

MỤC LỤC



CHƯƠNG 1	TỔNG QUAN VỀ LẬP TRÌNH.....	1
I.	Chương trình máy tính.....	1
I. 1.	Các bước viết chương trình.....	1
I. 2.	Ngôn ngữ lập trình	2
I. 3.	Giải thuật.....	2
I. 3. 1.	Khái niệm	2
I. 3. 2.	Đặc tính cần có của giải thuật	3
I. 3. 3.	Các hình thức mô tả giải thuật.....	3
II.	Cấu trúc dữ liệu.....	3
II. 1.	Khái niệm	3
II. 2.	Kiểu dữ liệu cơ sở.....	4
II. 3.	Kiểu dữ liệu có cấu trúc	4
III.	Lưu đồ giải thuật	4
III. 1.	Các ký hiệu cơ bản	4
III. 2.	Mô tả các cấu trúc điều khiển cơ bản	5
III. 2. 1.	Cấu trúc tuần tự	5
III. 2. 2.	Cấu trúc lựa chọn.....	6
(a)	Chỉ xét trường hợp khi điều kiện đúng.....	6
(b)	Xét cả hai trường hợp đúng hoặc sai	7
(c)	Xét nhiều trường hợp.....	8
III. 2. 3.	Cấu trúc lặp.....	9
(a)	Kiểm tra điều kiện trước khi lặp	9
(b)	Thực hiện lặp trước khi kiểm tra điều kiện	10
III. 3.	Một số ví dụ	11
IV.	Kết luận	14
V.	Bài tập vẽ lưu đồ giải thuật.....	14
V. 1.	Bài tập cơ bản	14
V. 2.	Bài tập luyện tập và nâng cao.....	15

CHƯƠNG 2 GIỚI THIỆU NGÔN NGỮ C.....	16
I. Giới thiệu công cụ Microsoft Visual Studio	16
II. Tạo dự án mới (project)	16
III. Tạo tập tin chứa lệnh.....	18
IV. Chương trình mẫu	21
IV. 1. Chương trình 1	21
IV. 1. 1. Lệnh mẫu	21
IV. 1. 2. Giải thích lệnh	21
IV. 1. 3. Biên dịch và thực thi chương trình.....	23
(a) <i>Biên dịch chương trình</i>	23
(b) <i>Một số lỗi cơ bản gặp phải và cách sửa</i>	23
(c) <i>Thực thi chương trình</i>	25
IV. 2. Chương trình 2	25
IV. 2. 1. Lệnh mẫu	25
IV. 2. 2. Giải thích lệnh	26
IV. 2. 3. Kết quả chương trình.....	27
V. Mở dự án có sẵn.....	28
VI. Một số chương trình mẫu	28
VI. 1. Chương trình mẫu 1	29
VI. 2. Chương trình mẫu 2	29
VII. Chạy từng bước xem kết quả hoạt động của chương trình	30
VIII. Kết luận.....	37
IX. Bài tập.....	37
CHƯƠNG 3 CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ C	39
I. Lịch sử ngôn ngữ C.....	39
II. Các khái niệm	39
II. 1. Lệnh và khối lệnh	39
II. 2. Từ khóa (key word)	40
II. 3. Tập các ký hiệu và ký tự của C.....	40
II. 4. Các đặt tên trong chương trình	40

II. 5.	Tạo ghi chú (chú thích) trong chương trình.....	41
II. 6.	Hằng số (constant)	42
II. 6. 1.	Hằng số nguyên	42
II. 6. 2.	Hằng số thực.....	42
II. 6. 3.	Hằng ký tự	43
II. 6. 4.	Hằng chuỗi	43
II. 7.	Kiểu dữ liệu cơ bản.....	43
II. 7. 1.	Kiểu số nguyên	44
II. 7. 2.	Kiểu số thực.....	44
II. 8.	Biến và khai báo biến.....	45
II. 8. 1.	Biến.....	45
(a)	<i>Khái niệm</i>	45
(b)	<i>Cách đặt tên biến</i>	45
II. 8. 2.	Khai báo biến.....	45
(a)	<i>Cú pháp</i>	45
(b)	<i>Khai báo nhiều biến cùng một kiểu dữ liệu</i>	45
(c)	<i>Khai báo và gán giá trị ban đầu cho biến</i>	46
(d)	<i>Ép kiểu cho biến</i>	46
(e)	<i>Phạm vi của biến</i>	46
(f)	<i>Định nghĩa tên hằng số</i>	47
III.	Ký hiệu các phép toán	47
III. 1.	Phép toán số học	47
III. 2.	Phép toán so sánh và kết hợp so sánh.....	48
III. 3.	Phép toán trên bit	49
III. 4.	Toán tử điều kiện	49
III. 5.	Viết tắt phép toán	50
III. 6.	Thứ tự ưu tiên các phép toán	50
IV.	Hàm nhập xuất dữ liệu: printf và scanf.....	51
IV. 1.	Chuỗi định dạng.....	51
IV. 2.	Hàm xuất dữ liệu ra màn hình: printf	52

(a)	<i>Xuất chuỗi</i>	52
(b)	<i>Xuất chuỗi kèm giá trị biến</i>	52
(c)	<i>Xuất các ký tự đặc biệt</i>	52
IV. 3.	Hàm nhập dữ liệu: scanf	53
IV. 4.	Ví dụ hàm nhập xuất.....	53
V.	Các hàm cơ bản khác	54
VI.	Kết luận	54
VII.	Bài tập.....	55
CHƯƠNG 4 CẤU TRÚC ĐIỀU KHIỂN		57
I.	Cấu trúc cơ bản của chương trình C	57
II.	Cấu trúc điều khiển	57
II. 1.	Cấu trúc rẽ nhánh.....	57
II. 1. 1.	Cấu trúc if.....	57
II. 1. 2.	Cấu trúc if ... else.....	59
II. 1. 3.	Cấu trúc if ... else lồng nhau.....	60
II. 2.	Cấu trúc lựa chọn switch...case	61
II. 3.	Cấu trúc lặp.....	63
II. 3. 1.	Cấu trúc lặp for.....	63
II. 3. 2.	Cấu trúc lặp while.....	64
II. 3. 3.	Cấu trúc lặp do...while	65
II. 4.	Lệnh break và continue.....	66
II. 4. 1.	Lệnh break.....	66
II. 4. 2.	Lệnh continue	67
III.	Phương pháp kiểm tra từng bước để tìm kết quả chương trình	67
IV.	Kết luận	68
V.	Bài tập	69
V. 1.	Bài tập cơ bản	69
V. 2.	Bài tập luyện tập và nâng cao	73

CHƯƠNG 5 CHƯƠNG TRÌNH CON.....	75
I. Các khái niệm	75
I. 1. Khái niệm hàm con (function)	75
I. 2. Ví dụ.....	75
I. 2. 1. Chương trình không sử dụng hàm con	76
I. 2. 2. Chương trình có sử dụng hàm con	77
I. 3. Cấu trúc chương trình C sử dụng hàm con	78
I. 3. 1. Khởi khai báo	79
I. 3. 2. Hàm chính (main())	79
I. 3. 3. Các hàm con	79
I. 4. Cấu trúc của một hàm	79
I. 4. 1. Kiểu dữ liệu của hàm.....	79
I. 4. 2. Tham số	80
(a) Tham số là tham trị	81
(b) Tham số là tham chiếu.....	81
I. 4. 3. Tên hàm.....	81
I. 5. Gọi hàm.....	81
II. Phương pháp xác định nguyên mẫu hàm.....	82
III. Một số ví dụ.....	83
IV. Kết luận	86
V. Bài tập	87
V. 1. Bài tập cơ bản	87
V. 2. Bài tập luyện tập và nâng cao	88
CHƯƠNG 6 MẢNG MỘT CHIỀU	90
I. Các khái niệm	90
I. 1. Khái niệm	90
I. 2. Khai báo mảng	90
I. 3. Khai báo và gán giá trị cho mảng.....	91
I. 4. Truy xuất phần tử của mảng.....	92

II.	Một số thao tác cơ bản trên mảng số nguyên	93
II. 1.	Hàm nhập xuất mảng một chiều	93
II. 2.	Liệt kê (xuất) những phần tử thỏa điều kiện cho trước	94
II. 3.	Đếm số lượng phần tử trong mảng	96
II. 4.	Tìm kiếm và trả về vị trí phần tử có giá trị lớn nhất.....	97
II. 5.	Tìm vị trí phần tử có giá trị x.....	98
II. 6.	Kiểm tra xem mảng có thỏa điều kiện cho trước.....	98
II. 7.	Tính tổng có điều kiện	99
II. 8.	Tính giá trị trung bình có điều kiện	100
II. 9.	Sắp xếp mảng theo thứ tự tăng	101
II. 10.	Xoá phần tử trong mảng	102
II. 11.	Chèn một phần tử x vào mảng	102
II. 12.	Tách và ghép mảng.....	103
II. 12. 1.	Kỹ thuật tách cơ bản.....	103
II. 12. 2.	Kỹ thuật ghép cơ bản.....	103
III.	Kết luận	104
IV.	Bài tập.....	105
IV. 1.	Bài tập cơ bản	105
IV. 2.	Bài tập luyện tập và nâng cao	108
TÀI LIỆU THAM KHẢO		112

CHƯƠNG 1 TỔNG QUAN VỀ LẬP TRÌNH

Tóm tắt: Giới thiệu các khái niệm cơ bản về lập trình, các bước xây dựng chương trình, các ký hiệu biểu diễn lưu đồ giải thuật, cách biểu diễn các cấu trúc điều khiển rẽ nhánh, cấu trúc lặp và các kỹ thuật liên quan đến lưu đồ giải thuật.

I. Chương trình máy tính

Chương trình máy tính (còn gọi là phần mềm) là tập hợp các lệnh chỉ dẫn từng bước cho máy tính quá trình thực hiện được một công việc hoặc đạt được một kết quả cụ thể. Những lệnh này do các lập trình viên viết ra thông qua một ngôn ngữ lập trình cụ thể. Ví dụ như phần mềm xử lý văn bản, phần mềm xử lý ảnh, phần mềm kế toán, phần mềm quản lý nhân sự, v.v... Tùy theo chức năng của mỗi chương trình thì chương trình được phân ra nhiều loại: chương trình hệ thống, chương trình tiện ích, chương trình ứng dụng.

Theo định nghĩa của Niklaus Wirth thì:

Chương trình = Giải thuật + Cấu trúc dữ liệu

Chương trình và giải thuật đều dựa trên ba cấu trúc điều khiển cơ bản:

- (1) Tuần tự (*Sequential*):** Thực hiện tuần tự từ trên xuống, mỗi lệnh đều thực hiện đúng một lần.
- (2) Lựa chọn, rẽ nhánh (*Selection*):** Thực hiện một hoặc một số lệnh trong dãy các lệnh của nhiều trường hợp.
- (3) Lặp lại (*Repetition*):** Thực hiện lệnh một số lần.

I. 1. Các bước viết chương trình

Để đưa bài toán đơn giản ngoài thực tế trên bằng máy tính (lập trình cho máy tính giải) thì chúng ta cần phải thực hiện qua các bước như:

- Mô tả các bước giải bài toán.

- Chọn ngôn ngữ lập trình và lập trình để tạo thành một chương trình hoàn chỉnh.
- Thực thi chương trình: nhập vào dữ liệu và nhận kết quả.
- Kiểm thử chương trình: kiểm tra xem chương trình có thực thi đúng theo yêu cầu cho tất cả các trường hợp hay không.

I. 2. Ngôn ngữ lập trình

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.

Một chương trình được viết bằng một ngôn ngữ lập trình cụ thể (ví dụ Pascal, C, Java, v.v...) gọi là chương trình nguồn, chương trình dịch làm nhiệm vụ dịch chương trình nguồn thành chương trình thực thi được trên máy tính.

Nhằm hỗ trợ thuận tiện cho người lập trình, một số phần mềm lập trình tích hợp các công cụ thiết kế, viết chương trình nguồn, biên dịch và kiểm tra.

I. 3. Giải thuật

I. 3. 1. Khái niệm

Giải thuật là một hệ thống chặt chẽ và rõ ràng các quy tắc nhằm xác định một dãy các thao tác trên những dữ liệu vào sao cho sau một số hữu hạn bước thực hiện các thao tác đó ta thu được kết quả của bài toán.

Nói một cách dễ hiểu, giải thuật là mô tả quá trình thực hiện để chuyển dữ liệu nhập vào thành kết quả mong muốn. Khi mô tả giải thuật luôn kèm theo mô tả hai thành phần dữ liệu đầu vào và kết quả thu được sau khi thực hiện giải thuật.

Ví dụ: Giải thuật tìm số lớn nhất (max) của hai số nguyên a và b

- Đầu vào: Hai số nguyên a và b

- Kết quả: Số lớn nhất của hai số nguyên a và b, ký hiệu là max

- Giải thuật:

① Bước 1: Giả sử $max = a$

② Bước 2: Nếu $b > max$ thì

$max = b$

③ Bước 3: In ra số max

I. 3. 2. Đặc tính cần có của giải thuật

- **Tính kết thúc:** Giải thuật phải dừng sau một số hữu hạn bước.
- **Tính xác định:** Các thao tác máy tính phải thực hiện được và các máy tính khác nhau thực hiện cùng một bước của cùng một giải thuật phải cho cùng một kết quả.
- **Tính tổng quát:** Giải thuật phải "vết" hết các trường hợp và áp dụng cho một loạt bài toán cùng loại.
- **Tính hiệu quả:** Một giải thuật được đánh giá là tốt nếu nó đạt hai tiêu chuẩn thực hiện nhanh và sử dụng ít tài nguyên hệ thống.

I. 3. 3. Các hình thức mô tả giải thuật

Có nhiều hình thức để mô tả giải thuật như: mã giả (Pseudo code), ngôn ngữ tự nhiên, lưu đồ (flow chart). Mỗi hình thức mô tả đều có những ưu điểm đặc trưng phù hợp cho từng loại bài toán. Trong khuôn khổ nội dung giáo trình này sẽ trình bày hình thức mô tả giải thuật bằng lưu đồ do tính đơn giản và trực quan, nhất là đối với người mới bắt đầu học lập trình.

II. Cấu trúc dữ liệu

II. 1. Khái niệm

Các thông tin lưu trữ trong máy tính gọi là dữ liệu (*data*). Mỗi dữ liệu thuộc một kiểu dữ liệu nào đó.

Kiểu dữ liệu là một tập hợp các giá trị có cùng một tính chất và tập hợp các phép toán thao tác trên các giá trị đó. Kiểu dữ liệu được chia ra làm hai loại: Kiểu dữ liệu cơ sở và kiểu dữ liệu có cấu trúc.

Cấu trúc dữ liệu là cách thức tổ chức các dữ liệu thành một đơn vị hoàn chỉnh biểu diễn được thông tin cần quan tâm. Cấu trúc dữ liệu có thể hiểu là kiểu dữ liệu có cấu trúc, bao gồm các thành phần dữ liệu và các thao tác tác động lên cấu trúc dữ liệu đó.

II. 2. Kiểu dữ liệu cơ sở

Kiểu dữ liệu cơ sở là kiểu dữ liệu mà giá trị của nó là đơn nhất. Các ngôn ngữ lập trình đều xây dựng sẵn các kiểu dữ liệu này. Kiểu dữ liệu cơ sở được dùng để tạo các kiểu dữ liệu có cấu trúc.

Ví dụ: Trong ngôn ngữ lập trình C, kiểu `int` gọi là kiểu cơ sở vì kiểu này bao gồm các số nguyên từ -32.768 đến 32.767 và các phép toán +, -, *, /, %, v.v...

II. 3. Kiểu dữ liệu có cấu trúc

Kiểu dữ liệu có cấu trúc là kiểu dữ liệu mà các giá trị của nó là sự kết hợp của các giá trị khác.

Ví dụ: Kiểu chuỗi ký tự trong ngôn ngữ lập trình C là một kiểu dữ liệu có cấu trúc.



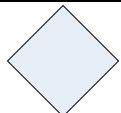
Các ngôn ngữ lập trình đều cho phép người lập trình định nghĩa kiểu dữ liệu có cấu trúc nhằm đáp ứng nhu cầu riêng của từng bài toán như danh sách liên kết, cấu trúc cây, v.v...

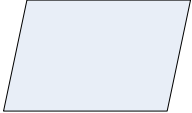


III. Lưu đồ giải thuật

Lưu đồ giải thuật là công cụ dùng để biểu diễn giải thuật một cách trực quan, việc mô tả dữ liệu đầu vào (*input*), kết quả (*output*) và luồng xử lý thông qua các ký hiệu hình học.

Phương pháp duyệt lưu đồ rất đơn giản: từ trên xuống và từ trái sang phải.

III. 1. Các ký hiệu cơ bản

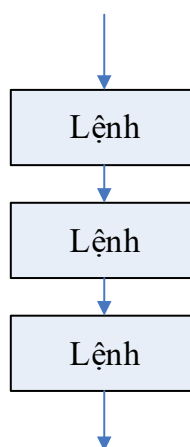
Stt	Ký hiệu	Ý nghĩa
1		Bắt đầu/ kết thúc
2		Hướng xử lý
3		Điều khiển lựa chọn/ rẽ nhánh

4		Nhập / xuất dữ liệu
5		Xử lý, tính toán hoặc gán
6		Điểm nối liên kết tiếp theo (Sử dụng khi lưu đồ vượt quá trang)

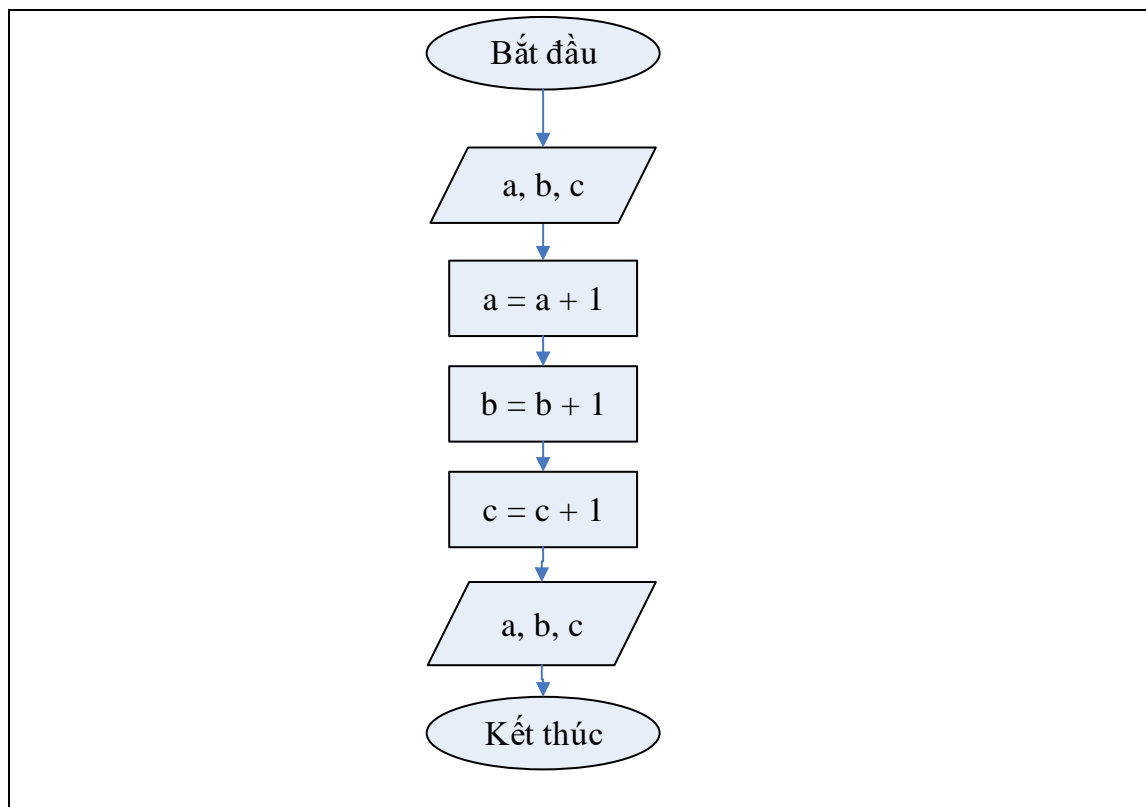
III. 2. Mô tả các cấu trúc điều khiển cơ bản

III. 2. 1. Cấu trúc tuần tự

Tuần tự thực thi tất cả các lệnh. Mỗi lệnh được thực thi một lần theo chuỗi từ trên xuống dưới, xong lệnh này rồi chuyển xuống lệnh kế tiếp.



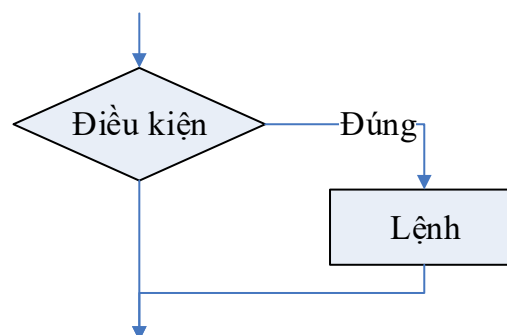
Ví dụ: Nhập vào 3 số nguyên a, b, c và xuất ra màn hình với giá trị của mỗi số tăng lên 1.



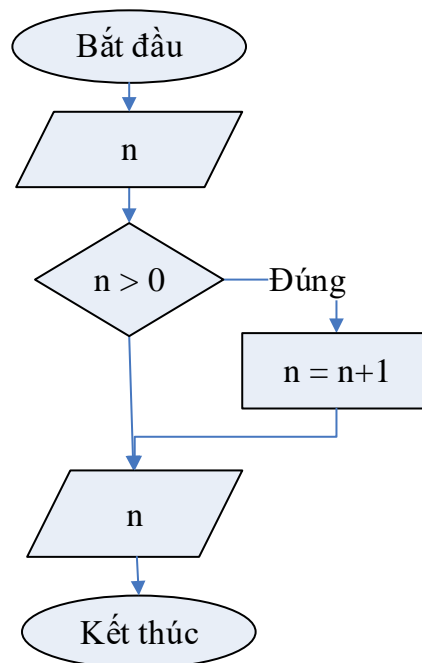
III. 2. 2. Cấu trúc lựa chọn

Điểm quyết định cho phép **chọn lệnh để thực hiện**. Tùy theo từng yêu cầu cụ thể của bài toán thì việc lựa chọn có thể gồm nhiều trường hợp: chỉ xét một trường hợp khi điều kiện đúng, xét cả hai trường hợp đúng hoặc sai, cũng có thể xét nhiều trường hợp.

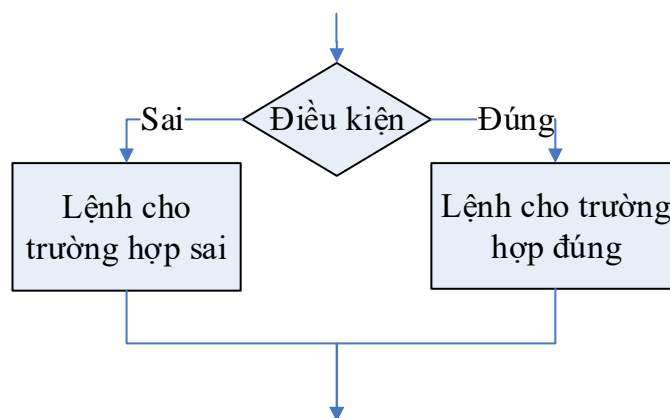
(a) Chỉ xét trường hợp khi điều kiện đúng



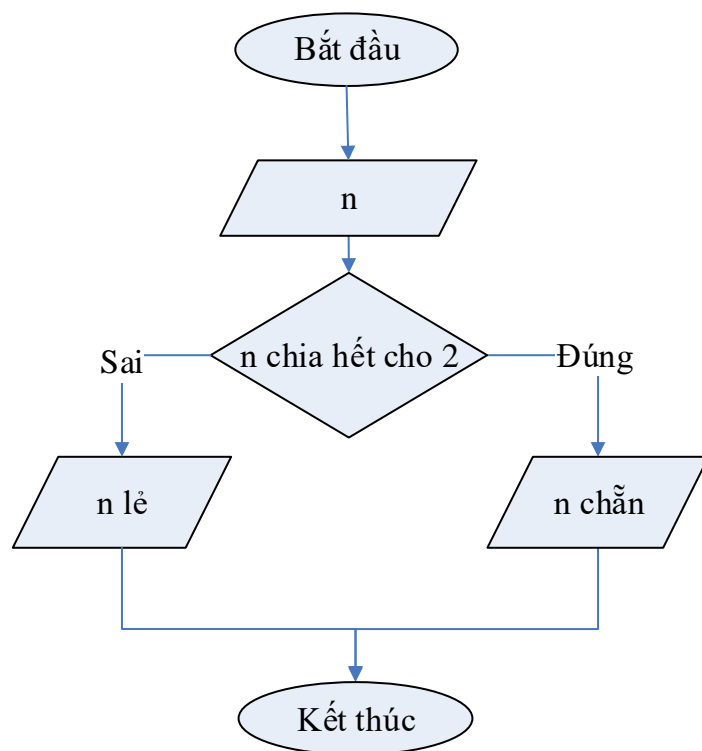
Ví dụ: Nhập vào số nguyên n . Kiểm tra nếu $n > 0$ thì tăng n lên 1 đơn vị. Xuất kết quả.



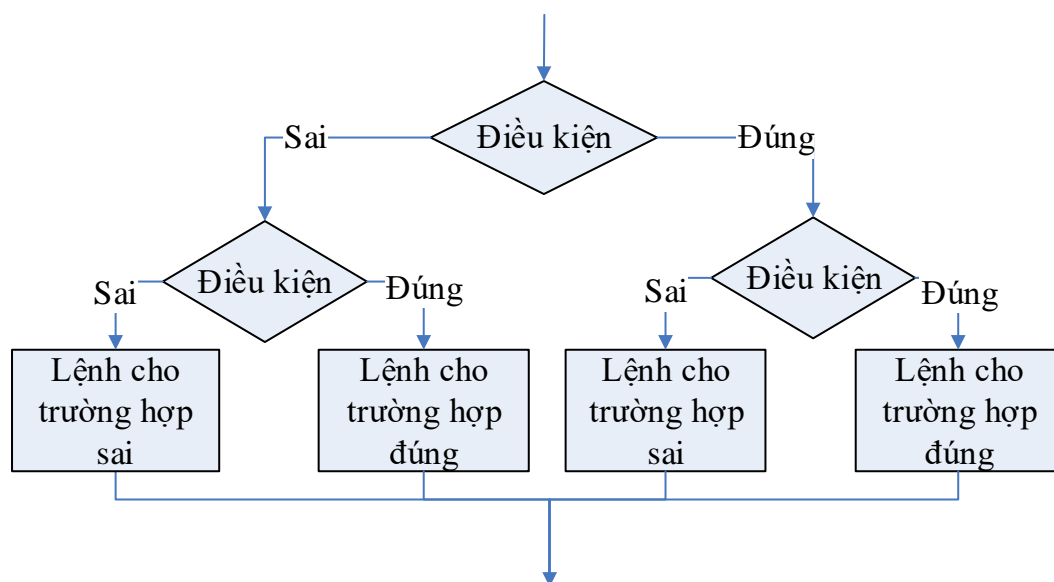
(b) Xét cả hai trường hợp đúng hoặc sai



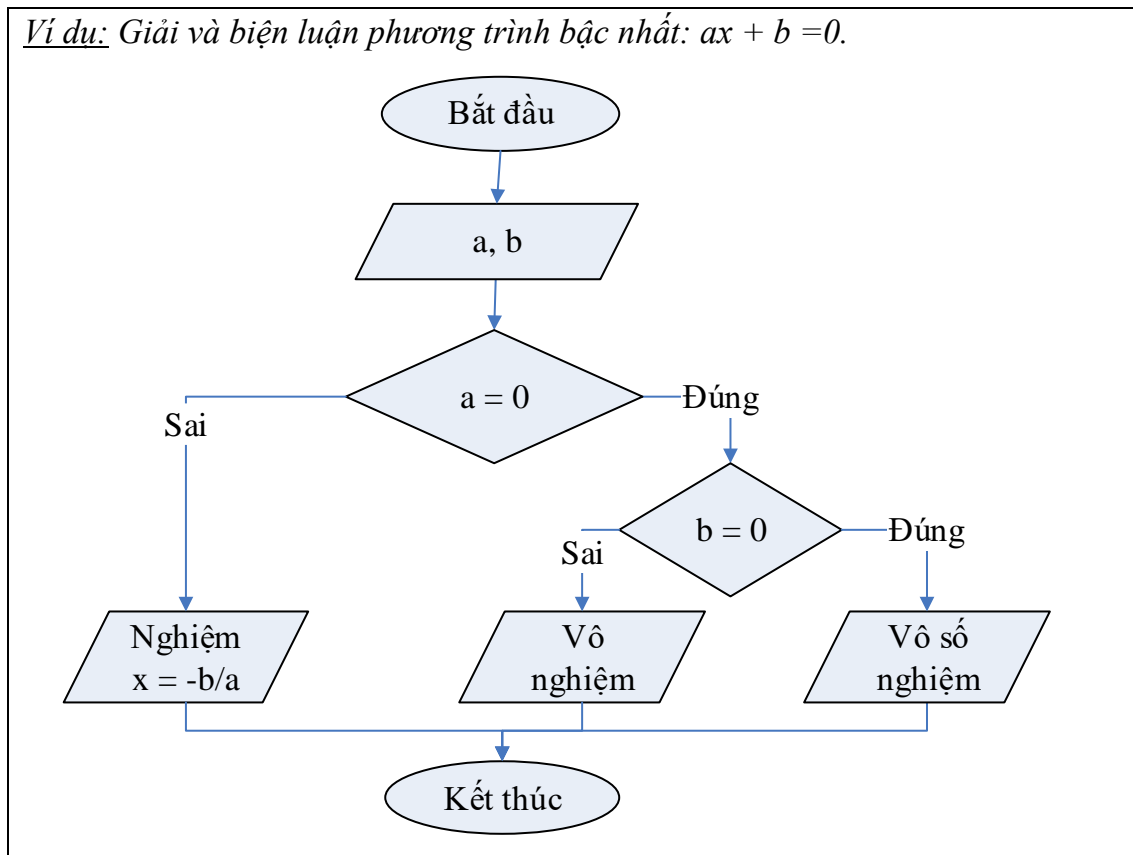
Ví dụ: Nhập vào số nguyên n . Kiểm tra nếu n chẵn xuất ra màn hình “ n chẵn”, ngược lại xuất “ n lẻ”.



(c) **Xét nhiều trường hợp**



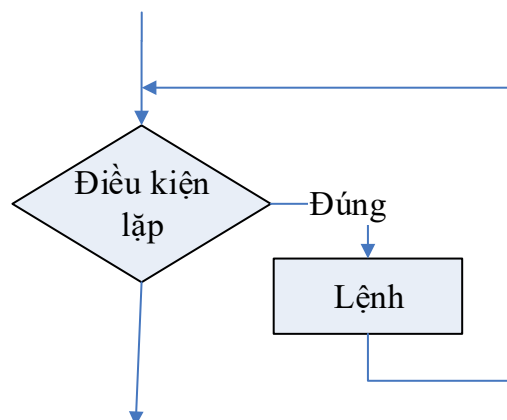
Ví dụ: Giải và biện luận phương trình bậc nhất: $ax + b = 0$.



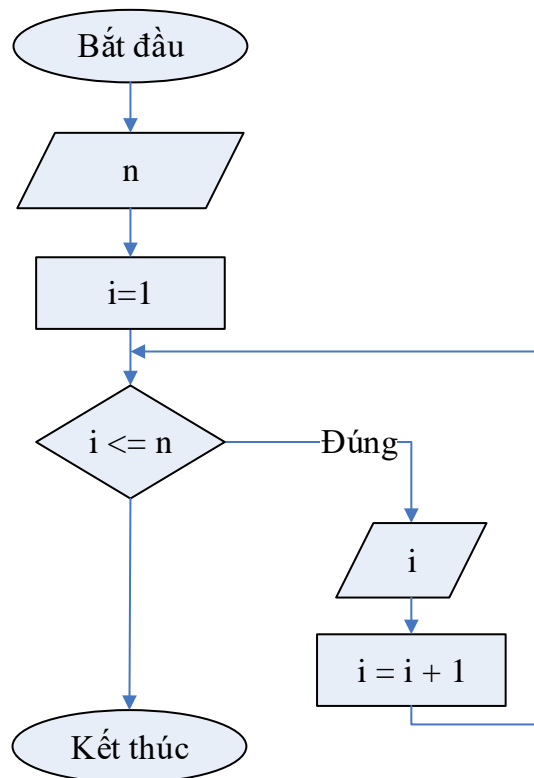
III. 2. 3. Cấu trúc lặp

Thực hiện liên tục 1 lệnh hay tập lệnh với số lần lặp dựa vào điều kiện. Quá trình lặp sẽ kết thúc khi điều kiện bị vi phạm.

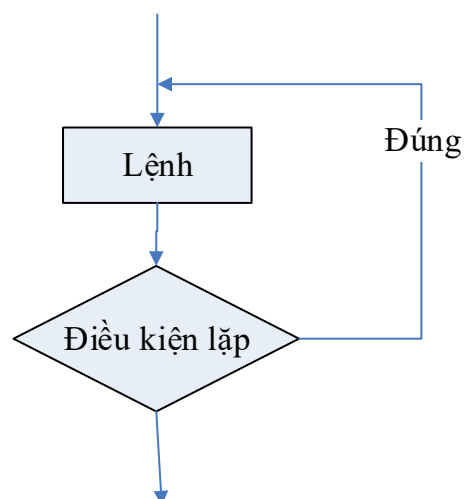
(a) Kiểm tra điều kiện trước khi lặp



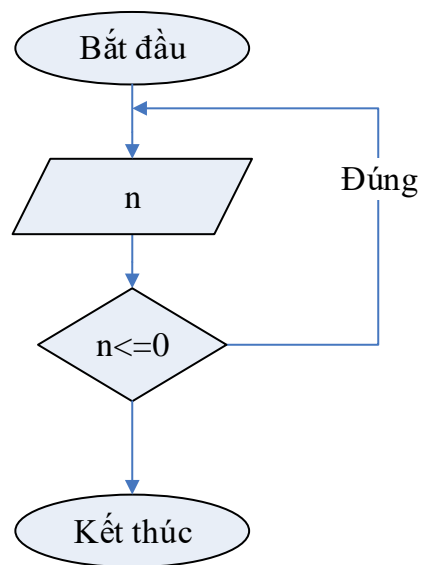
Ví dụ: Nhập vào số nguyên n . Xuất ra màn hình từ 1 đến n .



(b) **Thực hiện lặp trước khi kiểm tra điều kiện**

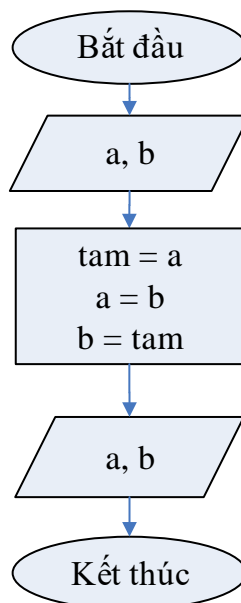


Ví dụ: Nhập vào số nguyên dương n . Nếu nhập sai yêu cầu nhập lại n .

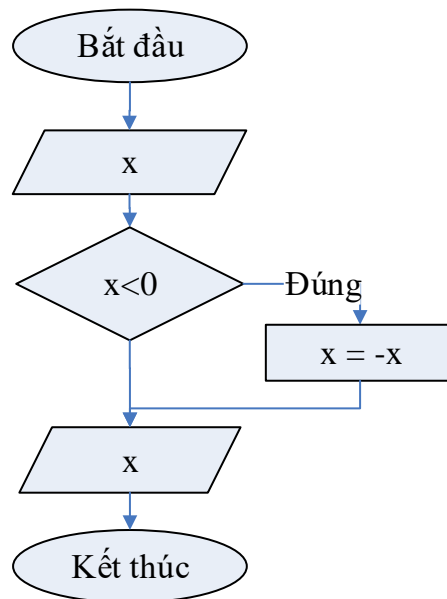


III. 3. Một số ví dụ

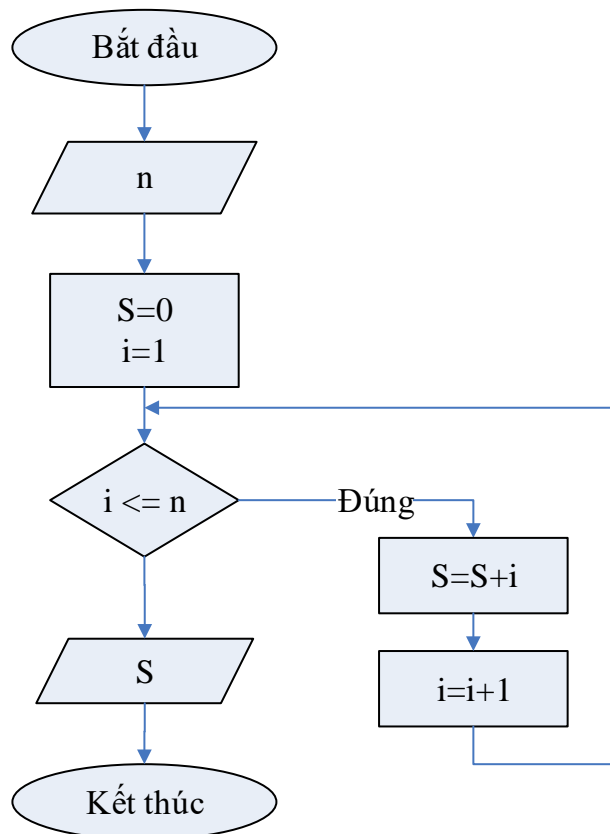
Ví dụ 1: Hoán vị hai số nguyên a và b



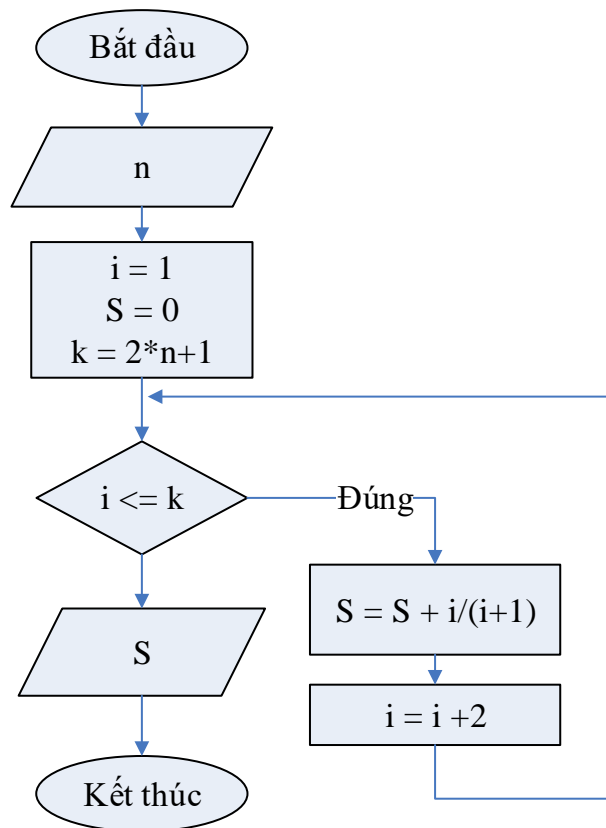
Ví dụ 2: Tính giá trị tuyệt đối của số nguyên x



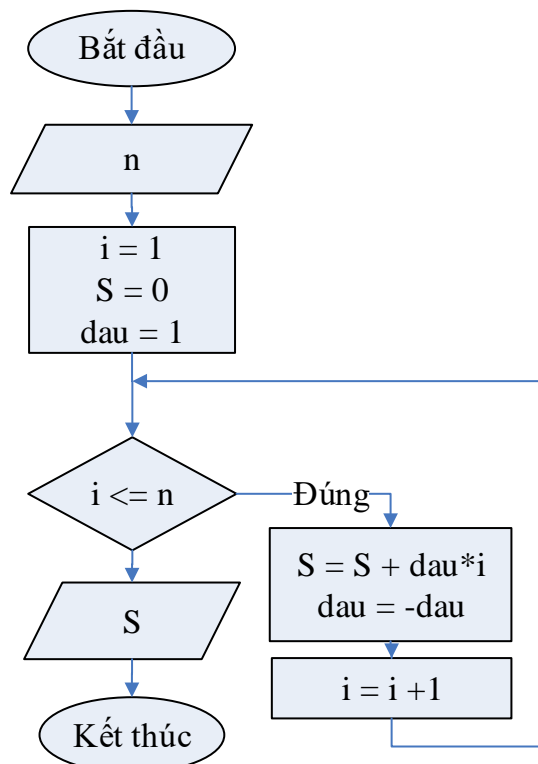
Ví dụ 3: Tính tổng: $S = 1 + 2 + 3 + \dots + n$, với $n > 0$



Ví dụ 4: Tính tổng: $S(n) = \frac{1}{2} + \frac{3}{4} + \frac{5}{6} + \dots + \frac{2n+1}{2n+2}$, với $n \geq 0$



Ví dụ 5: Tính tổng: $S(n) = 1 - 2 + 3 - 4 + \dots + (-1)^{n+1}n$, với $n > 0$



IV. Kết luận

Lưu đồ giải thuật rất hữu ích trong việc mô tả cách giải quyết của một bài toán.

Việc mô tả này rất trực quan thông qua các ký hiệu hình học, đây là giai đoạn đầu tiên trước khi bắt tay vào lập trình trên một ngôn ngữ lập trình cụ thể.

Khi xây dựng lưu đồ giải thuật, chúng ta cần chú ý một vài điểm sau:

- ❖ Một lưu đồ phải có điểm **bắt đầu** và điểm **kết thúc** (điều kiện kết thúc).
- ❖ Phải có **dữ liệu vào**, **dữ liệu ra** sau khi xử lý tính toán.
- ❖ Tại mỗi vị trí quyết định lựa chọn rẽ nhánh phải **ghi rõ điều kiện đúng hoặc sai** thì đi theo nhánh nào.

V. Bài tập vẽ lưu đồ giải thuật

V.1. Bài tập cơ bản

- C1.1. Nhập vào hai số x, y . Xuất ra màn hình tổng, hiệu, tích, thương của hai số trên.
- C1.2. Nhập vào số nguyên n , kiểm tra xem n chẵn hay lẻ và xuất ra màn hình.
- C1.3. Nhập vào ba cạnh a, b, c của tam giác. Xuất ra màn hình tam giác đó thuộc loại tam giác gì? (Thường, cân, vuông, đều hay vuông cân).
- C1.4. Nhập vào số nguyên n . Xuất ra n màn hình (Nếu n chẵn thì gấp đôi giá trị).
- C1.5. Nhập vào số nguyên n . Nếu $n > 5$ thì tăng n lên 2 đơn vị và trả về giá trị n , ngược lại trả về giá trị 0.
- C1.6. Tính $n!$, với $n \geq 0$
- C1.7. Tính $P(n) = 1 \times 3 \times 5 \times \dots \times (2n+1)$, với $n \geq 0$
- C1.8. Tính $S(n) = 1 + 3 + 5 + \dots + (2 \times n + 1)$, với $n \geq 0$
- C1.9. Tính $S(n) = 1 - 2 + 3 - 4 + \dots + (-1)^{n+1}n$, với $n > 0$
- C1.10. Tính $S(n) = 1 + (1 \times 2) + (1 \times 2 \times 3) + \dots + (1 \times 2 \times 3 \times \dots \times n)$, với $n > 0$
- C1.11. Tính $S(n) = 1^2 + 2^2 + 3^2 + \dots + n^2$, với $n > 0$
- C1.12. Tính $S(n) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$, với $n > 0$
- C1.13. Tính $P(x, y) = x^y$
- C1.14. Tính $S(n) = 1 + (1 + 2) + (1 + 2 + 3) + \dots + (1 + 2 + 3 + \dots + n)$, với $n > 0$

- C1.15. Cho số nguyên n . Tính trị tuyệt đối của n .
- C1.16. Cho số nguyên dương n gồm k chữ số. Tìm chữ số có giá trị lớn nhất.
- C1.17. Đếm số lượng ước số chẵn của số nguyên dương n .
- C1.18. In ra chữ số đầu tiên của số nguyên dương n gồm k chữ số.
- C1.19. Cho 2 số nguyên dương a, b . Tìm USCLN của a và b .
- C1.20. Cho 2 số nguyên dương a, b . Tìm BSCNN của a và b .
- C1.21. Cho số nguyên dương x . Kiểm tra xem x có phải là số nguyên tố không?
- C1.22. Cho số nguyên dương x . Kiểm tra x có phải là số chính phương không?
- C1.23. Cho số nguyên dương x . Kiểm tra xem x có phải là số hoàn thiện không?

V. 2. Bài tập luyện tập và nâng cao

- C1.24. Tính $S(n) = 1 + 2^2 + 3^3 + \dots + n^n$, với $n \geq 0$
- C1.25. Tính $S(n) = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \dots + \frac{n}{n+1}$, với $n > 0$
- C1.26. Tính $S(n) = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \dots + \frac{1}{1+2+3+\dots+n}$, với $n > 0$
- C1.27. Tính $S(n) = 1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}$, với $n > 0$
- C1.28. Tính $S(n) = 1 + \frac{1+2}{2!} + \frac{1+2+3}{3!} + \dots + \frac{1+2+3+\dots+n}{n!}$, với $n > 0$
- C1.29. Giải và biện luận phương trình: $ax^2 + bx + c = 0$
- C1.30. Tính $S(n) = \sqrt{n + \sqrt{(n-1) + \sqrt{(n-2) + \dots + \sqrt{1}}}}$, với $n > 0$
- C1.31. Tính $S(n) = \sqrt{1 + \sqrt{2 + \sqrt{3 + \dots + \sqrt{n}}}}$, với $n > 0$

CHƯƠNG 2 GIỚI THIỆU NGÔN NGỮ C

Tóm tắt: Hướng dẫn sử dụng công cụ Microsoft Visual Studio để viết chương trình bằng ngôn ngữ C và làm quen với một chương trình viết bằng ngôn ngữ C

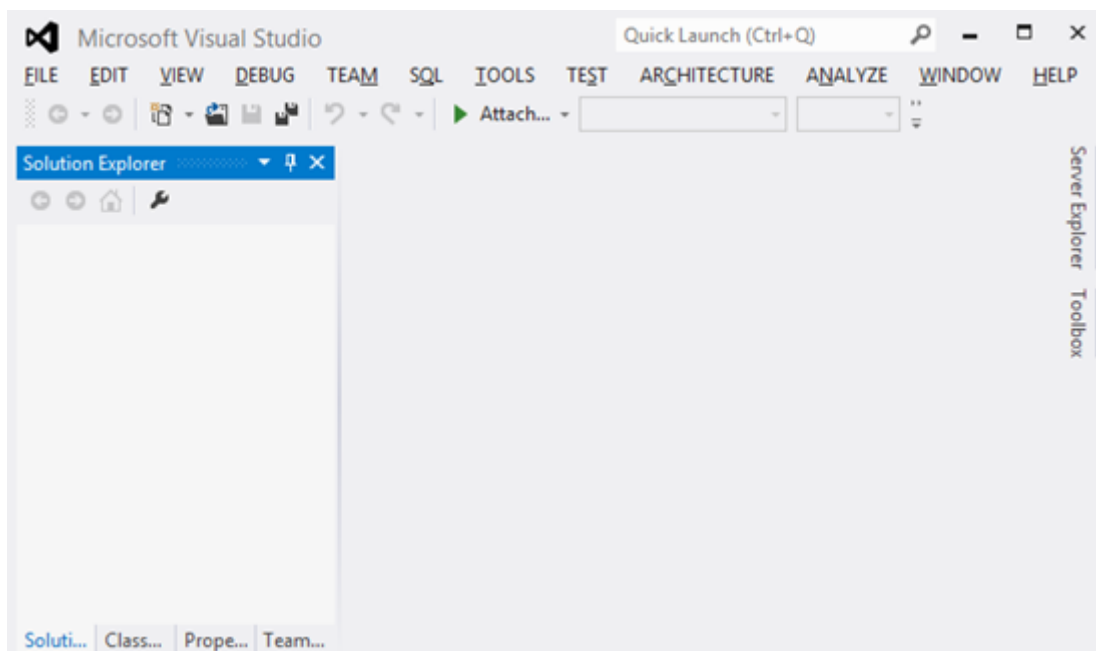
I. Giới thiệu công cụ Microsoft Visual Studio

Trước khi bắt đầu viết chương trình (dù đơn giản hay phức tạp), ta phải tạo dự án mới (nơi lưu trữ mọi thông tin của chương trình được cài đặt). Hiện tại có rất nhiều công cụ hỗ trợ cho việc lập trình bằng ngôn ngữ C, tùy theo công cụ sẽ có giao diện và thao tác khác nhau nhưng khi soạn thảo lệnh để lập trình là giống nhau.

Trong khuôn khổ tài liệu này sẽ giới thiệu công cụ Microsoft Visual Studio phiên bản 2012.

II. Tạo dự án mới (project)

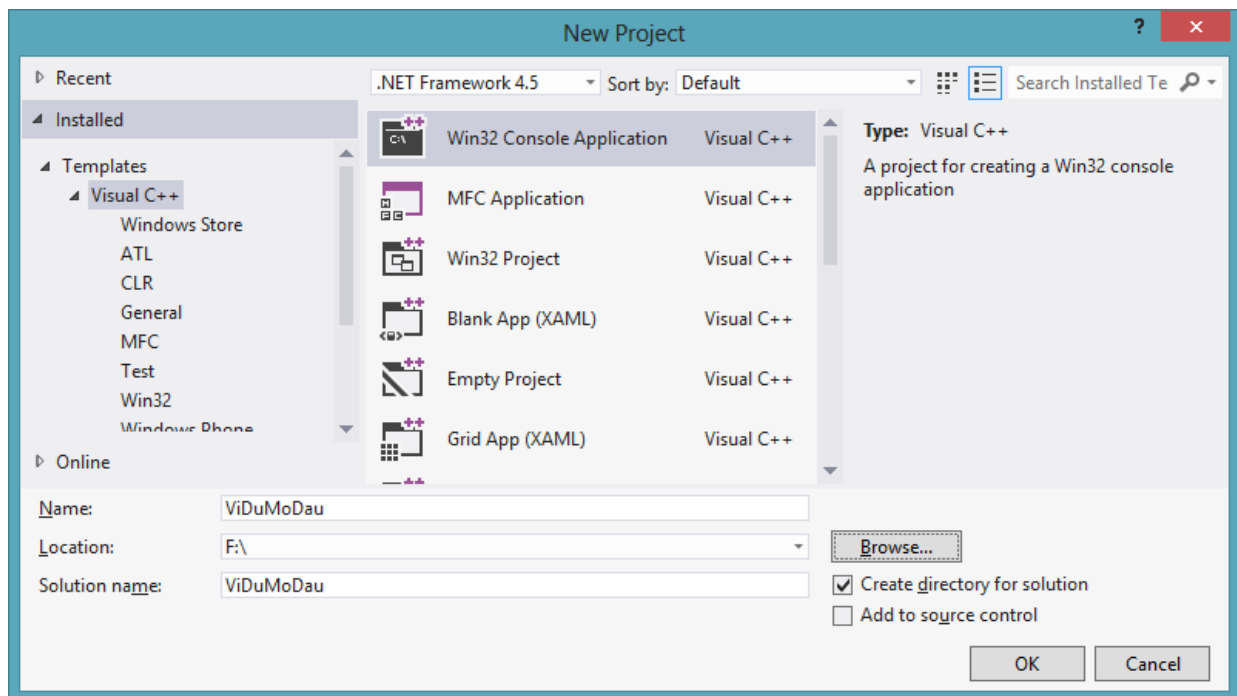
Bước 1: Khởi động công cụ Microsoft Visual Studio



Bước 2: Vào menu File, chọn New và sau đó chọn Project (hoặc nhấn phím tắt Ctrl+Shift+N), hộp thoại New Project sẽ xuất hiện. Lần lượt cung cấp các thông tin cho Project gồm:

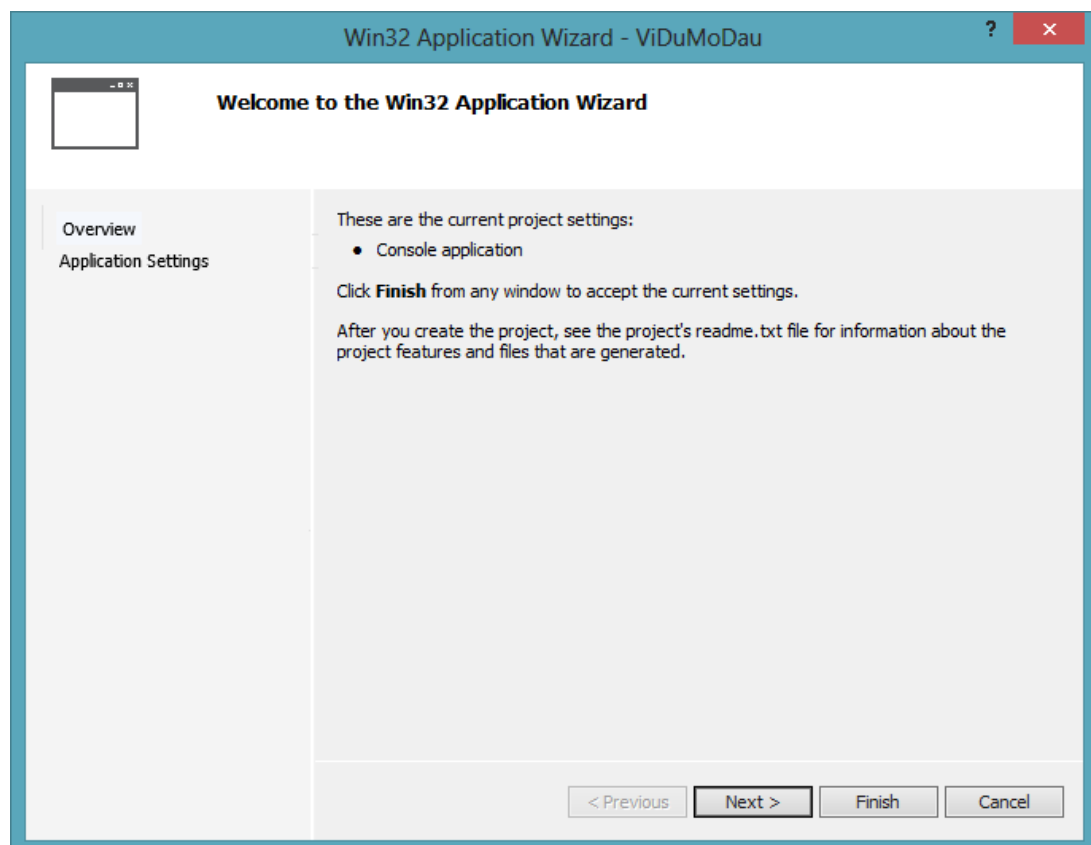
- Project type: Visual C++
- Templates: Win32 Console Application

- Name: Tên của Project
- Location: Vị trí lưu trữ Project
- Sau đó nhấn nút OK.

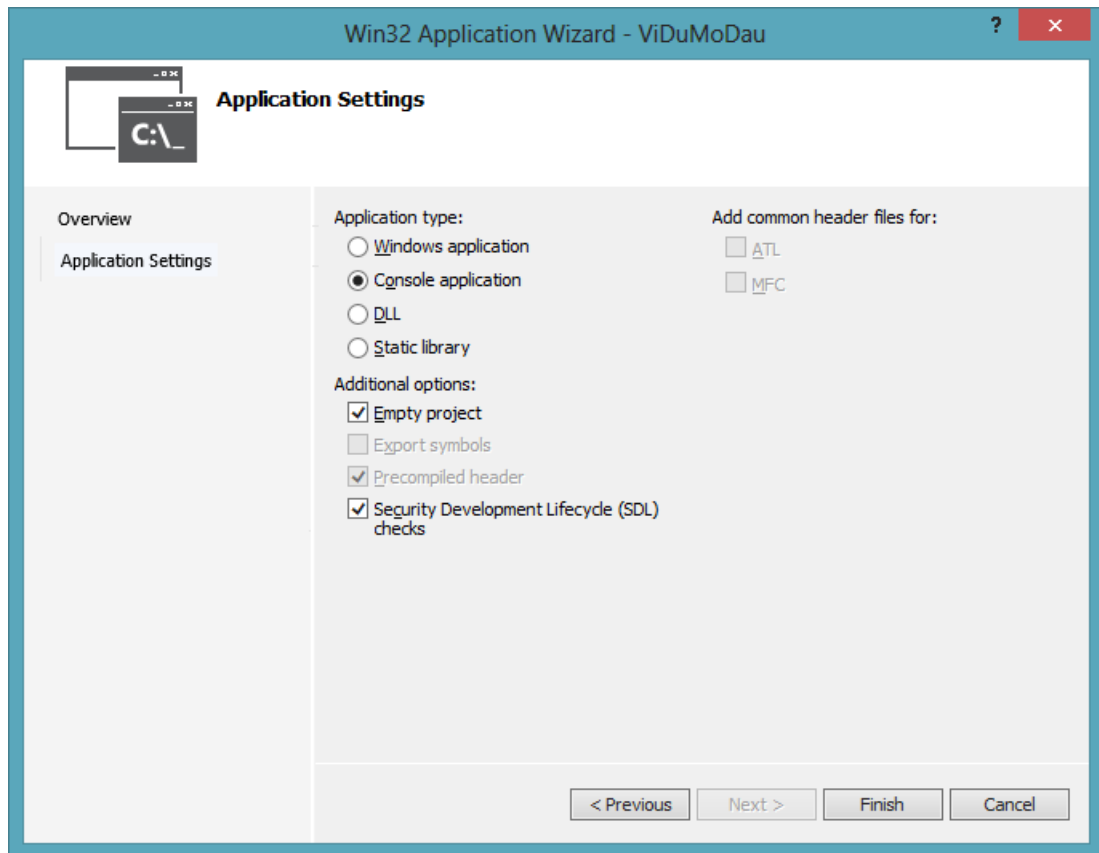


Tạo Project tên là ViDuMoDau được lưu tại ổ đĩa F:

Bước 3: Nhấn vào nút **Next** khi xuất hiện hộp thoại sau

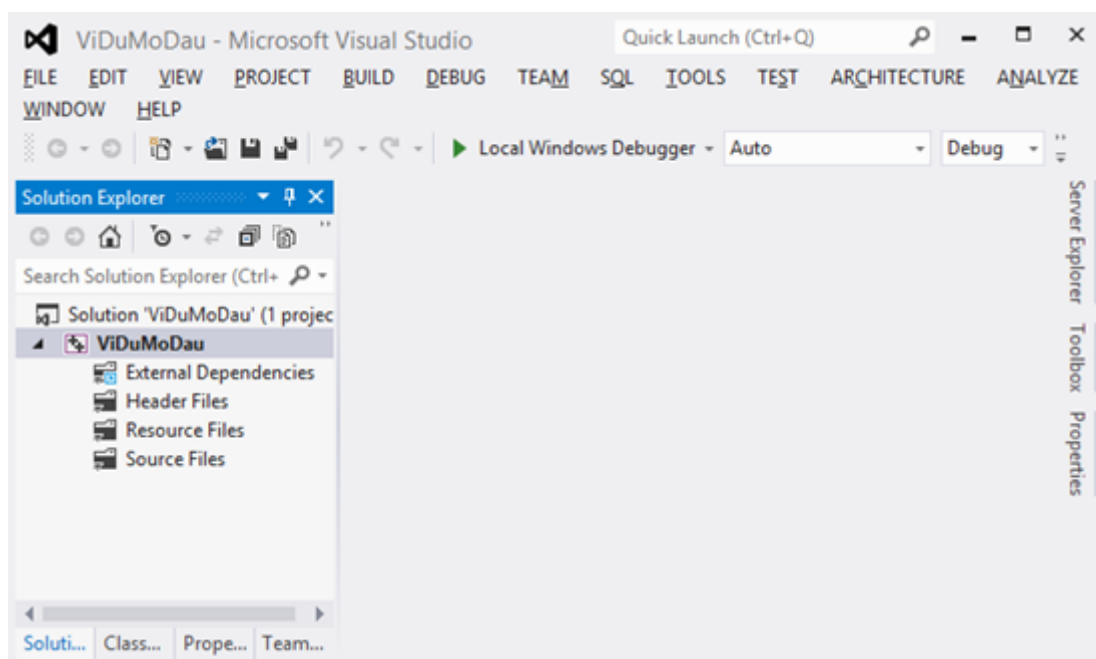


Bước 4: Nhấn chuột vào *check box Empty project* để tạo project rỗng và sau đó nhấn nút **Finish**, quá trình tạo project mới hoàn tất.



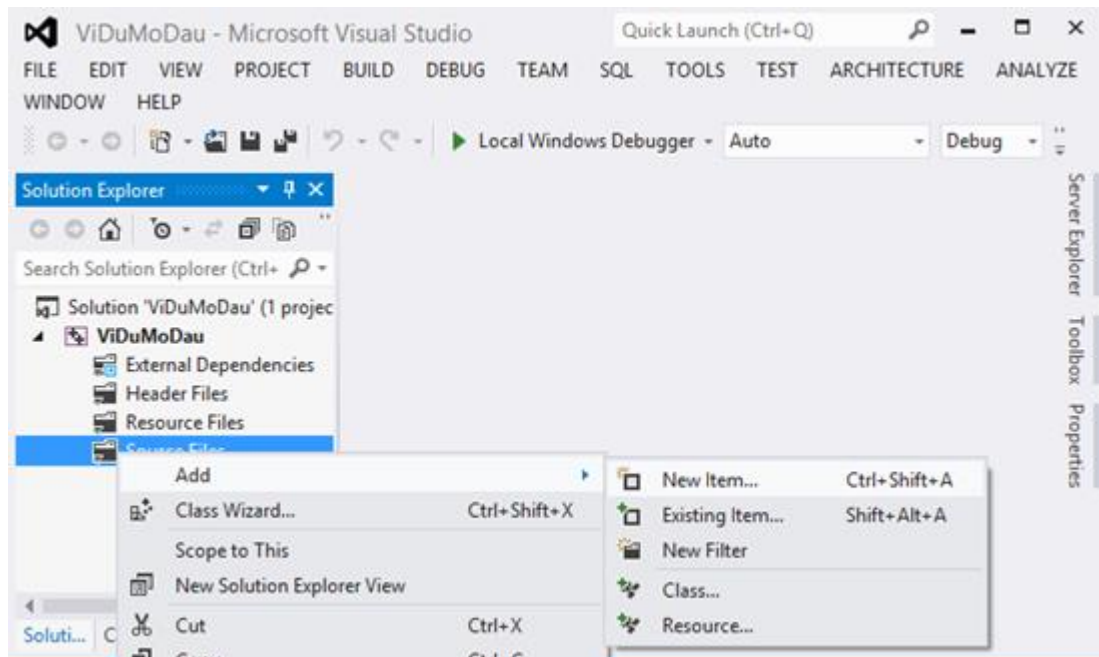
III. Tạo tập tin chứa lệnh

Sau khi tạo thành công project mới, cửa sổ ứng dụng của Microsoft Visual Studio sẽ xuất hiện như sau:

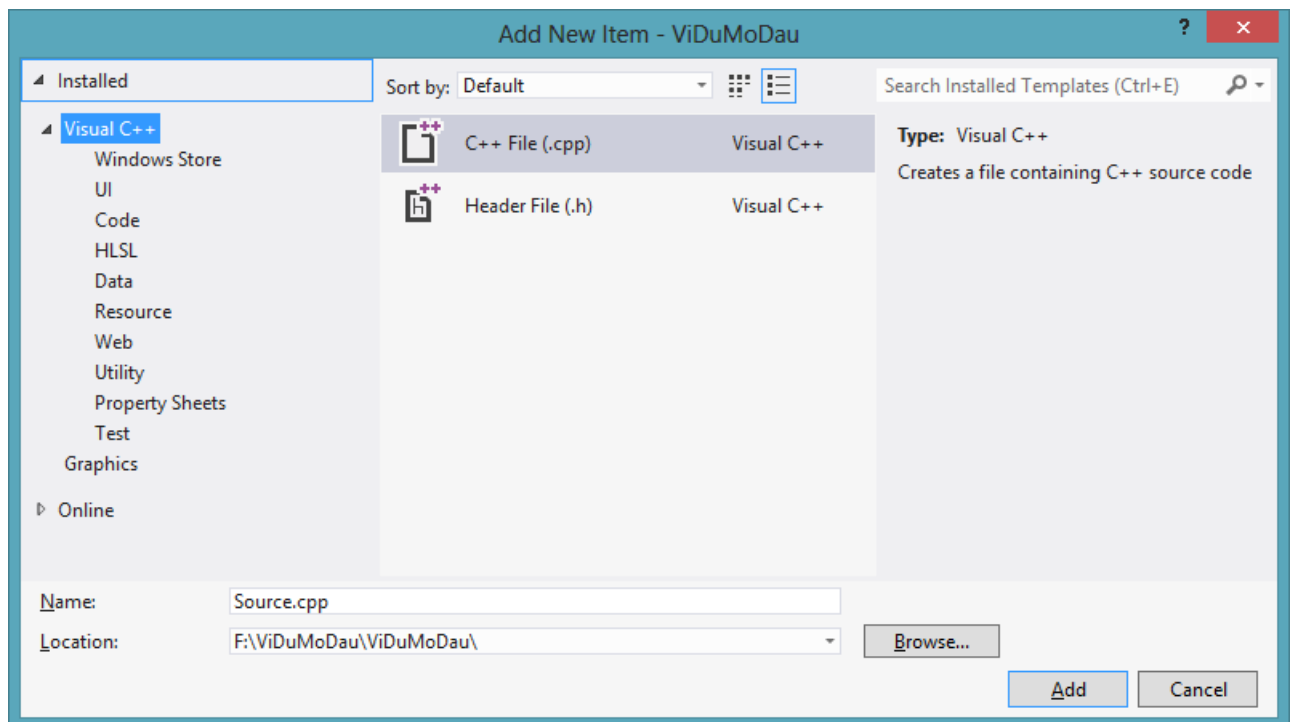


Trong Project ViDuMoDau tại phần cửa sổ Solution Explorer có các thư mục con như là External Dependencies, **Header Files**, Resource Files và **Source Files**. Mỗi thư mục này sẽ chứa các loại tập tin tương ứng, trong đó thư mục Source Files sẽ gồm các tập tin chứa mã lệnh của chương trình. Để có thể viết lệnh cho project thì ta phải tạo tập tin trong thư mục Source Files, cách làm như sau:

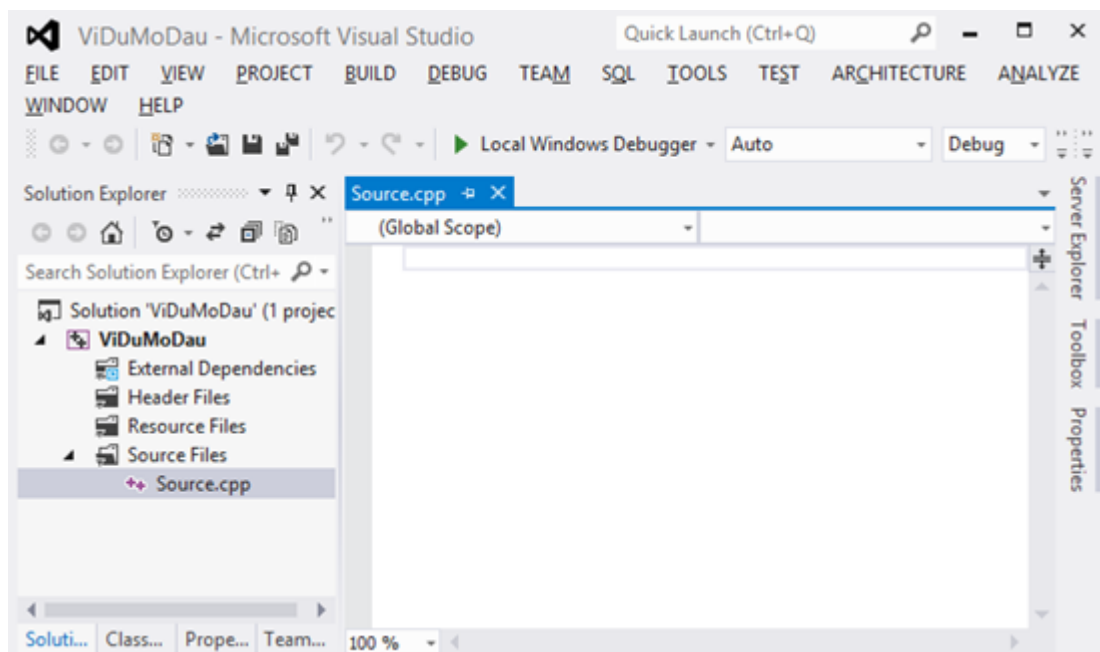
Bước 1: Nhấn chuột phải vào thư mục **Source Files**, chọn **Add** và sau đó chọn **New Item...**



Bước 2: Chỉ được phép thay đổi tên tập tin trong ô Name (ví dụ đặt tên tập tin là Source.cpp, những thông tin khác để mặc định), sau đó nhấn nút **Add** trên hộp thoại sau:



Sau khi tạo thành công tập tin Source.cpp trong thư mục Source Files, giao diện của dụng sẽ xuất hiện như dạng sau:



Bước 3: Chọn vào tập tin Source.cpp và viết lệnh.

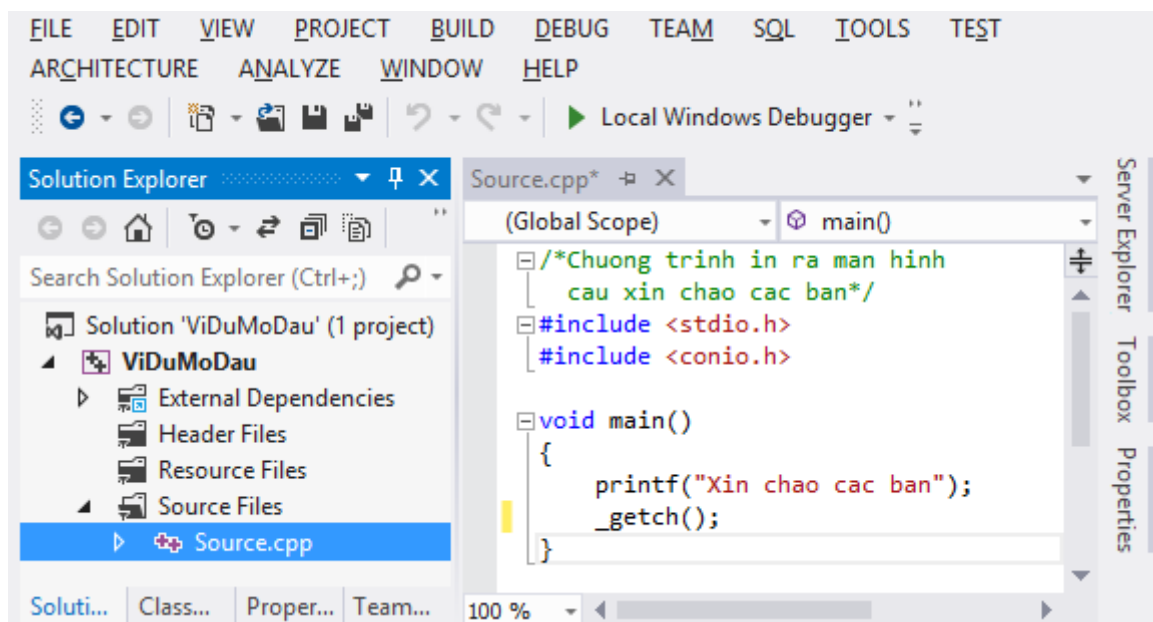
IV. Chương trình mẫu

IV. 1. Chương trình 1

Viết chương trình hiển thị trên màn hình câu thông báo: *Xin chào các bạn*

IV. 1. 1. Lệnh mẫu

Các lệnh sau được soạn thảo trong tập tin Source.cpp (sử dụng lại Project ViDuMoDau đã tạo ở mục II)



IV. 1. 2. Giải thích lệnh

Dòng	Lệnh
1	/*Chương trình in ra màn hình
2	câu xin chào các bạn*/
3	#include <stdio.h>
4	#include <conio.h>
5	
6	void main()
7	{
8	printf("Xin chào các bạn");
9	_getch();
10	}

Dòng 1 đến dòng 2 gọi là ghi chú, ghi chú không phải là lệnh, không có tác dụng trong chương trình mà là để giúp người lập trình ghi nhớ những gì cần thiết. Các ghi

chú được đặt bất kỳ đâu và phải được đặt trong cặp dấu `/* */` (hoặc dùng dấu `//` trước mỗi dòng ghi chú).

Dòng 3 đến dòng 4 gọi là khai báo tiền xử lý (hay còn gọi là khai báo thư viện hàm), do trong chương trình có sử dụng các hàm `printf()` và `_getch()`, các hàm này là các hàm thư viện có sẵn trong ngôn ngữ C và được định nghĩa trong hai tập tin là `stdio.h` và `conio.h`. Muốn khai báo thư viện hàm thì chỉ việc dùng từ khóa `#include <tên tập tin thư viện>`. Nếu dùng hàm thư viện mà không khai báo thư viện hàm thì chương trình sẽ báo lỗi.

Dòng 5 trống không có lệnh, mục đích là ngăn cách với hàm `main()` cho dễ nhìn, việc trình bày chương trình rõ ràng giúp cho người lập trình dễ quan sát và dễ sửa lỗi hơn.

Dòng 6 đến dòng 10 là các lệnh của hàm `main()`, dấu ngoặc tại dòng 7 và dòng 10 là thể hiện bắt đầu `{` và kết thúc hàm `}`, các lệnh sẽ viết trong cặp dấu `{}`.

Dòng 8 là lệnh xuất ra màn hình dòng chữ ***Xin chào các bạn***, chuỗi xuất nằm trong cặp dấu nháy kép ***“Xin chào các bạn”***.

Dòng 9 là lệnh tạm dừng chương trình chờ để người dùng xem kết quả hiển thị trên màn hình, cho đến khi nào người dùng nhấn phím bất kỳ trên bàn phím thì cửa sổ kết quả mới đóng lại (nếu dùng công cụ Turbo C, BC++ 3.1 hay Microsoft Visual C++ 6.0 thì sử dụng hàm `getch()` thay cho `_getch()`)

☞ Một chương trình C dù đơn giản hay phức tạp đều bắt buộc phải có hai phần: phần **khai báo thư viện hàm và hàm `main()`**. Hàm `main()` sẽ được thực hiện đầu tiên khi bắt đầu chạy chương trình.

☞ Các lệnh trong ngôn ngữ C đều phân biệt chữ **in HOA** và **in thường**, đòi hỏi chính xác các ký hiệu và tên lệnh nên khi lập trình sinh viên phải chú ý dùng ký hiệu và viết lệnh cho chính xác.

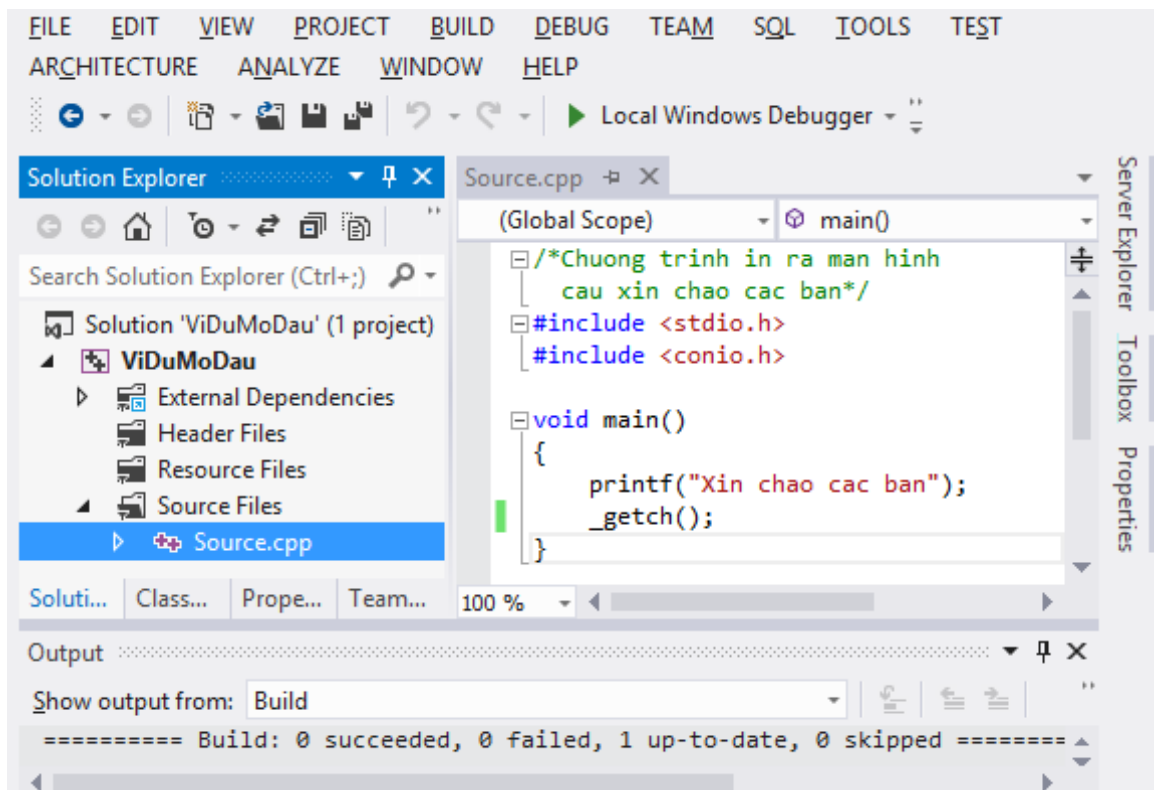
☞ Các lệnh nằm trong cặp dấu `{}` của hàm **`main()`** được viết thụt vào trong (bằng cách nhấn phím `tab` trên bàn phím) để dễ dàng phân biệt lệnh con của hàm.

IV. 1. 3. Biên dịch và thực thi chương trình

(a) Biên dịch chương trình

Mục đích của việc biên dịch là để kiểm tra xem trong quá trình viết lệnh có gặp các lỗi về cú pháp, lệnh, chưa khai báo, ...

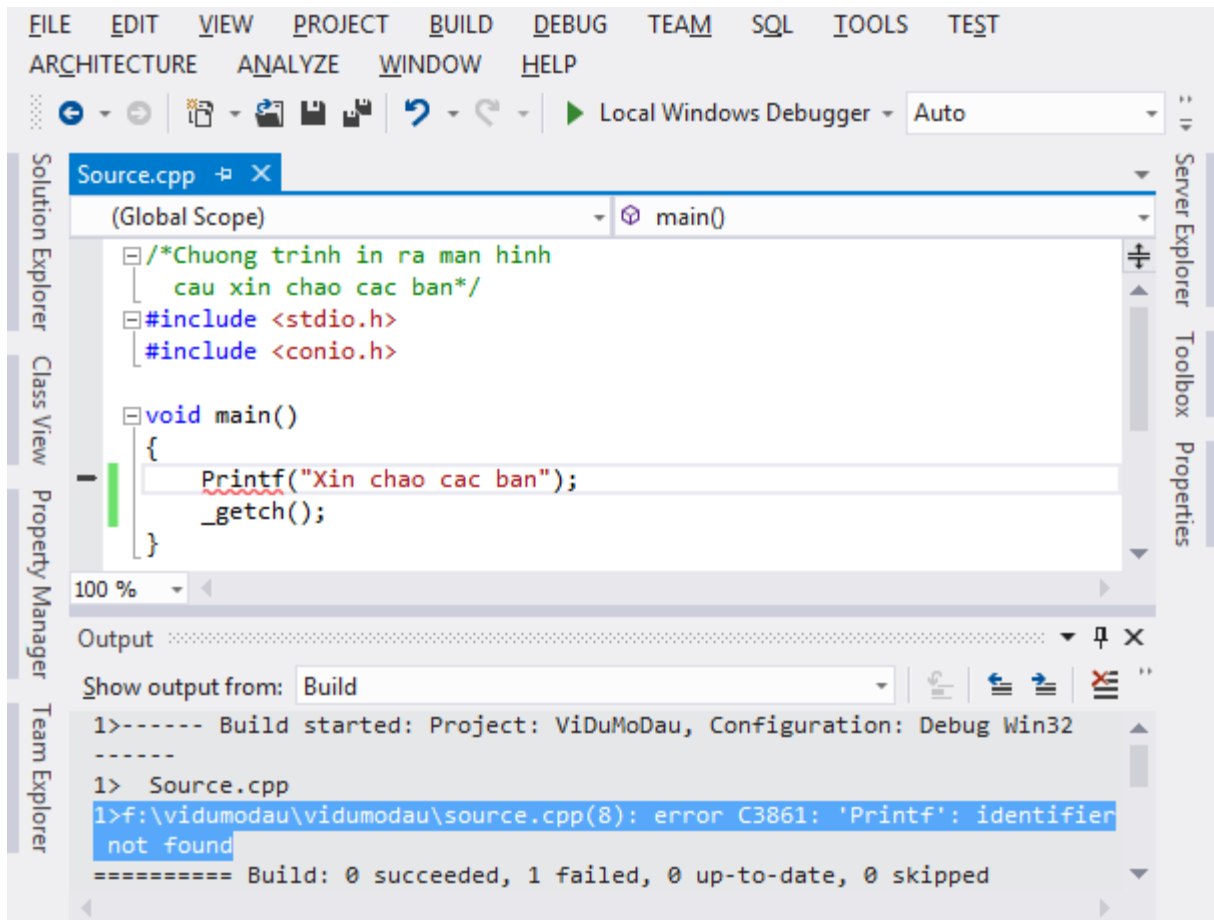
Nhấn phím **F7** để dịch chương trình và kiểm tra lỗi cú pháp (*error*) – quan sát trong cửa sổ Output. Nếu xảy ra lỗi thì phải kiểm tra đối chiếu với lệnh mẫu.



(b) Một số lỗi cơ bản gặp phải và cách sửa

Nếu xảy ra lỗi, trình biên dịch sẽ thông báo trong cửa sổ Output, khi nhấn chuột vào dòng báo lỗi thì công cụ sẽ hiển thị vị trí lỗi trên đoạn lệnh tương ứng.

Cách sửa lỗi là quan sát thông báo lỗi để hình dung trước đó là lỗi gì, sau đó nhấn chuột vào câu thông báo lỗi (theo thứ tự từ trên xuống dưới) và quan sát vị trí báo lỗi trên cửa sổ code và sửa lỗi.



Một số lỗi cơ bản:

Thông báo lỗi	Nguyên nhân	Cách sửa	Ví dụ
'Printf': identifier not found	Sai lệnh Printf do ký tự p viết in hoa. Ngôn ngữ C phân biệt hoa thường.	Viết lại lệnh printf là chữ thường	Sai: Printf("Xin chào"); Sửa thành: printf("Xin chào");
'printf': identifier not found	Viết đúng lệnh nhưng chưa khai báo thư viện hàm cho hàm printf()	Bổ sung khai báo: #include <stdio.h> ở đầu chương trình	
missing;	Thiếu dấu ; khi kết thúc khai báo hoặc lệnh	Bổ sung thêm dấu ;	Sai: printf("Xin chào") Sửa thành: printf("Xin chào");
end of file found before the left brace '{'	Thiếu dấu đóng ngoặc khi kết thúc hàm main()	Bổ sung thêm dấu }	void main() { //Các lệnh }

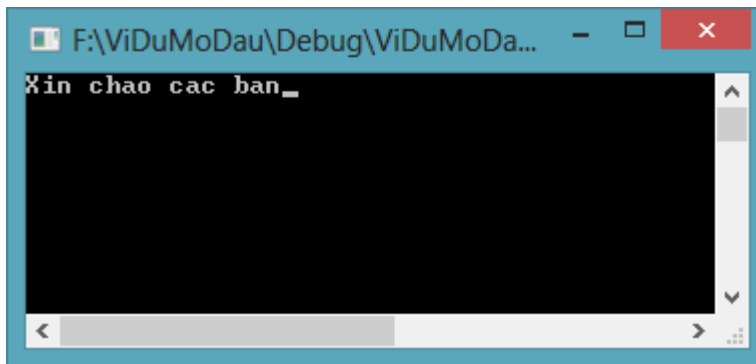
➤ Nếu đoạn chương trình viết chính xác theo mẫu mà vẫn báo lỗi thì có thể do những nguyên nhân sau:

- ✓ Tạo sai project (kiểm tra lại từng bước và thực hiện lại cho chính xác), chú ý chọn đúng Templates là **Win32 Console Application**
- ✓ Sai phần mở rộng của file source (phần mở rộng phải là **cpp** hoặc **c**)
- ✓ Tạo tập tin **source.cpp** không đúng thư mục **Source Files**

(c) **Thực thi chương trình**

Nếu không xảy ra lỗi thì nhấn phím **F5** để thực thi chương trình.

Kết quả sẽ hiển thị như sau:



Nhấn phím Enter để đóng cửa sổ kết quả và trở về cửa sổ soạn thảo code

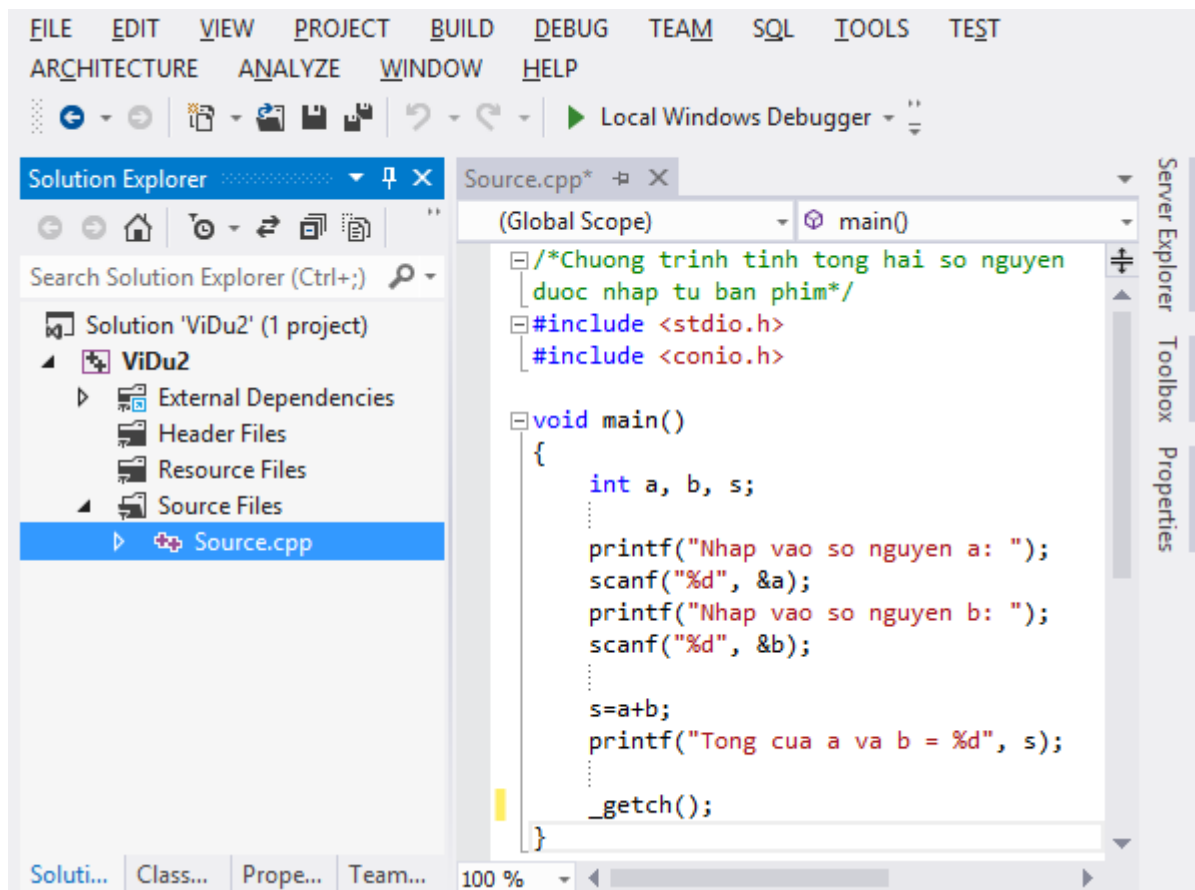
IV. 2. Chương trình 2

Viết chương trình nhập vào hai số nguyên từ bàn phím, tính tổng và hiển thị kết quả ra màn hình.

IV. 2. 1. **Lệnh mẫu**

Tạo mới project tên là **ViDu2**, tạo thêm tập tin **source.cpp** trong thư mục **Source Files**

Các lệnh sau được soạn thảo trong tập tin **Source.cpp**



IV. 2. 2. Giải thích lệnh

Dòng	Lệnh
1	/*Chương trình tính tổng hai số nguyên
2	được nhập từ bàn phím*/
3	#include <stdio.h>
4	#include <conio.h>
5	
6	void main()
7	{
8	int a, b, s;
9	
10	printf("Nhập vào số nguyên a: ");
11	scanf("%d", &a);
12	printf("Nhập vào số nguyên b: ");
13	scanf("%d", &b);
14	
15	s=a+b;
16	printf("Tổng của a và b = %d", s);
17	
18	_getch();
19	}

Dòng 8 gọi là khai báo biến, trong chương trình cần nhập vào hai số nguyên nên phải khai báo hai biến a và b có kiểu là int (kiểu số nguyên). Cách khai báo biến của C theo cú pháp: kiểu dữ liệu và phía sau là tên biến (nếu có nhiều biến cùng kiểu thì cách nhau bằng dấu phẩy). Biến s dùng để lưu giá trị tổng của hai số a và b.

Dòng 11 và dòng 13 dùng để nhập giá trị từ bàn phím cho biến a và b (hàm scanf() dùng để nhập giá trị cho biến từ bàn phím – trong thư viện hàm stdio.h). Ký hiệu “%d” là thể hiện định dạng của số nguyên.

Dòng 15 dùng để tính tổng hai số a và b lưu vào biến s.

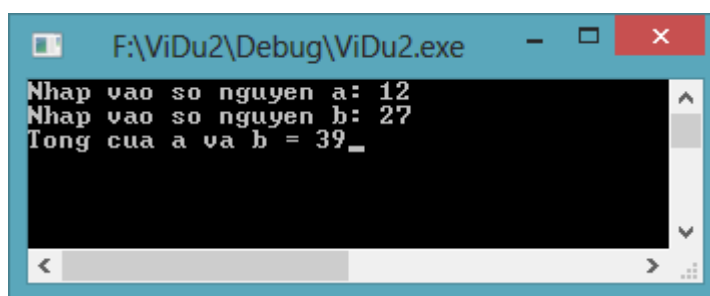
Dòng 16 dùng để xuất giá trị tổng s tính được.

IV. 2. 3. Kết quả chương trình

Nhấn phím **F7** để dịch chương trình, nếu không có lỗi thì nhấn phím **F5** để thực thi chương trình.

Quá trình thực thi chương trình như sau:

- Khi cửa sổ kết quả sẽ hiển thị câu thông báo: **Nhap vào so nguyen a: _** thì ta nhập vào một số nguyên a từ bàn phím (ví dụ nhập số 12) rồi sau đó nhấn phím Enter.
- Chương trình sẽ hiển thị tiếp câu thông báo: **Nhap vào so nguyen b: _** thì ta nhập vào một số nguyên b từ bàn phím (ví dụ nhập số 27) rồi sau đó nhấn phím Enter.
- Kết quả sẽ hiển thị như sau:



- Nhấn phím **Enter** để đóng cửa sổ kết quả và trở về cửa sổ soạn thảo code

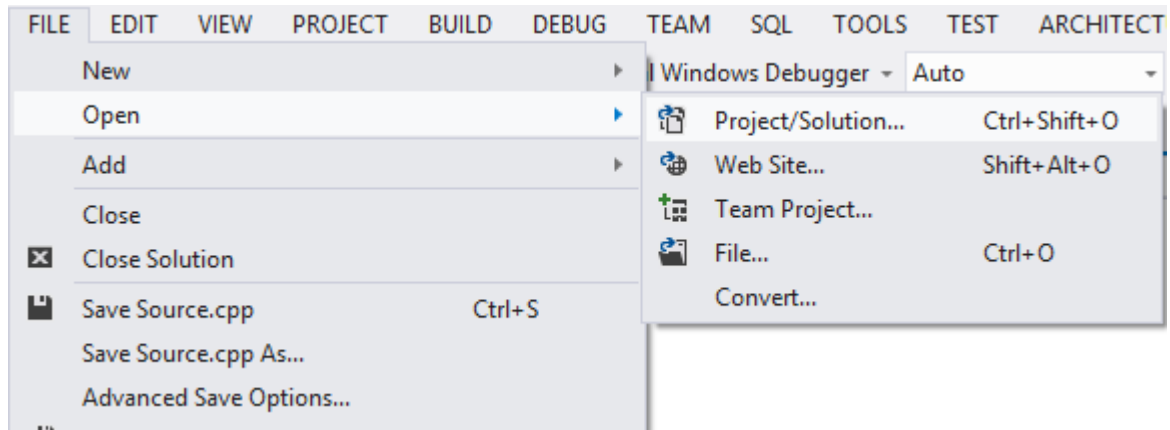
⚠ Nếu chương trình thực thi mà không nhận được kết quả mong muốn (kết quả sai) thì cần kiểm tra lại lệnh scanf. Lệnh scanf phải đảm bảo đầy đủ định dạng số nguyên “%d” (nằm trong cặp nháy kép) và dấu & trước tên biến.

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

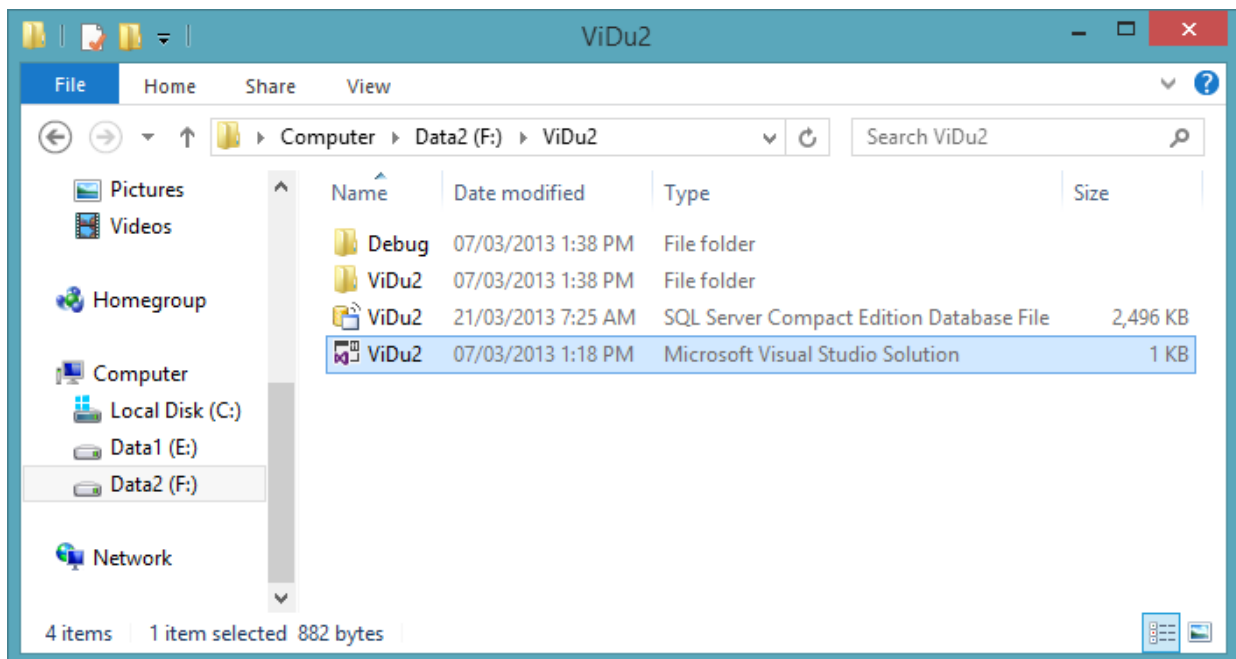
V. Mở dự án có sẵn

Khi cần mở dự án có sẵn ta thực hiện một trong hai cách sau:

Cách 1: Thông qua công cụ Visual Studio: Chọn *menu File* → *Open* → *Project/Solution...*



Cách 2: Hoặc chọn thư mục chứa dự án và chọn mở tập tin có phần mở rộng là *sln* (ví dụ: mở tập tin ViDu2.sln)



VI. Một số chương trình mẫu

Khi bắt đầu làm quen với ngôn ngữ lập trình, trước hết sinh viên phải làm quen với công cụ và viết một số chương trình theo mẫu. Bước đầu sinh viên sẽ nắm được một số thao tác cơ bản, nhận biết thông báo lỗi, sửa lỗi cơ bản và quan sát kết quả của chương trình trước khi học cú pháp cụ thể từng lệnh.

Sinh viên tạo mỗi dự án cho mỗi chương trình mẫu, viết lệnh, biên dịch, tìm cách sửa lỗi nếu có, thực thi chương trình và quan sát kết quả thực hiện.

VI. 1. Chương trình mẫu 1

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

void main()
{
    printf("Ho ten: Nguyen Thanh Tung\n");
    printf("MSSV: 0800139\n");
    printf("Email: nttung@gmail.com\tDien thoai: 0909123456\n");
    printf("Dia chi: 12 Trinh Dinh Thao, Tan Phu\n\n\n");
    printf("***het***");
    _getch();
}
```

VI. 2. Chương trình mẫu 2

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

void main()
{
    int a, b;
    float tb;

    printf("Nhap vao so nguyen a: ");
    scanf("%d", &a);
    printf("Nhap vao so nguyen b: ");
    scanf("%d", &b);

    tb=(a+b)/2.0;
    printf("Gia tri trung binh = %f", tb);
    _getch();
}
```

VII. Chạy từng bước xem kết quả hoạt động của chương trình

Khi gặp trường hợp chương trình thực thi cho kết quả sai (*chương trình không còn lỗi cú pháp và đã thực thi được*), chúng ta phải kiểm tra từng lệnh một xem có sai về mặt ngữ nghĩa, sai kiểu dữ liệu khai báo, sai về mặt giải thuật hay không. Những lỗi sai này rất khó phát hiện nếu chỉ quan sát lệnh bằng mắt thường nhất là đối với những mới học lập trình hoặc chưa có kinh nghiệm nhiều.

Để giải quyết vấn đề này, cách đơn giản nhất là sử dụng công cụ chạy từng bước của Visual Studio để phát hiện lệnh nào cho kết quả sai và tìm cách sửa cho phù hợp (hay còn gọi là debug chương trình).

Ví dụ: Cho đoạn chương trình thực hiện yêu cầu: nhập vào 2 số nguyên a và b, tính giá trị trung bình cộng của a và b, in kết quả ra màn hình.

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

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

```
void main()
```

```
{
```

```
    int a, b;
```

```
    float tb;
```

```
    printf("Nhập vào số nguyên a: ");
```

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

```
    printf("Nhập vào số nguyên b: ");
```

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

```
    tb=(a+b)/2;
```

```
    printf("Giá trị trung bình = %f", tb);
```

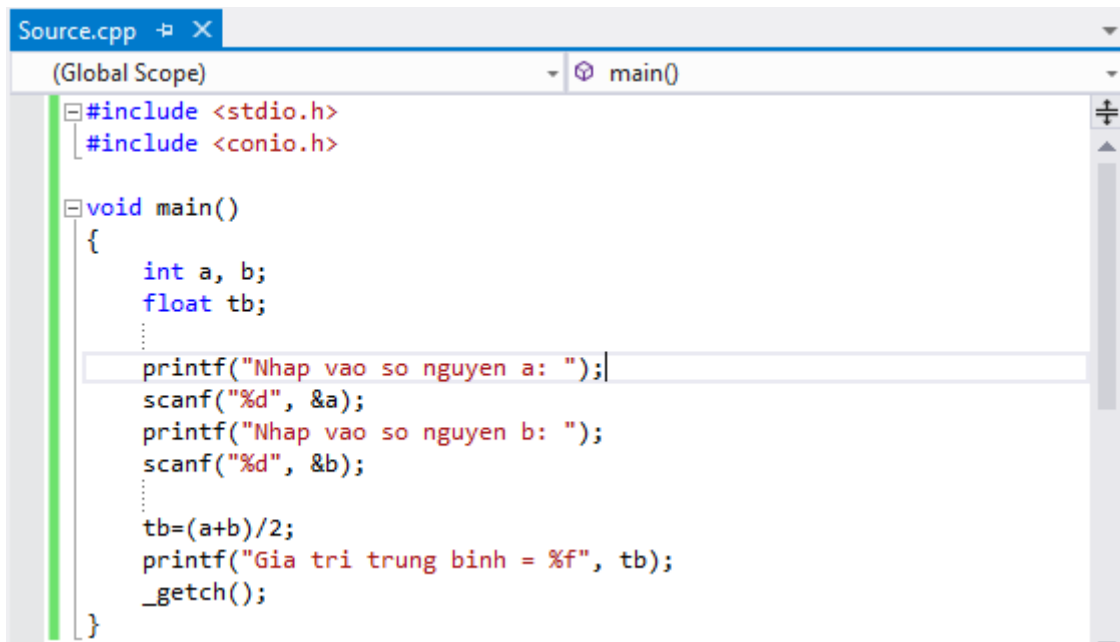
```
    _getch();
```

```
}
```

Tuy nhiên, khi chạy chương trình thì kết quả không đúng trong một số trường hợp. Hãy tìm và chỉ ra dòng lệnh gây ra lỗi sai.

Bước 1. Đặt dấu nhảy tại vị trí cần debug, ví dụ cần debug ngay từ dòng lệnh

printf("Nhập vào số nguyên a: ");



```
Source.cpp -# X
(Global Scope) main()
#include <stdio.h>
#include <conio.h>

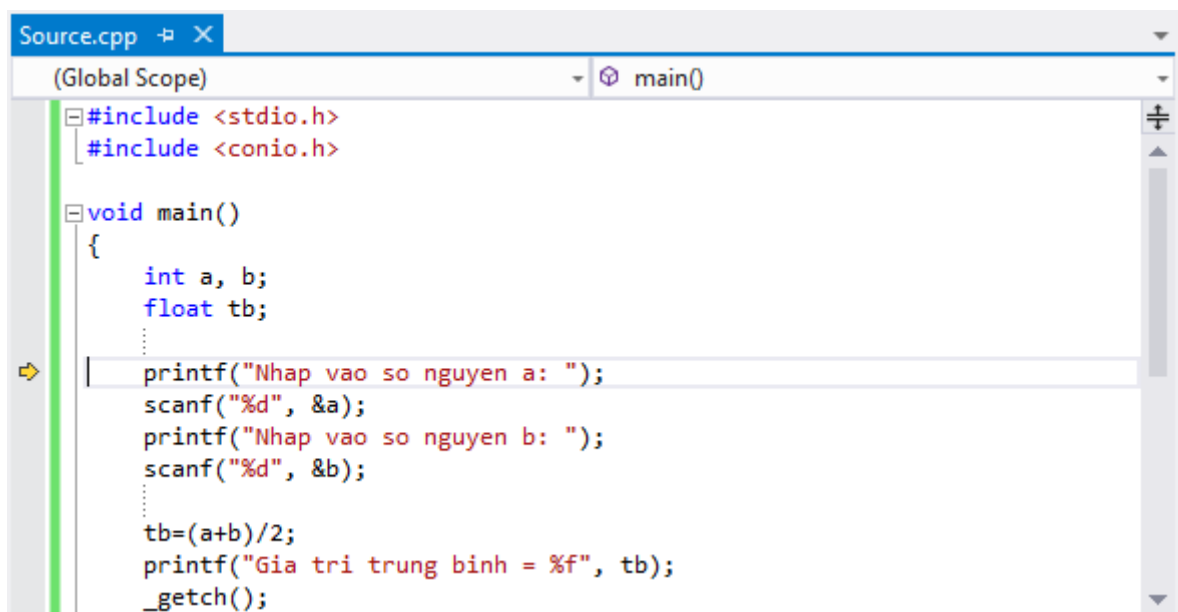
void main()
{
    int a, b;
    float tb;

    printf("Nhập vào số nguyên a: ");
    scanf("%d", &a);
    printf("Nhập vào số nguyên b: ");
    scanf("%d", &b);

    tb=(a+b)/2;
    printf("Giá trị trung bình = %f", tb);
    _getch();
}
```

Bước 2. Sau đó nhấn phím **Ctrl + F10** để chương trình chạy từng bước bắt đầu từ dấu nhảy đã đặt ở bước 1 (ta sẽ thấy một dấu mũi tên màu vàng xuất hiện tại đầu dòng)

⇒ | printf("Nhập vào số nguyên a: ");



```
Source.cpp -# X
(Global Scope) main()
#include <stdio.h>
#include <conio.h>

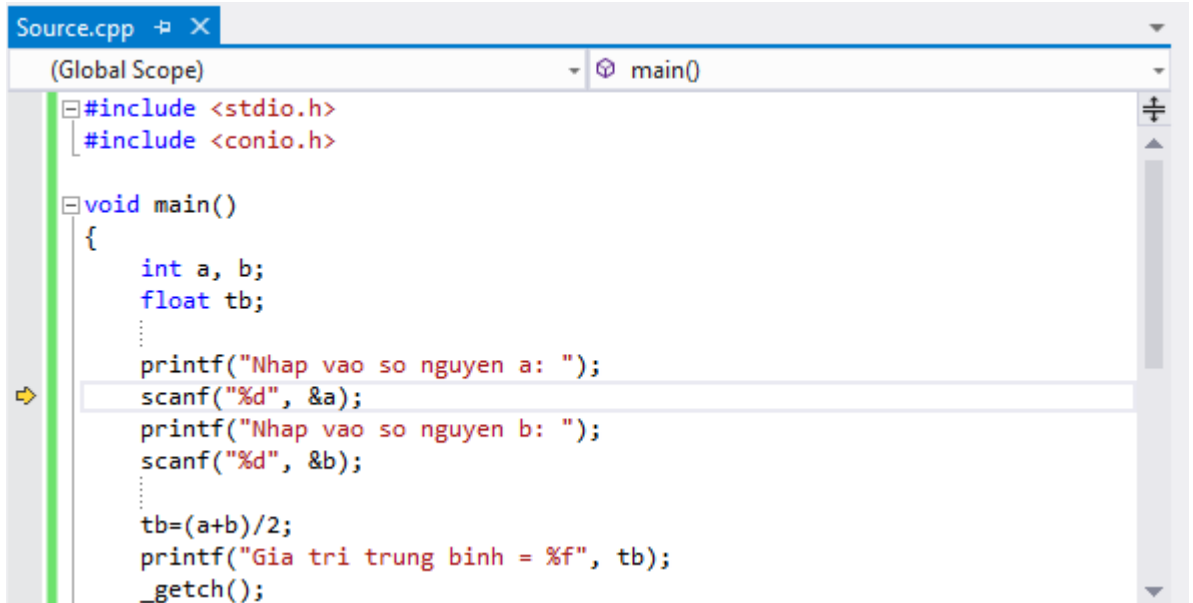
void main()
{
    int a, b;
    float tb;

    printf("Nhập vào số nguyên a: ");
    scanf("%d", &a);
    printf("Nhập vào số nguyên b: ");
    scanf("%d", &b);

    tb=(a+b)/2;
    printf("Giá trị trung bình = %f", tb);
    _getch();
}
```

Bước 3. Trong bước này, ta sẽ quan sát vị trí dấu mũi tên để biết được dòng lệnh sẽ thực hiện, kiểm tra giá trị của các biến và kết hợp với quan sát của sổ kết quả (của sổ console).

- (i) **Thực hiện từng lệnh:** tại cửa sổ lệnh, ta nhấn phím **F10**, cửa sổ lệnh sẽ thực thi dòng lệnh hiện tại (*có dấu mũi tên*), sau đó dấu mũi tên chuyển xuống lệnh kế tiếp.



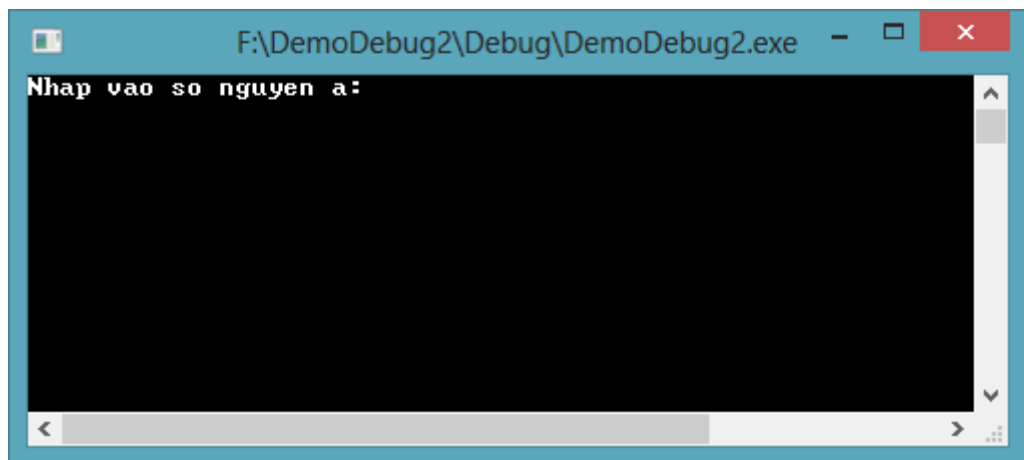
```
Source.cpp [X]
(Global Scope) [main()]
#include <stdio.h>
#include <conio.h>

void main()
{
    int a, b;
    float tb;

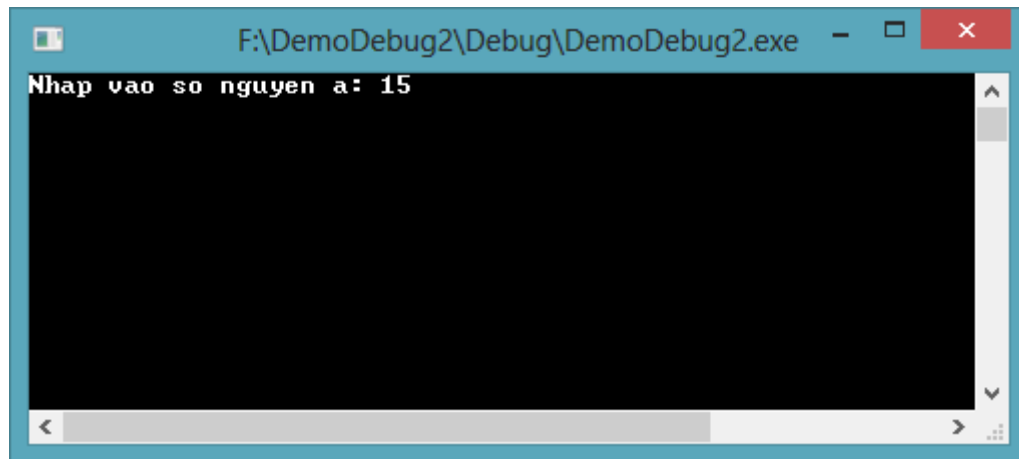
    printf("Nhap vao so nguyen a: ");
    scanf("%d", &a);
    printf("Nhap vao so nguyen b: ");
    scanf("%d", &b);

    tb=(a+b)/2;
    printf("Gia tri trung binh = %f", tb);
    _getch();
}
```

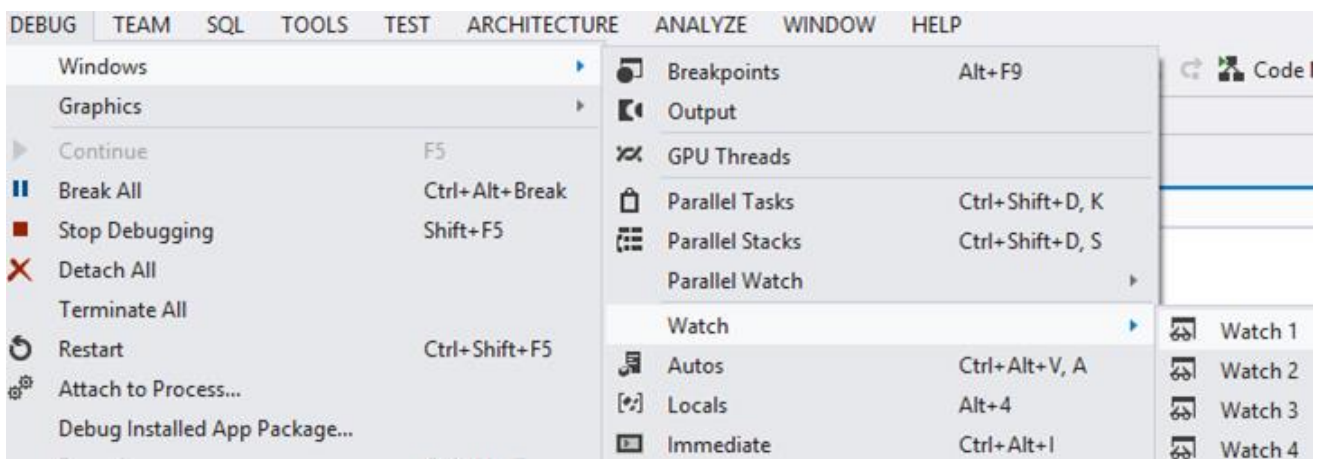
Bật cửa sổ console, ta sẽ thấy xuất hiện dòng “*Nhap vao so nguyen a:*”



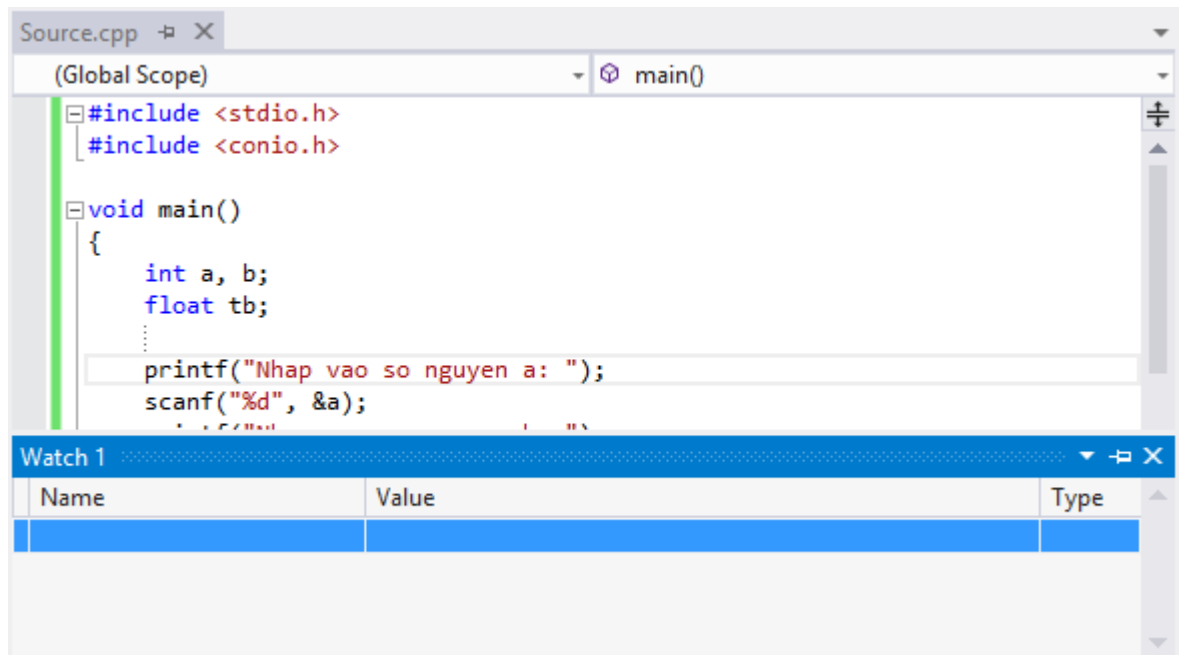
- (ii) Trở về cửa sổ lệnh, nhấn tiếp phím **F10**, chương trình sẽ thực hiện lệnh nhập cho số nguyên *a*. Cửa sổ console kết quả tự động xuất hiện, ta nhập vào một số nguyên từ bàn phím (*ví dụ số 15*) sau đó nhấn phím **Enter**.



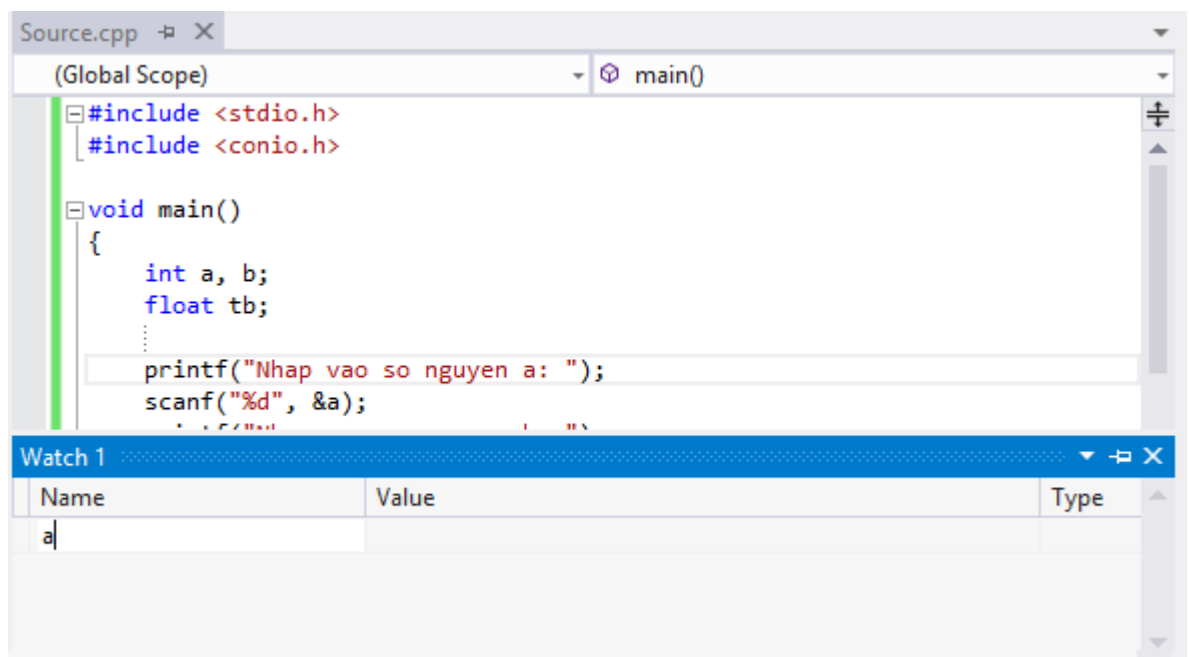
- (iii) **Kiểm tra giá trị của biến:** muốn kiểm tra xem biến a có được lưu trữ là số 15 như vừa nhập hay không, ta bật cửa sổ watch tại cửa sổ lệnh bằng cách chọn **menu Debug, Windows, Watch, chọn Watch 1**. Việc kiểm tra rất cần thiết do người dùng có thể nhập giá trị quá lớn, quá nhỏ hay khi lập trình thiếu dấu &, sai chuỗi định dạng trong hàm *scanf()*.




- Ta sẽ thấy xuất hiện cửa sổ Watch trong cửa sổ dòng lệnh (*mặc định sẽ xuất hiện phía bên dưới cửa sổ dòng lệnh*)



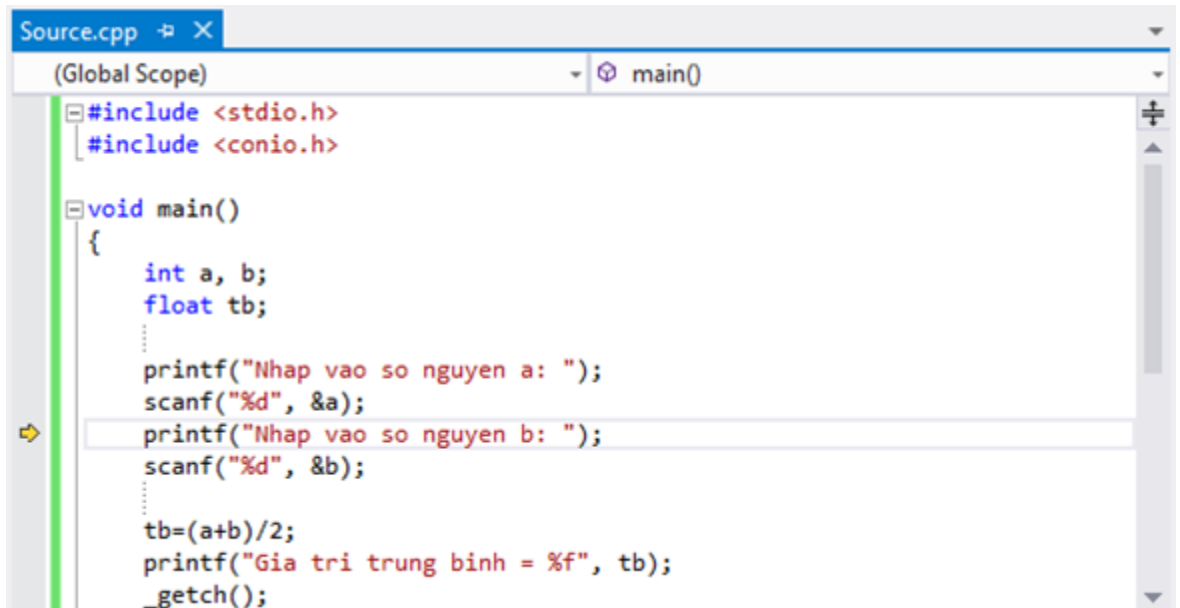
- Cửa sổ Watch gồm 3 phần: **Name** (tên biến), **Value** (giá trị hiện tại của biến) và **Type** (kiểu dữ liệu khai báo của biến).
- Để kiểm tra giá trị hiện tại của biến nào, ta chỉ việc chọn và nhập tên biến vào trong ô **Name**. Trong trường hợp này, ta cần biết giá trị của biến *a*.



- Giá trị và kiểu của biến *a* sẽ hiển thị chi tiết trong cửa sổ **Watch**, ta nhận thấy giá trị của *a* = 15 là đúng nên ta có thể kết luận lệnh nhập cho *a* là chính xác.

Watch 1		
Name	Value	Type
 a	15	int

- (iv) Tiếp tục nhấn phím **F10** tại cửa sổ lệnh, chương trình sẽ hiện tiếp dòng “*Nhap vao so nguyen b:* “ trong cửa sổ console



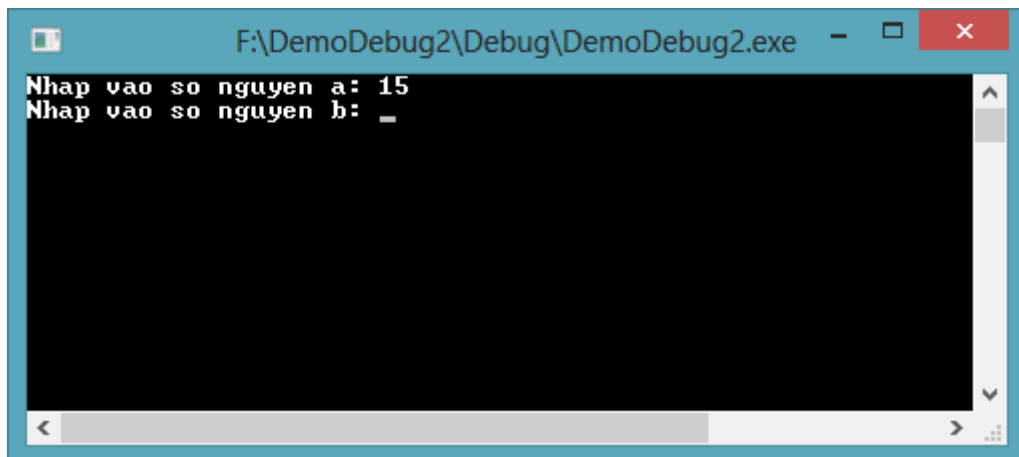
```

Source.cpp
(Global Scope) main()
#include <stdio.h>
#include <conio.h>

void main()
{
    int a, b;
    float tb;

    printf("Nhap vao so nguyen a: ");
    scanf("%d", &a);
    printf("Nhap vao so nguyen b: ");
    scanf("%d", &b);

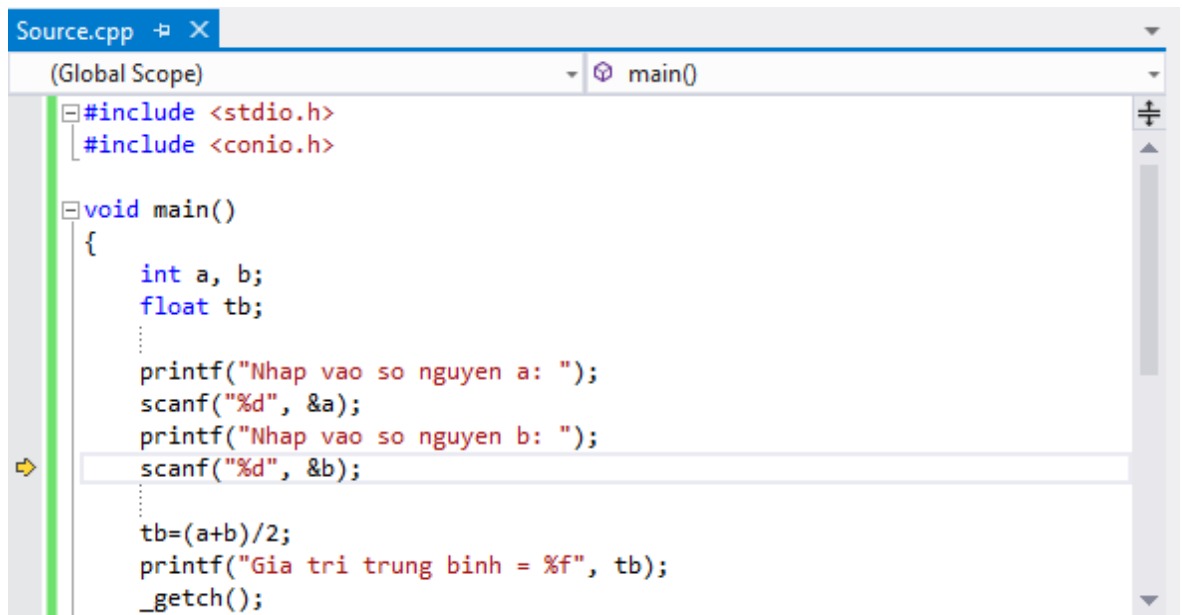
    tb=(a+b)/2;
    printf("Gia tri trung binh = %f", tb);
    _getch();
}
    
```



```

F:\DemoDebug2\Debug\DemoDebug2.exe
Nhap vao so nguyen a: 15
Nhap vao so nguyen b: _
    
```

- (v) Tiếp tục nhấn tiếp **F10** tại cửa sổ lệnh để chương trình thực hiện tiếp lệnh nhập vào số nguyên **b**.



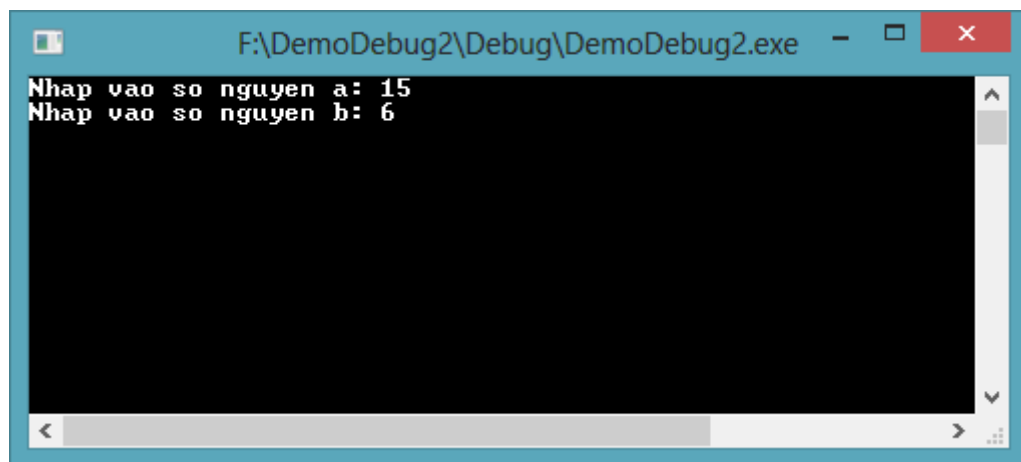
```
Source.cpp
(Global Scope) main()
#include <stdio.h>
#include <conio.h>

void main()
{
    int a, b;
    float tb;

    printf("Nhap vao so nguyen a: ");
    scanf("%d", &a);
    printf("Nhap vao so nguyen b: ");
    scanf("%d", &b);

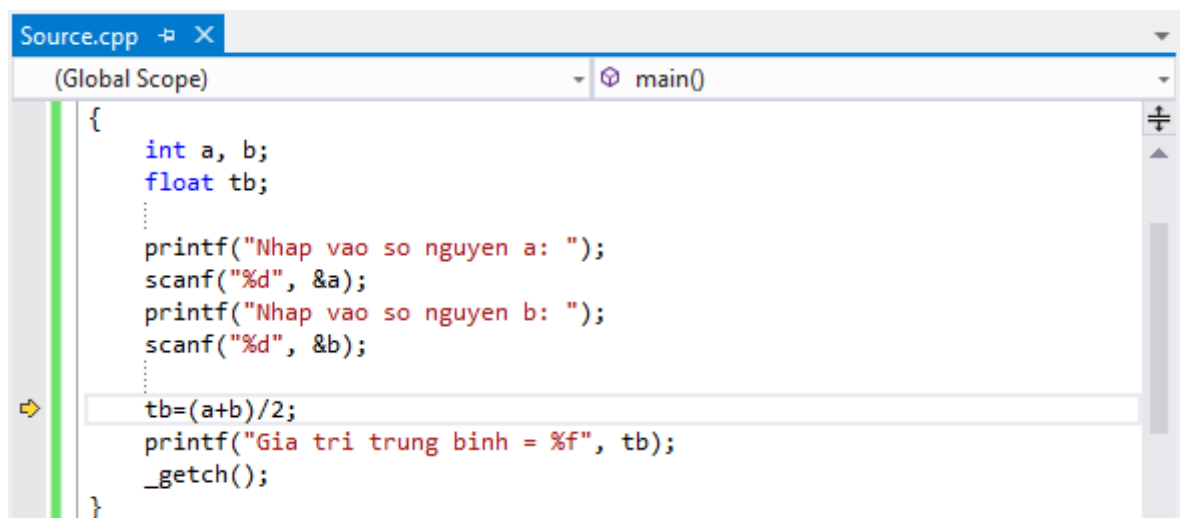
    tb=(a+b)/2;
    printf("Gia tri trung binh = %f", tb);
    _getch();
}
```

- Giả sử nhập **b** là số **6** tại cửa sổ console và nhấn **Enter**



```
F:\DemoDebug2\Debug\DemoDebug2.exe
Nhap vao so nguyen a: 15
Nhap vao so nguyen b: 6
```

- Tương tự như việc kiểm tra giá trị biến **a**, ta cũng tiến hành nhập thêm biến **b** vào cửa sổ **Watch** để chắc chắn rằng lệnh nhập cho **b** là đúng.
- Tiếp tục nhấn **F10** để thực hiện tiếp lệnh tính trung bình cộng.



```
Source.cpp
(Global Scope) main()
{
    int a, b;
    float tb;

    printf("Nhap vao so nguyen a: ");
    scanf("%d", &a);
    printf("Nhap vao so nguyen b: ");
    scanf("%d", &b);

    tb=(a+b)/2;
    printf("Gia tri trung binh = %f", tb);
    _getch();
}
```

- Tiến hành kiểm tra giá trị biến **tb** sau khi thực hiện lệnh **tb=(a+b)/2;**

The screenshot shows a C++ IDE with a source file named 'Source.cpp'. The code is as follows:

```

(Global Scope)
main()
{
    int a, b;
    float tb;

    printf("Nhap vao so nguyen a: ");
    scanf("%d", &a);
    printf("Nhap vao so nguyen b: ");
    scanf("%d", &b);

    tb=(a+b)/2;
}
    
```

Below the code, the 'Watch 1' window is open, displaying the following data:

Name	Value	Type
a	15	int
b	6	int
tb	10.0000000	float

- Sau khi kiểm tra, ta thấy biến **tb = 10**, thay vì **tb = 10.5** mới đúng.
- Do vậy, ta có thể kết luận lệnh **tb = (a+b)/2** là sai (vì đối với ngôn ngữ C, khi chia 2 số nguyên cho nhau thì kết quả thu được là số nguyên – phần nguyên sau khi chia). Lệnh này phải sửa thành **tb = (a+b)/2.0** hoặc cũng có thể viết theo cách khác **tb = (float)(a+b)/2** thì chương trình mới thực thi đúng (Chương 3 sẽ giải thích kỹ vấn đề này).

VIII. Kết luận

Sinh viên nên thực tập thực hành cài đặt các bài mẫu trên máy tính, sau đó biên dịch, quan sát kết quả.

Nếu có những thông báo lỗi khi biên dịch, sinh viên cố gắng đọc kỹ các thông báo và tìm hiểu đó là những lỗi gì và tìm cách sửa.

Với mỗi chương trình, sinh viên nên thực hiện chạy từng bước để quan sát và biết được kết quả thực hiện của từng lệnh trong chương trình

IX. Bài tập

- C2.32. Thay đổi các thông tin trong đoạn chương trình mẫu 1 thành thông tin của chính sinh viên.

C2.33. Thêm hoặc bỏ bớt ký tự `\n` hay `\t` trong chương trình mẫu 1, sau đó biên dịch, thực thi lại và quan sát kết quả thực hiện xem có thay đổi gì so với kết quả của chương trình mẫu 1.

C2.34. Dựa vào đoạn chương trình mẫu 1, xuất ra màn hình một bài thơ trình bày theo mẫu sau (không dấu tiếng Việt):

BÀI THƠ ĐI THI – Trần Tế Xương

*Táp tễnh người đi tớ cũng đi,
Cũng lều cũng chõng cũng vào thi.*

*Tiền chân, Cô mất hai đồng chân,
Sờ bụng, thầy không một chữ gì!
Lộc nước còn mong thêm giải ngạch
Phúc nhà nay được sạch trương qui.*

*Ba kỳ trọn vẹn thêm kỳ nữa,
ú ó u ơ ngọn bút chì*

C2.35. Thay đổi lệnh `tb = (a+b)/2.0` trong chương trình mẫu 2 thành `tb = (a+b)/2`, sau đó biên dịch lại, thực thi chương trình và xem kết quả.

C2.36. Thay đổi lệnh `printf("Gia tri trung binh = %f", tb)` trong chương trình mẫu 2 thành `printf("Gia tri trung binh = %d", tb)`, sau đó biên dịch lại, thực thi chương trình và xem kết quả.

C2.37. Thay đổi và bổ sung thêm lệnh trong chương trình mẫu 2 sao cho chương trình cho phép nhập vào điểm 3 môn học: toán, lý, hóa và tính điểm trung bình, in ra kết quả.

CHƯƠNG 3 CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ C

Tóm tắt: Giới thiệu các khái niệm, các từ khóa, các loại hằng số, các kiểu dữ liệu, khai báo biến, các ký hiệu phép toán và các hàm nhập xuất trong ngôn ngữ C.

I. Lịch sử ngôn ngữ C

Ngôn ngữ C do Dennis Ritchie xây dựng từ năm 1972 tại phòng thí nghiệm Bell Telephone với mục đích tạo ngôn ngữ để viết hệ điều hành UNIX, song nhờ có các tính năng ưu việt và tính mềm dẻo nên được giới tin học chấp nhận.

Năm 1978, xuất bản quyền sách “The C programming language” do B.W Kernighan và Dennis Ritchie viết và được phổ biến rộng rãi đến nay.

C là ngôn ngữ lập trình cấp cao, được sử dụng rất phổ biến để lập trình hệ thống cùng với Assembler và phát triển các ứng dụng do các đặc điểm sau:

- Bộ lệnh phù hợp với phương pháp lập trình có cấu trúc.
- Kiểu dữ liệu phong phú, cho phép định nghĩa thêm những kiểu dữ liệu mới phù hợp với yêu cầu của bài toán.
- Linh động về cú pháp và ít từ khóa.

II. Các khái niệm

II. 1. Lệnh và khối lệnh

Lệnh dùng để thực hiện một chức năng nào đó như: khai báo, gán, xuất, nhập, tính toán, v.v... và được kết thúc bằng dấu chấm phẩy (;).

Ví dụ:

```
int x;  
printf(“Xin chao”);
```

Khối lệnh là tập hợp nhiều lệnh và được đặt trong cặp dấu ngoặc { }.

Ví dụ:

```
for(int i=0; i<n; i++)  
{  
    k = k/i;  
    s = s + k;  
}
```

✎ Không nên viết nhiều lệnh trên một dòng.
 ✎ Lệnh quá dài có thể được viết thành nhiều dòng sao cho mỗi lệnh phải được quan sát trọn vẹn trong phạm vi của cửa sổ.

II. 2. Từ khóa (key word)

Từ khóa là từ có ý nghĩa xác định dùng để khai báo kiểu dữ liệu, lệnh, v.v... Tuyệt đối không được sử dụng các từ khóa này cho việc đặt tên theo mục đích riêng của người lập trình. Các từ khóa phải viết bằng chữ thường. Trong C có các từ khóa sau:

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

II. 3. Tập các ký hiệu và ký tự của C

Trong quá trình lập trình, đối với ngôn ngữ C chúng ta chỉ được sử dụng các ký hiệu và các ký tự (phân biệt chữ in hoa và in thường) như sau:

- 26 chữ cái từ a đến z (bao gồm chữ in hoa và chữ in thường).
- 10 chữ số từ 0 đến 9.
- Các ký hiệu toán học: +, -, *, /, =, <, >, (,)
- Các ký hiệu đặc biệt: : , ; " ' _ @ # \$! ^ [] { } ...
- Dấu cách hay khoảng trắng.

II. 4. Các đặt tên trong chương trình

Trong quá trình lập trình, ngoài việc sử dụng các từ khóa có sẵn của ngôn ngữ lập trình để viết lệnh, người lập trình cần sử dụng các tên riêng trong chương trình để thể hiện việc định danh cho biến, định nghĩa mới kiểu dữ liệu, các hằng, các hàm, các nhãn, v.v... Chiều dài tối đa của tên là 32 ký tự và phân biệt ký tự hoa và thường.

Tên được đặt theo quy tắc sau:

- Tên bắt đầu bằng một ký tự hoặc dấu gạch dưới (_).
- Các ký tự trong tên biến chỉ có thể là các ký tự chữ từ a đến z (ký tự in hoặc thường), số hoặc dấu gạch dưới.

- Không được đặt tên trùng với các từ khoá.
- Trong cùng phạm vi khai báo không được đặt cùng một tên.
- Tên dễ hiểu, súc tích và gợi nhớ.

✎ Ví dụ những tên hợp lệ: *tong*, *mu2*, *dientich*, *chuvi*, *giai_thua*, *_x*

✎ Những tên không hợp lệ:

<i>2mu_x</i>	<i>sai do ký tự số đứng đầu tên</i>
<i>int</i>	<i>sai do trùng với từ khóa</i>
<i>tien\$</i>	<i>sai do ký tự \$</i>
<i>thanh tien</i>	<i>sai do có khoảng trắng</i>
<i>binh-phuong</i>	<i>sai do có dấu gạch ngang (-)</i>
<i>f(x)</i>	<i>sai do có dấu ngoặc ()</i>

II. 5. Tạo ghi chú (chú thích) trong chương trình

Khi viết chương trình, người lập trình thường ghi chú lại những vấn đề cần lưu ý, giải thích về ý nghĩa các tên, ý nghĩa của các lệnh, v.v... nhằm mục đích để gọi nhớ lại cho quá trình kiểm tra lại lệnh sau này, hoặc cũng có thể giúp cho người khác có thể đọc và dễ hiểu những gì viết ra. Khi biên dịch chương trình, C sẽ bỏ qua những ghi chú này.

Nếu chỉ ghi chú trên cùng một dòng thì dùng dấu **//*nội dung ghi chú***, hoặc dùng cặp dấu **/* *nội dung ghi chú* */** đối với trường hợp cần ghi chú trên nhiều dòng.

Ví dụ:

```
/* Chương trình này dùng để nhập vào hai số nguyên a và b
   Tính tổng và xuất kết quả ra màn hình
   Viết vào ngày 20 tháng 03 năm 2010
*/
#include <stdio.h> //Khai báo thư viện cho hàm printf và scanf
#include <conio.h> //Khai báo thư viện cho hàm _getch

void main()
{
    int a, b; //Khai báo 2 số nguyên a và b
    //Lệnh nhập vào 2 số a và b
    printf("Nhập vào số nguyên a: ");
    scanf("%d", &a);
    printf("Nhập vào số nguyên b: ");
    scanf("%d", &b);
```

```
//Xuất tong ra màn hình
printf("Tong = %d", a + b);
_getch(); //Dung màn hình cho xem kết quả
}
```

II. 6. Hằng số (constant)

Hằng số là đại lượng không thể thay đổi trong suốt quá trình thực thi của chương trình. Hằng có thể là một chuỗi ký tự, một ký tự, một con số xác định.

II. 6. 1. Hằng số nguyên

Hằng số nguyên thể hiện những giá trị số rời rạc, hằng số nguyên 2 bytes được viết tự nhiên như một con số thông thường chúng ta đang sử dụng, bao gồm số âm và số dương.

Ví dụ:

1123 số dương một nghìn một trăm hai mươi ba

-768 số âm bảy trăm sáu mươi tám

Đối với hằng số nguyên 4 bytes thì viết kèm thêm ký tự (*l* hoặc *L*) vào cuối số

Ví dụ: *112300l (hoặc 112300L): số một trăm mười hai nghìn ba trăm*

Đối với những hằng số nguyên hệ bát phân (hệ 8), thập lục phân (hệ 16) thì có biểu diễn như sau:

. Hệ bát phân: 023 thêm số 0 vào trước số hệ bát phân

. Hệ thập lục phân: 0x23 thêm 0x vào trước số hệ thập lục phân

✎ Không được dùng dấu ngăn cách phần nghìn, ví dụ: số một triệu năm trăm nghìn phải viết là 1500000 (không được viết 1,500,000)

II. 6. 2. Hằng số thực

Hằng số thực thể hiện những giá trị số liên tục, giữa phần nguyên và phần lẻ cách nhau bằng dấu chấm (.).

Ví dụ:

1.23 số một phẩy hai mươi ba

-76.8 số âm bảy mươi sáu phẩy tám

Có thể biểu diễn theo hằng số mũ bằng cách thêm ký tự *e* (hoặc *E*) và số mũ sau số thực.

Ví dụ:

$123e6$ có giá trị là 123×10^6

$123e-2$ có giá trị là 123×10^{-2}

II. 6. 3. Hằng ký tự

Hằng ký tự là một ký tự riêng biệt được viết trong cặp dấu nháy đơn ('). Mỗi một ký tự tương ứng với một giá trị trong bảng mã ASCII.

Ví dụ: 'a', 'A', '0', '9', '.', ' ';

Do hằng ký tự cũng được xem như giá trị trị số nguyên nên chúng ta có thể dùng phép toán số học trên hai ký tự.

II. 6. 4. Hằng chuỗi

Hằng chuỗi ký tự là tập hợp các ký tự và được đặt trong cặp dấu nháy kép ("). Khi lưu chuỗi trong bộ nhớ thì cuối chuỗi có ký tự NULL gọi là ký tự kết thúc chuỗi '\0' có mã ascii là 0.

Ví dụ: "Ngon ngu lap trinh C", "Nguyen Van A"

Chuỗi có thể không có ký tự nào hoặc chỉ có một ký tự duy nhất.

Ví dụ:

"" chuỗi rỗng

"a" chuỗi chỉ có một ký tự a

" " chuỗi chỉ có một ký tự là khoảng trắng

Đối với chuỗi có ký tự đặc biệt như: dấu nháy đơn ('), dấu nháy kép ("), dấu '\', v.v... thì phải kèm thêm ký tự '\' phía trước

Ví dụ: chuỗi "C:\BaiTap\bt1.doc" thì phải viết là "C:\\BaiTap\\bt1.doc"

II. 7. Kiểu dữ liệu cơ bản

Kiểu dữ liệu của ngôn ngữ lập trình thường gồm hai dạng kiểu dữ liệu: kiểu dữ liệu cơ bản (tích hợp sẵn trong ngôn ngữ) và kiểu dữ liệu do người lập trình định nghĩa thêm.

Mỗi kiểu dữ liệu có kích thước và miền giá trị lưu trữ khác nhau, do vậy khi lập trình chúng ta cần nắm rõ các thông tin này để có thể sử dụng đúng kiểu và lưu trữ chính xác dữ liệu.

Đối với ngôn ngữ C, kiểu dữ liệu cơ bản bao gồm: kiểu số nguyên (giá trị rời rạc) và kiểu số thực (giá trị liên tục). Để thể hiện kiểu rỗng (không chứa giá trị gì cả) ta dùng kiểu **void**.

II. 7. 1. Kiểu số nguyên

Kiểu số nguyên là kiểu dữ liệu dùng để lưu các giá trị nguyên hay còn gọi là các giá trị rời rạc.

Stt	Tên kiểu	Ý nghĩa	Kích thước	Miền giá trị
1	char	Ký tự	1 byte	Từ -128 đến 127
		Số nguyên 1 byte	1 byte	
2	unsigned char	Số nguyên dương	1 byte	Từ 0 đến 255 (<i>tương đương 256 ký tự trong bảng mã ASCII</i>)
3	int	Số nguyên	2 bytes	Từ -32,768 đến 32,767
4	unsigned int	Số nguyên dương	2 bytes	Từ 0 đến 65,535
5	long	Số nguyên	4 bytes	Từ -2,147,483,648 đến 2,147,483,647
6	unsigned long	Số nguyên dương	4 bytes	Từ 0 đến 4,294,967,295

II. 7. 2. Kiểu số thực

Kiểu số thực là kiểu dữ liệu dùng để lưu các giá trị thực (các số có dấu chấm thập phân) hay còn gọi là các giá trị liên tục.

Mỗi kiểu số thực sẽ có độ chính xác khác nhau, nghĩa là số chữ số phía sau phần dấu chấm thập phân:

- Độ chính xác đơn: có tối đa 7 chữ số
- Độ chính xác kép: có tối đa 15 chữ số

Stt	Tên kiểu	Ý nghĩa	Kích thước	Miền giá trị
1	float	Số thực độ chính xác đơn	4 bytes	Từ 3.4×10^{-38} đến 3.4×10^{38}
2	double	Số thực độ chính xác kép	8 bytes	Từ 1.7×10^{-308} đến 1.7×10^{308}
3	long double	Số thực độ chính xác kép	10 bytes	Từ 3.4×10^{-4932} đến 1.1×10^{4932}

II. 8. Biến và khai báo biến

II. 8. 1. Biến

(a) Khái niệm

Biến là đại diện cho một vùng nhớ hay tập các vùng nhớ trên bộ nhớ chính của máy tính. Tên biến được dùng để tham khảo đến những vùng nhớ này.

Biến để lưu trữ các giá trị do người dùng nhập vào hoặc các giá trị tạm thời trong quá trình tính toán.

Mỗi biến sẽ có tên và kiểu dữ liệu tương ứng. Kiểu dữ liệu của biến xác định những giá trị kiểu nào có thể được lưu trong biến (ví dụ số hay chữ.v...).

PHẢI khai báo biến trước khi sử dụng.

(b) Cách đặt tên biến

Các biến được đặt tên theo quy ước đặt tên và theo quy tắc Hungarian Notation. Nghĩa là tên biến nên kèm theo ký tự đầu của kiểu dữ liệu, nhằm mục đích là để nhận biết kiểu dữ liệu của biến.

Tên biến thường viết bằng chữ in thường.

Ví dụ: *ituoi* có thể nhận biết được *tuoi* có kiểu số nguyên (ký tự *i* trong kiểu *int*), *fdientich* có thể nhận biết được *dientich* có kiểu số thực (ký tự *f* trong kiểu *float*)

II. 8. 2. Khai báo biến

(a) Cú pháp

<Kiểu dữ liệu> tênbiến;

Ví dụ:

- Khai báo biến *a* để lưu số nguyên 2 bytes
int ia;
- Khai báo biến *tb* để lưu số thực 4 bytes
float ftb;

(b) Khai báo nhiều biến cùng một kiểu dữ liệu

<Kiểu dữ liệu> tênbiến1, tênbiến2, tênbiến3;

Ví dụ:

- Khai báo các biến a, b, c cùng kiểu số nguyên 2 bytes
 $int\ ia, ib, ic;$
- Khai báo các biến x, y, z có cùng kiểu số thực 4 bytes
 $float\ fx, fy, fz;$

(c) **Khai báo và gán giá trị ban đầu cho biến**

Sử dụng khi khai báo biến và gán giá trị ban đầu cho biến, sử dụng hằng số để gán cho biến có kiểu dữ liệu tương ứng.

Cú pháp:

$\langle \text{Kiểu dữ liệu} \rangle\ \text{tênbiến} = \text{giá trị};$

Ví dụ:

```
int ia = 5;  
float fb = 5.4, c = 9.2;  
char cc = 'n';
```

(d) **Ép kiểu cho biến**

Mục đích là chuyển kiểu dữ liệu của một biến sang kiểu dữ liệu khác cho phù hợp với biến của kiểu dữ liệu cần gán.

Cú pháp:

$(\text{Kiểu dữ liệu})\ \text{tênbiến};$
--

Ví dụ:

```
float fk = 3.5;  
int ia = (int)fk; //Ép fk về phần nguyên nên ia sẽ có giá trị bằng 3.
```

(e) **Phạm vi của biến**

Tùy theo vị trí khai báo biến, mỗi biến sẽ có phạm vi (tác dụng khác nhau).

Phạm vi cục bộ: biến được khai báo bên trong hàm hay trong khối lệnh, những biến này chỉ có ảnh hưởng bên trong hàm hoặc bên trong khối lệnh.

Phạm vi toàn cục: biến được khai báo bên ngoài hàm, những biến này có ảnh hưởng đến toàn bộ chương trình.

(f) **Định nghĩa tên hằng số**

Hằng số là đại lượng không thay đổi giá trị trong quá trình thi hành chương trình. Có hai cách để định nghĩa hằng số: Dùng toán tử `define` hoặc `const`.

- Dùng toán tử **#define**

Cú pháp:

```
#define <tên_hằng> <giá trị hằng>
```

Ví dụ: Định nghĩa hằng số *MAX* có giá trị là 100

```
#define MAX 100
```

- Dùng từ khoá **const**

Cú pháp:

```
const <Kiểu dữ liệu> <tên_biến> = <giá trị hằng>;
```

Ví dụ: Định nghĩa hằng số *MAX* có giá trị là 100

```
const int MAX = 100;
```

☞ Tên hằng số nên đặt tên bằng chữ in HOA.

III. Ký hiệu các phép toán

III. 1. Phép toán số học

Được sử dụng trong các phép toán số học trên các kiểu dữ liệu cơ bản của C.

Stt	Ký hiệu	Ý nghĩa	Ghi chú
1	+	Phép cộng	
2	-	Phép trừ	
3	*	Phép nhân	
4	/	Phép chia	Đối với 2 số nguyên thì kết quả là chia lấy phần nguyên
5	%	Phép chia lấy phần dư	Chỉ áp dụng cho 2 số nguyên
6	++	Tăng một đơn vị nguyên	Nếu toán tử tăng giảm đặt trước thì tăng giảm trước rồi tính biểu thức hoặc ngược lại.
7	--	Giảm một đơn vị nguyên	

Một số ví dụ:

Ví dụ	Ý nghĩa
<code>int ix = 5/2;</code>	$ix = 2$, do 5 và 2 đều là số nguyên nên phép chia / được xem là phép chia lấy phần nguyên
<code>int ix = 5%2;</code>	$ix = 2$, do phép chia % là phép chia lấy phần dư
<code>float fy = 5/2;</code>	$fy = 2.0$, do 5 và 2 đều là số nguyên nên phép chia / được xem là phép chia lấy phần nguyên
<code>float fy = 5/2.0</code>	$fy = 2.5$, do 2.0 là hằng số thực nên phép chia / được xem là phép chia bình thường
<code>int ia = 5;</code> <code>ia++;</code>	$ia = ia + 1 = 6$
<code>int ia = 5;</code> <code>++ia;</code>	$ia = ia + 1 = 6$, do chỉ có một biến ia nên ký hiệu ++ nằm phía trước hay phía sau đều như nhau
<code>int ia = 5;</code> <code>int ib = ia++ + 7;</code>	<ul style="list-style-type: none"> - Tính $ib = ia + 7 = 5 + 7 = 12$ trước, do ký hiệu ++ nằm sau ia - Sau đó tính $ia++ = ia + 1 = 5 + 1 = 6$
<code>int ix = 5, iy = 11;</code> <code>int iz = --ix + iy++;</code>	<ul style="list-style-type: none"> - Tính $ix = ix - 1 = 4$ trước, do ký hiệu -- nằm trước ix - Sau đó tính $iz = ix + iy = 4 + 11 = 15$ - Cuối cùng tính $iy = iy + 1 = 11 + 1 = 12$, do dấu ++ nằm sau iy

III. 2. Phép toán so sánh và kết hợp so sánh

Được sử dụng trong các biểu thức điều kiện, kết quả của phép toán so sánh là đúng hoặc sai. Trong ngôn ngữ C mọi giá trị khác 0 được gọi là đúng, còn sai là 0.

Stt	Ký hiệu	Ý nghĩa
1	>	Lớn hơn
2	<	Nhỏ hơn
3	>=	Lớn hơn hoặc bằng

4	<=	Nhỏ hơn hoặc bằng
5	= =	Bằng nhau
6	!=	Khác nhau
7	!	Phủ định phép so sánh
8	&&	Kết hợp theo nguyên tắc AND các phép so sánh
9		Kết hợp theo nguyên tắc OR các phép so sánh

III. 3. Phép toán trên bit

Stt	Ký hiệu	Ý nghĩa
1	&	AND
2		OR
3	^	XOR
4	~	NOT
5	>>	Dịch phải
6	<<	Dịch trái

III. 4. Toán tử điều kiện

Cú pháp:

(Biểu thức điều kiện)? <Biểu thức 1>: <Biểu thức 2>;

Ý nghĩa:

- Nếu biểu thức điều kiện cho kết quả đúng thì sẽ thực hiện Biểu thức 1.
- Ngược lại thì sẽ thực hiện Biểu thức 2.

Ví dụ:

int n;

(n%2==0)? n++ : n--;

→ nếu n = 10 thì giá trị n = 11

→ nếu n = 21 thì giá trị n = 20

⚠ *Biểu thức biểu kiện phải dùng ký hiệu so sánh, nhất là ký hiệu so sánh phép bằng nhau (==), sinh viên thường hay sai do viết chỉ có 1 dấu bằng (dấu = là phép gán của C)*

III. 5. Viết tắt phép toán

Nhằm rút gọn biểu thức tính toán hai ngôi, ngôn ngữ C cho phép viết tắt nếu cả hai vế có cùng tên biến.

Cú pháp:

(Biến) = (Biến) (Toán tử) (Biểu thức);
có thể viết
(Biến) (Toán tử) = (Biểu thức);

Ví dụ:

$x = x + 5$; có thể viết tắt thành $x += 5$;

$y = y * 10$; có thể viết tắt thành $y *= 10$;

III. 6. Thứ tự ưu tiên các phép toán

Trong biểu thức có kết hợp nhiều phép toán thì ngôn ngữ C sẽ dựa vào bảng thứ tự ưu tiên để xác định thứ tự thực hiện.

Toán tử	Độ ưu tiên	Trình tự kết hợp
() [] ->	1	Từ trái qua phải
! ~ ++ -- - + * & sizeof	2	Từ phải qua trái
* / %	3	Từ trái qua phải
+ -	4	Từ trái qua phải
<< >>	5	Từ trái qua phải
< <= >= >	6	Từ trái qua phải
== !=	7	Từ trái qua phải
&	8	Từ trái qua phải
	9	Từ trái qua phải

^	10	Từ trái qua phải
&&	11	Từ trái qua phải
	12	Từ trái qua phải
? :	13	Từ phải qua trái
= += -= *= /= %=	14	Từ phải qua trái

IV. Hàm nhập xuất dữ liệu: printf và scanf

Dùng để nhập dữ liệu từ bàn phím và xuất kết quả ra màn hình. Khi nhập hay xuất dữ liệu, người lập trình phải mô tả định dạng chính xác của dữ liệu cần nhập hoặc xuất.

Các hàm này sử dụng thư viện hàm `<stdio.h>`, do vậy đầu chương trình phải khai báo thư viện hàm: `#include <stdio.h>`

IV. 1. Chuỗi định dạng

Stt	Kiểu	Chuỗi định dạng
1	char	%c
		%d
2	unsigned char	%d
3	int	%d
4	unsigned int	%u
5	long	%ld
6	unsigned long	%lu
7	char *	%s
8	float	%f
9	double	%lf
10	long double	%lf

IV. 2. Hàm xuất dữ liệu ra màn hình: printf

Xuất dữ liệu ra màn hình có hai dạng: Xuất chuỗi hoặc kết hợp xuất chuỗi với giá trị của các biến.

(a) Xuất chuỗi

Cú pháp:

```
printf("hàng chuỗi");
```

Ví dụ:

```
printf("Xin chao cac ban");
```

(b) Xuất chuỗi kèm giá trị biến

Cú pháp:

```
printf("chuỗi định dạng", đối số 1, đối số 2);
```

Ví dụ:

```
int ia=5;
```

```
float fb=2.7;
```

```
printf("Gia tri cua bien ia=%d, fb=%f", ia, fb);
```

Có thể thêm số nguyên, số thực phía trước chuỗi định dạng với mục đích là định dạng độ rộng hoặc là làm tròn số thực

Ví dụ: dành 4 vị trí để xuất giá trị ia, ib xuất 2 chữ số có số 0 đầu nếu nhỏ hơn 10 và fc xuất làm tròn 2 chữ số phần thập phân:

```
int ia=5, ib = 2;
```

```
float fc=2.793221;
```

```
printf("Gia tri cua bien ia=%2d, ib=%02d, fc=%3.2f", ia, ib, fc);
```

➤ Thứ tự chuỗi định dạng và thứ tự của biến cần xuất giá trị phải tương ứng nhau.

(c) Xuất các ký tự đặc biệt

Ký tự	Ý nghĩa	Ví dụ
\'	Xuất dấu nháy đơn	printf("\' "); Kết quả: ' ';

<code>\'</code>	Xuất dấu nháy đôi	<code>printf("\' ");</code> Kết quả: "
<code>\\</code>	Xuất dấu chéo ngược <code>"\"</code>	<code>printf("\\ ");</code> Kết quả: \
<code>\t</code>	Tab vào một đoạn ký tự trắng	<code>printf("xyz\tzyx");</code> Kết quả: xyz zyx
<code>\n</code>	Xuống dòng	<code>printf("xyz\nzyx");</code> Kết quả: xyz zyx

IV. 3. Hàm nhập dữ liệu: `scanf`

Dùng nhập dữ liệu vào biến từ bàn phím theo định dạng của kiểu dữ liệu.

Cú pháp:

```
scanf("chuỗi định dạng", &tên biến);
```

Ví dụ:

```
int x;
printf("Hay nhap vao so nguyen x: ");
scanf("%d", &x);
```

⚠ Khi sử dụng hàm `scanf()` nên có câu thông báo xuất phía trước để người sử dụng chương trình biết được dữ liệu cần nhập là gì.

IV. 4. Ví dụ hàm nhập xuất

Chương trình nhập vào số nguyên và in giá trị bình phương của số vừa nhập:

```
#include <stdio.h>
void main ()
{
    int n;
    printf("Nhap mot so nguyen: ");
    scanf("%d", &n);
    printf("Binh phuong so vua nhap = %d", n*n);
}
```

Kết quả minh họa:

Nhap mot so nguyen: 4

Binh phuong so vua nhap = 16

V. Các hàm cơ bản khác

Giới thiệu một số hàm cơ bản cần sử dụng trong một số bài tập của giáo trình. Các hàm này đều nằm trong thư viện <math.h>

Stt	Tên hàm	Ý nghĩa	Ví dụ	Kết quả
1	abs	Tính trị tuyệt đối của một số	<i>int iy = abs(-7);</i>	<i>iy = -7 = 7</i>
2	pow	Tính mũ	<i>int iy = pow(2, 3);</i>	<i>iy = 2³ = 8</i>
3	sqrt	Tính căn bậc hai của x	<i>float fcan = sqrt(4);</i>	<i>fcan = $\sqrt{4} = 2$</i>

VI. Kết luận

Với những khái niệm cơ bản của chương này, sinh viên có thể áp dụng viết những chương trình C cơ bản dùng cấu trúc tuần tự (thực thi các lệnh từ trên xuống dưới trong hàm main()).

Việc trình bày chương trình cũng có ý nghĩa rất quan trọng trong quá trình lập trình, chương trình phải thể hiện phân cấp theo cấu trúc lệnh con bên trong phải được thụt lùi vào trong một khoảng. Điều này giúp ích cho chúng ta sau này xem xét lại lệnh và phân tích lỗi nếu xảy ra sai sót về mặt xử lý.

Vị trí khai báo biến trong ngôn ngữ C có thể linh động khai báo bất kỳ vị trí nào, sao cho trước khi sử dụng biến là được. Tuy nhiên, để dễ quản lý các biến ta nên khai báo tập trung ở phần đầu của đoạn chương trình.

Khi khai báo biến phải dùng đúng kiểu dữ liệu, kích thước và định dạng phù hợp với dữ liệu cần lưu trữ của bài toán, tránh sử dụng phép gán cho các biến khác kiểu.

Các lỗi thường mắc phải trong khi sử dụng hàm printf() và scanf() là sai chuỗi định dạng của biến, thiếu dấu & trước tên biến trong hàm scanf()

Viết sai

*(Kiểu int chuỗi định dạng là %d
Hàm scanf thiếu dấu & trước ix)*

```
int ix;
printf("Nhap vao so nguyen: ");
scanf("%f", ix);

printf("Gia tri vua nhap: %f", ix);
```

Viết đúng

```
int ix;

printf("Nhập vào số nguyên: ");

scanf("%d", &ix);

printf("Giá trị vừa nhập: %d", ix);
```

VII. Bài tập

C3.38. Cho biết những lỗi và sửa lại cho đúng của đoạn chương trình sau:

```
int long = -100;
unsigned int i = -100;
int k = 2.9;
long m = 2, p = 4;
int 2k;
float y = y * 2;
char ch = "b";
```

C3.39. Viết chương trình in ra màn hình thông tin cá nhân của sinh viên theo mẫu sau:

```
Ho ten: NGUYEN VAN A
Lop: DHTH11E
Thông tin liên lạc:
    Địa chỉ: Số 12, Phường 4, Gò Vấp
    Số điện thoại: 0901234567
```

C3.40. Cho biết kết quả sau khi thực hiện lệnh sau:

```
int a, b;
b=a++ + ++a + --a;
Với a = 2 Kết quả: a=?, b=?
Với a = 9 Kết quả: a=?, b=?
```

C3.41. Viết chương trình xuất ra màn hình theo mẫu sau:

```

/*****/
/***** HUONG DAN CHEP TAP TIN *****/
/=>B1. Vào thư mục "C:\TUYENTAP\thotinh.txt"/
/=>B2. Click chuột phải vào tập tin thotinh.txt /
/=>B3. Chọn copy từ menu tại /
/=>B4. Chọn vị trí cần lưu, click phải chọn paste /
/*****/
```

- C3.42. Viết chương trình nhập vào tổng số giây, đổi sang giờ, phút, giây và xuất kết quả ra màn hình theo dạng giờ:phút:giây (nếu số có một chữ số thì xuất thêm số 0 ở đầu – Ví dụ: 03:20:04).
- C3.43. Viết chương trình nhập vào 2 số nguyên dương a và b, cho biết kết quả chia lấy phần nguyên và phần dư của a với b.
- C3.44. Viết chương trình nhập một số nguyên dương N có 2 chữ số từ bàn phím, xuất ra màn hình tổng các chữ số của N.
Ví dụ: Nhập N = 48, kết quả in ra màn hình là: 4+8=12
- C3.45. Viết chương trình cho phép nhập vào một số đo nhiệt độ theo độ Fahrenheit và xuất ra nhiệt độ tương đương của nó theo độ Celsius, sử dụng công thức chuyển đổi: $^{\circ}C = \frac{5}{9}(^{\circ}F - 32)$
- C3.46. Viết chương trình cho phép nhập vào giờ, phút và giây, hãy đổi sang giây và in kết quả ra màn hình.
- C3.47. Viết chương trình cho phép nhập vào thời gian của một công việc nào đó tính bằng giây. Hãy chuyển đổi và in ra màn hình thời gian trên dưới dạng bao nhiêu giờ, bao nhiêu phút, bao nhiêu giây.
- C3.48. Viết chương trình nhập vào 2 số nguyên a, b. Tính tổng, hiệu, tích, thương của 2 số trên và in kết quả ra màn hình.
- C3.49. Viết chương trình nhập vào 4 số nguyên a, b, c, d. Tính giá trị trung bình cộng của 4 số trên và in kết quả ra màn hình.

CHƯƠNG 4 CẤU TRÚC ĐIỀU KHIỂN

Tóm tắt: Tìm hiểu và cài đặt các cấu trúc rẽ nhánh, lựa chọn, lặp trong ngôn ngữ C. Mô tả cách hoạt động của các cấu trúc này và hướng dẫn chạy từng bước chương trình.

I. Cấu trúc cơ bản của chương trình C

Các thành phần	Mô tả	Bắt buộc
<code>#include <tên tập tin thư viện></code> <code>#define ...</code>	Tiền xử lý: - Mô tả thư viện hàm sẽ được dùng - Định nghĩa các hằng số	Có
<i>Khai báo và định nghĩa các kiểu dữ liệu (nếu có)</i>		Không
<code>void main()</code> <code>{</code> <i>Các lệnh;</i> <code>}</code>	Điều khiển mọi hoạt động của chương trình, bao gồm: khai báo biến, gọi hàm, v.v.... Các lệnh hoạt động theo thứ tự từ trên xuống dưới	Có

Như vậy, một chương trình cơ bản bắt buộc phải có hai thành phần là: Phần khai báo thư viện hàm và hàm main().

II. Cấu trúc điều khiển

II. 1. Cấu trúc rẽ nhánh

Cấu trúc rẽ nhánh chỉ cho máy tính chọn thực hiện một dãy lệnh nào đó dựa vào kết quả của biểu thức điều kiện (*biểu thức quan hệ hay biểu thức so sánh*)

II. 1. 1. Cấu trúc if

Chỉ xét một trường hợp cho biểu thức điều kiện đúng.

Cú pháp:

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

Nếu biểu thức điều kiện cho kết quả khác không thì thực hiện lệnh hoặc khối lệnh bên trong lệnh *if*.

Ví dụ:

```
#include <conio.h>  
#include <stdio.h>  
void main ()  
{  
    float fnumber ;  
  
    printf ( "Nhap mot so trong khoang tu 1 den 10 => " );  
    scanf ( "%f", &fnumber );  
    if (fnumber >5)  
    {  
        printf ( "So ban nhap lon hon 5. \n" );  
    }  
    printf ( "%f la so ban nhap. " , fnumber);  
}
```

✎ Mặc dù ngôn ngữ C chỉ quy định khi có nhiều lệnh bên trong một cấu trúc điều khiển thì mới dùng cặp dấu ngoặc {}, Tuy nhiên, nhằm mục đích trình bày chương trình cho rõ ràng ta nên sử dụng cặp dấu ngoặc {} cho lệnh đơn bên trong cấu trúc điều khiển.

- Ví dụ thay vì viết:

```
if (fnumber >5)  
    printf( "So ban nhap lon hon 5. \n");
```

- Chúng ta nên viết:

```
if (fnumber >5)  
{  
    printf( "So ban nhap lon hon 5. \n");  
}
```


II. 1. 2. Cấu trúc if ... else

Xét hai trường hợp của biểu thức điều kiện.

Cú pháp:

```
if (biểu thức điều kiện)  
{  
    <lệnh hoặc khối lệnh 1>;  
}  
else  
{  
    <lệnh hoặc khối lệnh 2>;  
}
```

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

Ví dụ: Nhập vào số nguyên a và b , nếu a là bội số của b thì in thông báo “ a là bội số của b ”, ngược lại in “ a không là bội số của b ”

```
#include <conio.h>  
#include <stdio.h>  
void main()  
{  
    int a, b;  
    printf("Nhap vao a: ");  
    scanf("%d", &a);  
    printf("Nhap vao b: ");  
    scanf("%d", &b);  
    if(a%b==0)  
    {  
        printf("a la boi so cua b");  
    }  
    else  
    {  
        printf("a khong la boi so cua b");  
    }  
}
```

II. 1. 3. Cấu trúc if ... else lồng nhau

Nếu cần xét nhiều trường thì có thể sử dụng cấu trúc if...else lồng nhau.

Ví dụ: Giải và biện luận phương trình bậc nhất: $ax+b=0$

```
#include <conio.h>
#include <stdio.h>
void main ()
{
    float a, b;
    printf ( "\n Nhập vào a: ");
    scanf ( "%f", &a);
    printf ( " Nhập vào b: ");
    scanf ( "%f", &b) ;
    if (a= = 0)
    {
        if (b= = 0)
        {
            printf ( " \n PTVSN");
        }
        else
        {
            printf ( " \n PTVN");
        }
    }
    else
    {
        printf ( " \n Nghiệm x=%f", -b/a);
    }
    _getch ();
}
```

II. 2. Cấu trúc lựa chọn switch...case

Sử dụng cấu trúc này khi cần thực hiện một khối lệnh của một trường hợp trong nhiều trường hợp, mỗi trường hợp tương ứng với một giá trị nguyên hay một ký tự cụ thể.

Cú pháp:

switch (biểu thức)

```
{  
    case n1:  
        các câu lệnh ;  
        break ;  
    case n2:  
        các câu lệnh ;  
        break ;  
    .....  
    case nk:  
        <các câu lệnh> ;  
        break ;  
    [default: các câu lệnh]  
}
```

- n_i là các **hằng số nguyên hoặc ký tự**.
- Phụ thuộc vào giá trị của biểu thức viết sau **switch**, nếu:
 - Giá trị này = n_i thì thực hiện câu lệnh sau case n_i .
 - Khi giá trị biểu thức không thỏa tất cả các n_i thì thực hiện câu lệnh sau **default** nếu có, hoặc thoát khỏi câu lệnh **switch**.
 - Khi chương trình đã thực hiện xong câu lệnh của **case** n_i nào đó thì nó sẽ thực hiện luôn các lệnh thuộc **case** bên dưới nó mà không xét lại điều kiện (do các n_i được xem như các nhãn) → Vì vậy, để chương trình thoát khỏi lệnh **switch** sau khi thực hiện xong một trường hợp, ta dùng lệnh **break**.

Ví dụ: Nhập vào một số nguyên tương ứng với tháng trong năm, in ra tên tháng bằng tiếng Anh.

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

void main()
{
    int ichon ;

    printf("***THUC DON***");
    printf("\n1. Lau thai!");
    printf("\n2. Nuoc ngot!");
    printf("\n3. Ca loc hap bau!");
    printf("\n4. Chuot dong!");
    printf("\nn ==>> Xin moi ban chon mon an: ");
    scanf ("%d",&ichon);

    switch (ichon)
    {
        case 1:
            printf("\nBan chon lau thai!");
            break;
        case 2:
            printf("\nBan chon nuoc ngot!");
            break;
        case 3:
            printf("\nBan chon ca loc hap bau!");
            break;
        case 4:
            printf("\nBan chon chuot dong!");
            break;
        default:
            printf("\nBan chon khong dung!");
    }

    getch();
}
```

II. 3. Cấu trúc lặp

Cấu trúc lặp dùng để thực hiện một lệnh hay khối lệnh nhiều lần tùy thuộc vào điều kiện của vòng lặp. Ngôn ngữ C có hai dạng cấu trúc lặp: lặp kiểm tra điều kiện trước khi lặp (for, while) và lặp kiểm tra điều kiện sau (do...while).

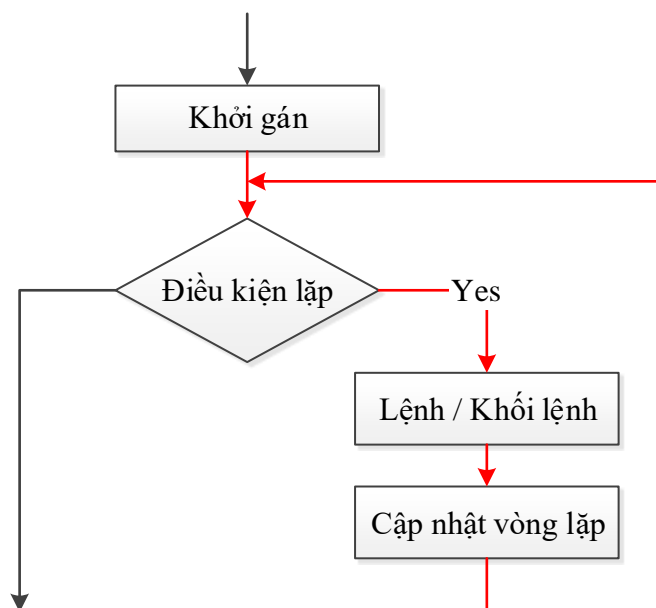
II. 3. 1. Cấu trúc lặp for

Cú pháp:

```
for (<Khởi gán>; <Điều kiện lặp>; <Cập nhật vòng lặp>)
{
    <khối lệnh>;
}
```

Bất kỳ biểu thức nào trong 3 biểu thức nói trên đều có thể vắng nhưng **phải giữ dấu chấm phẩy (;).**

Hoạt động của cấu trúc điều khiển for:



Bước 1: Khởi gán

Bước 2: Kiểm tra điều kiện lặp

- Nếu **đúng** thì cho thực hiện các lệnh của vòng lặp, thực hiện cập nhật vòng lặp. Quay trở lại bước 2.
- Ngược lại thoát khỏi lặp.

Ví dụ: In ra màn hình bảng mã ASCII từ ký tự số 33 đến 255.

```
#include<conio.h>
#include<stdio.h>
void main()
{
    for (int i=33;i<=255;i++)
    {
        printf("Ma ASCII cua %c: %d\t", i, i) ;
    }
    _getch() ;
}
```

II. 3. 2. Cấu trúc lặp while

Cú pháp:

```
< Khởi gán>;
while (<Điều kiện lặp>)
{
    lệnh/ khối lệnh;
    <Cập nhật vòng lặp>;
}
```

🔗 **Lưu ý:** Cách hoạt động của **while** giống như cấu trúc **for**

Ví dụ: Tính giá trị trung bình các chữ số của số nguyên n gồm k chữ số.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    long n, tong=0;
    int sochuso=0;
    float tb;

    printf("Nhap vao gia tri n gom k chu so");
    scanf("%ld",&n);
```

```
while(n>0)
{
    tong=tong+n%10;
    sochuso++;
    n=n/10;
}

tb=1.0*tong/sochuso;
printf("Gia tri trung binh la: %f", tb);

_getch ();
}
```

II. 3. 3. Cấu trúc lặp do...while

Cú pháp:

```
<Khởi gán>;
do
{
    < lệnh hoặc khối lệnh>;
    <Cập nhật vòng lặp>;
} while (<Điều kiện lặp>;
```

Thực hiện khối lệnh bên trong vòng lặp cho đến khi gặp điều kiện sai thì dừng.

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

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

void main()
{
    int ma ;
    do{
        ma=_getch();
        if (ma !=27)
        {
            printf ("Ma ASCII %c:%d\t", ma, ma);
        }
    }
```

```
    } while(ma!=27);  
  
    _getch();  
}
```

☞ Lặp **while** kiểm tra điều kiện trước khi thực hiện lặp, còn vòng lặp **do...while** thực hiện lệnh lặp rồi mới kiểm tra điều kiện. Do đó vòng lặp **do...while** thực hiện lệnh ít nhất một lần.

II. 4. Lệnh break và continue

II. 4. 1. Lệnh break

Dùng để kết thúc vòng lặp trực tiếp chứa nó khi thỏa một điều kiện nào đó.

Ví dụ: Cho phép người dùng nhập liên tục giá trị n cho đến khi nhập âm thì dừng.

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


II. 4. 2. Lệnh continue

Dùng để bỏ qua một lần lặp khi thỏa điều kiện nào đó.

Ví dụ: In ra màn hình giá trị từ 10 đến 20 trừ đi số 13 và số 17.

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

void main()
{
    for(int i=10 ; i<=20; i++)
    {
        if(i==13 || i==17)
        {
            continue;
        }
        printf("%d\t", i);
    }
    _getch();
}
```

III. Phương pháp kiểm tra từng bước để tìm kết quả chương trình

Bước 1: Xác định chương trình có sử dụng những biến nào.

Bước 2: Giá trị ban đầu của mỗi biến.

Bước 3: Những biến nào sẽ bị thay đổi trong quá trình chạy chương trình thì lập thành bảng có dạng sau:

Bước (Hoặc lần thực hiện)	Biến 1	Biến 2	...	Biến n	Kết quả in ra màn hình
0	Giá trị 0	Giá trị 0	...	Giá trị 0	
1	Giá trị 1	Giá trị 1	...	Giá trị 1	
2	Giá trị 2	Giá trị 2	...	Giá trị 2	
...	
...	

☞ Lưu ý xét kỹ từng lệnh và biểu thức điều kiện trong đoạn chương trình

Ví dụ: Cho biết kết quả của đoạn chương trình sau:

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

Chương trình gồm 2 biến i và a, chỉ có biến i có giá trị thay đổi trong quá trình chạy chương trình nên ta lập bảng sau:

a có giá trị là 4

Bước thực hiện	Giá trị của biến i	Kết quả in ra màn hình
0	0	0
1	1	0 1
2	2	0 1 2
3	3	0 1 2 3
4	4	Kết thúc

Tại bước 4, giá trị của i = 4 vi phạm điều kiện lặp (i < a) nên vòng lặp kết thúc.

Do đó kết quả in ra màn hình:

0
1
2
3

IV. Kết luận

Cấu trúc lặp và rẽ nhánh (lựa chọn) là hai cấu trúc chính hình thành nên chương trình. Dựa vào những cấu trúc điều khiển này ta có thể xây dựng thành những chương

trình phức tạp hơn. Vì vậy phải nắm rõ cách hoạt động của những cấu trúc điều khiển này để cài đặt đúng yêu cầu bài toán.

Khi sử dụng phải lưu ý điều kiện thực hiện hay kết thúc của một thao tác nào đó.

Bên trong một phát biểu điều khiển phải là một lệnh hay một khối lệnh (*khối lệnh được đặt bên trong cặp dấu ngoặc {}*).

Những biến không phụ thuộc vào vòng lặp nên đặt bên ngoài vòng lặp.

Khi sử dụng cấu trúc điều khiển lồng nhau phải lưu ý vị trí mở ngoặc hay đóng ngoặc cho hợp lý.

V. Bài tập

V.1. Bài tập cơ bản

C4.50. Cho biết kết quả của đoạn chương trình sau:

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

C4.51. Cho biết kết quả của đoạn chương trình sau:

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

C4.52. Cho biết kết quả của đoạn chương trình sau:

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

`printf("x=%d,y=%d,y/x=%d", x, y, y/x);`

- C4.53. Nhập vào hai số nguyên a, b. In ra màn hình giá trị lớn nhất.
- C4.54. Cho ba số a, b, c đọc vào từ bàn phím. Hãy tìm giá trị lớn nhất của ba số trên và in ra kết quả.
- C4.55. Cho ba số a, b, c đọc vào từ bàn phím. Hãy in ra màn hình theo thứ tự tăng dần các số. (Chỉ được dùng thêm hai biến phụ).
- C4.56. Viết chương trình nhập vào một số nguyên n gồm ba chữ số. Xuất ra màn hình chữ số lớn nhất ở vị trí nào?
Ví dụ: $n=291$. Chữ số lớn nhất nằm ở hàng chục (9).
- C4.57. Viết chương trình nhập vào số nguyên n gồm ba chữ số. Xuất ra màn hình theo thứ tự tăng dần của các chữ số.
Ví dụ: $n=291$. Xuất ra 129.
- C4.58. Nhập vào ngày, tháng, năm. Kiểm tra xem ngày, tháng, năm đó có hợp lệ hay không? In kết quả ra màn hình.
- C4.59. Nhập vào giờ, phút, giây. Kiểm tra xem giờ, phút, giây đó có hợp lệ hay không? In kết quả ra màn hình.
- C4.60. Viết chương trình nhập vào ngày, tháng, năm hợp lệ. Cho biết năm này có phải là năm nhuận hay không? In kết quả ra màn hình.
- C4.61. Viết chương trình tính diện tích và chu vi các hình: tam giác, hình vuông, hình chữ nhật và hình tròn với những thông tin cần được nhập từ bàn phím.
- C4.62. Viết chương trình tính tiền cước TAXI. Biết rằng:
- KM đầu tiên là 5000^d.
 - 200m tiếp theo là 1000^d.
 - Nếu lớn hơn 30km thì mỗi km thêm sẽ là 3000^d.
- Hãy nhập số km sau đó in ra số tiền phải trả.
- C4.63. Nhập vào 3 số nguyên dương a, b, c. Kiểm tra xem 3 số đó có lập thành tam giác không? Nếu có hãy cho biết tam giác đó thuộc loại nào? (Cân, vuông, đều, v.v...).
- C4.64. Viết chương trình nhập vào số nguyên dương n. Kiểm tra xem n có phải là số chính phương hay không? (số chính phương là số khi lấy căn bậc 2 có kết quả là nguyên).
- C4.65. Cho biết kết quả của đoạn chương trình sau:

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

C4.66. Cho biết kết quả của đoạn chương trình sau:

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

C4.67. Cho biết kết quả của đoạn chương trình sau:

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

C4.68. Cho biết kết quả của đoạn chương trình sau:

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

C4.69. Cho biết kết quả của đoạn chương trình sau:

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

}

C4.70. Cho biết kết quả của đoạn chương trình sau:

```
int a=10, s=0, i=0;
while(i<a)
{
    i++;
    if(i%2==0)
        continue;
    else
        s=s+i;
}
```

}

```
printf("s=%d", s);
```

C4.71. Cho biết kết quả của đoạn chương trình sau:

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

}

```
printf("%d", s);
```

C4.72. Viết chương trình in ra màn hình hình chữ nhật đặc và rỗng kích thước $m \times n$ (m, n nhập từ bàn phím).

Ví dụ: Nhập $m=5, n=4$

* * * * *	* * * * *
* * * * *	* * * * *
* * * * *	* * * * *
* * * * *	* * * * *
* * * * *	* * * * *

C4.73. Viết chương trình in ra màn hình tam giác vuông cân đặc và rỗng có độ cao h (h nhập từ bàn phím).

Ví dụ: Nhập $h=4$

```

      *
    *  *
  *  *  *
*  *  *  *

```

```

      *
    *  *
  *      *
*  *  *  *

```

C4.74. Viết chương trình in ra màn hình tam giác cân đặc và rỗng có độ cao h (h nhập từ bàn phím).

Ví dụ: Nhập $h=4$

```

      *
    *  *  *
  *  *  *  *
*  *  *  *  *

```

```

      *
    *      *
  *      *
*      *

```

C4.75. Viết chương trình nhập số nguyên dương n . Liệt kê n số nguyên tố đầu tiên.

C4.76. Viết chương trình đếm số ước số của số nguyên dương N .

Ví dụ: $N=12 \rightarrow$ số ước số của 12 là 6

C4.77. Một số hoàn thiện là một số có tổng các ước số của nó (không kể nó) bằng chính nó. Hãy liệt kê các số hoàn thiện nhỏ hơn 5000.

Ví dụ: số 6 là số hoàn thiện vì tổng các ước số là $1+2+3=6$.

C4.78. Nhập vào ngày, tháng, năm. Cho biết đó là ngày thứ mấy trong năm.

C4.79. In ra dãy số Fibonacci

$$f_1 = f_0 = 1; \quad f_n = f_{n-1} + f_{n-2}; \quad (n > 1)$$

C4.80. Cài đặt tất cả các lưu đồ đã vẽ ở chương 1.

V. 2. Bài tập luyện tập và nâng cao

C4.81. Nhập vào ngày, tháng, năm. Kiểm tra xem ngày, tháng, năm đó có hợp lệ hay không, nếu hợp lệ cho biết ngày sau đó là bao nhiêu.

Ví dụ: Nhập 31/12/2003

Ngày sau đó 01/01/2004

C4.82. Nhập vào ngày, tháng, năm. Kiểm tra xem ngày, tháng, năm đó có hợp lệ hay không, nếu hợp lệ cho biết ngày trước đó là bao nhiêu.

Ví dụ: Nhập 01/01/2003

Ngày trước đó 31/12/2002

- C4.83. (*) Nhập vào ngày, tháng, năm của năm 2003. Hãy kiểm tra xem dữ liệu có hợp lệ hay không? Nếu hợp lệ hãy cho biết đó là ngày thứ mấy trong tuần. (hai, ba, tư, v.v..., CN). (Hướng dẫn: lấy ngày 01 tháng 01 năm 2003 là ngày thứ tư làm mốc).
- C4.84. Nhập vào giờ, phút, giây. Kiểm tra xem giờ, phút, giây đó có hợp lệ hay không, nếu hợp lệ cho biết giờ sau đó 1 giây là bao nhiêu.
Ví dụ: *Nhập 01:59:59*
Giờ sau đó 1 giây là 02:00:00
- C4.85. Nhập vào giờ, phút, giây. Kiểm tra xem giờ, phút, giây đó có hợp lệ hay không, nếu hợp lệ cho biết giờ trước đó 1 giây là bao nhiêu.
Ví dụ: *Nhập 02:00:00*
Giờ trước đó 1 giây là 01:59:59
- C4.86. Viết chương trình in ra bảng cửu chương từ 2 đến 9.

CHƯƠNG 5 CHƯƠNG TRÌNH CON

Tóm tắt: Trình bày khái niệm về hàm con, các bước xây dựng cài đặt chương trình theo phương pháp thủ tục hàm và một số kỹ thuật liên quan.

I. Các khái niệm

I. 1. Khái niệm hàm con (function)

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

Lý do sử dụng hàm:

- Khi có một công việc trùng lặp (giống nhau) cần thực hiện ở nhiều vị trí.
- Khi cần chia một chương trình lớn phức tạp thành các đơn thể nhỏ (hàm con) để chương trình được trong sáng, dễ hiểu trong việc xử lý, quản lý việc tính toán và giải quyết vấn đề.

Có hai loại hàm: hàm có sẵn trong ngôn ngữ gọi là hàm thư viện và hàm do người lập trình tự định nghĩa thêm theo yêu cầu của bài toán.

I. 2. Ví dụ

Xét bài toán: Viết chương trình nhập vào thông tin của hai hình chữ nhật, tính chu vi và diện tích, sau đó xuất kết quả ra màn hình.

Chúng ta sẽ lập trình giải bài toán trên bằng hai phương pháp: không sử dụng hàm con và có sử dụng hàm con để thấy được lợi ích của việc viết chương trình bằng phương pháp hàm con.

I. 2. 1. Chương trình không sử dụng hàm con

```
#include <stdio.h>
#include <conio.h>
void main()
{
    unsigned int cd1, cr1, cd2, cr2;
    unsigned long s1, p1, s2, p2;

    printf("Nhap thong tin hinh chu nhat 1\n");
    printf("Nhap vao chieu dai: ");
    scanf("%u", &cd1);
    printf("Nhap vao chieu rong: ");
    scanf("%u", &cr1);

    printf("Nhap thong tin hinh chu nhat 2\n");
    printf("Nhap vao chieu dai: ");
    scanf("%u", &cd2);
    printf("Nhap vao chieu rong: ");
    scanf("%u", &cr2);

    p1 = (cd1+cr1)*2;
    s1 = cd1*cr1;
    p2 = (cd2+cr2)*2;
    s2 = cd2*cr2;

    printf("\n--Ket qua tinh cua hinh chu nhat 1\n");
    printf("*Chu vi = %lu", p1);
    printf("\n*Dien tich = %lu", s2);
    printf("\n--Ket qua tinh cua hinh chu nhat 1\n");
    printf("*Chu vi = %lu", p1);
    printf("\n*Dien tich = %lu", s2);
    _getch();
}
```

Đoạn chương trình trên có nhiều lệnh trùng lặp như nhập thông tin hình chữ nhật, công thức tính diện tích và chu vi, xuất kết quả tính được. Điều quan trọng nhất là đoạn chương trình trong hàm main() viết quá dài sẽ rất khó khăn trong việc phân tích chương trình.

I. 2. 2. Chương trình có sử dụng hàm con

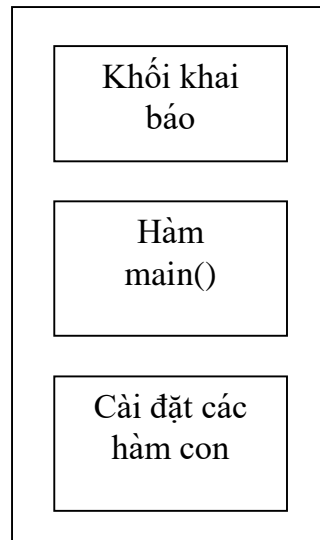
<i>#include <stdio.h></i> <i>#include <conio.h></i>	Phần khai báo thư viện hàm
<i>void NhapHCN(unsigned int &cd, unsigned int &cr);</i> <i>unsigned long TinhChuVi(unsigned int cd, unsigned int cr);</i> <i>unsigned long TinhDienTich(unsigned int cd, unsigned int cr);</i> <i>void XuatKetQua(unsigned long p, unsigned long s);</i>	Phần khai báo nguyên mẫu hàm (prototype)
<i>void main()</i> <i>{</i> <i> unsigned int cd1, cr1, cd2, cr2;</i> <i> unsigned long s1, p1, s2, p2;</i>	Khai báo các biến chứa thông tin chiều dài, chiều rộng, chu vi và diện tích cho hình chữ nhật 1 và hình chữ nhật 2
<i>printf("Nhap thong tin hinh chu nhac 1\n");</i> <i>NhapHCN(cd1, cr1);</i>	Nhập chiều dài và chiều rộng cho hình chữ nhật 1
<i>printf("Nhap thong tin hinh chu nhac 2\n");</i> <i>NhapHCN(cd2, cr2);</i>	Nhập chiều dài và chiều rộng cho hình chữ nhật 2
<i>p1 = TinhChuVi(cd1, cr1);</i> <i>s1 = TinhDienTich(cd1, cr1);</i>	Tính chu vi và diện tích hình chữ nhật 1
<i>p2 = TinhChuVi(cd2, cr2);</i> <i>s2 = TinhDienTich(cd2, cr2);</i>	Tính chu vi và diện tích hình chữ nhật 2
<i>printf("\n--Ket qua tinh cua hinh chu nhac 1\n");</i> <i>XuatKetQua(p1, s1);</i>	Xuất chu vi và diện tích của hình chữ nhật 1
<i>printf("\n--Ket qua tinh cua hinh chu nhac 2\n");</i> <i>XuatKetQua(p2, s2);</i> <i>_getch();</i> <i>}</i>	Xuất chu vi và diện tích của hình chữ nhật 2

<pre>void NhapHCN(unsigned int &cd, unsigned int &cr) { printf("Nhap vao chieu dai: "); scanf("%u", &cd); printf("Nhap vao chieu rong: "); scanf("%u", &cr); }</pre>	Hàm nhập chiều dài và chiều rộng của hình chữ nhật
<pre>unsigned long TinhChuVi(unsigned int cd, unsigned int cr) { unsigned long p = (cd+cr)*2; return p; }</pre>	Hàm tính chu vi hình chữ nhật với thông tin là chiều dài và chiều rộng
<pre>unsigned long TinhDienTich(unsigned int cd, unsigned int cr) { unsigned long s = cd*cr; return s; }</pre>	Hàm tính chu vi hình chữ nhật với thông tin là chiều dài và chiều rộng
<pre>void XuatKetQua(unsigned long p, unsigned long s) { printf("*Chu vi = %lu", p); printf("\n*Dien tich = %lu", s); }</pre>	Hàm xuất kết quả với thông tin cần xuất là chu vi và diện tích

Những lệnh trùng lặp được viết riêng thành một hàm riêng lẻ, khi sử dụng chỉ cần gọi lại hàm tương ứng. Hơn nữa các đoạn lệnh nằm độc lập ở các hàm khác nhau làm cho cấu trúc chương trình gọn gàng, dễ dàng trong việc kiểm tra và phân tích lệnh.

I. 3. Cấu trúc chương trình C sử dụng hàm con

Quan sát ví dụ ở mục I.2.2. chúng ta thấy cấu trúc tổng quát của một chương trình có sử dụng hàm con như sau:



I. 3. 1. Khởi khai báo

Bao gồm các khai báo về sử dụng thư viện, khai báo hằng số, khai báo hàm con (các nguyên mẫu hàm), khai báo các biến toàn cục và khai báo các kiểu dữ liệu tự định nghĩa.

I. 3. 2. Hàm chính (main())

Chứa các biến, các lệnh và các lời gọi hàm cần thiết trong chương trình.

I. 3. 3. Các hàm con

Viết lệnh cho các hàm con đã khai báo nguyên mẫu ở đầu chương trình.

➤ *Khối cài đặt hàm con và cài đặt hàm main() có thể hoán vị vị trí cho nhau chỉ khi có khai báo nguyên mẫu hàm.*

I. 4. Cấu trúc của một hàm

Khai báo nguyên mẫu hàm:

<Kiểu dữ liệu của hàm> TênHàm([danh sách các tham số]);

I. 4. 1. Kiểu dữ liệu của hàm

Xác định dựa vào kết quả của bài toán (Output). Gồm 2 loại:

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

```
void TênHàm (danh sách các tham số)
{
    Khai báo các biến cục bộ
    Các câu lệnh / khối lệnh hay lời gọi đến hàm khác.
}
```

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

```
<Kiểu dữ liệu> TênHàm ([danh sách các tham số])
{
    <Kiểu dữ liệu> kq;
    Khai báo các biến cục bộ
    Các câu lệnh / khối lệnh hay lời gọi đến hàm khác.

    return kq;
}
```

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

I. 4. 2. Tham số

Xác định dựa vào dữ liệu đầu vào của bài toán. Tham số khi cài đặt hàm còn được gọi là tham số hình thức vì các tham số này sẽ phụ thuộc vào giá trị hay biến truyền vào. Gồm 2 loại:

(a) Tham số là tham trị

Không thay đổi hoặc không cần lấy giá trị mới của tham số sau lời gọi hàm. Tham số dạng này chỉ mang ý nghĩa là **dữ liệu đầu vào**.

Ví dụ: hàm tính diện tích hình chữ nhật với tham số đầu vào là chiều dài (cd) và chiều rộng (cr)

unsigned long TinhChuVi(unsigned int cd, unsigned int cr);

(b) Tham số là tham chiếu

Có sự thay đổi giá trị của tham số trong quá trình thực hiện và cần lấy lại giá trị đó sau khi ra khỏi hàm.

Ứng dụng của tham số loại này có thể là **dữ liệu đầu ra** (kết quả) hoặc cũng có thể **vừa là dữ liệu đầu vào vừa là dữ liệu đầu ra**.

Dùng dấu **&** phía trước tên tham số khi cài đặt hàm.

Ví dụ: hàm hoán vị hai số nguyên a và b, kết quả giá trị của a và b sẽ hoán đổi cho nhau, nên a và b sẽ là tham số vừa làm đầu vào vừa lưu kết quả đầu ra

void HoanVi(int &a, int &b);

I. 4. 3. Tên hàm

Đặt tên theo **quy ước đặt tên trong C** sao cho tên gọi **đúng với chức năng hay mục đích thực hiện của hàm và gọi nhớ**. Ký tự đầu của mỗi từ trong tên hàm nên viết bằng chữ in hoa. Ví dụ: TinhLuong, TimKiem, TinhChuVi .

I. 5. Gọi hàm

Khi gọi hàm sẽ truyền tham số thực (có thể là giá trị hoặc tên biến), việc truyền tham số thực phải khớp kiểu và thứ tự theo ý nghĩa của dữ liệu đưa vào. Do hàm có hai dạng kiểu dữ liệu trả về nên sẽ có cách gọi khác nhau cho từng loại:

Hàm không có giá trị trả về

TênHàm(tên biến hoặc giá trị);

Ví dụ như hàm **void** *XuatKetQua(unsigned long p, unsigned long s)*; trong ví dụ I.2.2. sẽ được gọi thực hiện trong hàm main() là:

XuatKetQua(p1, s1);

XuatKetQua(p2, s2);

(Với *p1, s1, p2, s2* tương ứng với chu vi và diện tích của các hình chữ nhật)

Hàm có giá trị trả về

<kiểu dữ liệu trả về của hàm> tênbiến = TênHàm(tên biến hoặc giá trị);

Ví dụ như hàm **unsigned long** *TinhChuVi(unsigned int cd, unsigned int cr)*; trong ví dụ I.2.2. sẽ được gọi thực hiện trong hàm main() là:

unsigned long p1, p2;

p1=TinhChuVi(cd1, cr1);

p2=TinhChuVi(cd2, cr2);

II. Phương pháp xác định nguyên mẫu hàm

Trước hết phải phân tích và phân rã bài toán thành các chức năng độc lập. Ứng với mỗi chức năng độc lập sẽ xây dựng thành hàm con. Cách xác định để đưa ra được nguyên mẫu của hàm chúng ta cần phải xem xét và trả lời những câu hỏi sau:

- (1) **Câu hỏi 1: Hàm trả về gì?** → Xác định kiểu dữ liệu trả về của hàm
- (2) **Hàm thực hiện việc gì?** → Xác định tên hàm
- (3) **Cần những thông tin gì cần thiết để hàm xử lý?** → Xác định tham số

Với mỗi thông tin, xác định xem đã có giá trị trước khi vào hàm chưa? Có ba trường hợp (TH):

- **TH1:** Nếu chưa có → Tham số là tham chiếu (tham số đầu ra).
- **TH2:** Nếu có mà sau khi thực hiện xong hàm vẫn không thay đổi → Tham số là tham trị (tham số đầu vào)
- **TH3:** Nếu có mà sau khi thực hiện xong hàm thì giá trị cũng bị thay đổi theo → Tham số là tham chiếu (tham số đóng vai trò vừa là đầu vào vừa đầu ra).

III. Một số ví dụ

Ví dụ 1: Viết chương trình nhập số nguyên dương n và in ra màn hình các ước số của n

Phân tích bài toán:

- **Input:** n (Đề xác định tham số)
 - Kiểu dữ liệu: số nguyên dương (*unsigned int*).
 - Giá trị n không bị thay đổi trong quá trình tìm ước số \rightarrow Tham số của hàm là tham trị.
- **Output:** In ra các ước số của n (Đề xác định kiểu dữ liệu hàm)
 - Không trả về giá trị.
 - Kiểu dữ liệu của hàm là *void*.
- **Xác định tên hàm:** Hàm này dùng in ra các ước số của n nên có thể đặt là *LietKeUocSo*

Ta có nguyên mẫu hàm:

void LietKeUocSo (unsigned int n);

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

//Khai bao nguyen mau ham
void LietKeUocSo ( unsigned int n );

void main()
{
    unsigned int n;

    printf("Nhap n = ");
    scanf("%u",&n);
    printf("Cac uoc so cua n : " );
    LietKeUocSo(n);
    _getch( );
}

void LietKeUocSo (unsigned int n)
```

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

Ví dụ 2: Viết chương trình nhập số nguyên dương n và tính tổng

$$S = 1 + 2 + 3 + \dots + n, \text{ với } n > 0$$

Phân tích bài toán:

- **Input:** n (Để xác định tham số)
 - Kiểu dữ liệu: số nguyên dương (*unsigned int*).
 - Giá trị n không bị thay đổi trong quá trình tính tổng → Tham số của hàm là tham trị.
- **Output:** Tổng S (Để xác định kiểu dữ liệu hàm)
 - Trả về giá trị của S.
 - S là tổng các số nguyên dương nên S cũng là số nguyên dương → Kiểu trả về của hàm là *unsigned int* (hoặc *unsigned long* cho trường hợp giá trị của tổng lớn hơn 2 bytes).
- **Xác định tên hàm:** Hàm này dùng tính tổng S nên có thể đặt là TongS.

Ta có nguyên mẫu hàm:

unsigned long TongS (unsigned int n);

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

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

```
//Khai bao nguyen mau ham
```

```
unsigned long TongS ( unsigned int n );
```

```
void main()
```

```
{
```

```
    unsigned int n;
```

```
    unsigned long kq;
```

```
    printf("Nhap n = ");
```

```
scanf("%u",&n);
kq = TongS ( n );
printf("Tong can tinh la: %lu ", kq);
_getch( );
}
unsigned long TongS (unsigned int n)
{
    unsigned long S=0;
    unsigned int i=1;
    while(i<=n)
    {
        S+=i;
        i++;
    }
    return S;
}
```

Ví dụ 3: Viết chương trình nhập số nguyên a và b, hoán vị hai số đó

Phân tích bài toán:

- **Input:** a và b (Để xác định tham số)
 - Kiểu dữ liệu: số nguyên (*int*).
 - Giá trị a và b bị thay đổi sau khi hoán vị → Tham số của hàm là tham chiếu.
- **Output:** Trả về void, do giá trị a và b đã dùng tham chiếu để lưu giá trị mới sau khi hoán vị (Để xác định kiểu dữ liệu hàm)
- **Xác định tên hàm:** Hàm này dùng hoán vị nên có thể đặt là HoanVi.

Ta có nguyên mẫu hàm:

void HoanVi(int &a, int &b);

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

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

```
//Khai bao nguyen mau ham
```

```
void HoanVi(int &a, int &b);
```

```
void main()
```

```
{
```

```
    int a, b;
```

```
printf("Nhap a = ");
scanf("%d",&a);
printf("Nhap b = ");
scanf("%d",&b);

HoanVi(a, b);
printf("Sau khi hoan vi a=%d; b=%d", a, b);
_getch( );
}
void HoanVi(int &a, int &b)
{
    int tam = a;
    a = b;
    b = tam;
}
```

🔗 **Lưu ý cách gọi hàm:** Đối với hàm có kiểu dữ liệu hàm là **void** thì khi gọi **không cần phải gán giá trị vào biến, ngược lại phải gọi như trong ví dụ 2** (Phải khai báo **tương ứng kiểu** với kiểu dữ liệu hàm sẽ gọi và gán giá trị trả về vào biến đó).

IV. Kết luận

Trước khi xây dựng một hàm ta phải xác định mục đích của hàm là dùng để làm gì, trên cơ sở đó, ta mới xác định được các thành phần của hàm và xây dựng nguyên mẫu hàm.

Mỗi hàm phải thực hiện một chức năng độc lập và tách biệt với các hàm khác (*không được lồng nhau*).

Đối với hàm có giá trị trả về phải lưu ý kiểu dữ liệu phải tương ứng kiểu dữ liệu cả giá trị trả về và kiểu dữ liệu của biến được gán khi gọi hàm. Trường hợp hàm trả về từ hai loại giá trị trở lên thì phải có dòng chú thích cho trường hợp tương ứng để khi gọi hàm biết được kết quả (*chẳng hạn như tìm kiếm, kiểm tra, so sánh, v.v... giá trị trả về có 2 trường hợp: Có hoặc không có phần tử cần tìm, thỏa điều kiện kiểm tra hay không? Do vậy ta phải quy ước giá trị cho từng trường hợp*).

Nên đặt tên hàm sao cho gọi nhớ được chức năng, đặt tên theo quy tắc nhất định để tránh việc gọi sai tên hàm do lẫn lộn giữa ký tự hoa và thường, có dấu gạch nối giữa các từ trong hàm hay không?

Khi gọi hàm phải truyền đủ tham số, đúng kiểu dữ liệu và đúng thứ tự của tham số.

V. Bài tập

V.1. Bài tập cơ bản

- C5.87. Cài đặt lại tất cả các bài tập ở chương 4 theo phương pháp hàm.
- C5.88. Viết chương trình tính diện tích và chu vi hình tròn với bán kính được nhập từ bàn phím.
- C5.89. Nhập số nguyên dương n ($n > 0$). Liệt kê tất cả các số nguyên tố nhỏ hơn n .
- C5.90. Nhập số nguyên dương n ($n > 0$). Liệt kê n số chính phương đầu tiên.
- C5.91. Nhập số nguyên dương n ($n > 0$). Đếm xem có bao nhiêu số hoàn thiện nhỏ hơn n .
- C5.92. Nhập số nguyên dương n ($0 \leq n < 1000$) và in ra cách đọc của n .
Ví dụ: Nhập $n = 105$. In ra màn hình: *Mot tram le nam*.
- C5.93. Viết chương trình tính tiền thuê máy dịch vụ Internet và in ra màn hình kết quả. Với dữ liệu nhập vào là giờ bắt đầu thuê (GBD), giờ kết thúc thuê (GKT), số máy thuê (SoMay).
- Điều kiện cho dữ liệu nhập: $6 \leq \text{GBD} < \text{GKT} \leq 21$ (giả sử giờ là số nguyên).
 - Đơn giá: 2500đ cho mỗi giờ máy trước 17:00 và 3000đ cho mỗi giờ máy sau 17:00.
- C5.94. Viết chương trình tính tiền lương ngày cho công nhân, cho biết trước giờ vào ca, giờ ra ca của mỗi người.
- Giả sử rằng:**
- Tiền trả cho mỗi giờ trước 12 giờ là 6000đ và sau 12 giờ là 7500đ.
 - Giờ vào ca sớm nhất là 6 giờ sáng và giờ ra ca trễ nhất là 18 giờ (*Giả sử giờ nhập vào nguyên*).
- C5.95. Nhập vào 3 số thực a, b, c và kiểm tra xem chúng có thành lập thành 3 cạnh của một tam giác hay không? Nếu có hãy tính diện tích, chiều dài mỗi đường cao của tam giác và in kết quả ra màn hình.
- Công thức tính diện tích: $S = \sqrt{p \times (p - a) \times (p - b) \times (p - c)}$, với p là nửa chu vi của tam giác
 - Công thức tính các đường cao: $h_a = \frac{2S}{a}$; $h_b = \frac{2S}{b}$; $h_c = \frac{2S}{c}$

C5.96. Nhập vào 6 số thực a, b, c, d, e và f. Giải hệ phương trình sau :

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

C5.97. Viết chương trình nhập 2 số nguyên dương a, b. Tìm USCLN và BSCNN của hai số nguyên đó.

C5.98. Viết chương trình tính tổng nghịch đảo của n giai thừa.

C5.99. Cho 2 số nguyên a, b. Viết hàm hoán vị giá trị 2 số trên.

C5.100. (*) Viết chương trình nhập số nguyên dương n gồm 5 chữ số, kiểm tra xem các chữ số n có đối xứng hay không.

Ví dụ: *Đối xứng:* 13531

Không đối xứng: 13921

C5.101. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), đếm xem n có bao nhiêu chữ số chẵn và bao nhiêu chữ số lẻ.

C5.102. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), đếm xem n có bao nhiêu chữ số là số nguyên tố.

C5.103. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), tính tổng các ước số dương của n.

Ví dụ: *Nhập n=6*

Tổng các ước số từ 1 đến n: $1+2+3+6=12$.

C5.104. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), tìm ước số lẻ lớn nhất của n.

Ví dụ: *Ước số lẻ lớn nhất của 27 là 9.*

C5.105. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), kiểm tra xem các chữ số của n có toàn lẻ hay toàn chẵn không.

C5.106. (*) Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), sắp xếp các chữ số của n theo thứ tự tăng dần.

Ví dụ: *Nhập n=1536*

Kết quả sau khi sắp xếp: 1356.

V. 2. Bài tập luyện tập và nâng cao

C5.107. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), sau đó nhập một số nguyên x, tìm vị trí xuất hiện của chữ số có giá trị x trong n.

Ví dụ: Nhập $n=1526$, $x=2$

Kết quả: Chu số 2 o vị trí thu 3.

C5.108. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), kiểm tra xem các chữ số của n có được sắp thứ tự không.

Ví dụ: Nhập $n=1569$ hoặc $n=8521$

Kết quả: Có thứ tự.

C5.109. Viết chương trình nhập 2 số a, b sao cho: số lớn nhất trong 2 số phải là một số dương và chia hết cho 7. Nếu nhập sai phải yêu cầu nhập lại cho đến khi đúng.

C5.110. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), tính giá trị trung bình các chữ số chẵn trong n .

C5.111. (*) Viết chương trình in ra màn hình ngày/tháng/năm của ngày hiện tại, cho phép sử dụng các phím mũi tên lên, xuống để tăng hoặc giảm một ngày.

C5.112. (*) Viết chương trình in ra màn hình giờ:phút:giây hiện tại, cho phép sử dụng các phím mũi tên lên, xuống để tăng hoặc giảm một giây.

CHƯƠNG 6 MẢNG MỘT CHIỀU

Tóm tắt: Giới thiệu về cấu trúc mảng một chiều, khai báo dữ liệu, các thao tác nhập xuất, các kỹ thuật thao tác trên mảng. Ứng dụng các kỹ thuật này trong việc cài đặt các hàm tìm kiếm, kiểm tra, xây dựng mảng, tách và ghép mảng.

I. Các khái niệm

I. 1. Khái niệm

Đối với những kiểu dữ liệu cơ bản của C, mỗi biến chỉ lưu trữ được giá trị đơn. Khi cần lưu trữ nhiều giá trị cùng lúc, danh sách các giá trị thì chúng ta phải sử dụng kiểu dữ liệu mảng.

Mảng thực chất là một biến được cấp phát bộ nhớ liên tục và bao gồm nhiều biến thành phần.

Các thành phần của mảng là tập hợp các biến có cùng kiểu dữ liệu và cùng tên. Do đó để truy xuất các biến thành phần, ta dùng cơ chế chỉ mục.

I. 2. Khai báo mảng

Để khai báo một mảng, ta có 2 cách khai báo sau :

❖ Cách 1: Con trỏ hằng

< Kiểu dữ liệu > < Tên mảng > [< Số phần tử tối đa của mảng>];

Ví dụ:

int a[100]; //Khai báo mảng số nguyên a gồm 100 phần tử

float b[50]; //Khai báo mảng số thực b gồm 50 phần tử

❖ Cách 2: Sử dụng con trỏ

Ý nghĩa: Khi ta khai báo một mảng với kiểu dữ liệu bất kì (int, float, char,v.v...) thì tên của mảng thực chất là một **hằng địa chỉ của phần tử đầu tiên**.

< Kiểu dữ liệu > * < Tên mảng >;

Ví dụ :

*int *p; // khai báo con trỏ p*

int b[100];

$p = b;$ // p trỏ vào phần tử 0 của mảng b

Với cách viết như trên thì ta có thể hiểu các cách viết sau là tương đương

$$p[i] \Leftrightarrow *(p + i) \Leftrightarrow b[i] \Leftrightarrow *(b+i)$$

⚠ **Lưu ý:** Khi sử dụng biến con trỏ để truy xuất mảng, theo cách như trên thì thực chất con trỏ p chỉ chiếm 2 byte bộ nhớ để chứa địa chỉ mà thôi. Để tạo mảng chứa dữ liệu thành phần thì ta phải cấp phát vùng nhớ cho con trỏ p .
Dùng hàm : **malloc**, **calloc** trong thư viện **<stdlib.h>** để cấp phát vùng nhớ.

Ví dụ:

+ Cách 1: dùng malloc

$int *px;$ // Khai báo con trỏ px

$px = (int *) malloc (100);$ // Cấp phát 100 ô nhớ kiểu int cho con trỏ px

+ Cách 2: dùng calloc

$int *p;$ // khai báo con trỏ p

$p = (int *) calloc (100, sizeof (int));$ // cấp phát 10 ô nhớ mỗi ô chiếm 2bytes

Sau khi sử dụng xong thì nên giải phóng vùng nhớ bằng hàm free

Ví dụ : $free (p);$ // giải phóng vùng nhớ cho con trỏ p .

⚠ Nhằm thuận tiện cho việc viết chương trình, ta nên định nghĩa hằng số **MAX** ở đầu chương trình – là kích thước tối đa của mảng - như sau:

```
#define MAX 100
void main()
{
    int a[MAX], b[MAX];
    //Các lệnh
}
```

I. 3. Khai báo và gán giá trị cho mảng

- Gán từng phần tử, cú pháp:

<kiểu dữ liệu> tên mảng[số lượng] = {các giá trị};

Mỗi giá trị cách nhau bằng dấu phẩy.

Ví dụ:

```
int a[5] = {3, 6, 8, 1, 12};
```

Giá trị	3	6	8	1	12
Vị trí	0	1	2	3	4

- Gán toàn bộ phần tử có cùng một giá trị, cú pháp:

<kiểu dữ liệu> tên mảng[số lượng] = {giá trị};

Ví dụ:

```
int a[8] = {3};
```

Giá trị	3	3	3	3	3	3	3	3
Vị trí	0	1	2	3	4	5	6	7

I. 4. Truy xuất phần tử của mảng

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

Ví dụ: `int a[5]` // Khai báo mảng A gồm tối đa 5 phần tử nguyên.

a[0]	a[1]	a[2]	a[3]	a[4]
------	------	------	------	------

Truy xuất phần tử của mảng, cú pháp:

Gán giá trị: *tên mảng[vị trí] = giá trị hằng;*

Lấy giá trị: *tên biến = tên mảng[vị trí];*

Ví dụ:

```
void main()
{
    int a[5] = {3, 6, 8, 11, 12};
    cout<<"Gia tri mang tai vi tri 3 = "<<a[3];
}
Kết quả: Gia tri mang tai vi tri 3 = 11
```

II. Một số thao tác cơ bản trên mảng số nguyên

II. 1. Hàm nhập xuất mảng một chiều

#define MAX 100

- Nhập mảng một chiều số nguyên kích thước n

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

- Xuất mảng một chiều số nguyên kích thước n

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

- Sử dụng trong hàm main()

```
void main ( )
{
    int a[MAX] , n;
    printf("Nhap kích thước mảng: ");
    scanf("%d", &n);
    NhapMang (a,n);
    printf("Các giá trị của mảng vừa nhập: \n");
    XuatMang (a,n);
}
```

II. 2. Liệt kê (xuất) những phần tử thỏa điều kiện cho trước

Dùng để lọc những phần tử thỏa mãn điều kiện cho trước

Mẫu 1:

```
void LietKeXXX(int a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        if (a[i] thỏa điều kiện)
        {
            Xuất a[i];
        }
    }
}
```

Ví dụ 1: Liệt kê các phần tử có giá trị chẵn trong mảng

```
void LietKeChan(int a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        if (a[i] % 2 == 0)
        {
            printf("%d\t", a[i]);
        }
    }
}
```

Mẫu 2:

```
void LietKeXXX(int a[], int n, int x)
{
    for (int i = 0; i < n; i++)
    {
        if (a[i] thỏa điều kiện so với x)
        {
            Xuất a[i];
        }
    }
}
```

Ví dụ 2: Liệt kê các phần tử có giá trị lớn hơn x trong mảng

```
void LietKeLonHonX(int a[], int n, int x)
```

```
{
    for (int i = 0; i < n; i++)
    {
        if (a[i] > x)
        {
            printf("%d\t", a[i]);
        }
    }
}
```

Ví dụ 3: Chương trình nhập vào mảng một chiều số nguyên a, kích thước n. In ra các phần tử có giá trị lớn hơn x có trong mảng

```
#define MAX 100
void NhapMang(int a[], int n);
void XuatMang(int a[], int n);
void LietKeLonHonX(int a[], int n, int x);
void main()
{
    int a[MAX], n, x;
    printf("Nhap kích thước mảng: ");
    scanf("%d", &n);
    NhapMang(a, n);
    printf("Các phần tử của mảng:\n");
    XuatMang(a, n);
    printf("Nhap giá trị x: ");
    scanf("%d", &x);
    printf("Các phần tử có giá trị lớn hơn %d:\n", x);
    LietKeLonHonX(a, n, x);
}

void NhapMang(int a[], int n)
{
    //Cài đặt
}

void XuatMang(int a[], int n)
{
    //Cài đặt
}

void LietKeLonHonX(int a[], int n, int x)
{
    //Cài đặt
}
```

II. 3. Đếm số lượng phần tử trong mảng

Mẫu 1:

```
int DemXXX(int a[], int n)
{
    int d = 0;
    for (int i = 0; i < n; i++)
    {
        if (a[i] thỏa điều kiện)
        {
            d++;
        }
    }
    return d;
}
```

Ví dụ 1: Đếm các phần tử có giá trị là số nguyên tố

```
int LaSNT(int k)
{
    int d = 0;
    for (int i = 1; i <= k; i++)
    {
        if (k % i == 0)
        {
            d++;
        }
    }
    if (d == 2)
    {
        return 1;
    }
    return 0;
}
```

```
int DemSNT(int a[], int n)
{
    int d = 0;
    for (int i = 0; i < n; i++)
    {
        if (LaSNT(a[i]) == 1)
        {
            d++;
        }
    }
    return d;
}
```

Mẫu 2:

```
int DemXXX(int a[], int n, int x)
{
    int d = 0;
    for (int i = 0; i < n; i++)
    {
        if (a[i] thỏa điều kiện so với x)
        {
            d++;
        }
    }
    return d;
}
```

Ví dụ 2: Đếm các phần tử có giá trị nhỏ hơn x có trong mảng

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

II. 4. Tìm kiếm và trả về vị trí phần tử có giá trị lớn nhất

```
int TimVTMax(int a[], int n)
{
    int vtmax = 0;
    for (int i = 1; i < n; i++)
    {
        if (a[i] > a[vtmax])
        {
            vtmax = i;
        }
    }
    return vtmax;
}
```

II. 5. Tìm vị trí phần tử có giá trị x

Nếu x không xuất hiện trong mảng thì trả về -1

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

II. 6. Kiểm tra xem mảng có thỏa điều kiện cho trước

Trường hợp 1: kiểm tra tồn tại một phần tử trong mảng thỏa điều kiện nào đó cho trước → tìm phần tử thỏa điều kiện để kết luận.

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

Ví dụ: Kiểm tra xem mảng có tồn tại số lẻ không?

```
int KiemTraTonTaiLe(int a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        if (a[i] % 2 != 0)
            return 1;
    }
    return 0;
}
```


Trường hợp 2: kiểm tra tất cả các phần tử thỏa điều kiện nào đó cho trước → tìm phần tử không thỏa điều kiện để kết luận mảng không thỏa điều kiện.

```
int KiemTraXXX(int a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        if (a[i] không thỏa điều kiện)
        {
            return 0;
        }
    }
    return 1;
}
```

Ví dụ 2: Kiểm tra xem mảng có toàn bộ là giá trị âm không?

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

II. 7. Tính tổng có điều kiện

```
int TongXXX(int a[], int n)
{
    int s = 0;
    for (int i = 0; i < n; i++)
    {
        if (a[i] thỏa điều kiện)
        {
            s += a[i];
        }
    }
    return s;
}
```

Ví dụ: Tính tổng các phần tử có giá trị lẻ trong mảng

```
int TongLe(int a[], int n)
{
    int s = 0;
    for (int i = 0; i < n; i++)
    {
        if (a[i] % 2 != 0)
        {
            s += a[i];
        }
    }
    return s;
}
```

II. 8. Tính giá trị trung bình có điều kiện

```
float TrungBinhXXX(int a[], int n)
{
    int s = 0;
    int d = 0;
    for (int i = 0; i < n; i++)
    {
        if (a[i] thỏa điều kiện)
        {
            s += a[i];
            d++;
        }
    }
    if (d == 0)
    {
        return 0;
    }
    return (float) s / d;
}
```

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

Ví dụ: Tính giá trị trung bình các phần tử có giá trị âm trong mảng

```
float TrungBinhAm(int a[], int n)
```

```
{
    int s = 0;
    int d = 0;
    for (int i = 0; i < n; i++)
    {
        if (a[i] < 0)
        {
            s += a[i];
            d++;
        }
    }
    if (d == 0)
    {
        return 0;
    }
    return (float)s / d;
}
```

II. 9. Sắp xếp mảng theo thứ tự tăng

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

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

II. 10. Xoá phần tử trong mảng

Duyệt mảng từ trái sang phải. Xuất phát từ vị trí cần xoá tiến hành dời lần lượt các phần tử về phía trước cho đến khi kết thúc mảng, sau đó giảm kích thước mảng.

Vấn đề đặt ra là tìm vị trí cần xóa theo điều kiện bài toán rồi thực hiện xóa.

Ví dụ 1: Xoá phần tử đầu tiên của mảng

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

Ví dụ 2: Viết hàm xoá phần tử tại vị trí (vitri) cho trước trong mảng.

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

II. 11. Chèn một phần tử x vào mảng

Duyệt mảng từ phải sang trái. Xuất phát từ cuối mảng tiến hành đẩy lần lượt các phần tử về phía sau cho đến vị trí cần chèn, chèn phần tử cần chèn vào vị trí chèn và tăng kích thước mảng.

Trước khi chèn ta phải xác định vị trí cần chèn theo điều kiện bài toán.

Ví dụ 1: Thêm phần tử có giá trị x vào cuối mảng

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

Ví dụ 2: Chèn phần tử có giá trị X vào mảng tại vị trí (vitri) cho trước

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

II. 12. Tách và ghép mảng

II. 12. 1. Kỹ thuật tách cơ bản

Cho mảng a kích thước n (n chẵn). Tách mảng a thành 2 mảng b và c sao cho: b có $\frac{1}{2}$ phần tử đầu của mảng a, $\frac{1}{2}$ phần tử còn lại đưa vào mảng c.

```
void TachMang(int a[], int n, int b[], int &m, int c[], int &l)
{
    int k=n/2;

    m=l=0;
    for(int i=0; i<k; i++)
    {
        b[m++]=a[i];
        c[l++]=a[k+i]
    }
}
```

II. 12. 2. Kỹ thuật ghép cơ bản

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

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

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

Cách thực hiện: Đưa lần lượt từng phần tử của mảng a và mảng b vào mảng c , tăng chỉ số tương ứng. Nếu một trong hai mảng hết trước thì chép tất cả các phần tử còn lại của mảng chưa hết vào mảng c .

Đặt i là chỉ số của mảng a ; j : chỉ số của mảng b và k là chỉ số của mảng c .

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

III. Kết luận

Dữ liệu kiểu mảng dùng cho việc biểu diễn những thông tin có cùng kiểu dữ liệu liên tiếp nhau.

Khi cài đặt bài tập mảng một chiều nên xây dựng thành những hàm chuẩn để dùng lại cho các bài tập khác.

Các thao tác xử lý trên mảng đều theo quy tắc nhất định, chúng ta có thể ứng dụng mảng trong việc biểu diễn số lớn, dùng bảng tra, khử đệ qui, v.v...

Chỉ số mảng trong ngôn ngữ C được đánh số bắt đầu từ vị trí 0 (zero-base index).

IV. Bài tập

IV. 1. Bài tập cơ bản

- C6.113. Viết chương trình nhập xuất mảng một chiều các số thực.
- C6.114. Viết chương trình khởi tạo giá trị các phần tử là 0 cho mảng một chiều các số nguyên gồm n phần tử.
- C6.115. Viết chương trình phát sinh ngẫu nhiên mảng một chiều các số nguyên âm.
- C6.116. Viết chương trình phát sinh ngẫu nhiên mảng một chiều các số nguyên sao cho mảng có thứ tự tăng dần (*không sắp xếp*).
- C6.117. Viết chương trình nhập mảng các số thực và xuất các phần tử âm trong mảng.
- C6.118. Viết chương trình nhập mảng các số nguyên và xuất các phần tử lẻ có trong mảng.
- C6.119. Viết chương trình nhập vào mảng một chiều các số nguyên và xuất ra các phần tử chẵn nhỏ hơn 20.
- C6.120. Viết chương trình nhập vào mảng một chiều các số nguyên và xuất ra màn hình các phần tử là số nguyên tố.
- C6.121. Viết chương trình nhập vào mảng số nguyên kích thước n và số nguyên k , liệt kê các phần tử trong mảng là số nguyên tố nhỏ hơn k (*nếu mảng không tồn tại số nguyên tố nào nhỏ hơn k thì phải xuất ra một câu thông báo*).
- C6.122. Viết chương trình nhập vào mảng một chiều các số nguyên và xuất ra màn hình các phần tử là số chính phương nằm tại những vị trí lẻ trong mảng.
- C6.123. Viết hàm tìm vị trí phần tử có giá trị x xuất hiện cuối cùng trong mảng.
- C6.124. Viết hàm tìm vị trí của phần tử nhỏ nhất trong mảng các số nguyên.
- C6.125. Viết hàm tìm vị trí của phần tử lớn nhất trong mảng các số nguyên.
- C6.126. Viết hàm in vị trí các phần tử nguyên tố trong mảng các số nguyên.
- C6.127. Viết hàm in vị trí các phần tử nguyên tố lớn hơn 23.
- C6.128. Viết hàm tìm vị trí phần tử âm đầu tiên trong mảng. Nếu không có phần tử âm trả về -1 .
- C6.129. Viết hàm tìm vị trí phần tử âm lớn nhất trong mảng.
- C6.130. Viết hàm tìm vị trí phần tử dương đầu tiên trong mảng. Nếu không có phần tử âm trả về -1 .

- C6.131. Viết hàm tìm vị trí phần tử dương bé nhất trong mảng.
- C6.132. Viết hàm in các phần tử là bội của 3 và 5.
- C6.133. Viết hàm tìm số chẵn cuối cùng có trong mảng, nếu không tồn tại số chẵn hàm trả về -1.
- C6.134. Viết hàm tìm số lẻ lớn nhất có trong mảng, nếu không tồn tại số lẻ thì trả về -1.
- C6.135. Viết hàm tìm và đổi chỗ phần tử lớn nhất với phần tử nhỏ nhất trong mảng.
- C6.136. Nhập vào số nguyên X. Viết hàm in ra màn hình những phần tử có giá trị từ 1 đến X có trong mảng số nguyên.
- C6.137. Viết chương trình nhập vào một dãy số a gồm n số thực ($n \leq 100$), nhập vào dãy số b gồm m số thực ($m \leq 100$).
- In ra những phần tử chỉ xuất hiện trong dãy a mà không xuất hiện trong dãy b.
 - In ra những phần tử xuất hiện ở cả hai dãy.
- C6.138. Viết hàm đếm các phần tử âm, dương trong mảng.
- C6.139. Viết hàm đếm các phần tử chẵn, lẻ trong mảng.
- C6.140. Viết hàm đếm số lần xuất hiện của phần tử x trong mảng.
- C6.141. Viết hàm đếm các phần tử nhỏ hơn x trong mảng.
- C6.142. Viết hàm đếm các phần tử là số nguyên tố trong mảng.
- C6.143. Viết hàm đếm các phần tử là số hoàn thiện trong mảng.
- C6.144. Viết hàm đếm các phần tử là bội của 3 và 5 trong mảng các số nguyên.
- C6.145. Viết hàm tính tổng các phần tử chẵn trong mảng.
- C6.146. Viết hàm tính tổng các phần tử lẻ trong mảng các số nguyên.
- C6.147. Viết hàm tính tổng các phần tử nguyên tố trong mảng.
- C6.148. Viết hàm tính tổng các phần tử nằm ở vị trí chẵn trong mảng các số nguyên.
- C6.149. Viết hàm tính tổng các phần tử nằm ở vị trí nguyên tố trong mảng.
- C6.150. Viết hàm tính tổng các phần tử chia hết cho 5 có trong mảng.
- C6.151. Viết hàm tính tổng các phần tử cực đại trong mảng các số nguyên (*phần tử cực đại là phần tử lớn hơn các phần tử xung quanh nó*).
- Ví dụ: 1 5 2 6 3 5 1 8 6
- C6.152. Viết hàm tính tổng các phần tử cực tiểu trong mảng các số nguyên (*phần tử cực tiểu là phần tử nhỏ hơn các phần tử xung quanh nó*).

Ví dụ: 6 4 2 9 5 3 7 1 5 8

- C6.153. Viết hàm tính tổng các phần tử là bội của 3 và 5 trong mảng các số nguyên.
- C6.154. Viết hàm tính tổng các phần tử là số hoàn thiện trong mảng các số nguyên.
- C6.155. Viết hàm tính giá trị trung bình của các số hoàn thiện trong mảng các số nguyên.
- C6.156. Viết hàm sắp xếp mảng theo thứ tự giảm dần.
- C6.157. Viết hàm sắp xếp mảng theo thứ tự tăng dần của các phần tử là số nguyên tố.
- C6.158. Viết hàm sắp xếp các phần tử lẻ tăng dần.
- C6.159. Viết hàm sắp xếp các phần tử chẵn giảm dần.
- C6.160. Viết hàm sắp xếp các phần tử chẵn nằm bên trái theo thứ tự tăng dần còn các phần tử lẻ bên phải theo thứ tự giảm dần.
- C6.161. Viết hàm sắp xếp các phần tử âm giảm dần từ trái sang phải, phần tử dương tăng dần từ phải sang trái.
- C6.162. Viết hàm xoá phần tử tại vị trí i trong mảng.
- C6.163. Viết hàm xoá phần tử có giá trị lớn nhất trong mảng.
- C6.164. Nhập vào giá trị X . Viết hàm xoá tất cả các phần tử có giá trị nhỏ hơn X .
- C6.165. Nhập vào giá trị X . Viết hàm xoá phần tử có giá trị gần X nhất.
- C6.166. Viết hàm chèn giá trị X vào vị trí đầu tiên của mảng.
- C6.167. Viết hàm chèn giá trị X vào phía sau phần tử có giá trị lớn nhất trong mảng.
- C6.168. Viết hàm chèn phần tử có giá trị X vào trước phần tử có giá trị là số nguyên tố đầu tiên trong mảng.
- C6.169. Viết hàm chèn phần tử có giá trị X vào phía sau tất cả các phần tử có giá trị chẵn trong mảng.
- C6.170. Viết chương trình tách 1 mảng các số nguyên thành 2 mảng a và b , sao cho mảng a chứa toàn số lẻ và mảng b chứa toàn số chẵn.

Ví dụ: Mảng ban đầu: 1 3 8 2 7 5 9 0 10

 Mảng a : 1 3 7 5 9

 Mảng b : 8 2 10

- C6.171. Cho 2 mảng số nguyên a và b kích thước lần lượt là n và m . Viết chương trình nối 2 mảng trên thành mảng c sao cho chẵn ở đầu mảng và lẻ ở cuối mảng.

Ví dụ: *Mảng a:* 3 2 7 5 9
 Mảng b: 1 8 10 4 12 6
 Mảng c: 6 12 4 10 2 8 3 1 7 5 9

IV. 2. Bài tập luyện tập và nâng cao

C6.172. Viết chương trình nhập vào mảng a gồm n phần tử, trong quá trình nhập kiểm tra các phần tử nhập vào không được trùng, nếu trùng thông báo và yêu cầu nhập lại.

C6.173. Viết hàm tính tổng của từng dãy con giảm có trong mảng.

C6.174. (*) Cho mảng các số nguyên a gồm n phần tử ($n \leq 30000$) và số dương k ($k \leq n$). Hãy chỉ ra số hạng lớn thứ k của mảng.

Ví dụ: *Mảng a:* 6 3 1 10 11 18
 $k = 2$
 Kết quả: 10

C6.175. (*) Cho 2 dãy A, B các số nguyên (*kích thước dãy A nhỏ hơn dãy B*). Hãy kiểm tra xem A có phải là con của B hay không?

C6.176. Viết hàm liệt kê các bộ 4 số a, b, c, d trong mảng các số nguyên (*có ít nhất 4 phần tử và đôi một khác nhau*) sao cho $a + b = c + d$.

C6.177. (*) Viết chương trình tính trung bình cộng của các tổng các dãy tăng dần có trong mảng các số nguyên.

Ví dụ: 1 2 3 4 2 3 4 5 6 4 5 6 $\Rightarrow TB = 15$.

C6.178. Viết chương trình tính tổng tất cả các phần tử xung quanh trên mảng các số nguyên. (Phần tử xung quanh là hai phần tử bên cạnh cộng lại bằng chính nó (Ví dụ: 1 3 2 \rightarrow 1, 2 là hai phần tử xung quanh của 3).

Ví dụ: 1 3 2 5 3 9 6 \rightarrow tổng 17

C6.179. (**) Viết chương trình nhập vào hai số lớn a, b nguyên (a, b có từ 20 chữ số trở lên). Tính tổng, hiệu, tích, thương của hai số trên.

C6.180. Viết hàm tính tổng các phần tử là số Armstrong (số Armstrong là số có đặc điểm như sau: số có k ký số, tổng của các lũy thừa bậc k của các ký số bằng chính số đó.

Ví dụ: 153 là số có các ký số $1^3 + 5^3 + 3^3 = 153$ là một số Armstrong).

C6.181. Viết hàm tìm và xóa tất cả các phần tử trùng với x trong mảng một chiều các số nguyên, nếu không tồn tại phần tử x trong mảng thì trả về -1.

C6.182. Viết hàm xóa tất cả những phần tử trùng nhau trong dãy chỉ giữ lại một phần tử trong đó.

Ví dụ: 1 6 2 3 2 4 2 6 5 → 1 6 2 3 4 5

C6.183. (**) Viết hàm xóa những phần tử sao cho mảng kết quả có thứ tự tăng dần và số lần xóa là ít nhất.

C6.184. Cho dãy a gồm n số nguyên có thứ tự tăng dần. Nhập vào một phần tử nguyên X, viết hàm chèn X vào dãy sao cho dãy vẫn có thứ tự tăng dần (không sắp xếp).

C6.185. Viết chương trình tìm số lẻ nhỏ nhất lớn hơn mọi số chẵn có trong mảng.

C6.186. Viết hàm tìm giá trị chẵn nhỏ nhất nhỏ hơn mọi giá trị lẻ trong mảng các số nguyên.

C6.187. Viết hàm tìm phần tử xuất hiện nhiều nhất trong mảng các số nguyên.

C6.188. Viết chương trình đếm và liệt kê các mảng con tăng dần trong mảng một chiều các số nguyên.

Ví dụ: 6 5 3 2 3 4 2 7 các dãy con tăng dần là 2 3 4 và 2 7

C6.189. Viết chương trình tìm mảng con tăng dần có tổng lớn nhất trong mảng một chiều.

C6.190. (*) Viết chương trình nhập vào một dãy số a gồm n số nguyên ($n \leq 100$). Tìm và in ra dãy con tăng dài nhất

Ví dụ: Nhập dãy a : 1 2 3 6 4 7 8 3 4 5 6 7 8 9 4 5

Dãy con tăng dài nhất : 3 4 5 6 7 8 9

C6.191. (**) Viết chương trình tách 1 mảng các số nguyên thành 2 mảng a và b, sao cho kết quả thu được là:

- Mảng a chứa toàn số lẻ tăng dần.
- Mảng b chứa toàn số chẵn giảm dần.

(Không dùng sắp xếp)

Hướng dẫn: Tìm vị trí chèn thích hợp khi trích phần tử từ mảng ban đầu.

Ví dụ: Mảng ban đầu: 9 3 8 2 7 5 1 0 10

Mảng a: 1 3 5 7 9

Mảng b: 10 8 2

- C6.192. (**) Viết chương trình in ra tam giác Pascal (dùng mảng một chiều).
- C6.193. Viết chương trình nhập vào dãy số a gồm n số thực ($n \leq 100$), nhập vào dãy số b gồm m số thực ($m \leq 100$).
- Hãy sắp xếp hai dãy theo thứ tự tăng dần.
 - (*) Trộn 2 dãy trên thành dãy c sao cho dãy c vẫn có thứ tự tăng.
 - Xuất dãy a, b, c ra màn hình.
- C6.194. (*) Cho mảng C có n phần tử ($n < 200$), các phần tử là các chữ số trong hệ đếm cơ số 16 (Hexa) (điều kiện mỗi phần tử $\leq n$). Hãy tách mảng C ra các mảng con theo điều kiện sau: các mảng con được giới hạn bởi hai lần xuất hiện thứ hai của con số trong dãy.
- Ví dụ: Dãy **123A4518B23**
- có các dãy con là 123A451, 23A4518B2, 23A4518B23*
- C6.195. (**) Cho hai số nguyên dương A, B. Hãy xác định hai số C, D tạo thành từ hai số A, B sao cho C là số lớn nhất, D là số nhỏ nhất. Khi gạch đi một số chữ số trong C (D), thì các số còn lại giữ nguyên tạo thành A, các chữ số bỏ đi giữ nguyên tạo thành B.
- Ví dụ: $A = 52568, B = 462384 \rightarrow C = 54625682384, D = 45256236884$.
- C6.196. Viết chương trình nhập vào dãy số a gồm n số nguyên ($n \leq 100$). Hãy đảo ngược dãy đó.
- Ví dụ: Nhập a: 3 4 5 2 0 4 1
- Dãy sau khi đảo: 1 4 0 2 5 4 3*
- C6.197. Hãy kiểm tra xem dãy đã cho có thứ tự chưa (dãy được gọi là thứ tự khi là dãy tăng hoặc dãy giảm).
- C6.198. Cho mảng A có n phần tử hãy cho biết mảng này có đối xứng hay không.
- C6.199. (**) Hãy viết chương trình phát sinh ngẫu nhiên mảng các số nguyên gồm 10.000 phần tử, mỗi phần tử có giá trị từ 0 đến 32.000 và xây dựng hàm thống kê số lần xuất hiện các phần tử trong mảng, sau đó cho biết phần tử nào xuất hiện nhiều lần nhất.

Ví dụ: Giá trị của mảng: 5 6 11 4 4 5 4

5 xuất hiện 2 lần

6 xuất hiện 1 lần

11 xuất hiện 1 lần

4 xuất hiện 3 lần

Kết quả : 4 xuất hiện nhiều lần nhất

C6.200. Cho mảng A có n phần tử. Nhập vào số nguyên k ($k \geq 0$), dịch phải xoay vòng mảng A k lần.

Ví dụ: Mảng A: 5 7 2 3 1 9

Nhập k = 2

Dịch phải xoay vòng mảng A: 1 9 5 7 2 3

TÀI LIỆU THAM KHẢO

1. Phạm Văn Ất, *Kỹ thuật lập trình C: cơ sở và nâng cao*, nhà xuất bản Khoa Học Kỹ Thuật, 1996.
2. Lê Hoài Bắc, Lê Hoàng Thái, Nguyễn Tấn Trần Minh Khang, Nguyễn Phương Thảo, *Giáo trình ngôn ngữ C*, nhà xuất bản Đại Học Quốc Gia Tp. Hồ Chí Minh, 2003.
3. Nguyễn Tấn Trần Minh Khang, *Bài tập Kỹ thuật lập trình – Tập 1*, nhà xuất bản Đại Học Quốc Gia Tp. Hồ Chí Minh, 2004.
4. Nguyễn Đình Tê, Hoàng Đức Hải, *Giáo trình lý thuyết & Bài tập ngôn ngữ C*, nhà xuất bản mũi Cà Mau.
5. Huỳnh Tấn Dũng, Hoàng Đức Hải, *Bài tập ngôn ngữ C từ A đến Z*, nhà xuất bản Lao Động – Xã Hội.
6. Nguyễn Thanh Sơn, *Tập bài giảng Kỹ thuật lập trình*, 2004.
7. Trần Minh Thái, *Tập bài giảng Lập trình C cơ bản*, 2013.
8. John R. Hubbard, *455 Bài tập cấu trúc dữ liệu cài đặt bằng C++*, bản dịch của Minh Trung, Gia Việt, nhà xuất bản Thống Kê.