lf", *(pa + i));
}
return 0;
}

```

Kết quả:

Nhap so hang va so cot: 2 3

Nhap a[0][0] = 4.23

Nhap a[0][1] = 5.7

Nhap a[0][2] = 1.2

Nhap a[1][0] = 8.6

Nhap a[1][1] = 3.456

Nhap a[1][2] = 12

4.23 5.70 1.20

8.60 3.46 12.00

Ngoài ra, chúng ta có thể không cần dùng pa mà dùng ngay a là 1 con trỏ. Hoặc ta có thể nhập 1 cách tương tự như mảng 2 chiều bình thường như sau.

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

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

```
int main() {
```

```
    double a[10][10], *pa;
```

```
    int n, m, i, j;
```

```
    pa = (double *) a;
```

```
    printf("Nhap so hang va so cot: ");
```



```

scanf("%d %d", &n, &m);
for (i = 0 ; i < n; i++) {
    for (j = 0 ; j < m; j++) {
        printf("Nhap a[%d][%d] = ", i, j);
        scanf("%lf", pa + i * 10 + j);
    }
}
printf("\n"); // xuống dòng
for (j = 0 ; j < m; j++) {
    printf("%-5.2lf", *(pa + i * 10 + j));
}
}
return 0;
}

```

Lưu ý: Tại sao ta lại có **pa + i*10 + j**. Đó là vì ở hàm main() ta khai báo là a[10][10] nên máy cấp phát cho chúng ta các ô nhớ của mảng 2 chiều với 10 hàng, 10 cột mà nếu ta không dùng hết thì nó vẫn tồn tại.

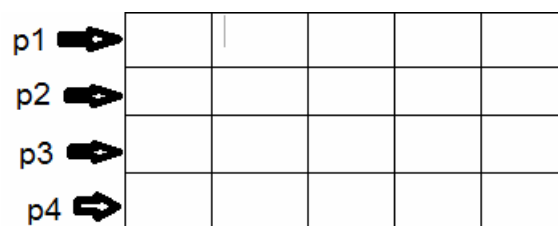
Ta có thể viết a[i][j] nhưng **không thể viết pa[i][j]** vì theo khai báo thì a là mảng 2 chiều nên ta có thể viết như vậy còn pa là 1 con trỏ và khi ta gán pa = a thì ta đã ngầm định coi a là mảng 1 chiều. Vì vậy nếu có thì chúng ta cũng chỉ được phép viết pa[i*10+j] (lấy giá trị) và &pa[i*10+j] (lấy địa chỉ).

9.5.4.2. Mảng các con trỏ

Mảng con trỏ là một mảng chứa tập hợp các con trỏ cùng một kiểu.

Ví dụ: Float *a[10]; // khai báo một mảng con trỏ. Gồm 10 con trỏ: a[0], a[1], ...a[9]; là 10 con trỏ.

Liên hệ với mảng 2 chiều thì ta có nhận xét sau: Mỗi hàng của mảng 2 chiều ta coi như 1 mảng 1 chiều. Mà mỗi con trỏ thì lại có mối quan hệ với 1 mảng 1 chiều, ta có hình vẽ mô tả như sau:



```

#include <stdio.h>
#include <stdlib.h>
int main() {
    double a[10][10], *pa[10];
    int n, m, i, j;
    printf("Nhap so hang va so cot: ");
    scanf("%d %d", &n, &m);
    for (i = 0 ; i < n; i++) {
        pa[i] = a[i]; // con tro thu i tro den hang thu i
        for (j = 0 ; j < m; j++) {
            printf("Nhap a[%d][%d] = ", i, j);
            scanf("%lf", &pa[i][j]);
        }
    }
    for (i = 0 ; i < n; i++) {
        printf("\n"); // xuống dòng
        for (j = 0 ; j < m; j++) {
            printf("%-5.2lf", pa[i][j]);
        }
    }
    return 0;
}

```

Sở dĩ ở ví dụ này ta viết được `pa[i][j]` đó là do mỗi `pa` là 1 con trỏ đến mảng một chiều. `pa[i][j]` tức là phần tử thứ `j` của con trỏ `pa[i]`.

Các ví dụ ở trên chúng ta đều xét khi mà khai báo mảng `a[][]` nên không cần cấp phát bộ nhớ cho con trỏ, bây giờ muốn cấp phát bộ nhớ cho con trỏ với mảng 2 chiều giống như cấp phát ở mảng 1 chiều thì ta làm như sau:

```

#include <stdio.h>
#include <stdlib.h>
int main() {
    double **pa;
    int n, m, i, j;
    printf("Nhap so hang va so cot: ");
    scanf("%d %d", &n, &m);

```

```

// cap phat n o nho cho n con tro (n hang)
pa = (double**) malloc(n * sizeof(double));
for (i = 0 ; i < n; i++) {
    // cap phat m o nho cho moi con tro (moi hang)
    pa[i] = (double *) malloc(m * sizeof(double));
    for (j = 0 ; j < m; j++) {
        printf("Nhap a[%d][%d] = ", i, j);
        scanf("%lf", &pa[i][j]);
    }
}
for (i = 0 ; i < n; i++) {
    printf("\n"); // xuong dong
    for (j = 0 ; j < m; j++) {
        printf("%-5.2lf", pa[i][j]);
    }
}
return 0;
}

```

Đây cũng có thể coi là mảng con trỏ, hoặc con trỏ trỏ đến con trỏ (con trỏ đa cấp).

Bài tập Chương 9

Thực hiện các yêu cầu sau sử dụng biến con trỏ:

1. Viết hàm tìm số lớn nhất, nhỏ nhất trong một mảng n số nguyên.
2. Viết hàm sắp xếp tăng dần, giảm dần của một dãy số cho trước.
3. Viết hàm tách tên và họ lót từ một chuỗi cho trước.
4. Viết hàm cắt bỏ khoảng trắng thừa ở giữa, hai đầu một chuỗi.
5. Viết hàm chuyển đổi 1 chuỗi sang chữ thường và 1 hàm chuyển đổi sang chữ HOA.
6. Viết hàm chuyển đổi 1 chuỗi sang dạng Title Case (kí tự đầu của mỗi từ là chữ hoa, các kí tự còn lại chữ thường)
7. Viết chương trình nhập vào 1 chuỗi và in ra chuỗi đảo ngược.
Ví dụ: Nhập vào chuỗi "Lap trinh C can ban"
In ra "nab nac C hnirt paL"
8. Viết chương trình nhập vào một chuỗi ký tự rồi đếm xem trong chuỗi đó có bao nhiêu chữ 'th'.
9. Viết chương trình nhập vào một số có 3 chữ số. Cho biết dòng chữ mô tả giá trị con số đó. Ví dụ 305 -> ba trăm lẻ năm.
10. Viết chương trình nhập vào một chuỗi sau đó in ra màn hình mỗi dòng là một từ. Ví dụ chuỗi "Lap trinh C". Kết quả in ra
Lap
trinh
C
11. Viết chương trình nhập vào một chuỗi các ký tự, ký số, khoảng trắng và dấu chấm câu. Cho biết chuỗi trên gồm bao nhiêu từ.

Chương 10: CÁC KIỂU DỮ LIỆU TỰ ĐỊNH NGHĨA

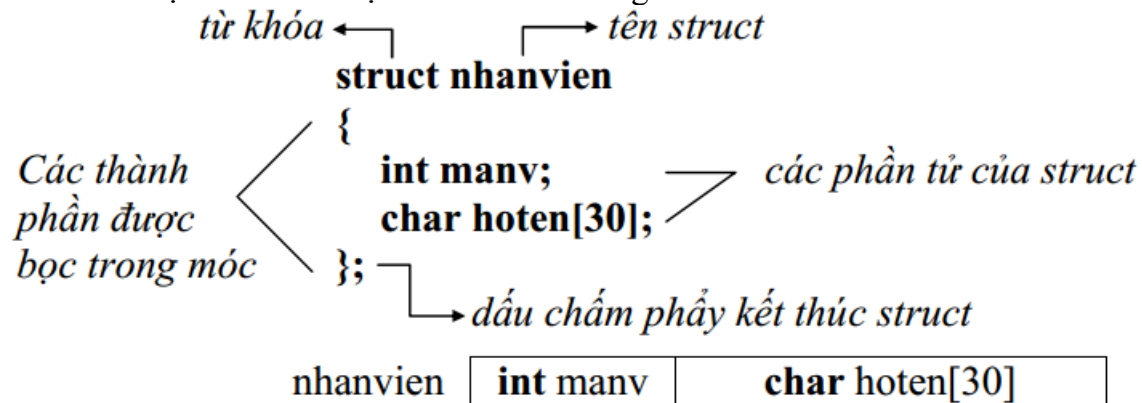
Cung cấp cho người đọc các kiến thức kỹ năng cơ bản sau: Ý nghĩa, cách khai structure, enum; nhập, xuất structure; khởi tạo structure, enum; một số kỹ thuật thao tác trên structure, enum; dùng struct làm tham số cho hàm.

10.1. Kiểu structure (kiểu cấu trúc)

Đối với mảng, chỉ có thể lưu nhiều thông tin có cùng kiểu dữ liệu. Nhưng với structure ta có thể lưu thông tin như một mảng có nhiều kiểu dữ liệu khác nhau.

10.1.1. Khai báo kiểu structure

Ví dụ: Khai báo một structure về thông tin nhân viên



Ví dụ trên định nghĩa kiểu dữ liệu mới có tên là struct nhanvien. Mỗi biến kiểu này gồm 2 phần tử: Biến nguyên có tên là manv và biến chuỗi có tên hoten.

Lưu ý: Từ khóa struct phải viết bằng chữ thường

10.1.2. Cách khai báo biến có kiểu structure

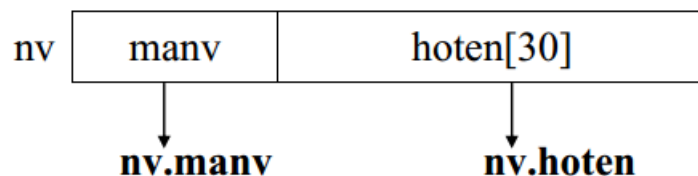
Ví dụ: struct nhanvien nv; hoặc nhanvien nv;

Khai báo biến nv có kiểu struct nhanvien

Lưu ý: Có thể vừa tạo structure nhanvien vừa khai báo biến nv

```
struct nhanvien
{
    int manv;
    char hoten[30];
} nv;
```

10.1.3. Tham chiếu các phần tử trong structure



Để tham chiếu đến manv trong nv ta viết như sau: nv.manv (là biến có kiểu int)

Lưu ý: Đối với biến khai báo kiểu con trỏ nhanvien *nv thì tham chiếu đến phần tử manv cần biểu diễn như sau:

nv -> manv

Ví dụ: Nhập và in danh sách nhân viên.

```
/* Danh sach nhan vien */
```

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

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

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

```
#define MAX 50
```

```
void main(void)
```

```
{
```

```
    struct nhanvien
```

```
    {
```

```
        int manv;
```

```
        char hoten[30];
```

```
    };
```

```
    struct nhanvien snv[MAX];
```

```
    char ctam[10];
```

```
    int i, in;
```

```
    printf("Nhap vao so nhan vien: ");
```

```
    gets(ctam);
```

```
    in = atoi(ctam); //Ham atoi dung de chuyen mot chuoi sang so (kieu int)
```

```
    //Nhap danh sach nhan vien
```

```
    for(i = 0; i < in; i++)
```

```
    {
```

```
        printf("Nhap vao ma nhan vien thu %d: ", i + 1);
```

```
        gets(ctam);
```

```
        snv[i].manv = atoi(ctam);
```

```
        printf("Nhap vao ho ten: ");
```

```
        gets(snv[i].hoten);
```

```
    }
```

```
    //in danh sach nhan vien
```

```
    for(i = 0; i < in; i++)
```

```

        printf("%5d  %s\n", snv[10].manv, snv[10].hoten);
    getch();
}

```

Kết quả in ra màn hình

```

Nhập vào số nhân viên: 2
Nhập vào mã nhân viên thu 1: 123
Nhập vào họ tên: Le Thuy Doan Trang
Nhập vào mã nhân viên thu 2: 35
Nhập vào họ tên: Le Nguyen Tuan Anh
123 Le Thuy Doan Trang
35 Le Nguyen Tuan Anh

```

10.1.4. Khởi tạo structure

Ví dụ: Nhập vào bảng số xe, cho biết xe đó đăng kí ở tỉnh nào

```

/* Xác định biên số xe */

```

```

#include <stdio.h>

```

```

#include <conio.h>

```

```

#include <stdlib.h>

```

```

#define MAX 6

```

```

void main(void)

```

```

{

```

```

    struct tinh

```

```

    {

```

```

        int ma;

```

```

        char *ten;

```

```

    };

```

```

    struct tinh sds[MAX] = {{60, "Dong Nai"}, {61, "Binh Duong"}, {62, "Long An"},
    {63, "Tien Giang"}, {64, "Vinh Long"}, {65, "Can Tho"}};

```

```

    char ctam[10];

```

```

    int i, in;

```

```

    printf("Nhập vào biên số xe: ");

```

```

    gets(ctam);

```

```

    in = atoi(ctam);

```

```

    for(i = 0; i < MAX; i++)

```

```

        if (sds[i].ma == in)
            printf("Xe dang ki o tinh %s.\n", sds[i].ten);

    getch();
}

```

Kết quả in ra màn hình

Nhap vao bien so xe: 62F5-1152

Xe dang ki o tinh Long An

10.1.5. Structure lồng nhau

Ví dụ: Nhập và in danh sách nhân viên.

```

/* Danh sach nhan vien */
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define MAX 50
void main(void)
{
    struct giacanh
    {
        char vo_chong[30];
        char con;
    };
    struct nhanvien
    {
        int manv;
        char hoten[30];
        struct giacanh canhan;
    };
    struct nhanvien snv[MAX];
    char ctam[10];
    int i, in;
    printf("Nhap vao so nhan vien: ");
    gets(ctam);
    in = atoi(ctam);
    //Nhap danh sach nhan vien
}

```



```

for(i = 0; i < in; i++)
{
    printf("Nhap vao ma nhan vien thu %d: ", i + 1);
    gets(ctam);
    snv[i].manv = atoi(ctam);
    printf("Nhap vao ho ten: ");
    gets(snv[i].hoten);
    printf("Cho biet ten vo (hoac chong): ");
    gets(snv[i].canhan.vo_chong);
    printf("So con: ");
    gets(ctam);
    snv[i].canhan.con=atoi(ctam);
}
//in danh sach nhan vien
for(i = 0; i < in; i++)
    printf("Ma so: %d\nHo ten: %s\n Ho ten vo (hoac chong): %s\nSo con: %d", snv[i].manv, snv[i].hoten, snv[i].canhan.vo_chong, snv[i].canhan.con);
getch();
}

```

Kết quả in ra màn hình

```

Nhap vao so nhan vien: 2
Nhap vao ma nhan vien thu 1: 123
Nhap vao ho ten: Le Thuy Doan Trang
Nhap vao ma nhan vien thu 2: 35
Nhap vao ho ten: Le Nguyen Tuan Anh
123 Le Thuy Doan Trang
35 Le Nguyen Tuan Anh

```

10.2. Kiểu Enum (kiểu liệt kê)

Một biến là kiểu dữ liệu enum thì biến đó có thể nhận được một giá trị nào đó trong các giá trị được liệt kê.

10.2.1. Định nghĩa kiểu enum

```

<enum> <tên enum>
{
    giá trị 1 = giá trị gì đó,
    giá trị 2 = giá trị gì đó,
    giá trị 3 = giá trị gì đó,
    ...
};

```

Ví dụ 1: Định nghĩa kiểu enum day

```

enum day { SUN, MON, TUE, WED, THU, FRI, SAT };

```

Annotations in the diagram:
 - từ khóa (keyword) points to **enum**
 - tên (name) points to **day**
 - các giá trị liệt kê (listed values) points to { SUN, MON, TUE, WED, THU, FRI, SAT }
 - dấu ; kết thúc enum (semicolon, ends enum) points to ;

⇒ Các tên thứ (SUN, MON ... SAT) trong day sẽ được đánh số lần lượt từ 0 đến 6 (SUN là 0, MON là 1... SAT là 6). Có thể đánh lại thứ tự như ví dụ sau:

Ví dụ 2:

```

enum NgayTrongTuan
{
    ThuHai = 2,
    ThuBa = 3,
    ThuTu = 4,
    ThuNam = 5,
    ThuSau = 6,
    ThuBay = 7,
    ChuNhat = 8,
};

```

Từ khóa **enum** phải viết bằng chữ thường

10.2.2. Cách khai báo biến có kiểu enum

Ví dụ 1: *enum day ngay;* (hoặc *day ngay;*)

(Đây là khai báo biến ngay có kiểu enum day)

Lưu ý: Có thể vừa tạo enum day vừa khai báo biến ngay enum day { SUN, MON, TUE, WED, THU, FRI, SAT } ngay;

Ví dụ 2:

```

enum color

```

```

{
    black,
    blue,
    green,
    cyan,
    red,
    purple,
    yellow,
    white,
}

color myColor;
myColor = blue;
if (myColor == green) myColor=red;

```

10.2.3 Sử dụng enum trong chương trình

Ví dụ: Sử dụng enum để in ra thứ ngày trong tuần.

```

#include <stdio.h>
enum eDayOfWeek
{
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
};

void PrintScreen(enum eDayOfWeek day)
{
    switch (day)
    {
        case MONDAY :
            printf(" Today is Monday");
            break;
        case TUESDAY:

```

```

        printf("Today is Tuesday");
        break;
case WEDNESDAY:
    printf("Today is Wednesday");
    break;
case THURSDAY:
    printf("Today is Thursday");
    break;
case FRIDAY:
    printf("Today is Friday");
    break;
case SATURDAY:
    printf("Today is Saturday");
    break;
case SUNDAY:
    printf("Today is Sunday");
    break;
default:
    break;
}
}
int main()
{
    enum eDayOfWeek nowDay = SATURDAY;
    PrintScreen(nowDay);
    return 0;
}

```

Bài tập chương 10

1. Định nghĩa 1 cấu trúc có thể được dùng làm danh bạ điện thoại gồm có tên, địa chỉ, số điện thoại, với số mẫu tin tối đa là 40. Viết chương trình với các chức năng sau: nhập thông mới, tìm kiếm số điện thoại, in danh sách theo quận (địa chỉ).
2. Viết chương trình đọc vào tên, địa chỉ, sắp xếp tên và địa chỉ theo thứ tự alphabet, sau đó hiển thị danh sách đã được sắp xếp.
3. Viết chương trình nhập vào các thông tin sau: Tên đội bóng, số trận thắng, số trận hòa, số trận thua. In ra đội bóng có số điểm cao nhất (với 1 trận thắng = 3 điểm, 1 trận hòa = 1 điểm và 1 trận thua = 0 điểm).
4. Xây dựng cấu trúc gồm: Họ tên, ngày sinh, tên trường, số báo danh, điểm thi. Trong đó, điểm thi là cấu trúc gồm 3 môn: Toán, Lý, Hóa. Nhập liệu vào khoảng 10 thí sinh, tìm và in ra các thí sinh có tổng điểm 3 môn ≥ 15 .
5. Viết chương trình tạo lập và tìm kiếm dữ liệu. Nội dung yêu cầu gồm: Nhập họ và tên, địa chỉ (gồm: Quận, phường, tổ), tuổi, lương. Tìm kiếm những người ở Quận 3 có tuổi dưới 30 thu nhập từ 500.000đ trở lên và in ra màn hình.

Chương 11: MỘT SỐ BÀI TẬP VẬN DỤNG CÓ LỜI GIẢI

Trình bày một số ví dụ phổ biến thường gặp trong lập trình C, giúp người đọc tăng cường khả năng ứng dụng những nội dung lý thuyết đã được học vào thực hành.

11.1. Viết chương trình nhập vào 2 số nguyên, sau đó in ra tổng bình phương của chúng

Hướng dẫn:

- Nhập vào 2 số nguyên từ bàn phím.
- Tính tổng theo công thức: $S = a*a + b*b$

Mã nguồn:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a, b;
    long S = 0;
    printf("\n Nhập vào số nguyên thứ nhất = ");
    scanf("%d", &a);
    printf("\n Nhập vào số nguyên thứ hai = ");
    scanf("%d", &b);
    S = a*a + b*b;
    printf("\n Tổng bình phương 2 số = %d^2 + %d^2 = %d ", a, b, S);
    getch();
}
```

11.2. Nhập số tự nhiên n rồi tính tổng (lưu ý phép chia các số nguyên)

$$S = 1 + 1/2 + 1/3 + 1/4 + \dots 1/n \quad (n > 0)$$

Hướng dẫn:

- Nhập vào số nguyên dương n ($n > 0$)
- Dùng vòng lặp for() để tính tổng S

Mã nguồn:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n,i;
```

```

float S = 0;
do
{
    printf("\nNhập n > 0: ");
    scanf("%d",&n);
}
while (n <= 0);
for(i = 1; i <= n; i++)
{
    S += 1/(float)i;
}
printf("\nS = %f", S);
getch();
}

```

11.3. Tính tổng $S(n) = 1+2+3+4+ \dots +n$

Hướng dẫn:

- Nhập vào số nguyên dương n
- Dùng vòng lặp for, do while để duyệt biến i từ 1 tới n
- Tính tổng: $s = s + i$

Mã nguồn:

```

#include <stdio.h>
#include <conio.h>
int main()
{
    int i, n, s = 0;
    do
    {
        printf("n = ");
        scanf("%d", &n);
    }while(n <= 0);

    for(i = 1; i <= n; i++)
    {
        s += i;
    }
}

```

```

printf("\nS(%d) = %d", n, s);
getch();
return 0;
}

```

11.4. Tính tổng $S(n) = 1^2 + 2^2 + \dots + n^2$

Hướng dẫn:

- Dùng vòng lặp for, do while để duyệt biến i từ 1 tới n
- Tính tổng: $s = s + i*i$

Mã nguồn:

```

#include <stdio.h>
#include <conio.h>
int main()
{
    printf("S(n) = 1^2 + 2^2 + ... + n^2\n");
    int i, n, s = 0;
    do
    {
        printf("n = ");
        scanf("%d", &n);
    }while(n <= 0);

    for(i = 1; i <= n; i++)
    {
        s += i*i;
    }
    printf("\nS(%d) = %d", n, s);
    getch();
    return 0;
}

```

11.5. Tính tổng $S(n) = 1/2 + 1/4 + 1/6 + 1/8 + 1/2n$

Hướng dẫn:

- Dùng vòng lặp for, do while để duyệt biến i từ 1 tới n
- Tính tổng: $s = s + 1/(2*i)$

Mã nguồn:

```

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

```



```

int main()
{
    printf("S(n) = 1/2 + 1/4 + 1/6 + 1/8 + 1/2n\n");
    int i, n;
    float s = 0.0;
    do
    {
        printf("n = ");
        scanf("%d", &n);
    }while(n <= 0);
    for(i = 1; i <= n; i++)
    {
        s += 1.0/(2*i);
    }
    printf("\nS(%d) = %f", n, s);
    getch();
    return 0;
}

```

11.6. Tính tổng $S(n) = 1 + 1/3 + 1/5 + 1/(2n+1)$

Hướng dẫn:

- Dùng vòng lặp for, do while để duyệt biến i từ 1 tới n
- Tính tổng: $s = s + 1/(2*i + 1)$

Mã nguồn:

```

#include <stdio.h>
#include <conio.h>
int main()
{
    printf("S(n) = 1 + 1/3 + 1/5 + 1/7 + ... + 1/(2n+1)\n");
    int i, n;
    float s = 1.0;
    do
    {
        printf("n = ");
        scanf("%d", &n);
    }

```

```

    }while(n < 0);
    for(i = 1; i <= n; i++)
    {
        s += 1.0/(2*i + 1);
    }
    printf("\nS(%d) = %f", n, s);
    getch();
    return 0;
}

```

11.7. Tính tổng $S(n) = 1/(1*2) + 1/(2*3) + \dots + 1/(n*(n+1))$

Hướng dẫn:

- Dùng vòng lặp for, do while để duyệt biến i từ 1 tới n
- Tính tổng: $s = s + 1/(i*(i + 1))$

Mã nguồn:

```

#include <stdio.h>
#include <conio.h>
int main()
{
    printf("S(n) = 1/(1*2) + 1/(2*3) + ... + 1/(n*(n+1))\n");
    int i, n;
    float s = 0.0;
    do
    {
        printf("n = ");
        scanf("%d", &n);
    }while(n <= 0);
    for(i = 1; i <= n; i++)
    {
        s += 1.0/(i*(i + 1));
    }
    printf("\nS(%d) = %f", n, s);
    getch();
    return 0;
}

```

11.8. Tìm giá trị lớn nhất của 4 số a, b, c, d

Hướng dẫn:

- Nhập 4 số a, b, c, d từ bàn phím
- Tìm giá trị lớn nhất
- Xây dựng hàm max(float a, float b) để tìm giá trị lớn nhất của 2 số.

Mã nguồn:

```
#include <stdio.h>
#include <conio.h>
float max(float x, float y);
void main()
{
    float MAX;
    float a, b, c, d;
    printf("\nNhập a: ");
    scanf("%f", &a);
    printf("\nNhập b: ");
    scanf("%f", &b);
    printf("\nNhập c: ");
    scanf("%f", &c);
    printf("\nNhập d: ");
    scanf("%f", &d);
    MAX = max(max(a, b), max(c, d));
    printf("\nMAX(%f,%f,%f,%f) = %f", a, b, c, d, MAX);
    getch();
}
float max(float x, float y)
{
    float max;
    if(x > y)
        max = x;
    else
        max = y;
    return max;
}
```

11.9. Tìm ước số chung lớn nhất và bội số chung nhỏ nhất của 2 số a,b

Hướng dẫn:

- Dùng thuật toán Euclid để tìm USC lớn nhất

- BSC nhỏ nhất = $a*b$ / USCLN(a, b)

Mã nguồn:

```
#include "stdio.h"
#include "conio.h"
int uscln(int a, int b);
int bscnn(int a, int b);
void main()
{
    int a = 6, b = 18;
    printf("USCLN(%d,%d) = %d\n", a, b, uscln(a, b));
    printf("BSCNN(%d,%d) = %d\n", a, b, bscnn(a, b));
    getch();
}
// dung thuat toan Euclid
int uscln(int a, int b)
{
    if(a == 0 && b == 0)
        return -1;
    else
        return (b == 0)? a : uscln(b, a%b);
}
int bscnn(int a, int b)
{
    if(a == 0 || b == 0)
        return -1;
    else
        return (a*b/uscln(a, b));
}
```

11.10. Tách các chữ số thuộc hàng trăm, hàng chục, hàng đơn vị

Yêu cầu:

Viết chương trình nhập vào số nguyên 3 chữ số (từ 100 đến 999), sau đó in ra các chữ số thuộc hàng trăm, hàng chục, hàng đơn vị.

Hướng dẫn:

- Nhập vào số nguyên ($100 \leq x \leq 999$).

- Số hàng trăm = $x / 100$, số hàng chục = $(x \% 100) / 10$, số hàng đơn vị = $(x \% 100) \% 10$

Mã nguồn:

```
#include "stdio.h"
#include "conio.h"
void main()
{
    int x;
    do
    {
        printf("\n Nhập vào số x = ");
        scanf("%d", &x);
    }
    while(x < 100 || x > 999 );
    printf("\n Chu số hàng trăm của %d là : %d", x, x/100);
    printf("\n Chu số hàng chục của %d là : %d", x, (x%100)/10);
    printf("\n Chu số hàng đơn vị của %d là : %d", x, (x%100)%10);
    getch();
}
```

11.11. Chương trình kiểm tra xem số N có phải là số nguyên tố

Hướng dẫn:

- Số nguyên tố là số chỉ chia hết cho chính nó và 1, ví dụ: 1, 3, 5, 7, 11, 13,... là số nguyên tố

- Nếu $(N \% i == 0)$ với mọi $i = (2 \text{ đến } N/2)$ thì N là số nguyên tố.

Mã nguồn:

```
#include "stdio.h"
#include "conio.h"
int so_nguyen_to(int N);
void main()
{
    int N;
    int ketqua;
    do
    {
        printf("\n Nhập vào số N = ");
```

```

    scanf("%d", &N);
}
while(N <= 0);
if(so_nguyen_to(N))
    printf("\n %d la so nguyen to", N);
else
    printf("\n %d la khong phai so nguyen to", N);
getch();
}
// Hamkiem tra so nguyen to
int so_nguyen_to(int N)
{
    int i;
    if(N == 1)
        return 1;
    else
    {
        for(i = 2; i < N/2; i++)
        {
            if(N % i == 0)
                return 0;
        }
        return 1;
    }
}
}

```

11.12. Giải hệ phương trình bậc nhất

Yêu cầu:

Viết chương trình giải hệ phương trình bậc nhất:

$$- ax + by = c$$

$$- dx + ey = f$$

Hướng dẫn:

Nghiệm của hệ: $x = Dx/D$, $y = Dy/D$

(Trong đó: $D = ae - bd$, $Dx = ce - bf$, $Dy = af - cd$)

Mã nguồn:

```
#include "stdio.h"
```

```

#include "conio.h"
void main()
{
    float a, b, c, d, e, f;
    float Dx, Dy, D;
    printf("\n Nhap vao cac he so a, b, c, d, e, f: \n");
    scanf("%f%f%f%f%f%f", &a, &b, &c, &d, &e, &f);
    D = a*e - b*d;
    Dx = c*e - b*f;
    Dy = a*f - c*d;
    printf("\n%fx + %fy = %f", a, b, c);
    printf("\n%fx + %fy = %f", d, e, f);
    if(D == 0 && Dx == 0)
        printf("\n He pt vo so nghiem");
    if(D == 0 && Dx != 0)
        printf("\n He pt vo nghiem");
    if(D != 0)
    {
        printf("\n He co nghiem");
        printf("\n x = %.3f", Dx/D);
        printf("\n y = %.3f", Dy/D);
    }
    getch();
}

```

11.13. Chuyển số nhị phân sang thập phân

Hướng dẫn:

Để chuyển từ số nhị phân sang thập phân. Chúng ta dùng công thức sau:

$$b_n b_{n-1} \dots b_0 = b_n \cdot 2^n + b_{n-1} \cdot 2^{n-1} + \dots + b_0 \cdot 2^0$$

Mã nguồn:

```

#include "stdio.h"
#include "conio.h"
void main()
{
    int num, binary_val, decimal_val = 0, base = 1, rem;
    printf("\n Nhap so nhi phan(1 & 0): ");
}

```

```

scanf("%d", &num); /* maximum five digits */
binary_val = num;
while (num > 0)
{
    rem = num % 10;
    decimal_val = decimal_val + rem * base;
    num = num / 10;
    base = base * 2;
}
printf("So nhi phan = %d \n", binary_val);
printf("Gia tri he thap phan = %d \n", decimal_val);
getch();
}

```

11.14. Đổi hệ thập phân sang nhị phân

Yêu cầu:

Đổi một số từ hệ thập phân sang nhị phân, ví dụ: 15 (hệ thập phân) = 1111 (hệ nhị phân)

Hướng dẫn:

Lấy số hệ thập phân chia cho 2 để tìm phần dư. Phép toán này được thực hiện liên tục tới khi nào phép chia bằng 0.

Mã nguồn:

```

#include "stdio.h"
#include "conio.h"
void main()
{
    unsigned long num, binary_val = 0, decimal_val, base = 1, rem;

    printf("\nNhap so nguyen he thap phan: ");
    scanf("%ld", &num); /* maximum five digits */
    decimal_val = num;
    while (num > 0)
    {
        rem = num % 2;
        binary_val = binary_val + rem * base;
        num = num / 2;
        base = base * 10;
    }
}

```



```

    }
    printf("So thập phân = %d \n", decimal_val);
    printf("So nhị phân = %d \n", binary_val);
    getch();
}

```

11.15. Xây dựng hàm kiểm tra năm nhuận

Hướng dẫn:

Năm nhuận là năm chia hết cho 400 hoặc chia hết cho 4 nhưng không chia hết cho 100.

Mã nguồn:

```

#include "stdio.h"
#include "conio.h"
#define IS_LEAP_YEAR(year) ((year%400 == 0) || (year%4 == 0 && year%100 != 0))?
    true : false
void main()
{
    int year;
    printf("\nNhập vào năm: ");
    scanf("%d", &year);
    if(IS_LEAP_YEAR(year))
        printf("%d là năm nhuận", year);
    else
        printf("%d không phải là năm nhuận", year);
    getch();
}

```

11.16. Xây dựng hàm đảo ngược các chữ số của một số nguyên

Hướng dẫn:

Dùng phép chia hết / và phép chia có dư % để tách từng chữ số ra.

Mã nguồn:

```

#include "stdio.h"
#include "conio.h"
void main()
{
    int n, reverse = 0, rem;
    printf("Nhập vào số nguyên: ");
    scanf("%d", &n);
    while(n > 0)

```

```

{
    rem = n % 10;
    reverse = reverse*10 + rem;
    n /= 10;
}
printf("So dao nguoc = %d", reverse);
getch();
}

```

11.17. Kiểm tra số có đối xứng

Ví dụ về số đối xứng: 123321 là số đối xứng, 12012 không phải số đối xứng.

Hướng dẫn:

Đảo các chữ số của số đầu vào. Nếu giá trị của số trước và sau khi đảo bằng nhau thì đó là số đối xứng.

Mã nguồn:

```

#include "stdio.h"
#include "conio.h"
void main()
{
    int n, reverse = 0, rem, temp;
    printf("Nhap vao so nguyen: ");
    scanf("%d", &n);
    temp = n;
    while(temp != 0)
    {
        rem = temp % 10;
        reverse = reverse*10 + rem;
        temp /= 10;
    }
    if(reverse == n)
        printf("%d la so doi xung", n);
    else
        printf("%d khong doi xung", n);
    getch();
}

```

11.18. Hướng dẫn đọc số tiếng anh

Yêu cầu:

Viết chương trình hướng dẫn đọc số tiếng anh

- Nhập vào 1 số có 3 chữ số
- Hiện thị cách đọc ra màn hình

Ví dụ: 201 -> two zero one

Hướng dẫn:

- Dùng switch case hoặc if/else

Mã nguồn:

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

const char *number_table[] = {"zero", "one", "two", "three", "four", "five", "six", "seven",
"eight", "nine", "ten"};

int main()
{
    int num;
    do
    {
        printf("\nnum = ");
        scanf("%d", &num);
    }while(num < 100 || num > 999);
    printf("%s    %s    %s",    number_table[num/100],number_table[num/10%10],
number_table[num%10]);

    getch();
    return 0;
}
```

11.19. Chương trình tính tiền cước TAXI

Yêu cầu:

Viết chương trình tính tiền cước TAXI. Biết rằng:

- Km đầu tiên là 11000đ
- 30Km tiếp theo là 10000đ
- Nếu lớn hơn 30Km thì mỗi Km thêm ra sẽ phải trả là 9000đ
- Hãy nhập số Km sau đó in ra số tiền phải trả.

Hướng dẫn:

- Dùng lệnh if để chia các mức giá khác nhau.

Mã nguồn:

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

```

#include <conio.h>
void main()
{
    float TotalKm;
    float Cost = 0;
    printf("\n Nhap vao so Km: ");
    scanf("%f", &TotalKm);
    if(TotalKm <= 1)
    {
        Cost = TotalKm * 11000;
    }
    else if(TotalKm <= 30 && TotalKm > 1)
    {
        Cost = (TotalKm - 1)*10000 + 1*11000;
    }
    else
    {
        Cost = 1*11000 + 29*10000 + (TotalKm - 30)*9000;
    }
    printf("\n So tien phai tra: %0.3f VND", Cost);
    getch();
}

```

11.20. Tìm số hoàn hảo

Yêu cầu:

Một số hoàn hảo là một số có tổng các ước số của nó bằng chính số đó. Hãy tìm số hoàn hảo nhỏ hơn 5000. Ví dụ: 6 có các ước số là 1, 2, 3 và $6 = 1 + 2 + 3$.

Hướng dẫn:

- Tìm các ước số của số đó. Sau đó tính tổng các ước số.
- So sánh tổng các ước số với số đó. In kết quả ra màn hình.

Mã nguồn:

```

#include "stdio.h"
#include "conio.h"
void main()
{
    int N;

```

```

    int i, j, sum;
    do
    {
        printf("\n Nhập vào số N = ");
        scanf("%d", &N);
    }
    while(N <= 0);
    for(i = 2; i <= N; i++)
    {
        sum = 1;
        for(j = 2; j <= i/2; j++)
        {
            if(i%j == 0)
                sum += j;
        }
        if(sum == i)
            printf("\n %d", i);
    }
    getch();
}

```

11.21. Tính sin(x)

Yêu cầu:

Lập chương trình tính sin(x) với độ chính xác 0.0001 theo công thức:
 $\sin(x) = x - x^3/3! + x^5/5! - \dots + (-1)^n \cdot x^{(2n+1)}/(2n+1)!$

Hướng dẫn:

Sử dụng while để lặp với điều kiện dừng là độ chính xác ≤ 0.0001 .

Mã nguồn:

```

#include "stdio.h"
#include "conio.h"
#include "math.h"
#define E 0.0001
#define PI 3.141592654
long giaiithua(int N);
void main()
{

```

```

float angle; // goc
float e = 1; // sai so
int n = 0;
float sinx = 0;
printf("\n Nhap vao so do goc (do) = ");
scanf("%f", &angle);
while(e > E)
{
    e = pow(float(angle*PI/180), (2*n+1))/giaithua(2*n+1); // tinh sai so
    if(n%2 == 0) // n chan
    {
        sinx = sinx + e;
    }
    else // n le
    {
        sinx = sinx - e;
    }
    n++;
}
printf("\n sin(%f) = %f", angle, sinx);
getch();
}

long giaithua(int N)
{
    if(N == 0 || N == 1)
        return 1;
    else
        return N*giaithua(N-1);
}

```

11.22. Tính cos(x)

Yêu cầu:

- Nhập một số $c > 0$ (ví dụ $c = 0.0001$) và một số thực x rồi tính cos theo công thức:

$$\cos(x) = 1 - x^2/2! + x^4/4! - \dots + (-1)^n x^{(2n)}/(2n)!$$

- Tổng được tính với n đủ lớn sao cho bất đẳng thức:

$$|x^{(2n)}/(2n)!| \leq c \text{ thỏa mãn}$$

Hướng dẫn:

Dùng vòng lặp for để tính biểu thức $\cos(x)$.

Mã nguồn:

```
#include "stdio.h"
#include "conio.h"
#include "math.h"
#define E 0.0001
#define PI 3.141592654
long giaithua(int N);
void main()
{
    float angle; // goc
    float e = 1; // sai so
    int n = 0;
    float cosx = 0;
    printf("\n Nhap vao so do goc (do) = ");
    scanf("%f", &angle);
    while(e > E)
    {
        e = pow(float(angle*PI/180), (2*n))/giaithua(2*n); // tinh sai so
        printf("\n e = %f", e);
        if(n%2 == 0)
        {
            cosx = cosx + e;
        }
        else
        {
            cosx = cosx - e;
        }
        n++;
    }
    printf("\n cos(%f) = %f", angle, cosx);
    getch();
}
long giaithua(int N)
```

```

{
    if(N == 0 || N == 1)
        return 1;
    else
        return N*giaithua(N-1);
}

```

11.23. Tính e^x

Yêu cầu:

Nhập 1 số $c > 0$ (sai số) và 1 số thực x rồi tính: $e^x = 1 + x/1! + x^2/2! + \dots + x^n/n!$

Tổng được tính với n đủ lớn sao cho $|x^n/n!| < c$ thỏa mãn.

Hướng dẫn:

Tương tự như tính $\sin x$, $\cos x$, chúng ta sử dụng vòng lặp `for()` để giải quyết bài toán này.

Mã nguồn:

```

#include "stdio.h"
#include "conio.h"
#include "math.h"
long giaithua(int N);
void main()
{
    float x; // goc
    float c; // sai so
    float e = 1;
    float exp = 0;
    int n = 0;
    printf("\n Nhap x = ");
    scanf("%f", &x);
    printf("\n Nhap sai so = ");
    scanf("%f", &c);
    while(e >= c)
    {
        e = pow(float(x),n)/giaithua(n); // tinh sai so
        exp += e;
        n++;
    }
    printf("\n e^%f = %f", x, exp);
}

```



```

    getch();
}
long giaithua(int N)
{
    if(N == 0 || N == 1)
        return 1;
    else
        return N*giaithua(N-1);
}

```

11.24. Bài tập sử dụng mảng hai chiều

Yêu cầu:

Viết chương trình nhập vào ngày, tháng năm và cho biết số thứ tự của ngày đó trong năm.

Hướng dẫn:

Sử dụng mảng hai chiều và cấu trúc tự định nghĩa

Mã nguồn:

```

#include <conio.h>
#include <stdio.h>
int ngay[2][12]={ {31,28,31,30,31,30,31,31,30,31,30,31},
                  {31,29,31,30,31,30,31,31,30,31,30,31}};
struct date
{
    int ngay;
    int thang;
    int nam;
};
int ngaynam(struct date *p);
int isdate(struct date *p);
void main()
{
    struct date d;
    printf("\nNhap ngay thang nam :");
    scanf("%d%d%d",&d.ngay,&d.thang,&d.nam);
    if (isdate(&d))
        printf("\nDo la ngay thu : %d ",ngaynam(&d));
    else

```

```

    printf("\nNhap sai ngay!");
    getch();
}

int isdate(struct date *p)
{
    int m,n,k;
    m=p->ngay;
    n=p->thang;
    k=p->nam;
    if (n==1 || n==3 || n==5 || n==7 || n==8 || n==10 ||
        n==12)
        return ((m<=31) && (m>0));
    else if (n==4 || n==6 || n==9 || n==11)
        return ((m<=30) && (m>0));
    else if ((n==2) && (k % 4 == 0 && k % 100 != 0 || k % 400 == 0))
        return ((m<=29) && (m>0));
    else
        return ((m<=28) && (m>0));
}

int ngaynam(struct date *p)
{
    int i,j,k,s;
    s=p->ngay;
    k=p->nam;
    j=((k % 4 == 0 && k % 100 != 0) || k % 400 == 0);
    for (i=0;i<p->thang-1;i++)
        s+=ngay[j][i];
    return (s);
}

```