



TRƯỜNG ĐẠI HỌC MỞ TP.HCM

GIÁO TRÌNH LẬP TRÌNH C++



LÊ PHÚ HIẾU

Biên soạn

TRƯỜNG ĐẠI HỌC MỞ TP.HCM

GIÁO TRÌNH LẬP TRÌNH C++

DÙNG CHO SINH VIÊN

THUỘC KHỐI NGÀNH KỸ THUẬT

KHÔNG CHUYÊN VỀ CÔNG NGHỆ THÔNG TIN

Tác giả Lê Phú Hiếu

THÀNH PHỐ HỒ CHÍ MINH

LỜI NÓI ĐẦU

C++ hiện là ngôn ngữ lập trình (NNLT) được ưa chuộng và sử dụng rộng rãi do tính năng mạnh mẽ, đa dụng của nó. Đây là NNLT được nhiều trường đại học và cao đẳng trong và ngoài nước sử dụng để giảng dạy lập trình ban đầu cho sinh viên.

Giáo trình “Lập trình C++” này được viết nhằm phục vụ cho đối tượng là sinh viên các hệ Cao đẳng và Đại học thuộc khối ngành Kỹ thuật (Không chuyên về Công nghệ Thông Tin). Mục tiêu của giáo trình nhằm cung cấp kiến thức cơ bản về ngôn ngữ lập trình C++, đủ để giải quyết những bài toán lập trình trong nhiều lĩnh vực khác nhau như: tính toán, kỹ thuật, sinh học, quản lý, ...

Với số lượng trang khiêm tốn, giáo trình này giới hạn, không đề cập đến cách tiếp cận lập trình hướng đối tượng mà chỉ nhấn mạnh đến kiểu lập trình truyền thống, lập trình đơn thể (module), theo tiếp cận hướng cấu trúc. Kiểu lập trình truyền thống là quan trọng và theo ý kiến cá nhân là bắt buộc phải giảng dạy cho người mới bắt đầu học lập trình trước khi chuyển sang học lập trình theo tiếp cận hướng đối tượng.

Giáo trình được tổ chức thành 8 chương, cuối mỗi chương đều có câu hỏi và bài tập để củng cố kiến thức phần lý thuyết. Phần lý thuyết được trình bày cô đọng, tập trung vào những kiến cốt lõi, cần thiết để áp dụng trong viết mã

chương trình. Phần giải bài tập sẽ được thực hiện trong tài liệu khác. Các chương trình ví dụ minh họa trong tài liệu đã được thử nghiệm và chạy tốt trong môi trường Visual C++ 6.0.

Giáo trình này có thể được dùng làm tài liệu cho sinh viên tự học. Giảng viên cũng có thể sử dụng để tham khảo trong giảng dạy môn lập trình.

Mặc dù có nhiều cố gắng nhưng không thể tránh khỏi thiếu sót, rất mong nhận được sự nhận xét, phê bình của đồng nghiệp và bạn đọc để tài liệu được hoàn thiện hơn.

Tác giả

Mục lục

<i>BỘ GIÁO DỤC VÀ ĐÀO TẠO</i>	<i>1</i>
<i>CHƯƠNG 1. MỘT SỐ KHÁI NIỆM CƠ BẢN VỀ LẬP TRÌNH</i>	<i>13</i>
1. Thuật toán (Algorithm)	13
1.1. Khái niệm	13
1.2. Các đặc trưng của thuật toán	13
1.3. Các công cụ biểu diễn thuật toán	14
2. Ngôn ngữ lập trình (NNLT)	15
3. Chương trình (máy tính)	16
4. Các bước xây dựng chương trình	17
5. Câu hỏi và bài tập	18
<i>CHƯƠNG 2. CÁC YẾU TỐ CƠ BẢN CỦA NNLT C/C++</i>	<i>21</i>
1. Bộ ký tự (character set)	21
2. Danh hiệu (identifier)	21
3. Từ khóa (keyword)	22
4. Chú thích (comment)	22
5. Các kiểu dữ liệu cơ bản (base type)	23
5.1. Số nguyên	23

5.2.	Số thực (độ chính xác đơn – 6 chữ số lẻ)	23
5.3.	Số thực (độ chính xác kép – 15 chữ số lẻ):	24
5.4.	Kí tự	24
6.	Hằng (constant)	25
	<i>Một số kí tự đặc biệt:</i>	26
7.	Biến (variable)	27
8.	Biểu thức (expression)	29
9.	Chuyển đổi kiểu (type conversion)	29
10.	Các toán tử (operator)	30
10.1.	Toán tử số học	30
10.2.	Toán tử quan hệ (so sánh)	30
10.3.	Toán tử logic	31
10.4.	Toán tử gán	31
10.5.	Toán tử điều kiện (? :)	32
10.6.	Toán tử tăng (++), giảm (—) một đơn vị	33
	<i>Biểu thức có dạng</i>	33
10.7.	Toán tử dịch chuyển số học	34
10.8.	Toán tử thao tác trên từng bit	34

10.9. Toán tử ép kiểu (đã nói ở trên)	34
10.10. Toán tử “lấy địa chỉ của” (&)	34
10.11. Toán tử sizeof:	34
11. Độ ưu tiên và trật tự kết hợp các toán tử (precedence and associativity)	35
12. Câu lệnh (statement, instruction)	36
13. Xuất / Nhập dữ liệu đơn giản trong C/C++	37
14. Một vài chương trình C/C++ đơn giản	39
15. Câu hỏi	46
16. Bài tập về phép toán và biểu thức	49
<i>CHƯƠNG 3. CÁC CẤU TRÚC ĐIỀU KHIỂN (CONTROL STRUCTURES)</i>	62
1. Cấu trúc rẽ nhánh (if ... then ... else ...)	62
2. Cấu trúc lựa chọn (switch ...)	66
3. Cấu trúc lặp while (...)	69
4. Cấu trúc lặp do ... while (...)	71
5. Cấu trúc lặp for (...)	74
6. Câu lệnh break	77
7. Câu lệnh continue (ít dùng)	78

8. Câu lệnh goto	78
9. Câu hỏi	79
10. Bài tập	91
10.1. Cấu trúc rẽ nhánh	91
10.2. Cấu trúc lựa chọn	92
10.3. Cấu trúc lặp	93
CHƯƠNG 4. HÀM (FUNCTION)	102
1. Khái niệm	102
2. Khai báo và định nghĩa hàm	103
3. Lệnh return	104
4. Các dạng hàm	104
4.1. Hàm không có tham số	104
4.2. Hàm với nhiều tham số	106
4.3. Hàm nguyên mẫu (function prototype)	107
4.4. Tổ chức một chương trình “C/C++” (trên cùng 1 tập tin)	108
5. Các phương pháp truyền tham số	109
6. Phạm vi (scope) và cấp lưu trữ (storage class) của các đối tượng	111
6.1. Phạm vi	111

6.2. Cấp lưu trữ	113
7. Cơ chế gọi hàm và bộ nhớ stack	120
8. Hàm có tham số với giá trị ngầm định (Function with default arguments)	123
9. Nạp chồng hàm (Function overloading)	124
10. Một số gợi ý khi thiết kế hàm	126
11. Một số ví dụ minh họa	127
12. Câu hỏi	127
13. Bài tập	131
<i>CHƯƠNG 5. KIỂU MẢNG (ARRAY)</i>	134
1. Khái niệm	134
2. Khai báo & khởi tạo giá trị các phần tử mảng trong “C/C++”	134
3. Truy xuất các phần tử của mảng	136
4. Truyền tham số mảng cho hàm	137
5. Các thao tác cơ bản trên mảng 1 chiều	138
5.1. Nhập giá trị cho các phần tử mảng.	138
5.2. Xuất giá trị các phần tử mảng (ra màn hình).	140
5.3. Thêm 1 phần tử vào mảng.	141

5.4. Xóa một phần tử ra khỏi mảng.	142
5.5. Tìm kiếm trên mảng.	143
5.6. Sắp xếp mảng.	145
6. Câu hỏi	149
7. Bài tập	149
<i>Mảng 1 chiều</i>	<i>149</i>
<i>Mảng 2 chiều</i>	<i>152</i>
CHƯƠNG 6. CON TRỎ (POINTER)	153
1. Khái niệm	153
2. Khai báo biến con trỏ	153
3. Truy xuất biến trỏ	154
4. Số học con trỏ	155
5. Liên hệ giữa con trỏ và mảng	156
6. Con trỏ đa cấp	157
7. Truyền tham số con trỏ cho hàm	158
8. Mảng các con trỏ	158
9. Từ khóa const với con trỏ	159
10. Cấp phát động	160

11. Con trỏ hàm _____	164
12. Con trỏ và chuỗi kí tự _____	165
13. Ứng dụng con trỏ _____	167
14. Sơ lược về kiểu tham chiếu (Reference) - Chỉ có trong C++. _	167
15. Bài tập _____	168
<i>CHƯƠNG 7. KIỂU STRUCT</i> _____	175
1. Khái niệm _____	175
2. Khai báo và truy xuất biến struct trong ‘C’ _____	175
3. Lệnh typedef _____	177
4. Kiểu enum _____	178
<i>Khai báo</i> _____	<i>178</i>
5. Mảng các struct _____	179
6. Con trỏ trỏ đến biến kiểu struct _____	180
7. Truyền tham số kiểu struct cho hàm _____	181
8. Kiểu union _____	181
9. Bài tập _____	182
<i>CHƯƠNG 8. File</i> _____	183
1. Giới thiệu chung _____	183

2. Thao tác trên kiểu file	183
2.1. Mở File (liên kết kênh nhập/xuất với file đĩa)	183
2.2. Xử lý File	184
2.3. Đóng File	185
2.4. Ví dụ Vào/Ra File định dạng	185
2.5. Ví dụ Vào/Ra kí tự	188
2.6. Ví dụ Vào/Ra File nhị phân	189
3. Bài tập	191
TÀI LIỆU THAM KHẢO	193

CHƯƠNG 1. MỘT SỐ KHÁI NIỆM CƠ BẢN VỀ LẬP TRÌNH

1. Thuật toán (Algorithm)

1.1. Khái niệm

- Thuật toán là khái niệm cơ sở của toán học và tin học.
- Thuật toán là phương pháp thể hiện lời giải của vấn đề – bài toán.
- Thuật toán là dãy các thao tác, các hướng dẫn rõ ràng, được sắp xếp theo một trình tự xác định, sao cho 2 bộ xử lý (người/máy) khác nhau, với cùng điều kiện đầu vào như nhau thì sau một số bước hữu hạn thực hiện, sẽ cho kết quả giống nhau mà không cần biết ý nghĩa của các thao tác này. Cần chú ý là không phải mọi dãy thao tác, chỉ dẫn nào cũng đều tạo ra thuật toán. Phương pháp nấu ăn, cách dùng thuốc,.. .. đều không phải là thuật toán do các thao tác, các chỉ dẫn là không xác định, không rõ ràng.

1.2. Các đặc trưng của thuật toán

Tính xác định: Các thao tác của thuật toán phải xác định, không được nhập nhằng, mơ hồ để có thể dễ dàng cài đặt trên một hệ tự động hóa.

Tính dừng: Thuật toán phải dừng sau một số hữu hạn bước thực hiện.

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

Tính phổ dụng: Thuật toán có thể được sử dụng lại để giải một lớp bài toán tương tự.

Tính hiệu quả: Thuật toán cần tối ưu về sử dụng bộ nhớ và đáp ứng yêu cầu của bài toán trong thời gian ngắn nhất có thể được. Thực tế rất khó đạt được cả 2 yêu cầu này trong một thuật toán.

1.3. Các công cụ biểu diễn thuật toán

Ngôn ngữ tự nhiên: là ngôn ngữ liệt kê các bước, mô tả thuật toán theo ngôn ngữ tự nhiên của con người.

Ví dụ: Thuật toán xác định trị lớn nhất trong 5 số nguyên.


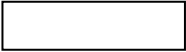
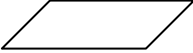
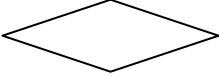
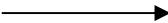
B1. Gọi a, b, c, d, e là 5 biến lưu trữ các trị nguyên cho trước (nhập từ bàn phím).

B2. Gọi max là biến lưu trữ trị lớn nhất trong 5 số nguyên trên, và giả sử a có trị lớn nhất.

B3. Lần lượt so sánh trị của max với các biến b, c, d, e còn lại. Nếu trị của max nhỏ hơn bất kỳ biến nào thì gán trị của biến đó cho max.

B4. Xuất kết quả trị biến max ra màn hình

Lưu đồ thuật toán hay sơ đồ khối (Flow chart): là công cụ cho phép biểu diễn thuật toán một cách trực quan. Thường chỉ có thể dùng công cụ lưu đồ đối với các thuật toán tương đối ngắn, có thể được biểu diễn trong một trang giấy. Các hình cơ bản sử dụng trong lưu đồ:

Hình oval mô tả điểm xuất phát / kết thúc.	
Hình chữ nhật mô tả một hay nhiều chỉ thị máy cần thực hiện.	
Hình bình hành mô tả thao tác nhập/xuất dữ liệu.	
Hình thoi mô tả sự rẽ nhánh, lựa chọn, phép kiểm tra điều kiện	
<i>Mũi tên chỉ hướng lưu chuyển của các thao tác.</i>	

Mã giả (Pseudo code) gần giống như ngôn ngữ tự nhiên, nhưng có sử dụng các cấu trúc chuẩn hóa (khai báo biến, chú thích, cấu trúc điều khiển, . . .) do người thiết kế quy định.

Ngôn ngữ lập trình (Programming language) là hệ thống các ký hiệu cho phép mô tả các quy trình tính toán dưới dạng văn bản.

2. Ngôn ngữ lập trình (NNLT)

Các thành phần cơ bản của NNLT bao gồm:

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

Hiện đã có hàng nghìn NNLT được thiết kế, và hàng năm lại có thêm nhiều NNLT mới xuất hiện. Sự phát triển của NNLT gắn liền

với sự phát triển của ngành tin học. Mỗi loại NNLT phù hợp với một số lớp bài toán nhất định.

Phân loại NNLT:

- *Ngôn ngữ máy* (machine language) hay còn gọi là NNLT cấp thấp có tập lệnh phụ thuộc vào một hệ máy cụ thể. Chương trình viết bằng ngôn ngữ máy sử dụng bảng chữ cái chỉ gồm 2 kí tự 0, 1. Chương trình ngôn ngữ máy được nạp trực tiếp vào bộ nhớ và thực hiện ngay.
- *Ngôn ngữ lập trình cấp cao* nói chung không phụ thuộc vào loại máy tính cụ thể. Chương trình viết bằng NNLT cấp cao sử dụng bộ kí tự phong phú hơn, và phải được chuyển đổi sang dạng mã máy để máy tính có thể hiểu được bằng chương trình dịch. Một số NNLT cấp cao thông dụng hiện nay: Pascal, C, C++, Java, Smalltalk, Basic, Ruby, Fortran, Algol, Lisp, Prolog, Cobol, ...

3. Chương trình (máy tính)

Là tập hợp hữu hạn các chỉ thị máy được bố trí, sắp xếp theo một trật tự xác định, nhằm giải quyết yêu cầu của bài toán đặt ra. Chương trình được viết bằng một NNLT cụ thể nào đó.

Các chương trình C/C++ (trong môi trường DOS) được tạo ra bằng 1 trình soạn thảo văn bản (EDITOR) như: SK, NC Editor, VRES . . . Hiện nay, các chương trình dịch đều tích hợp sẵn editor riêng cho phép USER soạn thảo, biên dịch, kiểm lỗi, liên kết và thực hiện chương trình một cách dễ dàng.

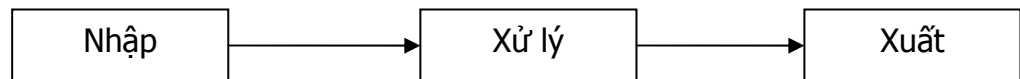
Các chương trình này (mã nguồn – source code), thực chất là ở dạng ngôn ngữ tự nhiên, do đó phải được biên dịch lại dưới dạng mã máy (object code) mà máy tính có thể hiểu được. Việc này được thực hiện bởi chương trình dịch.

Có 2 loại chương trình dịch:

Trình thông dịch (interpreter): mỗi lệnh được dịch sang mã máy và cho thực hiện ngay.

Trình biên dịch (compiler): toàn bộ chương trình nguồn được dịch sang mã máy (tập tin.obj), sau đó trình liên kết (linker) sẽ kết nối các module chương trình để tạo thành tập tin EXE.

Nói chung, một chương trình máy tính được bố cục thành 3 phần:



4. Các bước xây dựng chương trình

B1. **Xác định đúng yêu cầu của bài toán:** Cần xác định phạm vi, các giới hạn, ràng buộc, các giả thiết của bài toán. Đặc biệt cần khắc phục sự ì về mặt tâm lý trong quá trình tìm hiểu bài toán.

B2. **Xây dựng thuật giải:** Cần có kiến thức liên quan đến vấn đề đang giải quyết. Cần xác định rõ thuật toán sẽ tác động trên những đối tượng (thông tin) nào? Có bao nhiêu đối tượng (biến) cần xử lý? Mỗi biến có thể được lưu trữ dưới dạng nào, kiểu gì? Giá trị ban đầu của các biến? Hình dung trước kết xuất DL sau khi xử lý sẽ như thế nào?

B3. Thể hiện thuật giải bằng *lưu đồ thuật toán* (nếu được).

B4. *Cài đặt thuật toán bằng một NNLT cụ thể*: Dùng một trình soạn thảo VB để tạo chương trình nguồn (source code) theo một NNLT nào đó.

B5. *Thử nghiệm thuật toán*, nếu sai quay lại B2. Cần xác định lỗi của thuật toán thuộc loại nào: lỗi về mặt cú pháp (syntax error), lỗi lúc thực hiện chương trình (run-time error), và lỗi logic. Lỗi cú pháp xảy ra lúc biên dịch chương trình, do vi phạm các quy định về mặt cú pháp của NNLT đang sử dụng. Lỗi này tương đối dễ khắc phục. Lỗi run-time error như: divide by zero, stack overflow, không đủ bộ nhớ, ... Lỗi logic (logic error) khó phát hiện hơn nhiều.

B6. Kết thúc.

5. Câu hỏi và bài tập

- Thuật toán (Algorithm)
 - Thuật toán là gì?
 - Nêu các đặc trưng cần có của một thuật toán.
 - Các cách biểu diễn một thuật toán?
- Ngôn ngữ lập trình (Programming language) là gì?
- Nêu các bước xây dựng thuật toán.
- Danh hiệu (Identifier)
 - Danh hiệu được dùng để làm gì?
 - Như thế nào là một danh hiệu hợp lệ?
 - Nguyên tắc sử dụng danh hiệu?

- Từ khóa là gì?
 - Đặc điểm của các từ khóa trong NNLT “C/C++”?
- Kiểu dữ liệu (Data type)
 - Trình bày các kiểu dữ liệu đơn giản mà Anh (Chị) đã biết.
- Hằng (Constant)
 - Hằng là gì?
 - Hằng được sử dụng khi nào?
 - Cho biết cách thức khai báo một hằng?
 - Cho ví dụ về cách biểu diễn hằng nguyên, thực, ký tự, chuỗi kí tự.
- Biến (Variable)
 - Biến là gì?
 - Biến được sử dụng để làm gì?
 - Cho biết cách thức khai báo một biến?
 - Hãy cho biết cách thức làm thay đổi nội dung (giá trị) của một biến?
- Biểu thức (Expression)
 - Biểu thức là gì?
 - Kiểu của biểu thức do quyết định?
- Khi nào xảy ra việc ép kiểu tự động?
 - Khi nào cần phải sử dụng toán tử ép kiểu?

- Trình bày nguyên tắc NNLT “C/C++” tính trị các biểu thức?
- Toán tử
 - Hãy trình bày các toán tử mà Anh (Chị) biết.
 - Cho ví dụ về toán tử điều kiện (...?: ...).
- Hãy viết biểu thức tương đương không có toán tử ‘!’:
 - $!(x \leq 5)$
 - $!(x > 5)$
 - $!(x > 2 \ \&\& \ y \neq 3)$
 - $!(x > 2 \ || \ y == 3)$
 - $!(x == 1 \ \&\& \ (x \neq 3) \ || \ x > 10)$
 - $!(x > 100 \ || \ x < 0 \ \&\& \ (x == 0))$
- Câu lệnh (Statement, Instruction)
 - Cho biết các loại câu lệnh trong “C/C++” ? Cho ví dụ.

CHƯƠNG 2. CÁC YẾU TỐ CƠ BẢN CỦA NNLT C/C++

1. Bộ ký tự (character set)

NNLT C/C++ chỉ chấp nhận các ký tự sau:

- Các ký tự chữ hoa: A, B, C, . . . , Z
- Các ký tự chữ thường: a, b, c, . . . , z
- Các chữ số: 0, 1, . . . , 9
- Các ký tự dấu: , . ! ? : . . .
- Các ký tự trắng: ENTER, BACKSPACE, khoảng trắng.
- Các ký tự đặc biệt khác: + - * / ^ | # \$ & % () [] _ = ~ ‘ “ . . .

2. Danh hiệu (identifier)

Dùng để đặt tên cho các đối tượng như hằng, biến, hàm, . . .

Độ dài tối đa của 1 danh hiệu (tùy theo chương trình dịch) thường là 31-32 ký tự.

Danh hiệu hợp lệ được bắt đầu bằng một ký tự chữ cái hoặc dấu gạch nối (underscore), tiếp theo sau là dãy các ký tự chữ hoặc số hoặc dấu gạch nối, và không phép có khoảng trắng ở giữa.

Nên đặt danh hiệu theo các gợi ý sau:

- Đặt tên cho các đối tượng một cách gợi nhớ (Mnemonic).
- Tên hằng được đặt bằng chữ hoa.

Tên biến, tên hàm được đặt như sau: từ đầu tiên bằng chữ thường, các từ còn lại bắt đầu bằng chữ hoa.

Tên kiểu dữ liệu do USER định nghĩa được bắt đầu bằng chữ hoa.

Chú ý:

- Mọi danh hiệu phải được khai báo trước khi sử dụng. TRÌNH BIÊN DỊCH sẽ báo lỗi undefined symbol trong quá trình biên dịch chương trình nếu vi phạm nguyên tắc này.
- “C/C++” phân biệt chữ hoa và chữ thường (Case sensitive).
- Trong cùng một phạm vi (scope), không được đặt trùng danh hiệu..

3. Từ khóa (keyword)

Là từ dành riêng và có ngữ nghĩa xác định do NNLT quy định. Mọi NNLT đều có một bộ từ khóa riêng. Ngôn ngữ lập trình “C/C++” (version 3.1) thể hiện các từ khóa dưới dạng các kí tự màu trắng, các danh hiệu dưới dạng các kí tự màu vàng, số dưới dạng màu xanh lơ, . . .

4. Chú thích (comment)

Được dùng để làm cho chương trình dễ đọc, dễ hiểu, dễ bảo trì hơn.

Chương trình dịch sẽ bỏ qua những nội dung nằm trong phần chú thích.

Có 2 loại chú thích:

- **Chú thích trên một dòng:**

// . . . phần chú thích (cho đến cuối dòng)

- **Chú thích trên nhiều dòng:**

/*... phần chú thích (có thể trải dài trên nhiều dòng)... */

5. Các kiểu dữ liệu cơ bản (base type)

5.1.Số nguyên

Tên kiểu: **int**

Kích thước: 2 bytes và có phạm vi biểu diễn giá trị $-32768 \dots 32767$

Các phép toán áp dụng được trên kiểu int::

- Các phép toán số học: +, -, *, /, %
- Các phép toán so sánh: <, <=, >, >=, ==, !=
- Các phép toán dịch chuyển số học: >>, và <<
- Các phép toán trên bit: ~ (not bit), & (and bit), | (or bit), ^ (xor bit)

5.2.Số thực (độ chính xác đơn – 6 chữ số lẻ)

Tên kiểu: **float**

Kích thước: 4 bytes và có phạm vi biểu diễn giá trị $-3.4E -38 \dots 3.4E +38$

Các phép toán áp dụng được trên kiểu float:

- Các phép toán số học: +, -, *, / (không có phép toán %)
- Các phép toán so sánh: <, <=, >, >=, ==, !=

5.3.Số thực (độ chính xác kép – 15 chữ số lẻ):

Tên kiểu: **double** (Mặc định trong Visual C++)

Kích thước: 8 bytes và có phạm vi biểu diễn giá trị $-1.7E-308 \dots 1.7E+308$

Các phép toán áp dụng được trên kiểu double: như kiểu float

5.4.Kí tự

Tên kiểu: **char**

Kích thước: 1 byte và có phạm vi biểu diễn giá trị $-128 \dots 127$

Các phép toán áp dụng được trên kiểu char:

- Các phép toán số học: +, -, *, /, %
- Các phép toán so sánh: <, <=, >, >=, ==, !=

Để mở rộng các kiểu dữ liệu cơ sở, “C/C++” đưa thêm các tiền tố (prefix): **short, long, unsigned, signed** vào trước tên các kiểu cơ sở như sau:

Tên kiểu	Kích thước
unsigned char	1 byte (0 .. 256)
Char	1 byte (-128 .. 127)
unsigned int (unsigned)	2 bytes (0 .. 65535)
short int	2 bytes (-32768 .. 32767)
Int	2 bytes (-32768 .. 32767)

unsigned long int (unsigned long)	4 bytes (0 .. 4294967295)
long int (long)	4 bytes (-2147483648 .. 2147483647)
float	4 bytes (-3.4E -38 .. 3.4E +38)
double	8 bytes (-1.7E -308 .. 1.7E +308)
long double	10 bytes (-3.4 E -4932 .. 1.1E+4932)

6. Hằng (constant)

Là đại lượng có giá trị không thay đổi trong suốt thời gian tồn tại của nó.

Tên hằng phải là một danh hiệu hợp lệ và phải “được khai báo trước khi sử dụng”.

Hằng được dùng để thay thế cho các con số tối nghĩa trong chương trình. Việc sử dụng hằng những lúc cần thiết là một phong cách lập trình tốt và cần được khuyến khích.

Hằng số nguyên: có thể được biểu diễn dưới dạng thập phân, nhị phân, bát phân, và thập lục phân.

Hằng số thực: có thể được biểu diễn dưới dạng kí pháp thông thường hoặc dạng kí pháp khoa học.

- Kí pháp thông thường (còn gọi là số thực dấu phẩy tĩnh) gồm 2 phần, được phân cách bởi dấu chấm thập phân. Ví dụ: 1234.5
- Kí pháp khoa học (còn gọi là số thực dấu phẩy động) gồm phần định trị (là một số thực) và phần mũ (là một số

nguyên). Hai phần này được phân cách bởi chữ e hoặc E. Ví dụ: 1.2345 E+03.

Hằng kí tự: được đặt trong cặp nháy đơn và có thể được biểu diễn bằng:

- Kí hiệu trong bảng mã ASSCI. Ví dụ: 'A'.
- Escape character (kí tự thoát) bao gồm cặp kí tự \n, với n là số thứ tự của kí tự trong bảng mã ASSCII. Ví dụ: '\65', '\7', '\n'

Một số kí tự đặc biệt:

Kí hiệu	Ý nghĩa
\n	Kí tự xuống dòng
\t	Kí tự TAB
\0	Kí tự NULL
\'	Dấu nháy đơn
\”	Dấu nháy kép
\\	Dấu sổ chéo ngược (Backslash)
.....	

Hằng chuỗi kí tự: được đặt trong cặp nháy đôi. Thực chất đó là mảng các kí tự có kí tự kết thúc chuỗi là kí tự NULL, kí hiệu '\0'

Khai báo hằng: thường được đặt trong phần khai báo toàn cục ở đầu chương trình, ngay sau các khai báo chỉ thị tiền xử lý. Có 2 cách khai báo hằng:

- Dùng chỉ thị tiền xử lý: #define <tên hằng> <chuỗi thay thế>
- Dùng từ khoá const: const <tên kiểu> <tên hằng>=<giá trị>;

Ví dụ:

```
const int MAX = 10;
```

hay

```
#define MAX 10
```

7. Biến (variable)

Là đại lượng có giá trị có thể thay đổi bằng toán tử gán (kí hiệu =). Biến được dùng để lưu trữ thông tin về các đối tượng và có giá trị cụ thể, xác định tại mỗi thời điểm trong chương trình.

Tên biến phải là một danh hiệu hợp lệ và không được đặt trùng với từ khóa. Nên đặt tên biến sao cho có tính gợi nhớ, không dài quá và cũng không nên quá ngắn. Nên tránh đặt tên biến trùng với tên các hàm thư viện (sin, cos, . . .).

“C/C++” cho phép khai báo biến ở khắp mọi nơi trong chương trình, miễn sao đảm bảo nguyên tắc “Mọi danh hiệu trước khi sử dụng phải được khai trước”.

Cách khai báo biến:

```
<type> <variables list>;
```

Trong đó:

<type> là một kiểu dữ liệu hợp lệ bất kỳ có trong “C/C++”.

<variables list> là một hay nhiều biến, được phân cách bằng dấu ‘,’.

Chú ý: lệnh khai báo biến luôn được kết thúc bằng dấu ‘;’

Ví dụ:

Để khai báo biến kí tự

```
char ch; // khai báo biến kí tự
```

Để khai báo 2 biến nguyên:

```
int x, y; // khai báo 2 biến kiểu nguyên
```

hoặc

```
int x;
```

```
int y;
```

Có thể khai báo và đồng thời khởi tạo giá trị cho biến như sau:

```
<tên kiểu> <tên biến> = <giá trị>;
```

Ví dụ:

```
int tong = 0, x = 5, y = 7;
```

8. Biểu thức (expression)

Là công thức tính toán bao gồm các toán hạng và các toán tử tương ứng. Các toán hạng có thể là một biến, hằng, lời gọi hàm. Bản thân các toán hạng cũng có thể là một biểu thức con khác đặt trong cặp ngoặc đơn, hình thành nên một biểu thức phức hợp.

Có các loại biểu thức thông dụng sau: biểu thức gán, biểu thức số học, biểu thức logic. Đặc biệt, biểu thức logic trong “C/C++” được xem là có giá trị nguyên. Biểu thức có trị khác 0 (kể cả số âm) tương ứng với mệnh đề logic TRUE, và biểu thức có trị = 0 tương ứng với mệnh đề logic FALSE.

Trong “C/C++”, biểu thức luôn trả về một giá trị.

Kiểu của biểu thức phụ thuộc vào kiểu của giá trị trả về.

9. Chuyển đổi kiểu (type conversion)

Chuyển đổi kiểu ngầm định: Trong cùng 1 biểu thức, nếu các toán hạng không cùng kiểu với nhau thì trước khi tính toán giá trị của biểu thức, chương trình dịch sẽ thực hiện việc chuyển đổi kiểu ngầm định (nếu được) theo nguyên tắc “Kiểu có phạm vi giá trị biểu diễn nhỏ hơn sẽ được chuyển sang kiểu có phạm vi giá trị biểu diễn lớn hơn”. Sơ đồ chuyển đổi kiểu ngầm định:

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

Ép kiểu (type casting): Trong một số trường hợp, ta bắt buộc phải sử dụng đến toán tử ép kiểu để tạo ra một biểu thức hợp lệ như sau:
<tên kiểu> (<biểu thức>) hoặc (<tên kiểu>) <biểu thức>

Ví dụ, để tạo biểu thức số học hợp lệ sau `8.0 % 3`, ta cần thực hiện ép kiểu như sau:

```
(int) 8.0 % 3
```

Hay

```
int (8.0 % 3)
```

10. Các toán tử (operator)

“C/C++” rất giàu về toán tử. Sau đây là một số toán tử thông dụng:

10.1. Toán tử số học

Kí hiệu	Ý nghĩa	Số ngôi	Toán hạng
+	Cộng	2	int, float, double, char
-	Trừ	2	- nt -
*	Nhân	2	- nt -
/	Chia	2	- nt -
%	Modulo	2	int, char (không có kiểu số thực)

10.2. Toán tử quan hệ (so sánh)

Kí hiệu	Ý nghĩa	Số ngôi	Toán hạng
<	Nhỏ hơn	2	int, float, double, char
<=	Nhỏ hơn hoặc bằng	2	- nt -

>	Lớn hơn	2	- nt -
>=	Lớn hơn hoặc bằng	2	- nt -
==	So sánh bằng	2	- nt -
!=	So sánh khác nhau	2	- nt -

10.3. Toán tử logic

Kí hiệu	Ý nghĩa	Số ngôi	Toán hạng
!	NOT logic	1	int, float, double, char
&&	AND logic	2	- nt -
	OR logic	2	- nt -

10.4. Toán tử gán

Toán tử gán dùng để thay đổi trị của một biến bằng trị của một biểu thức.

Kí hiệu: =

Biểu thức gán có dạng: <biến> = <biểu thức>, trong đó <biến> là một danh hiệu hợp lệ (nằm ở vế trái), và <biểu thức> (nằm ở vế phải) là biểu thức có cùng kiểu với kiểu của <biến>.

Cách tính trị của biểu thức gán: TRÌNH BIÊN DỊCH tính trị của <biểu thức>, sau đó gán trị này cho <biến>. Toàn bộ biểu thức gán này cũng trả về một giá trị là trị của <biểu thức> vừa tính được.

Ví dụ:

```
x = 1; // gán trị 1 cho biến nguyên x
```

```
x = 2*y; // tính trị 2*y, sau đó gán kết quả tính được cho biến x
```

`z = x + 2*y; // tính trị x+2*y, sau đó gán kết quả tính được cho biến z`

`a = sin(2*b); // gọi hàm thư viện tính sin(2*b), sau đó gán kết quả cho a`

C/C++ cho phép **viết gọn các biểu thức gán** bằng các toán tử gán sau:

Dạng viết thông thường	Dạng viết thu gọn	Ý nghĩa
<code>i = i+<bt></code>	<code>i += <bt></code>	Tự cộng
<code>i = i-<bt></code>	<code>i -= <bt></code>	Tự trừ
<code>i = i*<bt></code>	<code>i *= <bt></code>	Tự nhân
<code>i = i/<bt></code>	<code>i /= <bt></code>	Tự chia
<code>i = i%<bt></code>	<code>i %= <bt></code>	Tự modulo
...		

10.5. Toán tử điều kiện (? :)

Là toán tử 3 ngôi, và có dạng:

`<BT1> ? <BT2> : <BT3>`

Trong đó:

- `<BT1>` thường là một biểu thức so sánh hay một biểu thức logic
- `<BT2>`, và `<BT3>` là một biểu thức thông thường nào đó

Kiểu của biểu thức điều kiện phụ thuộc vào kiểu của `<BT2>`, `<BT3>`.

Cách tính giá trị biểu thức điều kiện

B1: Tính trị của <BT1>

B2: Nếu <BT1> có trị $\neq 0$ thì toàn bộ biểu thức điều kiện sẽ nhận trị cho bởi <BT2>. Ngược lại thì chương trình dịch sẽ dùng trị của <BT3> làm trị của toàn bộ biểu thức điều kiện.

Ví dụ

Biểu thức điều kiện trả về trị nhỏ nhất giữa 2 số nguyên cho trước x và y

$$(x < y) ? (x) : (y)$$

Biểu thức điều kiện trả về trị lớn nhất trong 3 số nguyên cho trước a, b, c

$$(x > y) ? (x > z ? x : z) : (y > z ? y : z)$$

10.6. Toán tử tăng (++), giảm (—) một đơn vị

Biểu thức có dạng

$$\begin{array}{ll} \langle \text{biến} \rangle ++ & \langle \text{biến} \rangle -- \\ ++\langle \text{biến} \rangle & --\langle \text{biến} \rangle \end{array}$$

Sự khác nhau giữa 2 dạng tiền tố và hậu tố chỉ nảy sinh khi biểu thức nằm trong một biểu thức khác như sau:

- **Dạng tiền tố:** trị của <biến> được thay đổi trước khi tham gia biểu thức chung

- **Dạng hậu tố:** biểu thức chung sử dụng trị cũ của <biến>, sau đó <biến> mới được thay đổi trị.

10.7. Toán tử dịch chuyển số học

Kí hiệu	Ý nghĩa	Số ngôi	Toán hạng
<<	Dịch trái	2	int
>>	Dịch phải	2	- nt -

10.8. Toán tử thao tác trên từng bit

Kí hiệu	Ý nghĩa	Số ngôi	Toán hạng
!	Đảo bit	1	int
&	AND bit	2	- nt -
	OR bit	2	- nt -

10.9. Toán tử ép kiểu (đã nói ở trên)

10.10. Toán tử “lấy địa chỉ của” (&)

Kí hiệu: &

Bộ nhớ trong của máy tính bao gồm các ô nhớ liên tiếp được đánh số từ 0 đến giới hạn của bộ nhớ. Mỗi biến khai báo trong chương trình sẽ chiếm 1 vùng nhớ có kích thước qui định bởi kiểu đã khai báo cho biến và có địa chỉ là byte đầu tiên được cấp phát. Ta có thể xác định địa chỉ của biến nhờ vào toán tử & như sau:

& <biến>

10.11. Toán tử sizeof:

sizeof(<biểu thức>) trả về kích thước (số bytes) của biểu thức.

sizeof(<kiểu>) trả về kích thước (số bytes) của kiểu.

11. Độ ưu tiên và trật tự kết hợp các toán tử (precedence and associativity)

Trong quá trình tính trị của các biểu thức, NNLT “C/C++” căn cứ vào độ ưu tiên và trật tự kết hợp của các toán tử như sau:

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

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

12. Câu lệnh (statement, instruction)

Có 3 loại câu lệnh:

Câu lệnh đơn là một biểu thức được kết thúc bằng dấu ‘;’ và có thể trải dài trên nhiều dòng. Câu lệnh đơn có dạng: <biến> = <biểu thức>; hay <biểu thức>; hay <lời gọi hàm>;

Câu lệnh ghép hay còn gọi là khối lệnh bao gồm từ 2 câu lệnh trở lên, và được đặt trong cặp ngoặc nhọn { } như sau:

```
{  
  
<lệnh 1>;  
  
<lệnh 2>;  
  
...  
  
<lệnh n>;  
  
}
```

Câu lệnh điều khiển được xây dựng từ các cấu trúc điều khiển như:

- **Cấu trúc rẽ nhánh**: câu lệnh if ... else ...
- **Cấu trúc lựa chọn**: câu lệnh switch (...)
- **Cấu trúc lặp**: câu lệnh for (...), câu lệnh while(...), câu lệnh do...while(...).

Chú ý:

- Câu lệnh ‘;’ là một câu lệnh rỗng, nhưng hợp lệ.

- Khai báo biến, khai báo hàm nguyên mẫu cũng được kết thúc bằng dấu chấm phẩy ‘;’)

13. Xuất / Nhập dữ liệu đơn giản trong C/C++

Xuất dữ liệu ra màn hình:

- Trong “C/C++”, dùng hàm printf(. . .) khai báo trong tập tin tiêu đề <stdio.h> (xem tài liệu)
- Trong “C++”, dùng đối tượng cout và toán tử << khai báo trong <iostream.h> như sau:

```
cout<< BT1[<<BT2<<BT3 . . .];
```

trong đó:

BT1, BT2, . . . là các biểu thức hợp lệ (kiểu char, int, float, double, char*).

Ví dụ:

```
Cout << 65;
```

```
Cout << 'A';
```

```
Cout << "A";
```

```
Cout << 65 << ' ' << ch << ' ' << 'A' << ' ' << "A\n";
```

Nhập dữ liệu từ bàn phím:

- Trong “C/C++”, ta dùng hàm scanf(...) khai báo trong tập tin tiêu đề <stdio.h> (xem tài liệu)

- Trong “C++”, ta dùng đối tượng `cin` và toán tử `>>` khai báo trong `<iostream.h>` như sau:

```
cin >> var1 [ >> var2 >> var3 ...];
```

trong đó:

`var1`, `var2`, ... là các biến hợp lệ (kiểu `char`, `int`, `float`, `double`, `char*`).

Lệnh `cin >> var` yêu cầu đọc các giá trị từ bàn phím cho biến `var` với các qui ước sau:

Các giá trị số được phân cách bởi các kí tự trắng (SPACE BAR, TAB, ENTER). Khi gặp một kí tự không hợp lệ (dấu ‘.’ đối với số nguyên, chữ cái đối với số, . . .) sẽ kết thúc việc đọc từ `cin`. Các kí tự không hợp lệ này sẽ được xem xét trong lần đọc sau.

Đối với giá trị kí tự, và xâu kí tự dấu phân cách là SPACE BAR, TAB, ENTER. Không có khái niệm kí tự không hợp lệ trong việc nhập các giá trị kí tự hay xâu kí tự.

Ví dụ:

Giả sử có các khai báo sau:

```
int x,y;  
  
double z;  
  
char ch;
```

Ta có thể viết các lệnh xuất/nhập DL:

```
cin >> x >> y;
```

```
cin >> z;
```

```
cin >> ch;
```

```
cout << x << ' ' << y << ' ' << ch << ' ' << z << "\n";
```

Thông thường, để nhập dữ liệu cho biến ta dùng cặp câu lệnh sau:

```
cout << "Nhap tri cho bien :"; // câu dẫn hướng
```

```
cin >> var ; // nhập giá trị từ bàn phím cho biến var
```

Chú ý:

Lệnh sau đây là câu lệnh hợp lệ:

```
cin >> x, y, z; // ý nghĩa chỉ nhập trị cho biến x
```

Để nhập đầy đủ trị cho 3 biến x, y và z:

```
cin >> x >> y >> z;
```

14. Một vài chương trình C/C++ đơn giản

Chương trình rỗng chỉ có hàm main()

```
// Null.cpp
```

```
void main(){  
  
}
```

Giải thích:

Một chương trình “C/C++” là tập hợp nhiều hàm, trong đó hàm `main()` là không thể thiếu.

Dòng đầu tiên là tiêu đề hàm `main()`. `void` là từ khóa cho biết hàm `main()` không trả về giá trị nào. Có thể viết hàm `main()` có trị trả về như sau:

```
int main(){  
  
    // ... các câu lệnh ...  
  
    return 0;  
  
}
```

Phần đặt trong cặp dấu “{” và dấu “}” là thân của hàm `main()`.

Chương trình trên là rỗng, không thực hiện thao tác nào (Phần nằm giữa cặp dấu {} không chứa bất kỳ câu lệnh nào). Tuy nhiên đó là chương trình hợp lệ.

Trong môi trường Visual C++:

- Để biên dịch: ấn tổ hợp phím CTRL+F.
- Để tạo ra tập tin .exe: ấn phím F7.
- Để chạy chương trình: ấn tổ hợp phím CTRL+F5.

Chương trình in ra màn hình dòng chữ “HELLO C!”.

```
// Hello.cpp
```



```
#include <iostream.h> // de su dung cout

void main() {

    cout << "HELLO C++\n";

}
```

Giải thích:

Trong C/C++ có 2 dạng chú thích:

- Chú thích trên dòng: // . . . phần chú thích . . .
- Chú thích trên 1 hoặc nhiều dòng: /* . . . phần chú thích . . .
*/

Chú thích thường được dùng để nêu rõ chức năng, nhiệm vụ của 1 hàm, biến, hằng. Phần chú thích chỉ có ý nghĩa đối với programmer, chương trình dịch sẽ bỏ qua không biên dịch những nội dung nằm trong phần chú thích.

Dòng đầu tiên là chỉ thị tiền xử lý, yêu cầu chương trình dịch chèn nội dung của tập tin <iostream.h> tại vị trí khai báo chỉ thị tiền xử lý. Tập tin <iostream.h> được gọi là tập tin tiêu đề, và có chứa các khai báo, định nghĩa hằng, biến, hàm liên quan đến việc xuất nhập dữ liệu trong đó có đối tượng cout và toán tử <<.

Trong phần thân của hàm main() chỉ có 1 câu lệnh duy nhất, yêu cầu chương trình dịch thực hiện việc in dòng chữ HELLO C! đặt trong cặp nháy đôi.

Chương trình mô phỏng máy tính đơn giản.

```
// Calculator.cpp
```

```
#include <iostream.h>

void main() {

    int x,y;

    cout << "Nhap tri cho cac bien x, y : "; cin >> x >> y;

    cout << x << '+' << y << '=' << x+y;

    cout << x << '-' << y << '=' << x-y;

    cout << x << '*' << y << '=' << x*y;

    //chu y, neu y=0 thì Chương trình dịch sẽ báo lỗi divide by zero

    cout << x << '/' << y << '=' << x/y;

}
```

Chương trình nhập vào 3 số nguyên, sau đó in ra màn hình cho biết số min, max của 3 số này.

```
// MinMax.cpp
```

```
#include <iostream.h>

void main() {

    int a, b, c;

    cout << "Nhap 3 so a, b, c : "; cin >> a >> b >> c;
```

```
cout << "MAX = " << (a>b ? (a>c ? a : c) : (b>c ? b : c));

cout << "MIN = " << (a<b ? (a<c ? a : c) : (b<c ? b : c));

}
```

Chương trình nhập 1 kí tự chữ thường/hoa, sau đó in ra kí tự chữ hoa/thường tương ứng.

// Kitu.cpp

```
#include <iostream.h>

#include <assert.h> //de su dung ham thu vien assert(<bt logic>)

void main() {

    int kt;

    cout << "Nhap 1 ki tu chu(thuong/hoa) : "; cin >> kt;

    assert(kt>='A' && kt<='Z' || kt>='a' && kt<='z');

    cout << "Ki tu tương ung la ";

    cout << char((kt>='a' && kt<='z')?(kt-'a'+'A'):(kt-'A'+'a'));

}
```

Chương trình nhập một số chỉ tháng, và một số chỉ năm. Sau đó in ra cho biết tháng đó có bao nhiêu ngày?

// NumDays.cpp

```
#include <iostream.h>
```

```

void main() {

    int m, y;

    cout << "Nhap vao thang m = "; cin >> m;

    cout << "Nhap vao nam y = "; cin>>y;

    // Kiem tra DL nhap tai day . . .

    int m30 = (m==4 || m==6 || m==9 || m==11);

    int leapyear = (y%400==0 || y%4==0 && y%!=100);

    int numDays = (m==28 ? 28+leapyear : m30 != 0 ? 30 : 31);

    cout << "So ngay cua thang " << m << " nam " << y

        <<" la " << numDays;

}

```

Chương trình nhập vào giờ, phút, giây và tính thời điểm 1 giây sau đó.

```

// NextSec.cpp

#include <iostream.h>

```

```

void main() {

    int h, m, s;

    cout << "Nhap gio, phut, giay : "; cin >> h >> m >> s;

```

```

// Kiem tra DL nhap tai day ...

++s>59 ? (++m>59 ? ++h : 0) : 0;

s %= 60;

m %= 60;

h %= 24;

cout << "Thoi diem 1 giây sau : ";

cout << h/10<<h%10 << ':' << m/10<<m%10 << ':' << s/10<<
s%10;

}

```

Chương trình nhập vào ngày, tháng, năm và tính ngày hôm sau, ngày hôm trước?

```
// NextDays.cpp
```

```

#include <iostream.h>

void main() {

    int d, m, y;

    cout << "Nhap ngay, thang, nam : "; cin >> d >> m >> y;

    // Kiem tra DL nhap tai day ...

    int m30 = (m==4 || m==6 || m==9 || m==11);

```

```

int leapyear = (y%400==0 || y%4==0 && y%100!=0);

int numDays = (m==28 ? 28+leapyear : m30 != 0 ? 30 : 31);

++d>numDays ? (d=1,++m>12 ? m=1,++y : 0) : 0;

cout << "Hôm sau là ngày " << d<<'/'<<m<<'/'<<y<<endl;

}

```

15. Câu hỏi

1) Một chương trình “C/C++” chứa các khai báo sau:

```

int i, j;

long ix;

short s;

float x;

double dx;

char c;

```

Hãy xác định kiểu của các biểu thức sau:

```

i + c

x + c

(int) dx + ix

i + x

```

$$s + j$$

$$ix + j$$

$$s + c$$

$$ix + c$$

2) Cho 2 số nguyên M, N . Hãy mô tả các vị từ logic sau bằng các biểu thức logic (theo M, N) tương ứng trong NNLT “C/C++”:

- N là số âm.
- N là số dương.
- N là số chẵn.
- N là số lẻ.
- N là số âm chẵn.
- N là số âm lẻ.
- N là số dương chẵn.
- N là số dương lẻ.
- N là bội số của M .
- N là ước số của M .
- N là số tự nhiên gồm 3 chữ số thỏa tính chất $P = \text{”Tổng của các chữ số bằng Tích của các chữ số”}$.

3) Cho a, b, c là 3 số tự nhiên đại diện cho 3 cạnh của tam giác T . Hãy mô tả các mệnh đề dưới đây bằng biểu thức logic (theo a, b, c) tương ứng trong NNLT “C/C++”:

- T là tam giác thường.
- T là tam giác cân.

- T là tam giác vuông.
 - T là tam giác vuông cân.
 - T là tam giác đều.
- 4) Gọi N là số tự nhiên chỉ năm. Hãy mô tả mệnh đề định nghĩa năm nhuận bằng biểu thức logic (theo N) tương ứng trong NNLT “C/C++” như sau: “Năm nhuận là năm chia chắn cho 400 hoặc nếu nó chia chắn cho 4 thì đồng thời không được chia chắn cho 100”.
- 5) Cho M là ma trận vuông cấp $n \times n$, $n > 0$. Gọi i, j là các chỉ số hàng và cột của ma trận. Hãy mô tả các vị từ logic dưới đây bằng các biểu thức logic (theo các chỉ số i, j) tương ứng trong NNLT “C/C++”:
- Các phần tử nằm trên hàng đầu tiên.
 - Các phần tử nằm trên hàng cuối cùng.
 - Các phần tử nằm trên cột đầu tiên.
 - Các phần tử nằm trên cột cuối cùng.
 - Các phần tử nằm trên đường chéo chính.
 - Các phần tử nằm trên đường chéo phụ.
 - Các phần tử nằm cao hơn đường chéo chính.
 - Các phần tử nằm cao hơn đường chéo phụ.
 - Các phần tử nằm phía dưới đường chéo chính.
 - Các phần tử nằm phía dưới đường chéo phụ.
 - Các phần tử nằm trên một đường song song với đường chéo chính.
 - Các phần tử nằm trên một đường song song với đường chéo phụ.

16. Bài tập về phép toán và biểu thức

Mục tiêu

- Làm quen với môi trường Visual Studio 6.0
- Sử dụng phép toán và biểu thức để giải một số bài toán đơn giản.

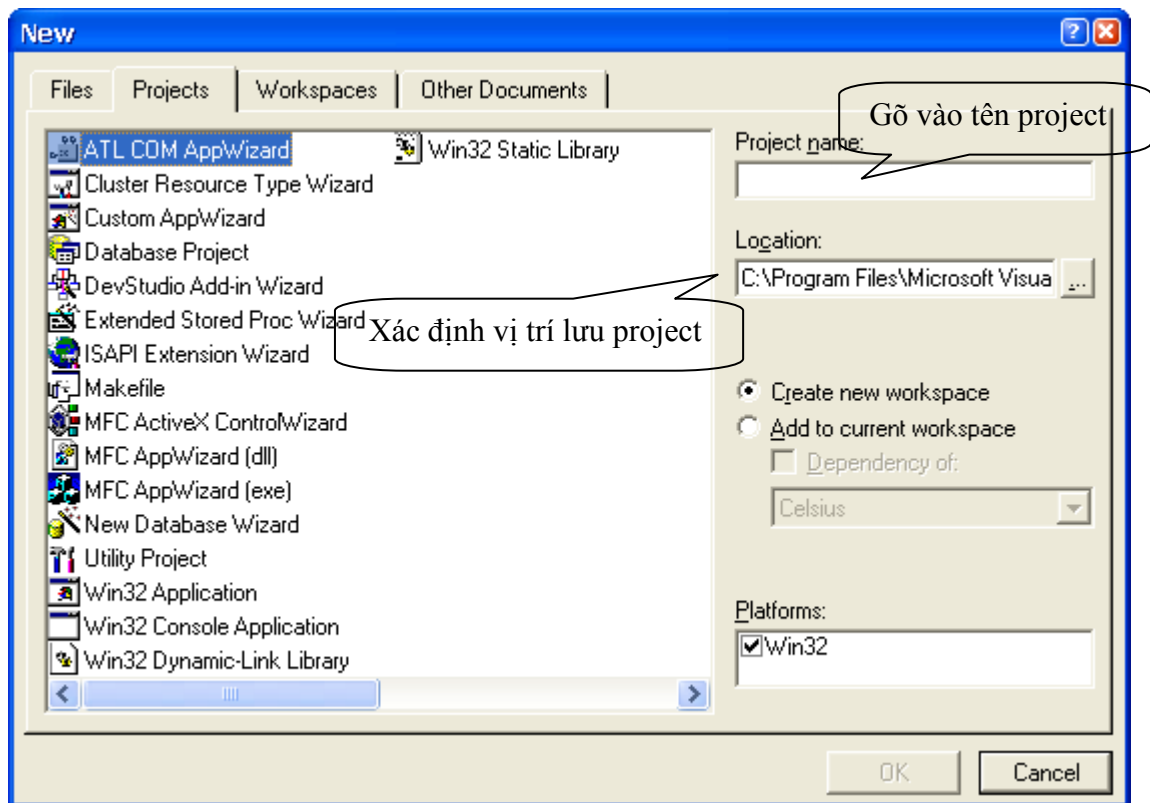
Yêu cầu

- Biết cách tạo mới chương trình, mở chương trình đã có trên đĩa, lưu trữ chương trình, ...
- Hiểu được cấu trúc của một chương trình C/C++ đơn giản.
- Viết được một số chương trình có sử dụng phép toán và biểu thức để giải một số bài toán đơn giản.

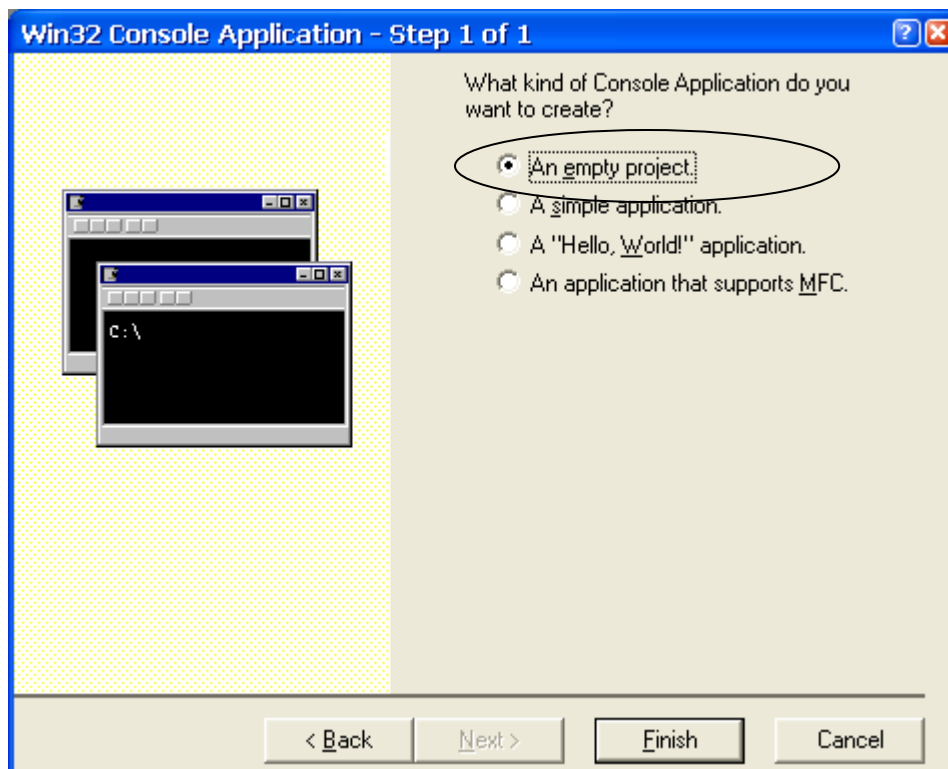
Tạo mới chương trình C/C++:

Cách 1:

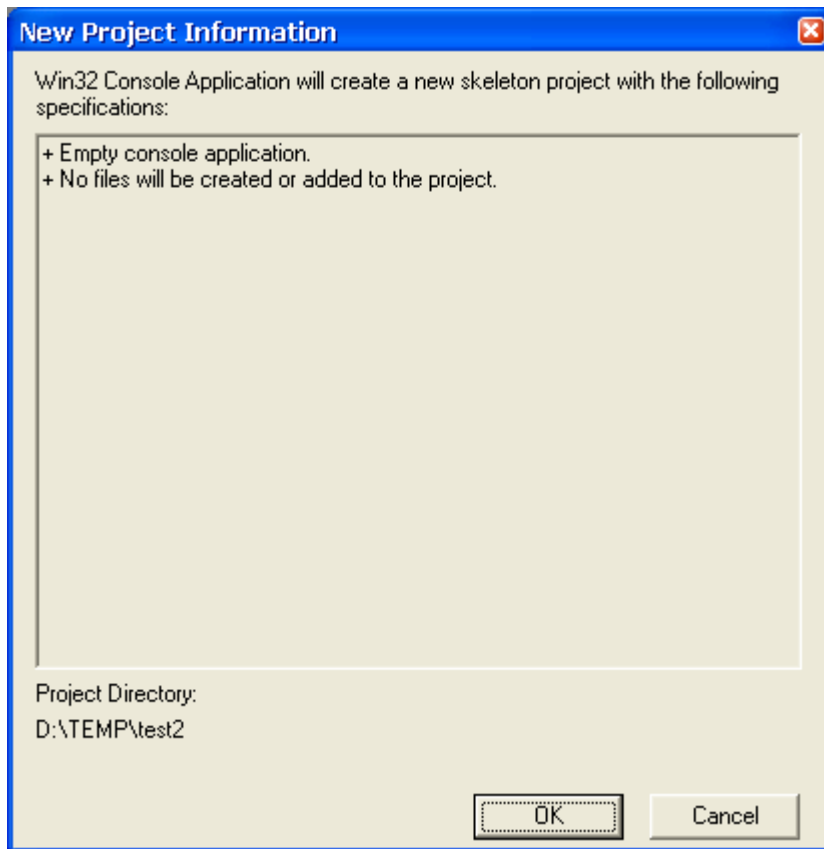
- Chọn menu File→New: (có thể bỏ bước này)
- Xuất hiện hộp thoại New.



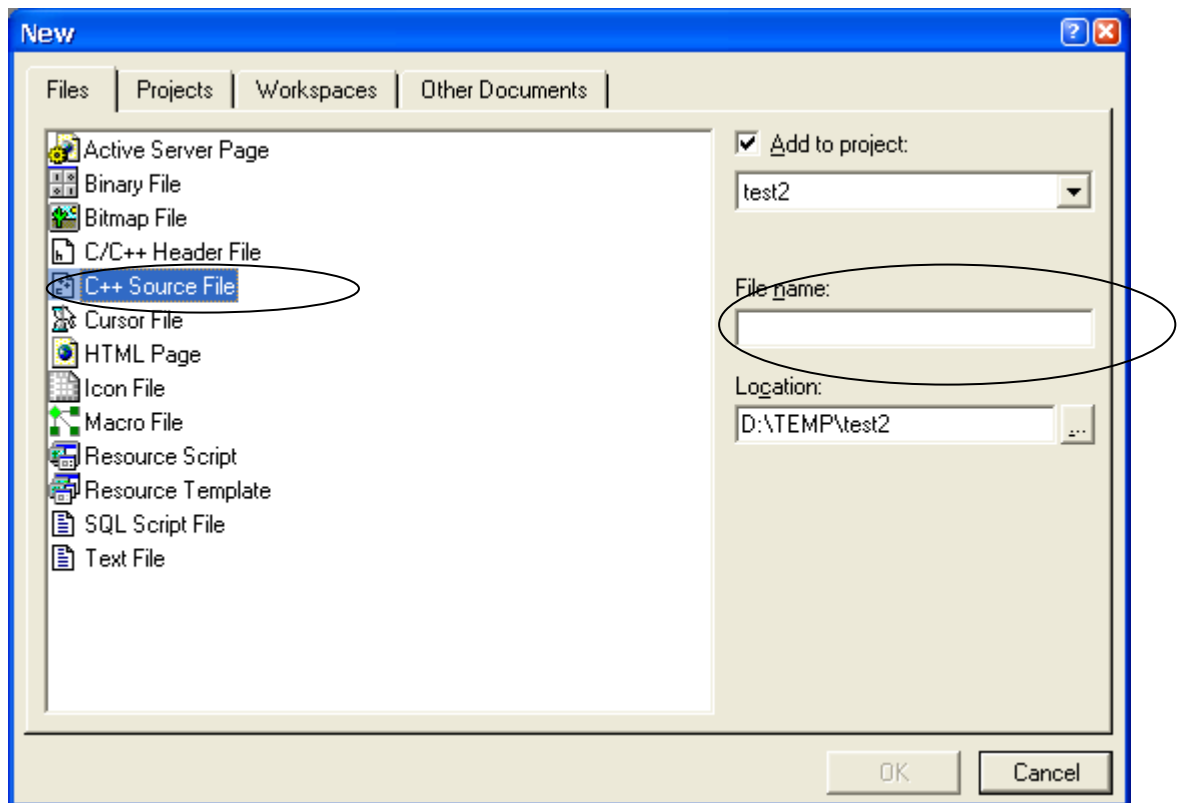
- Chọn mục Win32 Console Application, sau đó cung cấp tên và thư mục nơi sẽ lưu project và click vào nút OK.



- Chọn mục An empty project, sau đó ấn nút Finish.



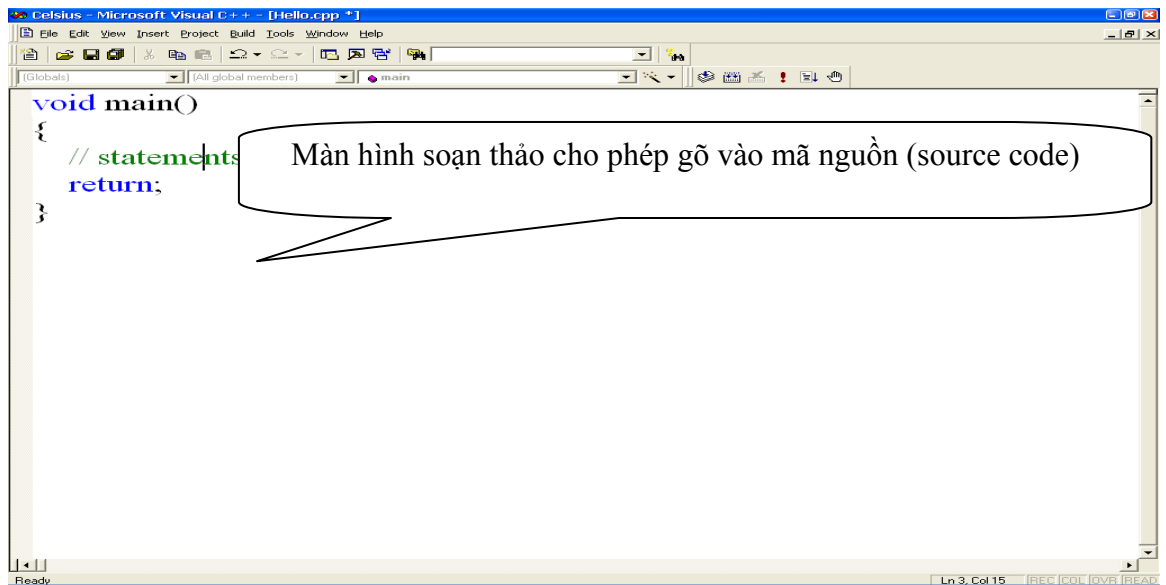
- *Tạo file mới* có tên dạng *.cpp, chọn menu File→New



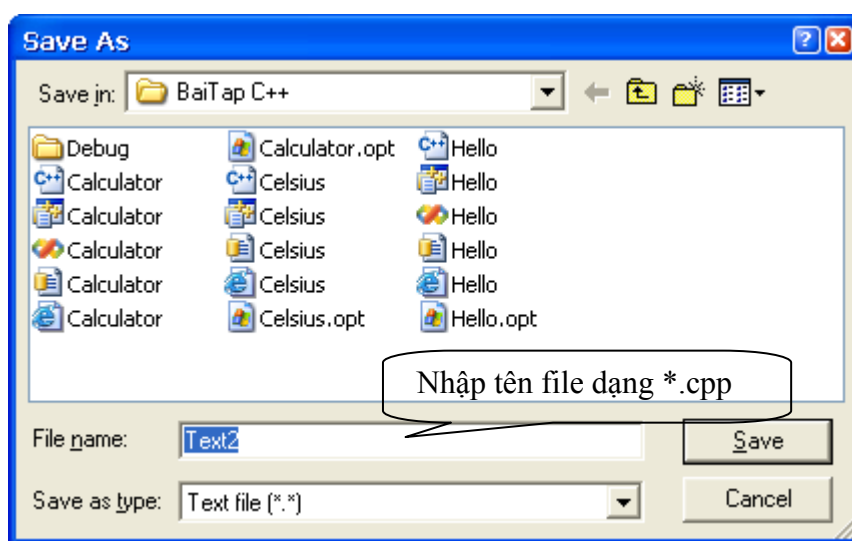
- Chọn mục C++ Source File, và đặt tên file trong hộp văn bản File name (Không cần gõ phần mở rộng .cpp), sau đó chọn OK.

Cách 2:

- Click vào biểu tượng “New Text File”



- Chọn menu File→Save, hoặc ấn tổ hợp phím CTRL+S, hoặc Click vào biểu tượng “Save”.



Lưu tập tin, nhớ chú ý tên phần mở rộng phải là “.cpp”

Gõ vào chương trình đầu tiên sau đây:

```
void main()

{

// statements
```

```
return; // có thể bỏ qua  
  
}
```

Hoặc

```
int main()  
  
{  
  
    // statements  
  
return 0;  
  
}
```

Để biên dịch
(BD):

(Cần thiết phải
thực hiện lần đầu tiên)

Để biên dịch và
liên kết:

(Có thể bỏ qua)

- Chọn menu Build→Compile (tạo file __.obj)
- Ấn tổ hợp phím CTRL+F7
- Click biểu tượng trên thanh Toolbar
- Chọn menu Build→Build (tạo file __.obj và file __.exe)
- Chọn menu Build→Rebuild All
- Ấn phím F7
- Click biểu tượng trên thanh Toolbar

Để biên dịch, liên kết và chạy chương trình:

(Chỉ có thể thực hiện sau khi đã BD)

Chọn menu Build→Execute

Ấn tổ hợp phím CTRL+F5

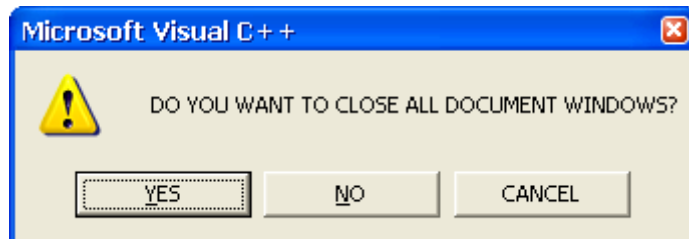
Click biểu tượng trên thanh Toolbar

Để lưu trữ file chương trình:

Dùng tổ hợp phím CTRL+S, hoặc dùng menu File->Save, hoặc dùng biểu tượng save trên thanh toolbar.

Để tạo mới chương trình trước tiên cần phải đóng chương trình đang chạy (nếu có) như sau:

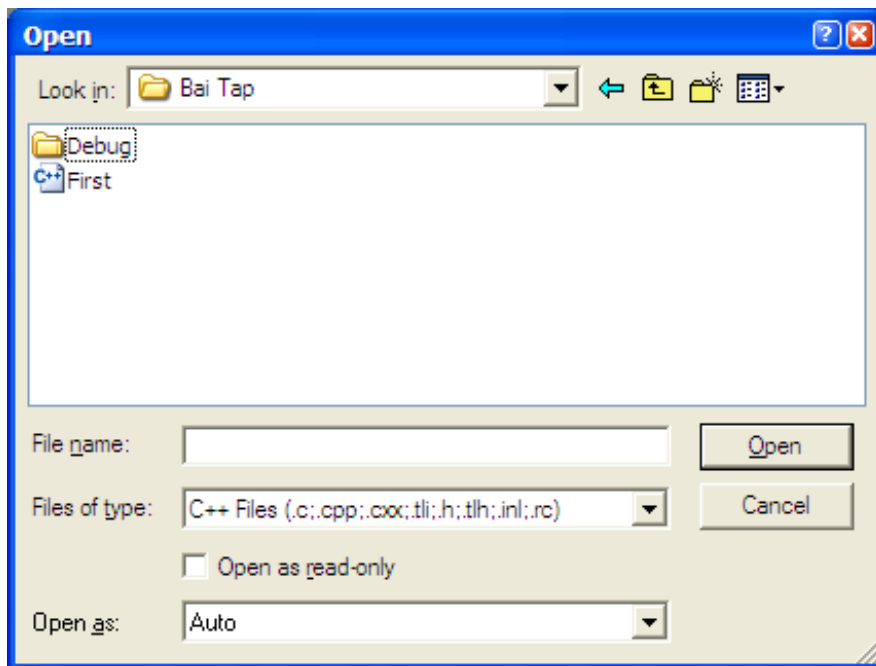
- Chọn menu File→Close Workspace



- Chọn Yes để đóng tất cả các cửa sổ tài liệu lại.

Để mở file đã tạo trước đó:

- Chọn biểu tượng Open, hoặc Ấn tổ hợp phím CTRL+O, hoặc chọn menu File→Open
- Chọn file muốn mở, sau đó chọn nút Open



Gõ vào chương trình thứ hai sau đây:

```
#include <iostream.h>

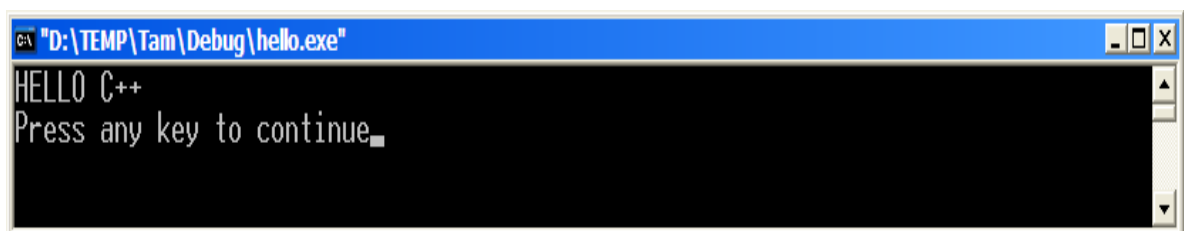
void main()

{

    cout << "HELLO C++\n";

}
```

Kết quả sau khi biên dịch (CTRL+F7) và cho thực hiện chương trình (CTRL+F5):



Câu hỏi: Có thể sử dụng chương trình trên để viết thêm được các chương trình gì?

Gõ vào chương trình sau:

```
#include <iostream.h>

void main()

{

    cout << "1 + 2 = " << 1 + 2 << "\n";

    cout << "1.0 / 2 = " << 1.0 / 2 << "\n";

    cout << "1 / 2.0 = " << 1 / 2.0 << "\n";

    cout << "1.0 / 2.0 = " << 1.0 / 2.0 << "\n";

    cout << "11 % 8 = " << 11 % 8 << "\n";

    cout << "-11 % 8 = " << -11 % 8 << "\n";

    cout << "11 % -8 = " << 11 % -8 << "\n";

    cout << "-11 % -8 = " << -11 % -8 << "\n";

    cout << "(1 + 2 > 1 * 2) = " << (1 + 2 > 1 * 2) << "\n";

    cout << "(1 + 2 <= 1 * 2) = " << (1 + 2 <= 1 * 2) << "\n";

    cout << "('B' > 'a' && 1 + 2 > 1 * 2) = " << ('B' > 'a' && 1 +
2 > 1 * 2) << "\n";
```

```

    cout << “(‘B’ > ‘a’ || 1 + 2 > 1 * 2) = ” << (‘B’ > ‘a’ || 1 + 2 > 1
* 2) << “\n”;

    cout << “sizeof(int) = ” << sizeof(int) << “\n”;

    cout << “sizeof(double) = ” << sizeof(double) << “\n”;

    cout << “sizeof(long int) = ” << sizeof(long int) << “\n”;

    cout << “sizeof(1.0 / 2) = ” << sizeof(int) << “\n”;

    cout << “(4 & 5) = “ << (4 & 5) << “\n”;

    cout << “(4 | 5) = “ << (4 | 5) << “\n”;

    cout << “(4 ^ 5) = “ << (4 ^ 5) << “\n”;

    cout << “(2 << 3) = “ << (2 << 3) << “\n”;

    cout << “(8 >> 3) = “ << (8 >> 3) << “\n”;

}

```

Kết quả xuất ra màn hình:

```

c:\ "D:\TEMP\Tam\Debug\hello.exe"
1 + 2 = 3
1.0 / 2 = 0.5
1 / 2.0 = 0.5
1.0 / 2.0 = 0.5
11 % 8 = 3
-11 % 8 = -3
11 % -8 = 3
-11 % -8 = -3
(1 + 2 > 1 * 2) = 1
(1 + 2 <= 1 * 2) = 0
('B' > 'a' && 1 + 2 > 1 * 2) = 0
('B' > 'a' || 1 + 2 > 1 * 2) = 1
sizeof(int) = 4
sizeof(double) = 8
sizeof(long int) = 4
sizeof(1.0 / 2) = 4
(4 & 5) = 4
(4 | 5) = 5
(4 ^ 5) = 1
(2 << 3) = 16
(8 >> 3) = 1
Press any key to continue_

```

Câu hỏi: mục tiêu của chương trình trên là gì ?

Gõ vào chương trình sau:

// chương trình đổi độ Fahrenheit sang độ Celsius

```

#include <iostream.h>

void main()

{

    // B1. Nhập dữ liệu

    double f, c;

    cout << "Nhập độ Fahrenheit f = "; cin >> f;

    // B2. Xử lý

    c = 5.0 / 9 * (f-32);

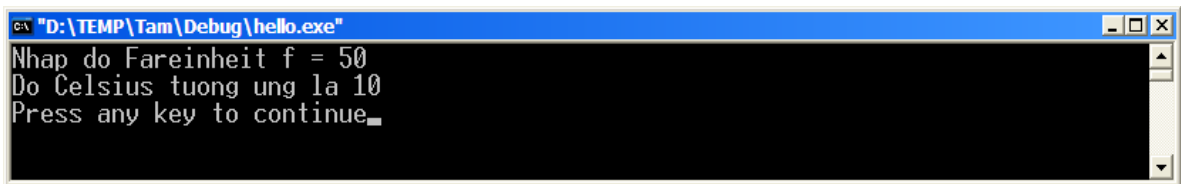
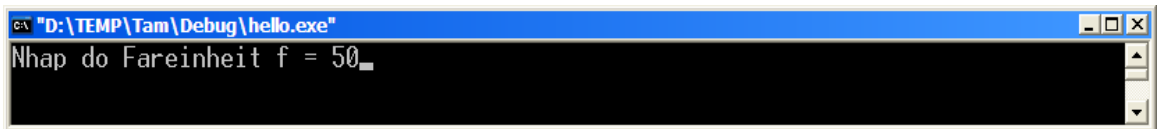
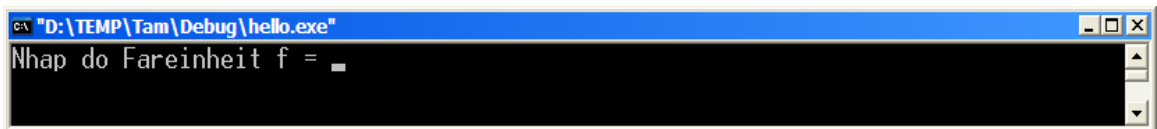
```

```
// B3. Xuất kết quả c

cout << "Độ Celsius tương ứng là " << c << "\n";

}
```

Kết quả sau khi nhập trị 50 cho biến f:



Câu hỏi: Có thể phát triển chương trình trên để viết thêm được các chương trình gì ?

Sinh viên tự viết các chương trình sau:

- 6) Viết chương trình làm calculator đơn giản (+, -, *, /, %)
- 7) Viết chương trình tính chu vi, diện tích hình tròn.
- 8) Viết chương trình tính chu vi, diện tích hình tam giác theo 3 cạnh.
- 9) Viết chương trình in trị đảo ngược của số nguyên gồm 3 chữ số (chữ số hàng đơn vị khác 0). Ví dụ, nếu nhập vào 483 thì in ra 384.
- 10) Viết chương trình hoán đổi trị của 2 số nguyên.

- 11) Viết chương trình nhập vào 3 số nguyên. In ra cho biết số lớn thứ nhất, số lớn thứ hai, và số nhỏ nhất.
- 12) Viết chương trình nhập vào 1 kí tự. In ra cho biết:
 - Mã ASCII của kí tự đó
 - Kí tự đứng sau và mã ASCII của kí tự đó.
 - Kí tự đứng trước và mã ASCII của kí tự đó.
- 13) Viết chương trình nhập vào 1 kí tự. Sau đó in ra kí tự hoa/thường tương ứng ngược lại.
- 14) Viết chương trình thử nghiệm toán tử ++, --.
- 15) Viết chương trình nhập vào 1 thời điểm (giờ, phút, giây). In ra cho biết thời điểm 1 giây sau ? / 1 giây trước ?
- 16) Viết chương trình nhập năm y. In ra cho biết năm y có nhuận hay không ?
- 17) Viết chương trình nhập tháng m, năm y. In ra cho biết tháng m, năm y có tối đa bao nhiêu ngày ?
- 18) Viết chương trình nhập ngày d, tháng m, năm y. In ra cho biết ngày vừa nhập có hợp lệ hay không ?
- 19) Viết chương trình nhập vào 1 bộ ngày tháng năm. In ra cho biết ngày hôm sau là ngày mấy?
- 20) Viết chương trình nhập vào 1 bộ ngày tháng năm. In ra cho biết ngày hôm trước là ngày mấy?

CHƯƠNG 3. CÁC CẤU TRÚC ĐIỀU KHIỂN (CONTROL STRUCTURES)

Hầu hết các thuật toán đều có thể được cài đặt bằng 1 trong 3 cấu trúc cơ bản sau:

Cấu trúc tuần tự: Các câu lệnh được thực hiện lần lượt theo thứ tự tuyến tính từ trên xuống và từ trái sang phải.

Cấu trúc rẽ nhánh hoặc lựa chọn: một trong những đặc điểm cơ bản của một chương trình máy tính là tính “dự đoán và đáp ứng”. Chương trình phải dự đoán tất cả các trường hợp có thể có của bài toán và phải đáp ứng đầy đủ cho từng trường hợp này. Nếu không thuật toán sẽ cho kết quả không đúng.

Cấu trúc lặp cho phép thể hiện một cách ngắn gọn một số câu lệnh lặp đi lặp lại về bản chất. Khi sử dụng cấu trúc lặp, cần phải bảo đảm tính dừng, tính kết thúc của vòng lặp, tránh cho vòng lặp bị quẩn, bị lặp vô tận (LOOP).

1. Cấu trúc rẽ nhánh (if ... then ... else ...)

Cú pháp (syntax)

```
if (<ĐK>
    <lệnh1> ;
[else
    <lệnh2>;]
```

Trong đó

- <ĐK> thường là biểu thức logic. Tổng quát, <ĐK> có thể là 1 biểu thức nguyên tính (integral type)
- Phần else là không bắt buộc phải có. Câu lệnh if không có phần else được gọi là câu lệnh “if thiếu”.
- <lệnh1>, <lệnh2> là câu lệnh hợp lệ bất kỳ: câu lệnh đơn, hoặc câu lệnh ghép, hoặc câu lệnh điều khiển.

Ý nghĩa

Nếu <ĐK> thỏa thì thực hiện <lệnh1> ở phần if, còn ngược lại thì thực hiện <lệnh2> ở phần else.

Như vậy chỉ <lệnh1> hoặc <lệnh2> được thực hiện mà thôi.

Sự thực hiện câu lệnh (Xem lưu đồ cú pháp)

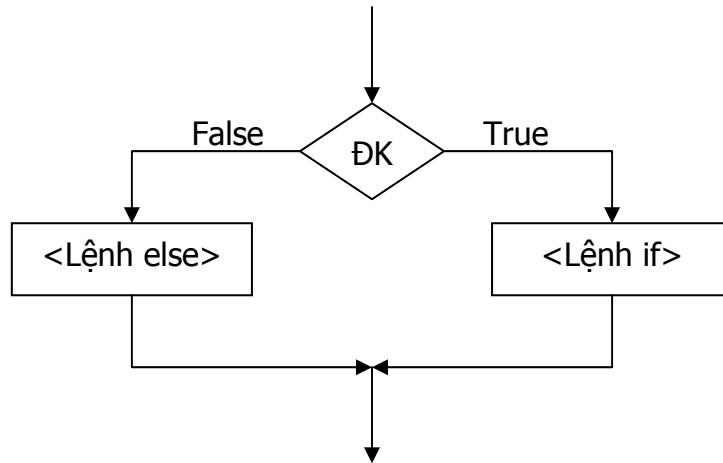
B1: <ĐK> được tính trị.

B2: Nếu <ĐK> có trị $\neq 0$ (tương ứng với mệnh đề TRUE) thì <lệnh1> trong phần if được thực hiện.

Còn ngược lại, nếu <ĐK> có trị = 0 (tương ứng với mệnh đề FALSE) thì <lệnh2> trong phần else được thực hiện (nếu có).

B3: Chuyển quyền điều khiển sang câu lệnh kế tiếp sau lệnh if.

Lưu đồ cú pháp câu lệnh if ... else ...



Chú ý

- Coi chừng quên cặp dấu (<ĐK>) sau từ khóa if.

```
if ĐK
```

```
  S1;
```

- Coi chừng dư dấu ‘;’

```
if (ĐK) ;
```

```
  S1;
```

- Coi chừng thiếu cặp dấu bao quanh {<khối lệnh>}

```
if (ĐK)
```

```
  S1;
```

```
  S2;
```

```
  ...
```


Một số dạng cấu trúc if ... else ... lồng nhau

Dạng 1

```
if (ĐK1)
    S1;
else
    if (ĐK2)
        S2;
    else
        S3;
```

Dạng 2

```
if (ĐK1)
    if (ĐK2)
        S1;
    else
        S2;
else
    S3;
```

Dạng 3

```
if (ĐK1)

    if (ĐK2)

        S1;

    else

        S2;
```

Dạng 4

```
if (ĐK1)

{

    if (ĐK2) S1;

}

else

    S2;
```

2. Cấu trúc lựa chọn (switch ...)

Cú pháp (syntax)

```
switch(<BT>)

{

    case <val1>: [<lệnh1>; break;]

    case <val2>: [<lệnh2>; break;]
```

```
....  
  
case <valn>: [<lệnh>; break;]  
  
[default: <lệnh>; break;]  
  
}
```

Ý nghĩa

Tùy theo giá trị của <BT> so trùng với <val> của nhãn case nào thì thực hiện <lệnh> tương ứng cho đến khi gặp lệnh break thì thoát khỏi câu lệnh switch(...).

Đây là cấu trúc tổng quát

Sự thực hiện câu lệnh

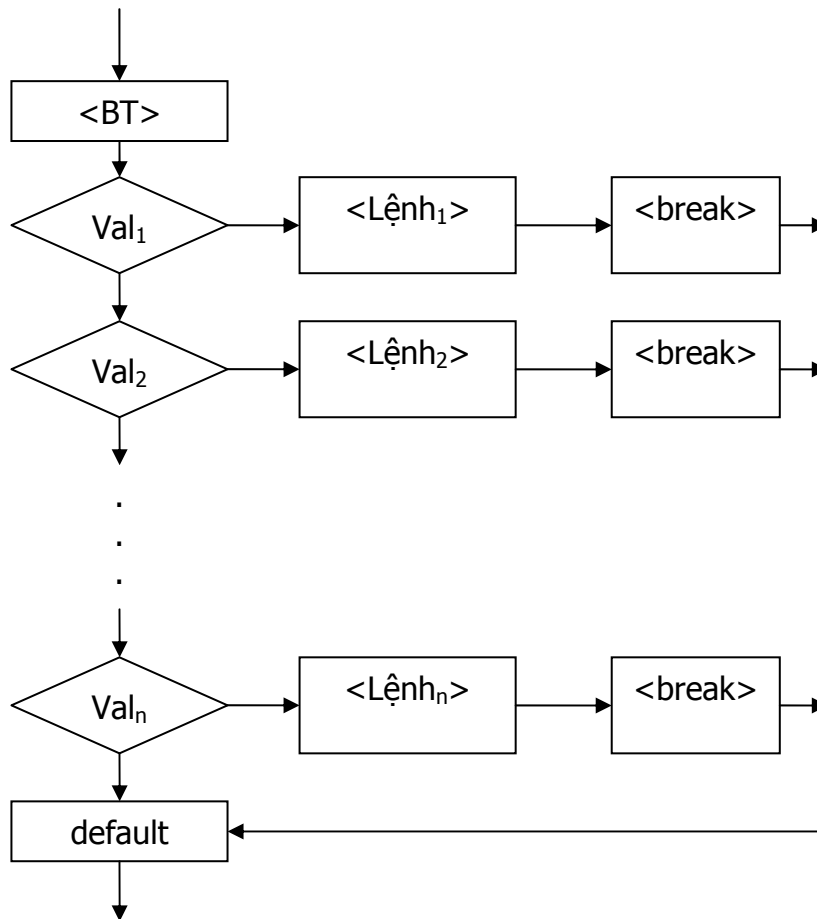
B1: <BT> được tính trị.

B2: Lần lượt so sánh trị của <BT> vừa tính được với các giá trị <val₁>, <val₂>, ... sau nhãn case. Nếu so trùng thì <lệnh_i> tương ứng được thực hiện, cho đến khi gặp lệnh break.

Nếu không có nhãn giá trị nào so trùng với trị của <BT> thì <lệnh> ở phần default được thực hiện (nếu có).

B3: Thoát khỏi lệnh switch(...), và chuyển quyền điều khiển sang câu lệnh kế tiếp.

Lưu đồ cú pháp câu lệnh switch ...



Chú ý

- Coi chừng quên lệnh break;
- Nếu muốn thực hiện <lệnh> trong một số trường hợp giá trị <val_i> sau nhãn case, ta đặt các <val_i> đó gần nhau, và ghi lệnh tại <val_i> cuối cùng của nhóm.
- Cách viết câu lệnh switch:

```
switch()
{
}
```

3. Cấu trúc lặp while (...)

Cú pháp (syntax)

```
while (<ĐK>)
```

```
<lệnh>;
```

trong đó:

- <ĐK> thường là biểu thức logic.
- <lệnh> có thể là câu lệnh đơn, khối lệnh, hoặc câu lệnh điều khiển.

Ý nghĩa

Chừng nào mà điều kiện còn thỏa thì thực hiện <lệnh>.

Sự thực hiện câu lệnh (Xem lưu đồ cú pháp)

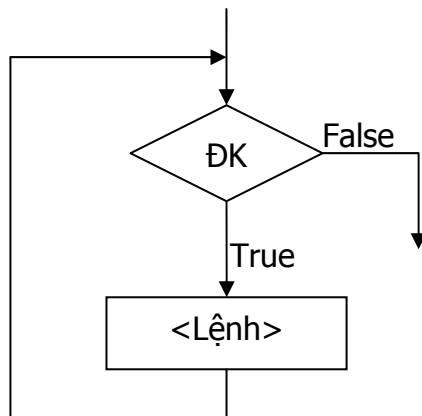
B1: Tính trị của <ĐK>.

B2: Nếu trị của <ĐK> != 0 (kể cả số âm – tương đương với mệnh đề logic TRUE), thực hiện <lệnh>, sau đó quay về B1.

Còn ngược lại (<ĐK> = 0) thì chuyển sang B3.

B3: Thoát khỏi vòng lặp, và chuyển quyền điều khiển sang câu lệnh kế tiếp sau lệnh while.

Lưu đồ cú pháp câu lệnh while ...



Nhận xét

- Do <ĐK> được kiểm tra trước, nên phần <lệnh> của vòng lặp while có thể không được thực hiện lần nào.
- Để bảo đảm cho vòng lặp while(...) không bị quẩn, thì phải có ít nhất một câu lệnh trong phần thân vòng lặp có tác dụng làm thay đổi việc đánh giá lại <ĐK> thoát sau mỗi lần lặp.

Chú ý

- Coi chừng quên cặp dấu (<ĐK>) sau từ khóa while.

```
while ĐK // sai do thiếu (ĐK)
```

```
{
```

```
// ...
```

```
}
```

- Coi chừng dư dấu ‘;’ sau cặp dấu (...)

```
while (ĐK) ;
```

```
{
```

```
    // ...
```

```
}
```

- Coi chừng thiếu cặp dấu bao quanh khối lệnh {<khối lệnh>}

```
while (ĐK)
```

```
    S1;
```

```
    S2;
```

```
    ...
```

4. Cấu trúc lặp do ... while (...)

Cú pháp (syntax)

```
do{
```

```
    <lệnh>;
```

```
}while (<ĐK>;
```

trong đó:

- <ĐK> thường là biểu thức logic.
- <lệnh> có thể là câu lệnh đơn, khối lệnh, hoặc câu lệnh điều khiển.

Ý nghĩa

Thực hiện <lệnh> cho đến khi <ĐK> không còn được thỏa.

Sự thực hiện câu lệnh:

B1: Thực hiện <lệnh>.

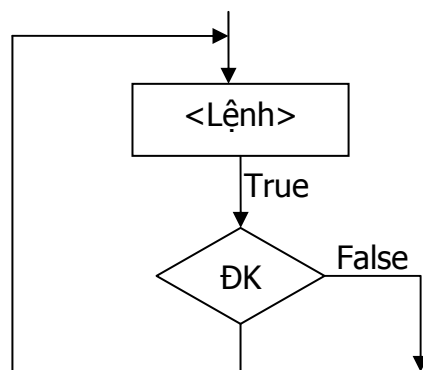
B2: Tính trị của <ĐK>.

B3: Nếu trị của <ĐK> $\neq 0$ (kể cả số âm – tương đương với mệnh đề logic TRUE), thì quay về B1.

Còn ngược lại (<ĐK> có trị = 0) thì chuyển sang B4.

B4: Thoát khỏi vòng lặp, và chuyển quyền điều khiển sang câu lệnh kế tiếp sau lệnh while.

Lưu đồ cú pháp câu lệnh do ... while ...



Nhận xét

- Khác với vòng lặp while(...), phần <lệnh> trong do ... while(...) luôn được thực hiện ít nhất là 1 lần, do <ĐK> được kiểm tra sau.

- Trong trường hợp vòng lặp do . . . while (...) bị quẩn, ta nên xem xét và hiệu chỉnh lại các câu lệnh trong phần thân vòng lặp do ... while(...) có liên quan đến <ĐK> thoát của vòng lặp.

Chú ý

- Coi chừng quên cặp dấu ngoặc tròn bao quanh <ĐK> sau từ khóa while.

```
do {
    // ...
} while ĐK // sai do thiếu (ĐK)
```

- Coi chừng thiếu dấu ‘;’ sau cặp dấu (...)

```
do {
    // ...
} while (ĐK) // sai do thiếu ;
```

- Coi chừng thiếu cặp dấu bao quanh khối lệnh {<khối lệnh>}

```
do
    S1;
    S2;
    ...
```

while (ĐK);

5. Cấu trúc lặp for (...)

Cú pháp (syntax)

```
for ([<phần khởi tạo>] ; [<ĐK>] ; [<phần hiệu chỉnh>])
```

```
<lệnh>;
```

Trong đó:

- <phần khởi tạo> là một hay nhiều biểu thức gán (được phân cách bởi dấu ‘,’) có nhiệm vụ khởi tạo giá trị ban đầu cho các biến đếm.
- <ĐK> thường là biểu thức logic.
- <phần hiệu chỉnh> là một hay nhiều biểu thức gán (được phân cách bởi dấu ‘,’) có nhiệm vụ thay đổi trị của các biến ở <phần khởi tạo>.
- <lệnh> có thể là câu lệnh đơn, khối lệnh, hoặc câu lệnh điều khiển.

Ý nghĩa

Tương tự như cấu trúc while, và thường được dùng cho câu lệnh với số lần lặp xác định.

Sự thực hiện câu lệnh

B1: Thực hiện <phần khởi tạo> (nếu có).

B2: Tính trị của <ĐK>.

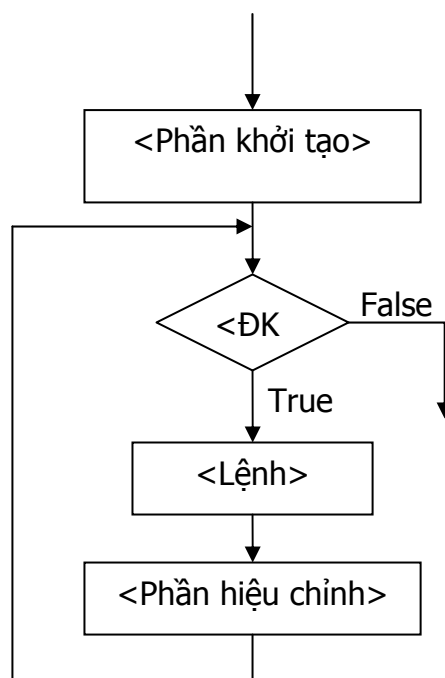
B3: Nếu trị của <ĐK>≠0 (kể cả số âm – tương đương với mệnh đề logic TRUE), thì thực hiện <lệnh>, sau đó thực hiện <phần khởi tạo> (nếu có) và quay về B2.

Còn ngược lại (<ĐK> có trị = 0) thì chuyển sang B4.

B4: Thoát khỏi vòng lặp, và chuyển quyền điều khiển sang câu lệnh kế tiếp sau lệnh for.

Lưu đồ cú pháp câu lệnh for ...

Nhận



xét

- <phần khởi tạo> chỉ được thực hiện duy nhất 1 lần.

- Trong trường hợp vòng lặp for bị quẩn, ta nên xem xét và hiệu chỉnh lại các câu lệnh trong phần thân vòng lặp for có liên quan đến <ĐK> thoát của vòng lặp.
- Có thể chuyển đổi qua lại giữa 2 cấu trúc lặp while (...) và for (...).

Các dạng khác của câu lệnh for (...)

Chú ý

- Các phần của vòng lặp for được phân cách bằng dấu chấm phẩy ‘;’.

```
for (<KT>,<ĐK>,<HC>) // sai dùng dấu , để phân cách các phần
{
    // ...
}
```

- Coi chừng dư dấu ‘;’ sau cặp dấu (...)

```
for (<khởi tạo>; <ĐK>; <hiệu chỉnh>); // dư dấu ;
{
    // ...
}
```

- Coi chừng thiếu cặp dấu bao quanh khối lệnh {<khối lệnh>}

```
for (<khởi tạo>; <ĐK>; <hiệu chỉnh>)
```

```
S1;
```

```
S2;
```

```
...
```

6. Câu lệnh break

- Dùng để thoát khỏi câu lệnh switch().
- Dùng để thoát khỏi vòng lặp trong cùng chứa nó. Chương trình dịch sẽ bỏ qua các câu lệnh sau lệnh break;

Ví dụ 1:

```
while (1)
{
    ...
    if (ĐK)
        break; // nhảy khỏi vòng lặp while(1)
    ...
}
```

Ví dụ 2:

```
while (1)
{
    ...
}
```

```
for (...; ...; ...)  
  
{  
  
    if (ĐK)  
  
        break; // chỉ thoát khỏi vòng lặp for (...; ...; ...) chứa nó  
  
    ...  
  
}  
  
...  
  
}
```

7. Câu lệnh continue (ít dùng)

- Khi gặp câu lệnh continue trong 1 vòng lặp, chương trình dịch bỏ qua các lệnh còn lại trong thân vòng lặp này để bắt đầu một lần lặp mới.
- Thông thường, ta có thể thay thế thuật toán có sử dụng lệnh break, hoặc lệnh continue bằng một thuật toán sử dụng cấu trúc điều khiển khác.

8. Câu lệnh goto

- Khi gặp câu lệnh goto case, với case là nhãn (có dạng *identifier*;) được đặt tại vị trí nào đó, chương trình dịch chuyển quyền điều khiển đến câu lệnh đặt tại vị trí nhãn và thực hiện lệnh này.
- Câu lệnh goto làm cho chương trình khó theo dõi, khó kiểm soát, làm mất tính cấu trúc của chương trình.

- Câu lệnh goto ít được dùng và có thể được thay thế bằng các câu lệnh sử dụng các cấu trúc điều khiển khác.

9. Câu hỏi

- Trình bày cú pháp, ý nghĩa, lưu đồ và cho ví dụ về các cấu trúc:
 - if ... else ...
 - switch (...)
 - while (...)
 - do ... while (...)
 - for (... ; ... ; ...)
- Câu nào sau đây là phần đầu của câu lệnh if hợp lệ? Đối với câu lệnh không hợp lệ, hãy cho biết tại sao?
 - if (delta > 0) ...
 - if delta < 0 ...
 - if (delta == 0) ...
 - if (delta = = 0) ...
 - if (delta =< 0) ...
 - if (delta <= 0) ...
 - if (delta => 0) ...
 - if (delta >= 0) ...
 - if (delta != 0) ...
 - if (delta ! = 0) ...
 - if (1 + 2) ...
 - if (2 - 2) ...

- Hãy rút gọn đoạn chương trình sau:

```
if ( a > b )  
  
{  
  
    b = x;  
  
    a = 1;  
  
}  
  
else  
  
{  
  
    b = x;  
  
    a = 0;  
  
}
```

- Hãy rút gọn đoạn chương trình sau:

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



```
{  
  
    a += b;  
  
    c = x;  
  
}
```

- Hãy rút gọn đoạn chương trình sau:

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

- Hãy rút gọn đoạn chương trình sau:

```
if ( a > b )  
  
{  
  
    a = 1;  
  
}
```

```
    c = x;

}

else

{

    b = 2;

    c = x;

}
```

- Hãy chuyển đổi câu lệnh for thành câu lệnh while:

```
for ( int i = 0; i < 100; i++ )

cout << setw( 5 ) << i ;
```

- Hãy chuyển đổi câu lệnh for thành câu lệnh while:

```
for ( int i = 2000; i > 0; i /= 2 )

cout << setw( 5 ) << i ;
```

- Hãy chuyển đổi câu lệnh for thành câu lệnh while:

```
for ( int i = 0; i < 100; i++ )

{

    for ( int j = 0; j < 200; j++ )
```

```
    cout << setw( 5 ) << i * j;  
  
    cout << endl ;  
  
}
```

- Hãy chuyển đổi câu lệnh while thành câu lệnh for:

```
int count = 0;  
  
while ( count < 100 )  
{  
  
    cout << count;  
  
    count++;  
  
}
```

- Hãy chuyển đổi câu lệnh while thành câu lệnh for:

```
int count = 0;  
  
while ( count < 100 )  
{  
  
    count++;  
  
    cout << count;  
  
}
```

- Hãy chuyển đổi câu lệnh while thành câu lệnh for:

```
int x = 10, i = x, y = 0, j = 0;

while ( i > j )

{

    y += i;  i--;

}
```

- Hãy chuyển đổi câu lệnh while thành câu lệnh do . . . while:

```
int x = 10, i = x, y = 0, j = 0;

while ( i > j )

{

    y += i;  i--;

}
```

- Xét các đoạn lệnh sau. Hãy cho biết kết quả xuất ra màn hình của các biến x, y. Giải thích.

```
...

int x = 0, y = 1;

for ( int i = 0; i < 5; i++ )

{

    for ( int j = 0; j < 5; j++ )
```

```
if (( i + j ) % 2 == 0 )  
  
    x++;  
  
y += x;  
  
}  
  
...
```

- Như bài tập trên

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

- Như bài tập trên

```
...
```

```

int x = 0, y = 1;

for ( int i = 0; i < 5; i++ ) {

    for ( int j = 0; j < 5; j++ )

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

        y += x;

}

...

```

- Như bài tập trên

```

...

int x = 0, y = 1;

for ( int i = 0; i < 5; i++ )

    for ( int j = 0; j < 5; j++ )

        {

            if ( ( i + j ) % 2 == 0 )

                x++;

            else

                y += x;

        }

```

```
}
```

```
...
```

- Như bài tập trên

```
...
```

```
int i = 0, x = 0;
```

```
do{
```

```
    if ( x %5 == 0 )
```

```
    {
```

```
        x++;
```

```
        cout << setw( 4 ) << x;
```

```
    }
```

```
    i++;
```

```
} while ( i < 20 );
```

```
...
```

- Như bài tập trên

```
...
```

```
int i = 0, x = 0;
```

```
while ( i < 20 )
```

```
{  
  
    if ( x % 5 == 0 )  
  
        {  
  
            x += i;  
  
            cout << setw( 4 ) << x;  
  
        }  
  
    i++;  
  
}  
  
...
```

- Như bài tập trên

```
...  
  
int i, x = 0;  
  
for ( i = 1; i < 10; i*=2 )  
  
{  
  
    x++;  
  
    cout << setw( 4 ) << x;  
  
}  
  
...
```


- Như bài tập trên

```
...  
  
int i, x = 0;  
  
for (i = 1; i < 10; i++)  
  
{  
  
    if (i % 2 == 1 )  
  
        x += i;  
  
    else  
  
        x— ;  
  
    cout << setw( 4 ) << x;  
  
}  
  
...
```

- Như bài tập trên

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

```

k = (i + j - 1);

if ( k % 2 == 0 )

    x += k;

else

    if ( k % 3 == 0 )

        x += k - 2;

cout << setw( 4 ) << x;

}

...

```

- Như bài tập trên

```

...

int i, j, k, x = 0;

for ( i = 1; i < 5; i++ )

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

        {

            switch ( i + j - 1 )

                {

                    case -1:

```

```
case 0: x += 1; break;

case 1:

case 2:

case 3: x += 2; break;

default: x += 3;

}

cout << setw( 4 ) << x;

}

...
```

10. Bài tập

10.1. Cấu trúc rẽ nhánh

- 1) Viết chương trình nhập vào một số nguyên và cho biết đó số âm hay dương, chẵn hay lẻ.
- 2) Viết chương trình hoán đổi trị của 2 biến nguyên.
- 3) Viết chương trình giải phương trình bậc 1: $Ax + B = 0$.
- 4) Viết chương trình giải phương trình bậc 2: $Ax^2 + Bx + C = 0$.
- 5) Viết chương trình giải phương trình trùng phương $Ax^4 + Bx^2 + C = 0$.
- 6) Viết chương trình nhập 3 số nguyên dương đại diện cho 3 cạnh của 1 tam giác. Hiển thị ra màn hình cho biết đó là tam giác đều, vuông cân, cân, vuông, thường, hay không phải là 3 cạnh của tam giác.

7) Viết chương trình nhập 3 số nguyên. Sau đó hiển thị ra màn hình cho biết số lớn nhất, số lớn thứ hai, và số nhỏ nhất bằng 2 cách:

- Cách 1: Dùng cấu trúc IF . . . ELSE . . . lồng nhau
- Cách 2: Dùng toán tử điều kiện: $exp1 ? exp2 : exp3$

10.2. Cấu trúc lựa chọn

- 1) Viết chương trình tính số ngày tối đa của tháng m, năm y cho trước.
- 2) Cho trước 2 số nguyên và các kí hiệu phép toán sau: (+, -, *, /, %). Viết chương trình tính và xuất kết quả phép tính ra màn hình (có lưu ý đến lỗi chia cho giá trị 0).
- 3) Viết chương trình xếp hạng theo điểm như sau (giả sử không có điểm lẻ):
 - Điểm 1 → 4: Kém
 - Điểm 5: Trung bình
 - Điểm 6: Trung bình khá
 - Điểm 7: Khá
 - Điểm 8 → 9: Giỏi
 - Điểm 10: Xuất sắc
- 4) Viết chương trình nhập vào 2 số nguyên và một phép toán (+, -, *, /, %). In kết quả của phép toán ra màn hình (có chú ý đến lỗi chia không).
- 5) Bộ bài tây 52 lá được đánh số như sau:

	Ách Hai Ba Bốn Năm Sáu Bảy Tám Chín Mười Bồi Đâm Già
--	---

Cơ	1	2	3	4	5	6	7	8	9	10	11
	12	13									
Rô	14	15	16	17	18	19	20	21	22	23	24
	25	26									
Chuồn	27	28	29	30	31	32	33	34	35	36	37
	38	39									
Bích	40	41	42	43	44	45	46	47	48	49	50
	51	52									

Viết chương trình nhập một số có giá trị trong đoạn [1..52], sau đó in ra cho biết số đó tương ứng với lá bài nào? VD: với $N = 31$ thì hiển thị ra màn hình “Năm Chuồn”.

6) Tương tự như bài trên, các quân cờ được đánh số như sau:

	Tướng	Sĩ	Tượng	Xe	Pháo	Mã	Chốt
Đen	1	2	3	4	5	6	7
Trắng	8	9	10	11	12	13	14

Viết chương trình nhập một số có giá trị trong đoạn [1..14], sau đó in ra cho biết số đó tương ứng với quân cờ nào? VD: với $N = 11$ thì hiển thị ra màn hình “Xe Trắng”.

10.3. Cấu trúc lặp

- Viết chương trình in ra các ký tự và thứ tự của các ký tự này theo bảng mã ASCII (Chú ý chỉ in ra các ký tự có mã $ASCII > 32$).
- Viết chương trình hiển thị ra màn hình tất các số trong đoạn [100 . . 999] thỏa tính chất $P = \text{”Tổng các chữ số} = \text{Tích các chữ số”}$. VD: 123 là số thỏa tính chất P.

- 3) Viết chương trình tìm tất cả các số gồm 3 chữ số thỏa tính chất “có chữ số hàng chục bằng nửa tổng của chữ số hàng trăm và chữ số hàng đơn vị.
- 4) Viết chương trình in các số Armstrong (là số có tổng lập phương của các chữ số bằng chính nó).
- 5) Cho trước số nguyên $n > 0$. Viết chương trình tìm tất cả các bộ số Pythagore (x, y, z) thỏa mãn:

$$\begin{cases} x^2 + y^2 = z^2 \\ x < y < z \leq n \end{cases}$$

- 6) Cho trước số tự nhiên n (với $1500 < n < 2000$). Viết chương trình cho hiện ra màn hình các cách phân tích số n thành dạng tổng lập phương của 2 số tự nhiên x, y như sau:
 $n = x^3 + y^3$.
- 7) Số “hoàn thiện” (perfect number) là số tự nhiên có tổng các ước số (kể cả 1) bằng chính nó. VD: số tự nhiên 28 là số hoàn thiện. Viết chương trình hiển thị ra màn hình tất cả các số hoàn thiện < 1000 .
- 8) Viết chương trình tìm số tự nhiên bé nhất có ít nhất 2 cách biểu diễn nó thành tổng của 4 số chính phương đôi một khác nhau. Ví dụ, $78 = 9 + 16 + 9 + 4 = 36 + 25 + 16 + 1$
- 9) Viết chương trình nhập vào số $n > 0$. Sau đó tính các biểu thức tổng gồm n số hạng như sau:

- $S1 = 1 + 2 + 3 + \dots$
- $S2 = 1 - 2 + 3 - 4 + \dots$
- $S3 = 1 + 3 + 5 + 7 + \dots$
- $S4 = 1 + 2 + 4 + \dots$
- $S5 = 1! + 2! + 3! + \dots$

10) Cho số tự nhiên n . Hãy tính:

- 2^n
- $n!$
- $\left(1 + \frac{1}{1^2}\right)\left(1 + \frac{1}{2^2}\right) \cdots \left(1 + \frac{1}{n^2}\right)$
- $\sqrt{2 + \sqrt{2 + \sqrt{2 + \dots + \sqrt{2}}}}$ (n dấu căn)
- $n!! = \begin{cases} 1!.3!.5! \dots & \text{neu } n \text{ le} \\ 2!.4!.6! \dots & \text{neu } n \text{ chan} \end{cases}$

11) Cho số thực a và số tự nhiên n . Hãy tính:

- a^n
- $a(a+1) \dots (a+n-1)$
- $\frac{1}{a} + \frac{1}{a(a+1)} + \dots + \frac{1}{a(a+1)K(a+n)}$

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

13) Cho trước số tự nhiên n . Viết chương trình tính tổng của các chữ số của số n .

14) Cho trước số tự nhiên n . Viết chương trình tính và in trị đảo ngược của số n .

15) Cho trước số tự nhiên n . Viết chương trình kiểm tra tính đối xứng của số n . Ví dụ, với $n = 12321$ thì hiển thị thông báo số 12321 là số đối xứng.

16) Cho trước số tự nhiên n . Viết chương trình kiểm tra tính tuần hoàn và in ra số chu kỳ (nếu có) của số n . Ví dụ, với $n=12121212$ thì in ra số 12121212 tuần hoàn với các chu kỳ 12, 1212.

- 17) Cho trước số tự nhiên n . Viết chương trình tính số đối ứng của số n . Ví dụ, với $n = 1047$ thì số đối ứng là 9063.
- 18) Cho trước số tự nhiên n . Hãy cho biết:
- n có bao nhiêu chữ số ?
 - Tính tổng các chữ số của n .
 - Tính tích các chữ số của n .
 - Tìm chữ số đầu tiên của n .
- 19) Cho 2 số tự nhiên a, b . Hãy:
- Tìm ước số chung lớn nhất của chúng.
 - Tìm bội số chung nhỏ nhất của chúng.
- 20) Cho số tự nhiên n . Hãy xây dựng số m bằng cách loại bỏ đi trong số n tất cả các chữ số 0 và chữ số 5 (thứ tự ban đầu của các chữ số được giữ nguyên). Ví dụ, với $n = 50915509$ thì $m = 919$.
- 21) Viết chương trình in ra màn hình bảng cửu chương theo chiều dọc và theo chiều ngang.
- 22) Trăm trâu ăn cỏ, trâu đực ăn 5. trâu nài ăn 3, ba trâu già ăn 1. Viết chương trình hiển thị số trâu theo từng loại.
- 23) Viết chương trình kiểm tra xem 2 số tự nhiên a, b có nguyên tố cùng nhau?
- 24) Viết chương trình nhập số nguyên $n > 0$ và một số nguyên h chỉ hệ (có giá trị 2, 8 hoặc 16). Sau đó hiển thị ra màn hình trị tương ứng của N dưới dạng chỉ định bởi hệ h (nhị phân, bát phân, hoặc thập lục phân).

VD: nhập $n = 65, h = 2$ In ra : 0000000001000001 B

- 25) Viết chương trình kiểm tra xem số $n > 0$ nhập vào từ bàn phím có phải là số nguyên tố?
- 26) Viết chương trình nhập vào số $n > 0$ và hiển thị ra màn hình tất cả các số nguyên tố $< n$.
- 27) Viết chương trình nhập vào số $n > 0$ và hiển thị ra màn hình n số nguyên tố đầu tiên.
- 28) Dãy số Fibonacci: 1 1 2 3 5 8 13 21 34 55 ... Viết chương trình kiểm tra xem số N nhập vào có phải là số Fibonacci?
- 29) Viết chương trình nhập vào số $n > 0$ và hiển thị ra màn hình các số Fibonacci $< n$.
- 30) Viết chương trình nhập vào số $n > 0$ và hiển thị ra màn hình n số Fibonacci đầu tiên.
- 31) Viết chương trình vẽ các hình sau đây với h là chiều cao nhập vào từ bàn phím: VD: với $h = 5$

```

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

```

```

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

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

```

32) Viết chương trình nhập vào số $n > 0$ và hiển thị ra màn hình theo dạng sau:

VD: với $n = 5$

1 2 3 4 5	A B C D E
2 3 4 5 1	B C D E A
3 4 5 1 2	C D E A B
4 5 1 2 3	D E A B C

5 1 2 3 4

E A B C D

33) Viết chương trình lần lượt nhập vào các trị nguyên cho đến khi nhập trị 0 thì dừng. Hãy tính:

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

34) Viết chương trình phân tích số nguyên $n > 0$ thành tích các thừa số nguyên tố theo các dạng sau: Ví dụ, với $n = 120$ thì:

- Dạng 1: $120 = 2 * 2 * 2 * 3 * 5$
- Dạng 2: $120 = 2 ^ 3 * 3 * 5$

35) Viết chương trình nhập vào cặp tọa độ $x, y < 10$. Sau đó vẽ dấu ‘*’ tại tọa độ (x,y) vừa nhập như sau: Ví dụ, nhập $x = 8, y = 5$

1 2 3 4 5 6 7 8 9 10

1 .

2 .

3 .

- 4 .
- 5 *
- 6
- 7
- 8
- 9
- 10

36) Viết chương trình tính căn bậc 2, căn bậc 3, căn bậc 4, căn bậc N của số nguyên A, ($N > 0, A > 0$) theo hệ thức truy hồi sau:

– Căn bậc 2:

$$\begin{cases} x_0 = 1 \\ x_n = \frac{x_{n-1} + \frac{A}{x_{n-1}}}{2} \end{cases}$$

– Căn bậc 3:

$$\begin{cases} x_0 = 1 \\ x_n = \frac{2x_{n-1} + \frac{A}{x_{n-1}^2}}{3} \end{cases}$$

– Căn bậc 4:

$$\begin{cases} x_0 = 1 \\ x_n = \frac{3x_{n-1} + \frac{A}{x_{n-1}^3}}{4} \end{cases}$$

– Một cách tổng quát với căn bậc N:

$$\begin{cases} x_0 = 1 \\ x_n = \frac{(n-1)x_{n-1} + \frac{A}{x_{n-1}^{n-1}}}{n} \end{cases}$$

37) Cho trước độ sai số ϵ . Viết chương trình tính E^x , $\text{Sin}(x)$, $\text{Cos}(x)$ theo các công thức khai triển sau:

$$E^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

CHƯƠNG 4. HÀM (FUNCTION)

1. Khái niệm

Để giải quyết một vấn đề tương đối lớn, phức tạp, ta thường có khuynh hướng phân chia vấn đề ban đầu thành những vấn đề con nhỏ hơn. Và từ những vấn đề con này, ta lại tiếp tục phân rã thành những vấn đề con nhỏ hơn nữa (nếu được) ... cho đến khi gặp vấn đề con sơ cấp có thể thấy ngay cách giải quyết. Cách tiếp cận này áp dụng tư tưởng “chia để trị” và được gọi là phương pháp phân tích từ trên xuống (Top – Down Analysis). Mỗi vấn đề con là một đơn thể (module) đảm trách giải quyết một chức năng cụ thể, xác định. Các đơn thể liên lạc với nhau thông qua các thông số.

Ta có thể xem hoạt động của hàm như hoạt động của một hộp đen. Cùng một dữ kiện vào sẽ cho ra cùng kết quả.

Hầu hết các NNLT cấp cao (C, C++, PASCAL, FOX . . .) đều hỗ trợ phương pháp lập trình cấu trúc cho phép tổ chức chương trình máy tính thành nhiều đơn thể (module). Mỗi đơn thể chương trình tương ứng với 1 hàm (trong C/C++).

Có 2 loại hàm trong NNLT “C/C++”:

- Hàm thư viện do chương trình dịch cung cấp.
- Hàm tự tạo do USER định nghĩa.

Hàm thư viện (library functions): chương trình dịch “C/C++” cung cấp một thư viện đồ sộ các hàm tiện ích giúp giảm nhẹ công sức lập trình và cho phép USER tập trung nhiều hơn vào logic của vấn đề đang giải quyết. Để sử dụng các hàm này trong chương trình, cần chèn vào đầu chương trình các tập tin tiêu đề chứa các khai báo và định

nghĩa hằng, biến, hàm nguyên mẫu, . . . bằng các chỉ thị tiền xử lý `#include <tên tập tin>`.

Hàm tự tạo: Ngoài ra, “C/C++” cho phép USER định nghĩa thêm các hàm khác phục vụ cho nhu cầu lập trình của mình.

2. Khai báo và định nghĩa hàm

Gồm có 2 phần: dòng tiêu đề và phần thân hàm

```
[<kiểu giá trị trả về>] <tên hàm>([<danh sách các tham số>])  
  
{  
  
    // các khai báo cục bộ  
  
    . . . .  
  
    // các câu lệnh  
  
    . . . .  
  
}
```

Trong đó:

- <kiểu giá trị trả về>:
 - Mô tả kiểu dữ liệu của giá trị mà hàm sẽ trả về cho nơi gọi nó. Nói chung, giá trị trả về của hàm có thể thuộc một kiểu dữ liệu hợp lệ bất kỳ, ngoại trừ “C/C++” không cho phép trả về một biến kiểu mảng.
 - Nếu ta bỏ qua, không mô tả <kiểu giá trị trả về> thì chương trình dịch sẽ ngầm hiểu đó là kiểu int.

– Mô tả <kiểu giá trị trả về> là void nếu ta muốn hàm không trả về giá trị nào (hàm này tương đương với thủ tục trong PASCAL).

- <tên hàm>: là danh hiệu hợp lệ
- <danh sách các tham số> có dạng:

<tên kiểu> <tên biến>, <tên kiểu> <tên biến>, ...

- Phần thân hàm đặt trong cặp dấu { }, và chứa các câu lệnh mô tả hoạt động của hàm.

3. Lệnh return

Hàm sau khi đã được khai báo và định nghĩa, có thể được gọi thực hiện bằng lời gọi hàm trong chương trình. Hoạt động của được kết thúc theo 1 trong 3 cách như sau:

- *Trường hợp hàm có giá trị trả về là void*: hàm kết thúc hoạt động sau khi thực hiện câu lệnh cuối trước dấu } sau cùng, hoặc nếu ta muốn hàm kết thúc sớm trong điều kiện nào đó thì dùng lệnh return;
- *Trường hợp hàm có giá trị trả về khác void*: Khi gặp lệnh **return <biểu thức>**; chương trình dịch sẽ kết thúc hoạt động của hàm, và trị của <biểu thức> được trả về cho nơi gọi hàm.

4. Các dạng hàm

4.1. Hàm không có tham số

Ví dụ 1: hàm in dòng chữ “HELLO C”


```
void PrintHello()
{
    cout << "Hello C++\n";
}
```

Cách gọi hàm:

```
PrintHello();
```

Ví dụ 2: hàm giải phương trình bậc 2:

```
void GiaiPTBac2()
{
    double a, b, c;

    cout << "Nhap cac he so a, b, c : "; cin >> a >> b >> c;

    double delta = b*b-4*a*c;

    if (delta<0)

        cout << "Phuong trinh vo nghiem\n";

    else if (delta==0)

        cout << "Phuong trinh co nghiem kep X = " << -b/2/a;

    else // delta>0

        cout << "Phuong trinh co 2 nghiem phan biet:"
```

```
<< "\nX1 = " << (-b-sqrt(delta))/2/a
<< "\nX2 = " << (-b+sqrt(delta))/2/a;
}
```

Cách gọi hàm:

```
GiaiPTBac2();
```

4.2.Hàm với nhiều tham số

Ví dụ 1, hàm tính số max của 2 số nguyên:

```
int Max(int a, int b)
{
    return (a > b ? a : b);
}
```

Cách gọi hàm:

```
Max(x, y); // voi x, y co kieu int
```

Ví dụ 2, hàm vẽ hình vuông đặc chiều cao h:

```
void Draw(int h)
{
    for (int i=1; i<=h; ++i) // lap h dong
```

```
{  
  
    for (int j=1; j<=h; ++j) // lap h cot  
  
        cout << setw(3) << '*';  
  
    cout << endl;  
  
}  
  
}
```

Cách gọi hàm:

```
Draw(5)
```

Hoặc

```
Draw(h), với h có kiểu int
```

4.3.Hàm nguyên mẫu (function prototype)

Hàm nguyên mẫu được dùng để cung cấp thông tin cho chương trình dịch về tên hàm, kiểu giá trị trả về, số lượng, thứ tự và kiểu của các tham số của hàm. Chương trình dịch căn cứ vào các thông tin này để kiểm tra các lời gọi hàm trong chương trình.

Hàm nguyên mẫu được đặt sau phân khai báo toàn cục và ngay trước hàm main() hoặc có thể đặt trong tập tin khác.

Khai báo:

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

Trong đó:

- <kiểu giá trị trả về> mô tả kiểu của giá trị trả về bởi hàm
- <tên hàm> là danh hiệu hợp lệ và nên đặt tên sao cho thể hiện được chức năng của hàm
- <danh sách các tham số> mô tả các kiểu dữ liệu và tên biến đại diện của từng tham số., mỗi cặp kiểu dữ liệu và tên biến được phân cách bởi dấu phẩy ‘,’

Ví dụ: Khai báo hàm nguyên mẫu có chức năng xác định trị min giữa 2 số nguyên.

```
int Min(int, int) ;
```

hay

```
int Min(int a, int b) ; // nên dùng cách khai báo này
```

4.4. Tổ chức một chương trình “C/C++” (trên cùng 1 tập tin)

Cách 1: chương trình gồm 3 phần

PHẦN KHAI BÁO TOÀN CỤC

PHẦN KHAI BÁO VÀ ĐỊNH NGHĨA HÀM

HÀM main()

Cách 2: chương trình gồm 4 phần (nên dùng cách này)

PHẦN KHAI BÁO TOÀN CỤC

PHẦN KHAI BÁO HÀM NGUYÊN MẪU

HÀM main()

PHÂN ĐỊNH NGHĨA HÀM

5. Các phương pháp truyền tham số

Tham số thực (actual parameter) là tham số trong lời gọi hàm.

Tham số hình thức (formal parameter) là tham số trong phân khai báo và định nghĩa. Tham số hình thức chỉ là tên đại diện cho tham số thực tương ứng. Kiểu của tham số hình thức sẽ qui định kiểu của tham số thực.

Có 2 cách truyền tham số cho hàm:

- Truyền bằng giá trị (chủ yếu trong “C/C++”): Trong phương pháp này, chương trình dịch cấp phát vùng nhớ riêng cho từng tham số hình thức, sau đó sao chép giá trị của tham số thực tương ứng vào các tham số hình thức. Sau khi kết thúc thực hiện hàm, chương trình dịch sẽ thu hồi các vùng nhớ đã cấp phát cho các tham số hình thức, và các biến cục bộ khai báo bên trong hàm. Như vậy, mọi sự thay đổi trị của các tham số hình thức đều không ảnh hưởng đến các tham số thực bên ngoài hàm.
- Truyền bằng tham chiếu (còn gọi là truyền bằng địa chỉ): Chỉ có trong C++. Thay vì cấp phát vùng nhớ riêng như trong phương pháp truyền bằng giá trị, chương trình dịch sẽ truyền địa chỉ của các tham số thực tương ứng cho các tham số hình thức. Điều đó có nghĩa là ta có thể xem tham số hình thức cũng chính là tham số thực, hay nói cách khác tham số hình thức là tên gọi khác của tham số thực. Như vậy, mọi sự thay

đổi trị của tham số hình thức bên trong hàm chính là thay đổi trị của tham số thực bên ngoài hàm.

- **Để phân biệt cách truyền tham chiếu**, với cách truyền trị, ta đặt kí hiệu & trước tên biến hình thức.

Ví dụ: Khai báo hàm nguyên mẫu có chức năng hoán đổi trị của 2 biến nguyên:

```
void Swap(int &,int &); // truyền bằng tham chiếu
```

hay

```
void Swap(int & a,int & b); // truyền bằng tham chiếu
```

```
void F(int, int ); // truyền bằng trị
```

hay

```
void F(int a, int b); // truyền bằng trị
```

Chú ý:

- Trong cách truyền tham chiếu, tham số thực tương ứng phải là một biến. Còn trong cách truyền trị, tham số thực tương ứng có thể là biến, hằng, lời gọi hàm, hoặc một biểu thức cùng kiểu với tham số hình thức.
- Các tham số hình thức trong cách truyền bằng giá trị được gọi là tham trị. Còn các tham số hình thức trong cách truyền bằng tham chiếu được gọi là tham biến.

6. Phạm vi (scope) và cấp lưu trữ (storage class) của các đối tượng

Để tối ưu hóa việc sử dụng bộ nhớ và quản lý việc truy xuất các đối tượng (biến, hằng, hàm) trong chương trình, “C/C++” đưa ra các khái niệm phạm vi và cấp lưu trữ.

6.1. Phạm vi

Phạm vi là vùng chương trình mà đối tượng được nhận biết và có thể được sử dụng. Nói chung, phạm vi của một đối tượng trải dài từ nơi nó được khai báo đến cuối khối, hàm, hay tập tin chứa đối tượng đó. Có các loại phạm vi sau:

- Phạm vi cục bộ (local scope)
 - Phạm vi khối (Block scope)
 - Phạm vi hàm (Function scope)
- Phạm vi toàn cục (global scope)
 - Phạm vi tập tin (File scope)
 - Phạm vi chương trình (Program scope)

Ví dụ về phạm vi:

```
#include <iostream.h>

int x=3; // biến x toàn cục

const int MAX = 10;

void fct(int x);
```

```

void main(){

    int x=1; // x cục bộ của hàm main()

    fct(x);

    {

    int y=4;

    x += y;

    }

    cout<<"Y = "<<y; // Sai vì biến y đã bị hủy, không thể truy
xuất

    cout<<"X của hàm main() = "<<x;

    cout<<"X toàn cục = "<<::x;

}

void fct(int x){

    x += MAX;

    cout<<"X bên trong hàm fct() = "<<x<<endl;

}

```

Biến khai báo bên trong khối hay hàm được gọi là **biến cục bộ** (local variable) của khối, hàm đó và chỉ có thể được truy xuất bên trong phạm vi khối hay hàm đó mà thôi. Còn biến khai báo bên ngoài

mọi hàm, mọi khối được gọi là **biến toàn cục** (global variable) và có thể được truy xuất ở mọi nơi trong chương trình.

Các biến cục bộ có thời gian tồn tại tương đối ngắn. Chúng sẽ bị hủy mỗi khi ra khỏi khối hay kết thúc thực hiện hàm chứa nó.

Ngược lại, các biến toàn cục có thời gian tồn tại là thời gian của chương trình.

6.2. Cấp lưu trữ

Cấp lưu trữ (storage class) là cách thức NNLT cấp phát vùng nhớ và lưu trữ biến. Cấp lưu trữ của một biến được xác định bằng các từ khóa sau: *auto*, *register*, *static*, *extern*.

Biến auto (còn gọi là biến tự động, biến cục bộ)

Mọi biến khai báo bên trong một khối hay hàm mặc nhiên có tính chất auto, trừ khi xác định rõ cấp lưu trữ khác.

Ví dụ, khai báo *int x;* tương đương với khai báo *auto int x;*

Biến auto có phạm vi cục bộ bên trong hàm hay khối và có thời gian tồn tại ngắn, do được cấp phát trong vùng nhớ STACK.

Biến register (ít dùng)

Để tăng tốc độ truy xuất biến (thường là các biến đếm trong vòng lặp for), ta đặt từ khóa register trước khai báo biến, yêu cầu chương trình dịch lưu trữ biến trong thanh ghi. Chú ý, chương trình dịch có

thể bỏ qua không đáp ứng lời yêu cầu này nếu có quá nhiều lời đề nghị loại này hoặc nếu không còn đủ thanh ghi để cấp phát.

Biến static (còn gọi là biến tĩnh)

Là biến được cấp phát trong vùng nhớ DATA (trình bày sơ đồ sử dụng bộ nhớ của một chương trình “C/C++”), và do đó có tính chất cố định, lâu dài. Thời gian tồn tại của biến static là thời gian của chương trình. Như vậy biến static khai báo bên trong một khối, hay hàm sẽ không bị hủy khi ra khỏi khối hay hàm đo, và vẫn lưu giữ giá trị cũ của lần gọi hàm trước. Chú ý, biến static phải được khởi tạo giá trị khi khai báo. Chương trình dịch sẽ chỉ khởi tạo giá trị cho biến static duy nhất một lần trong lần gọi hàm đầu tiên.

Biến extern

Một chương trình C/C++ có thể được tổ chức trên nhiều tập tin khác nhau. Và phạm vi của một biến nào đó trong chương trình có thể được trải dài trên nhiều tập tin nhờ vào từ khóa extern như sau: (... vẽ hình ...)

Từ khóa extern đứng trước khai báo biến có ý nghĩa là biến đó đã được khai báo và định nghĩa ở nơi khác trong chương trình (trong cùng tập tin hay ở trong tập tin khác), chương trình dịch sẽ không cấp phát thêm vùng nhớ cho biến có khai báo extern mà sử dụng chung vùng nhớ đã cấp phát trước đó.

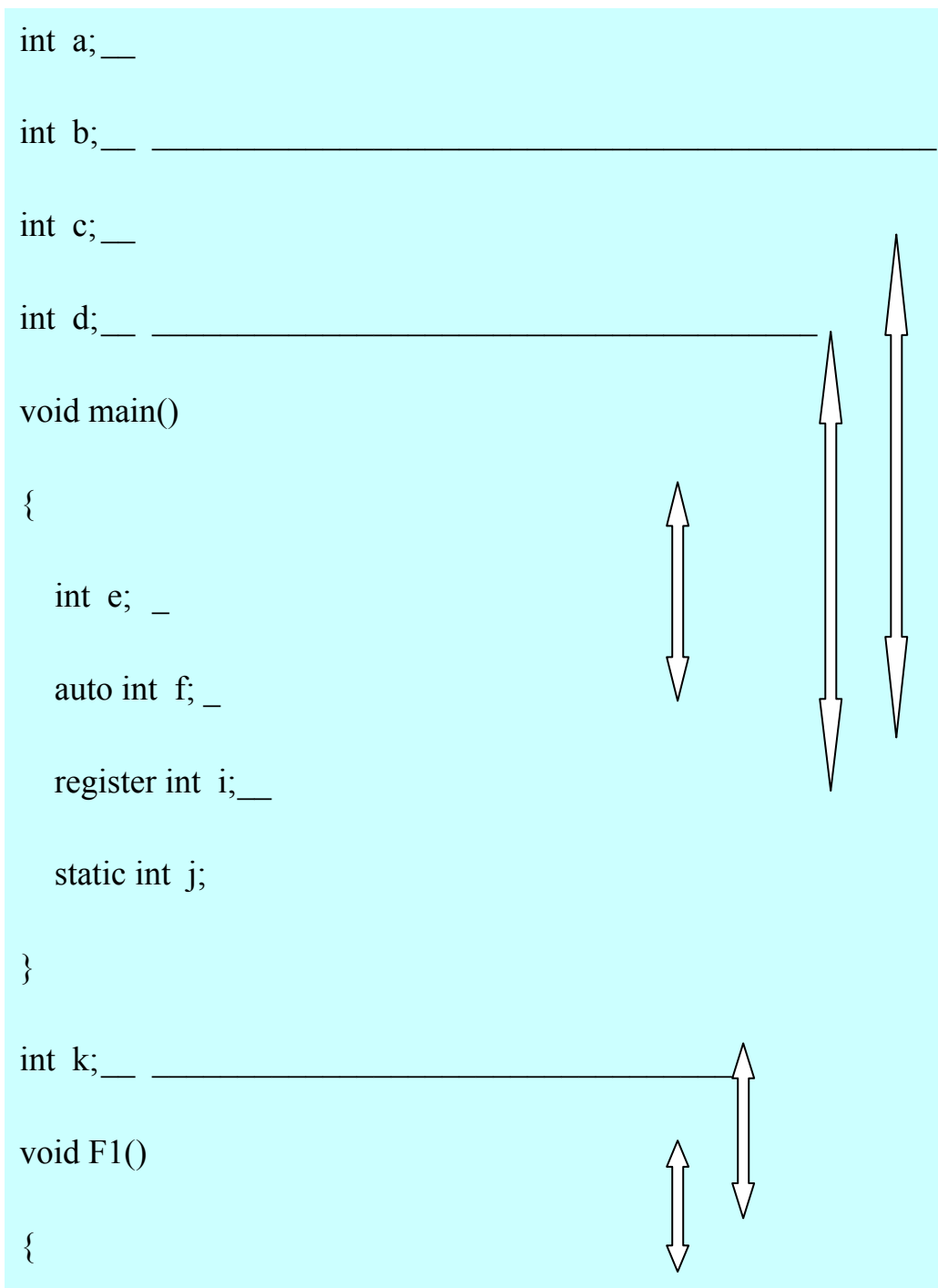
Tóm tắt về cấp lưu trữ biến (Storage class)

Vị trí khai báo biến	Từ khóa	Thời gian tồn tại	Phạm vi
Hàm	auto (default)	Hàm	Hàm
Hàm	register	Hàm	Hàm
Hàm	static	Chương trình	Hàm
Bên ngoài hàm	static	Chương trình	File
Bên ngoài hàm	-----	Chương trình	File
Bên ngoài hàm	extern	Chương trình	Nhiều File

Ví dụ 1:

Minh họa tầm vực, phạm vi của biến (**Visibility (Scope) of variables**)

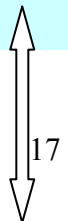
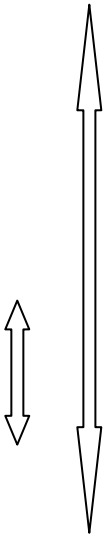
File1.cpp



```
int m; _____  
}
```

File2.cpp

```
extern int a; _  
  
void F2()  
{  
    extern int b;  
}  
  
extern int c; _
```



```
void F3()
{
    extern int d;
}
```



Ví dụ 2:

```
#include <iostream.h>

#include "file3.cpp"

const int MAX=10;

int x=3;

void main()
{
    const int MAX = 5;

    int x = 1;

    x += ::MAX;
```

```

cout<<"X cuc bo trong ham main() = "<<x<<endl;

cout<<"X toan cuc = "<<::x<<endl;

f1();

f2();

}

```

File3.cpp

```

#include <iostream.h>

extern int x; // dung chung bien x da khai bao o noi khac

extern const int MAX; // dung chung hang MAX da khai bao o noi khac

void F1()

{

    int x=2; // x cuc bo

    cout<<"X cuc bo trong ham f1() = "<<x<<endl;

    cout<<"X toan cuc da khai bao trong ham main() = "<<::x<<endl;

}

void F2()

```

```

{

const int MAX=15; // hang cuc bo

cout<<"Hang MAX cuc bo trong ham f2() = "<<MAX<<endl;

cout<<"Hang MAX toan cuc da khai bao trong File1 = "

<<::MAX<<endl;

cout<<"Bien x toan cuc da khai bao trong File1 = "
"<<::x<<endl;

}

```

7. Cơ chế gọi hàm và bộ nhớ stack

Một chương trình “C/C++” là tập hợp các hàm trong đó hàm main() là quan trọng nhất và không thể thiếu do đây là điểm đầu vào của chương trình. Khi ta cho thực hiện chương trình thì hàm main() được gọi thực hiện trước tiên. Khi đó, hàm main() có thể gọi chạy các hàm khác, và các hàm này lại có thể gọi chạy các hàm khác nữa, ...

Khi một hàm được gọi thì “C/C++” sẽ tạo ra một khung kích hoạt (activation frame) trong vùng nhớ stack. Khung kích hoạt chứa các biến cục bộ và mẫu tin hoạt động (active record) của hàm. Mẫu tin hoạt động chứa địa chỉ trở về (return address) của hàm gọi nó và các thông số khác. Sau khi kết thúc thực hiện hàm, trình biên dịch sẽ xóa các biến cục bộ đã cấp phát cho hàm (các tham số hình thức trị, các biến cục bộ khai báo bên trong hàm) và chuyển điều khiển về dòng lệnh tại địa chỉ trở về của hàm gọi nó.

Ví dụ, xét chương trình sau:


```
void A();

void B();

void C();

void D();

void E();

void main(){

    ...

    D();

    E();

    ...

}

void A()

{

    ...

    B();

    ...

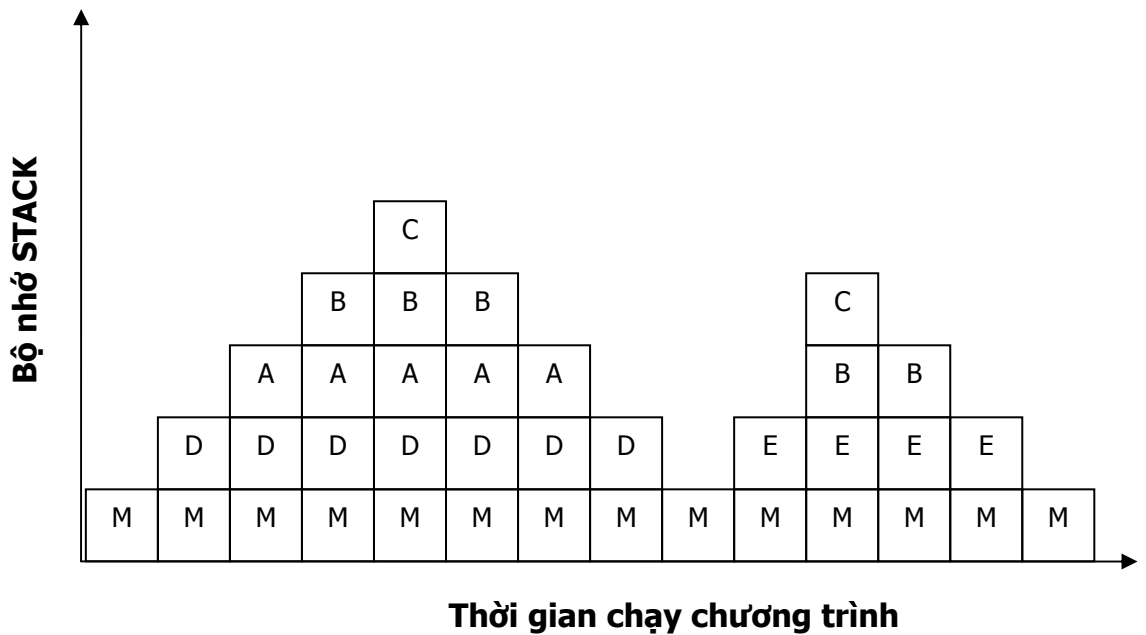
}

void B()
```

```
{  
  
    ...  
  
    C();  
  
    ...  
  
}  
  
void C()  
  
{  
  
    ...  
  
}  
  
void D()  
  
{  
  
    ...  
  
    A();  
  
    ...  
  
}  
  
void E()  
  
{  
  
    ...
```

```
C();  
  
...  
  
}
```

Ta có hình ảnh sau:



8. Hàm có tham số với giá trị ngầm định (Function with default arguments)

C++ cho phép cài đặt hàm với tham số có giá trị ngầm định. Các tham số này phải được khai báo liên tục và theo thứ tự từ phải sang trái.

Khai báo:

```
<return type> <function name>(<type> <param1>=<val1>,  
<type> <param>=<val2>,... );
```

Phần định nghĩa được thực hiện như hàm thông thường.

```
<return type> <function name>(<type> <param1>, <type>
<param2>,...)

{

    // các câu lệnh ...

}
```

9. Nạp chồng hàm (Function overloading)

C++ cho phép dùng chung một danh hiệu để đặt tên cho các hàm khác nhau. Ta gọi đó là *nạp chồng hàm*.

Thông thường, ta chỉ nạp chồng hàm đối với những hàm giống nhau về bản chất, nhưng khác nhau ở số lượng, và kiểu dữ liệu của các tham số.

Ví dụ, để tìm trị max của 2 số nguyên, 3 số nguyên, ... , hoặc 2 số thực, 3 số thực, ... trong C chuẩn, ta viết nhiều hàm khác nhau, mỗi hàm phải được đặt tên khác nhau. Trong C++, ta có thể nạp chồng, i.e., ta có thể dùng chung tên hàm (VD max) cho tất cả các hàm này.

Các hàm được nạp chồng có thể giống nhau về kiểu của trị trả về, nhưng bắt buộc phải khác nhau ở số lượng, hoặc kiểu của các tham số. Chương trình dịch sẽ căn cứ vào các yếu tố này để xác định hàm cần thực hiện mỗi khi gặp lời gọi hàm được nạp chồng.

Chú ý

Khả năng nạp chồng hàm kết hợp với hàm có tham số với giá trị ngầm định có thể gây ra tình trạng nhập nhằng, mơ hồ như sau:

```
void F(int=2, double=3.0);

void F(int);

void F(double);

void main()
{
    double x = 20.0;

    int y = 10;

    F(x, y); // mơ hồ! chương trình dịch không biết gọi hàm nào
}

void F(int, double)
{
}

void F(int)
{
```

```
}  
  
void F(double)  
  
{  
  
}
```

10. Một số gợi ý khi thiết kế hàm

- Xác định rõ **chức năng, nhiệm vụ** của hàm.
- Chỉ **nên thiết kế hàm** theo phương châm “mỗi hàm chỉ thực hiện một nhiệm vụ duy nhất”, và nên thiết kế sao cho có thể sử dụng lại hàm để hỗ trợ cho các việc khác (reusable).
- **Đặt tên hàm** sao cho có tính gợi nhớ (Mnemonic)
- **Nên đặt chú thích**, ghi rõ các thông tin về hàm như chức năng, điều kiện dữ liệu vào, xác định dữ liệu ra của hàm, . . .
- **Xác định trị trả về**: hàm có cần trả về giá trị? Nếu có, xác định rõ kiểu trả về. Đối với các hàm có chức năng nhập/xuất dữ liệu, trị trả về thường là void (tương đương với thủ tục trong Pascal). Còn đối với loại hàm kiểm tra một tính chất P nào đó, ta thường trả về giá trị 0 hoặc 1, i.e. trả về trị của một biểu thức logic.
- Xác định **số lượng tham số** và **kiểu** của từng tham số: hàm có nhận tham số hay không? Bao nhiêu tham số? Kiểu của từng tham số?
- Xác định rõ **phương pháp truyền tham số**: nếu không có nhu cầu làm thay đổi trị của tham số thực truyền vào cho

hàm thì áp dụng phương pháp truyền bằng giá trị. Còn ngược lại thì áp dụng cách truyền bằng tham chiếu.

11. Một số ví dụ minh họa

12. Câu hỏi

- Nêu cách khai báo hàm, định nghĩa hàm, cách gọi hàm.
- Phạm vi của một đối tượng (biến, hằng) trong chương trình ?
- Trình bày các phương pháp truyền tham số cho hàm (truyền bằng giá trị, bằng tham chiếu, và bằng tham trở)
- Nêu cách thiết kế hàm mà theo Anh (Chị) cho là đạt yêu cầu.
- Cho ví dụ về hàm xuất/nhập.
- Cho ví dụ về hàm kiểm tra một giá trị nguyên thỏa tính chất “P” nào đó.
- Cho ví dụ về hàm xác định giá trị nguyên thỏa tính chất “P” nào đó.
- Trình bày cách tổ chức một chương trình “C/C++”.
- Trình bày cấp lưu trữ của một đối tượng (auto, register, static, extern).
- Khai báo các hàm nguyên mẫu sau:
 - Giải phương trình bậc 2 $ax^2 + bx + c = 0$
 - Giải phương trình bậc 4 $ax^4 + bx^2 + c = 0$
 - In bảng cửu chương theo chiều dọc.
 - In bảng cửu chương theo chiều ngang.
 - Vẽ hình tam giác với chiều cao h dạng ...
 - Phân tích số $n > 0$ thành tích các thừa số nguyên tố theo dạng $n = 120 = 2*2*2*3*5$.

- Phân tích số $n > 0$ thành tích các thừa số nguyên tố theo dạng $n = 120 = 2^3 * 3 * 5$.
- Kiểm tra năm y có phải là năm nhuận ?
- Kiểm tra 1 bộ ngày, tháng, năm có hợp lệ hay không ?
- Kiểm tra số tự nhiên $n > 0$ có phải là số nguyên tố ?
- Kiểm tra 2 số tự nhiên có nguyên tố cùng nhau ?
- Kiểm tra số tự nhiên $n > 0$ có phải là số thuộc dãy số Fibonacci ?
- Kiểm tra số $n > 0$ có phải là số đối xứng ?
- Kiểm tra số n^a [100 ... 999] có phải là số thỏa tính chất P = “có tổng các chữ số bằng tích các chữ số” ?
- Kiểm tra số $n > 0$ có phải là số hoàn thiện ?
- Tính trị max của 2 số nguyên.
- Tính trị min của 2 số nguyên.
- Tính USCLN của 2 số tự nhiên.
- Tính tổng S của $n > 0$ số hạng đầu tiên theo công thức cho trước.
- Tính trị đảo của số tự nhiên.
- Tính trị đối ứng bù 10 của số tự nhiên.
- Tính số ngày tối đa của tháng m , năm y .
- Tính ngày hôm sau của 1 bộ ngày tháng năm.
- Tính khoảng cách giữa 2 bộ ngày, tháng, năm.
- Định nghĩa các hàm nguyên mẫu trên, và viết chương trình ứng dụng.

- Cho chương trình sau đây:

```
#include <iostream.h>
```



```

void mul(int&,int,int&);

int main() {

    int x = 4, y = 3, z = 2;

    mul( y, z, x );

    cout << "\nX = " << x;

    cout << "\nY = " << y;

    cout << "\nZ = " << z;

    return 0;

}

void mul( int & x,int y, int & z ) {

    x *= y;

    y *= z;

    z *= x;

}

```

Hãy cho biết trị in ra màn hình của các biến x, y, và z. Giải thích theo mẫu bảng sau:

Câu lệnh	Hàm main()			Hàm mul()		
	X	Y	Z	X	Y	Z

- Cho chương trình sau đây:

```
#include <iostream.h>

void f( int x, int & y, int z );

void g( int & x, int y, int & z );

int main() {

    int x = 2, y = 3, z = 4;

    f( y, z, x );

    cout << "\nX = " << x;

    cout << "\nY = " << y;

    cout << "\nZ = " << z;

    return 0;

}

void f( int x, int & y, int z ){

    g( z, y, x );

    x += y;

    y += z;

    z += x;

}
```

```

void g( int & a, int b, int & c ) {

    a *= b;

    b *= c;

    c *= a;

}

```

Hãy cho biết trị in ra màn hình của các biến x, y, và z. Giải thích theo mẫu bảng sau:

Câu lệnh	Hàm main()			Hàm f()			Hàm g()		
	X	Y	Z	X	Y	Z	A	B	C

13. Bài tập

- Viết hàm xác định số Max trong 2 số nguyên. Sau đó viết chương trình áp dụng để in ra màn hình số Max trong 5 số nguyên.
- Viết hàm hoán đổi trị 2 biến nguyên. Sau đó viết chương trình ứng dụng.
- Viết hàm xác định USCLN của 2 số tự nhiên. Sau đó viết chương trình áp dụng nhập vào 2 số tự nhiên và cho biết 2 số đó có nguyên tố cùng nhau ? Sau đó tính BSCNN của 2 số này.
- Cho trước số tự nhiên n. Viết hàm xác định số nguyên tố thứ n. Sau đó viết chương trình áp dụng.
- Cho trước số tự nhiên n. Viết hàm xác định số Fibonacci thứ n. Sau đó viết chương trình áp dụng.

- 6) Cho trước số tự nhiên n . Viết hàm kiểm tra số n có phải là số nguyên tố hay không ? Sau đó viết chương trình áp dụng.
- 7) Viết hàm kiểm tra 2 số có nguyên tố cùng nhau hay không ? Sau đó viết chương trình áp dụng.
- 8) Cho trước số tự nhiên n . Viết hàm kiểm tra số n có phải là số Fibonacci hay không ? Sau đó viết chương trình áp dụng.
- 9) Cho trước số tự nhiên n . Viết hàm kiểm tra số n có phải là số đối xứng hay không ? Sau đó viết chương trình áp dụng.
- 10) Cho tính chất $P =$ "Tổng các chữ số của 1 số nguyên bằng tích của các chữ số của nó". Viết hàm kiểm tra một số (gồm 3 chữ số) có thỏa tính chất P hay không ? Sau đó viết chương trình áp dụng.
- 11) Viết hàm in ra màn hình bảng cửu chương dọc (ngang). Sau đó viết chương trình áp dụng.
- 12) Viết các hàm in ra màn hình các loại tam giác đặc và rỗng với chiều cao h . Sau đó viết chương trình áp dụng tạo một menu cho phép chọn dạng hình tam giác để in ra màn hình.
- 13) Viết chương trình bao gồm các hàm sau:
 - Hàm kiểm tra năm y cho trước có nhuận hay không ?
 - Hàm xác định số ngày tối đa của tháng m trong năm y cho trước.
 - Hàm kiểm tra tính hợp lệ của một bộ ngày, tháng, năm cho trước.
 - Hàm xác định ngày kế tiếp của một bộ ngày, tháng, năm cho trước.

- Hàm xác định N ngày kế tiếp của một bộ ngày, tháng, năm cho trước.

Viết chương trình áp dụng các hàm trên.

- 14) Viết chương trình tính khoảng cách ngày giữa 2 bộ ngày tháng năm.

CHƯƠNG 5. KIỂU MẢNG (ARRAY)

1. Khái niệm

Kiểu mảng cho phép giải quyết nhiều bài toán lập trình liên quan đến một lượng lớn dữ liệu một cách gọn, súc tích. Ví dụ: bài toán xác định số min, max của nhiều số nguyên, tìm kiếm, sắp xếp trên 1 dãy các số liệu, . . .

Mảng là kiểu dữ liệu có cấu trúc bao *gồm nhiều phần tử cùng kiểu* và được *bố trí vùng nhớ liên tục*.

Kiểu của các phần tử mảng gọi là *kiểu cơ sở*. Mỗi phần tử mảng là một biến có kiểu cơ sở.

Mảng có *kích thước* là số phần tử trong mảng. Kích thước mảng bắt buộc phải là biểu thức hằng nguyên để có thể cấp phát vùng nhớ lúc biên dịch.

Mảng có thể có 1 chiều hay nhiều chiều. Mảng n chiều ($n > 1$) có thể được coi như mảng 1 chiều mà mỗi phần tử là mảng n-1 chiều. Số phần tử của mảng nhiều chiều bằng tích của kích thước các chiều.

Ví dụ: mảng các số nguyên, các số thực, các kí tự, . . .

2. Khai báo & khởi tạo giá trị các phần tử mảng trong “C/C++”

```
<kiểu cơ sở> <tên biến mảng> [<kích thước>];
```

Trong đó:

- <kiểu cơ sở> có thể là kiểu dữ liệu hợp lệ bất kỳ trong C/C++.

- <tên biến mảng> là 1 danh hiệu hợp lệ và có giá trị là địa chỉ của vùng nhớ của phần tử đầu tiên của mảng.
- <kích thước> là một giá trị hằng nguyên hoặc một biểu thức hằng nguyên (không thể là 1 biến) và được đặt trong cặp dấu []. Trường hợp mảng có nhiều chiều, thì mỗi chiều phải được xác định rõ kích thước bằng [<kích thước 1>][<kích thước 2>]. . .

Ví dụ:

Để khai báo mảng nguyên 1 chiều có tên arr1D gồm 5 phần tử.

```
int arr1D[5];
```

Để khai báo mảng nguyên 2 chiều có tên arr2D gồm 6 phần tử (2 dòng 3 cột)

```
int arr2D[2][3];
```

Ta có thể khởi tạo giá trị ban đầu cho các phần tử của mảng 1 chiều như sau:

```
int arr1D[5] = {3,5,4,6,2};
```

```
int arr1D[ ] = {3,5,4,6,2};
```

```
int arr1D[5] = {3}; //phần tử đầu = 3, các phần tử còn lại = 0
```

Để khai báo và khởi tạo mảng 1 chiều với tất cả các phần tử có trị = 0:

```
int arr1D[5] = {0};
```

Ta có thể khởi tạo giá trị ban đầu cho các phần tử của mảng 2 chiều như sau:

```
int arr2D[2][3] = {3,5,6,2,4,1};  
  
int arr2D[ ][3] = {3,5,6,2,4,1};  
  
int arr2D[2][3] = {{3,5,6},{2,4,1}};  
  
int arr2D[ ][3] = {{3,5,6},{2,4,1}};
```

Để khai báo và khởi tạo mảng 2 chiều với tất cả các phần tử có trị = 0:

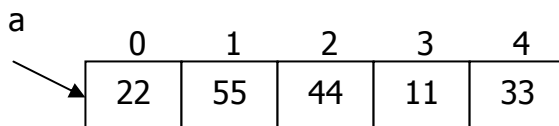
```
int arr2D[2][3] = {0};
```

3. Truy xuất các phần tử của mảng

Các phần tử mảng có thể được truy xuất thông qua chỉ số của nó trong mảng.

Các phần tử mảng được đánh số thứ tự bắt đầu từ 0, số thứ tự này gọi là chỉ số mảng. Các phần tử mảng có thể được truy xuất như sau: <tên biến mảng>[chỉ số]

Ví dụ: mảng 1 chiều



Ví dụ: mảng 2 chiều

a

	0	1	2
0	22	55	44
1	11	33	66

Chú ý

- Chương trình dịch “C/C++” không kiểm tra việc vi phạm biên mảng.
- Kích thước của mảng phải là biểu thức hằng

4. Truyền tham số mảng cho hàm

Trong phần khai báo và định nghĩa hàm, đối với mảng 1 chiều ta ghi cặp dấu [] ngay sau tên kiểu cơ sở của mảng. Đối với mảng nhiều chiều, ta phải ghi đầy đủ số lượng cặp dấu [] bằng đúng số chiều và đặt ngay sau tên kiểu cơ sở, các chiều phải ghi rõ kích thước, ngoại trừ kích thước của chiều đầu tiên để trông như đối với mảng 1 chiều.

Trong lời gọi hàm, ta chỉ cần ghi tên biến mảng tại vị trí tương ứng với tham số mảng hình thức.

Thực chất của việc truyền tham số mảng cho hàm là sự truyền giá trị địa chỉ của tham số thực cho tham số mảng hình thức. Điều này có nghĩa là trước khi cho thực hiện hàm chương trình dịch sẽ cấp phát vùng nhớ riêng cho tham số mảng hình thức như một biến cục bộ. Tham số cục bộ này nhận trị ban đầu là bản sao giá trị địa chỉ của vùng nhớ đầu tiên của mảng truyền vào cho hàm và sẽ bị hủy khi kết thúc thực hiện hàm. Tuy nhiên, do tham số thực truyền địa chỉ của nó

cho tham số hình thức nên mọi sự thay đổi trị của các phần tử mảng trong hàm sẽ làm thay đổi trị của các phần tử mảng bên ngoài hàm.

5. Các thao tác cơ bản trên mảng 1 chiều

- *Nhập giá trị cho các phần tử mảng.*
- *Xuất giá trị các phần tử mảng (ra màn hình).*
- *Thêm 1 phần tử vào mảng.*
- *Xóa một phần tử ra khỏi mảng.*
- *Tìm kiếm trên mảng.*
- *Sắp xếp mảng.*

Các thao tác trên thực chất là duyệt mảng và xử lý trên từng phần tử mảng nhờ vào cấu trúc lặp một cách tổng quát như sau:

Đối với mảng 1 chiều gồm MAX phần tử:

```
for (i=0; i<MAX; i++)  
  
    xử lý phần tử a[i];
```

Đối với mảng 2 chiều gồm ROWS dòng, COLS cột:

```
for (i=0; i<ROWS; i++)  
  
    for (j=0; j<COLS; j++)  
  
        xử lý phần tử a[i][j];
```

5.1. Nhập giá trị cho các phần tử mảng.

Giả sử đã khai báo mảng 1 chiều các số nguyên gồm MAX=20 phần tử

Hàm nhập giá trị cho các phần tử mảng từ bàn phím

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

Hàm tạo giá trị ngẫu nhiên cho các phần tử mảng trong đoạn [-M ... M]

```
void InitArray(int a[], int n)
{
    //hàm khởi động bộ tạo số ngẫu nhiên khai báo trong <stdlib.h>
    srand(time(0));
    for (int i=0; i<n; i++)
        a[i] = rand()(2*M+1) - M;
}
```

Hàm tạo giá trị ngẫu nhiên tăng dần cho các phần tử mảng, phần tử đầu tiên có trị trong đoạn [x ... y], với $0 < x < y$.

```

void InitArray(int a[], int n, int x, int y)
{
    srand(time(0));

    a[0] = rand()%(b-a+1) + a;

    for (int i=1; i<n; i++)

        a[i] = a[i-1]+ rand()%10;
}

```

5.2. Xuất giá trị các phần tử mảng (ra màn hình).

Hàm xuất giá trị cho các phần tử mảng 1 chiều ra màn hình

```

void Output(const int a[], int n)
{
    for (int i=0; i<n; i++)

        cout << setw(4) <<a [i];

    cout << endl;
}

```

// Hàm xuất giá trị cho các phần tử mảng 2 chiều gồm ROWS dòng, COLS cột ra màn hình

```

void Output(const int a[][COLS], int m, int n)

```

```

{
    for (int i=0; i<m; i++)
    {
        for (int j=0; j<n; j++)

            cout << setw(4) << a[i][j];

        cout << endl;
    }
}

```

5.3. Thêm 1 phần tử vào mảng.

Hàm thêm giá trị x vào cuối mảng

```

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

```

Hàm thêm giá trị x vào mảng tại vị trí có chỉ số pos thứ tự tương quan ban đầu của các phần tử mảng là không quan trọng

```

void Insert(int a[], int &n, int x, int pos)

```

```

{
    a[n] = a[pos];

    a[pos] = x;

    n++;
}

```

Hàm thêm giá trị x vào mảng tại vị trí có chỉ số pos thứ tự tương quan ban đầu của các phần tử mảng không thay đổi

```

void Insert(int a[], int &n, int x, int pos)
{
    for (int i=n; i>pos; i--)

        a[i] = a[i-1];

    a[pos] = x;

    n++;
}

```

5.4.Xóa một phần tử ra khỏi mảng.

Hàm xóa phần tử tại vị trí có chỉ số pos ra khỏi mảng, thứ tự mảng là không quan trọng

```

void Remove(int a[], int &n, int pos)
{

```

```
a[pos] = a[n];  
  
n--;  
  
}
```

Hàm xoá phần tử tại vị trí có chỉ số pos ra khỏi mảng, thứ tự mảng là quan trọng

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

5.5. Tìm kiếm trên mảng.

Hàm tìm kiếm giá trị x, trả về chỉ số của phần tử đầu tiên có trị = x, nếu không tìm thấy thì trả về trị -1 (hoặc n).

Hàm tìm kiếm tuyến tính trên mảng chưa có thứ tự

```
int LinearSearch(const int a[], int n, int x)  
  
{  
  
    for (int i=0; i<n; i++)  
  
        if (a[i]==x)
```

```
        return i;

    return -1;

}
```

Hàm tìm kiếm giá trị x , trả về chỉ số của phần tử đầu tiên có trị= x , trả về trị -1 (hoặc n) nếu không tìm thấy, mảng đã có thứ tự tăng dần

Tìm kiếm tuyến tính

```
int LinearSearch(const int a[], int n, int x)

{

    for (int i=0; i<n && a[i]<x; i++)

        if (a[i]==x) return i;

    return -1;

}
```

Tìm kiếm nhị phân

```
int BinarySearch(const int a[], int n, int x)

{

    int first=0, last=n-1, mid;

    while(first<=last) {

        mid = (first + last) / 2;
```



```
if (a[mid]<x)

    first= mid + 1; // tìm x ở phần nửa sau của mảng

else if (a[mid]>x) last = mid - 1;

else // a[mid]==x

    return i;

}

return -1;

}
```

5.6.Sắp xếp mảng.

Để sắp xếp mảng gồm n phần tử, ta tìm cách đặt (n-1) phần tử vào đúng vị trí của nó theo tiêu chuẩn sắp xếp. Ở lần xếp phần tử thứ i, ta so sánh phần tử này với các phần tử còn lại của mảng và thực hiện đổi chỗ khi cần thiết để thỏa mãn tiêu chuẩn sắp xếp. Cuối cùng ta có một mảng đã được xếp thứ tự.

a. Phương pháp sắp xếp đơn giản (simple sort)

Nội dung phương pháp: Ở bước thứ i (i=0, 1, . . . , n-2) ta so sánh phần tử a[i] với các phần tử a[j] còn lại (j=i+1, . . . , n-1) để xác định phần tử nhỏ nhất, sau đó đổi chỗ phần tử nhỏ nhất này với a[i].

```
void Swap (int &x, int &y)

{
```

```

int z = x;

x = y; y = z;

}

void SimpleSort(int a[], int n)

{

int i, j;

for (i=0; i<n-1; i++)

    for (j=i; j<n; j++)

        if (a[i] > a[j])

            swap(a[i],a[j]);

}

```

b. Phương pháp sắp xếp lựa chọn (selection sort)

```

void SelectionSort(int a[], int n)

{

int i, j, min, tam;

for (i=0; i<n-1; i++)

{

tam = a[i];

```

```
min = i;

for (j=i; j<n; j++)

    if (a[min] > a[j]) min = j;

a[i] = a[min];

a[min] = tam;

}

}
```

c. Phương pháp sắp xếp nổi bọt (bubble sort)

Nội dung phương pháp: Ở bước thứ i ($i=0, 1, \dots, n-2$) ta lần lượt so sánh từng cặp phần tử $a[j], a[j-1]$, với ($j=i+1, \dots, n-1$), sau đó đổi chỗ 2 phần tử này nếu $a[j-1]>a[j]$.

```
void BubbleSort(int a[], int n)

{

    int i, j;

    for (i=0; i<n-1; i++)

        for (j=n-1; j>i; j--)

            if (a[j-1] > a[j])

                swap(a[j-1],a[j]);

}
```

d. Phương pháp sắp xếp chèn (insertion sort)

Nội dung phương pháp: Giả sử dãy con ($a[0] \dots a[i-1]$) đã được sắp. Ở bước thứ i ($i=1, \dots, i < n-1$), ta xác định vị trí thích hợp của $a[i]$ để chèn vào dãy con đã được sắp thứ tự bằng phương pháp tìm kiếm tuần tự từ $a[i]$ trở về $a[0]$.

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

e. Phương pháp sắp xếp “lẻ tăng chẵn giảm”

Nội dung phương pháp: Cách tiến hành giống như thuật toán simple sort, chỉ khác ở **tiêu chuẩn so sánh** để thực hiện việc hoán đổi trị của các phần tử.

6. Câu hỏi

- Nêu lợi ích của việc dùng mảng.
- Nêu cách khai báo và khởi tạo giá trị cho biến mảng một chiều, biến mảng hai chiều.
- Nêu cách truyền tham số mảng cho hàm, cách gọi hàm có tham số mảng.
- Trình bày các thao tác cơ bản trên kiểu mảng (1 chiều):
 - Nhập/Xuất giá trị cho các phần tử mảng
 - Thêm phần tử mới vào mảng
 - Xóa một phần tử trong mảng thỏa tiêu chuẩn P nào đó.
 - Tìm kiếm trên mảng
 - Sắp xếp mảng.

7. Bài tập

Mảng 1 chiều

- 1) Cho trước $n > 0$. Liệt kê tất cả các số nguyên tố $\leq n$ dùng phương pháp sàng Erathosthene.
- 2) Cho trước mảng nguyên kích thước $MAX=100$. Cho trước tiêu chuẩn “P” (Ví dụ: “Là số chẵn”, “Là số dương”, “Là số chính phương”, “Là số nguyên tố”, ...). Xây dựng các hàm sau đây và viết chương trình áp dụng:

- Liệt kê tất cả các phần tử mảng thỏa tiêu chuẩn “P”.
- Đếm số lượng các phần tử mảng thỏa tiêu chuẩn “P”.
- Tính tổng các phần tử mảng thỏa tiêu chuẩn “P”.
- Tính trung bình tổng các phần tử mảng thỏa tiêu chuẩn “P”.
- Cho trước mảng nguyên kích thước $MAX=100$. Viết chương trình thống kê số lần xuất hiện các phần tử trong mảng.

3) Cho trước mảng nguyên có kích thước gồm $MAX=100$.
Viết các hàm sau đây:

- Khởi tạo giá trị cho các phần tử của mảng (nhập từ bàn phím).
- Khởi tạo giá trị ngẫu nhiên cho các phần tử của mảng, mỗi phần tử có trị trong đoạn $[a...b]$, với $0 < a < b$.
- Khởi tạo giá trị ngẫu nhiên cho các phần tử của mảng, sao cho mảng có thứ tự tăng dần.
- Xuất giá trị của các phần tử của mảng ra màn hình.
- Kiểm tra mảng có thứ tự tăng ? giảm ? hay không có thứ tự?
- Đảo ngược thứ tự các phần tử trong mảng.
- Xoay trái/phải các phần tử trong mảng $k > 0$ lần.
- Tìm kiếm giá trị x trong mảng.
- Xóa phần tử đầu tiên trong mảng thỏa tiêu chuẩn “P”.
- Xóa tất cả các phần tử trong mảng thỏa tiêu chuẩn “P”.
- Sắp xếp mảng theo thứ tự “tăng dần”.
- Sắp xếp mảng theo thứ tự “lẻ tăng chẵn giảm”.

- Sắp xếp theo thứ tự tăng dần và loại bỏ các phần tử trùng nhau.
- Đếm số dãy con tăng dần trong mảng và xuất các dãy con này ra màn hình, mỗi dãy con trên 1 dòng.
- Xuất dãy con tăng dần có số lượng phần tử nhiều nhất.
- Xuất dãy con tăng dần có tổng các phần tử lớn nhất.

Viết chương trình áp dụng các hàm đã xây dựng ở trên.

- 4) Cho trước mảng nguyên có kích thước gồm MAX=100. Viết chương trình sắp xếp mảng theo thứ tự tăng, đồng thời loại bỏ các phần tử trùng nhau.
- 5) Viết chương trình trộn 2 mảng nguyên đã có thứ tự tăng/giảm dần, thành mảng nguyên mới cũng có thứ tự tăng/giảm dần.
- 6) Viết hàm vẽ biểu đồ đứng, và hàm vẽ biểu đồ ngang. Viết chương trình áp dụng.

Ví dụ, với mảng nguyên $\text{int } a[5]=\{4, 7, 10, 6, 3\}$ ta có:

Biểu đồ ngang

Biểu đồ đứng

* * * *

*

* * * * * *

*

* * * * * * * * * *

*

* * * * *

* *

* * *

* * *

```

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

```

Mảng 2 chiều

- 1) Viết chương trình in ma phương bậc lẻ.
- 2) Viết chương trình in mảng 2 chiều kích thước $MAX * MAX$ theo thứ tự xoắn ốc sau:

Ví dụ, với $MAX = 4$:

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

- 3) Tương tự như bài trên, viết chương trình sắp xếp mảng 2 chiều theo thứ tự xoắn ốc với các phần tử mảng có trị ngẫu nhiên.
- 4) Viết chương trình xác định các phần tử “yên ngựa” (nếu có) của mảng 2 chiều cho trước. Phần tử “yên ngựa” có giá trị **min dòng và max cột** hoặc **max dòng và min cột**.

CHƯƠNG 6. CON TRỎ (POINTER)

1. Khái niệm

- Con trỏ (Pointer) là kiểu dữ liệu đặc biệt, có giá trị là địa chỉ vùng nhớ của một đối tượng (biến, hàm).
- Tương ứng với mỗi kiểu dữ liệu sẽ có một kiểu biến trỏ riêng biệt. VD ta có con trỏ char*, int*, float*, double*, . . . chứa địa chỉ của biến char, int, float, double.
- Tùy theo hệ máy, kích thước của biến trỏ là 2 bytes (hệ máy PC) hoặc 4 bytes (hệ máy tính lớn).
- Biến trỏ cho phép truy xuất đối tượng một cách gián tiếp, i.e. thm chiếu đến 1 đối tượng khác thông qua địa chỉ của nó.
- Việc cấp phát động cho mảng được thực hiện thông qua con trỏ.
- Để nắm bắt kiểu con trỏ, cần phân biệt **nội dung** của vùng nhớ và **địa chỉ** của nó.

2. Khai báo biến con trỏ

```
<type> * <tên biến trỏ>;
```

trong đó:

<type> là kiểu dữ liệu của biến mà con trỏ đang trỏ đến,.

<tên biến trỏ> là danh hiệu hợp lệ.

Như vậy, <type> * là kiểu con trỏ.

Cũng như các kiểu dữ liệu khác, ta có thể khai báo đồng thời khởi tạo giá trị cho biến trỏ như sau:

```
int x;
```

```
int *px = &x; // px chứa địa chỉ của biến x.
```

Chú ý:

```
int *px, x, *py, y;
```

3. Truy xuất biến trở

Sau khi khai báo biến trở, ta có thể truy xuất nó thông qua tên biến như một biến thông thường. Khi đó, ta được giá trị (nội dung) của biến trở là địa chỉ của một vùng nhớ nào đó. Nếu muốn truy xuất nội dung của vùng nhớ mà biến trở đang trỏ đến, ta dùng toán tử * (gọi là “khử tham chiếu” – dereference) đặt trước tên biến trở như sau:

```
*<tên biến trở>
```

Giả sử có các khai báo sau:

```
int x =5, y= 7;
```

```
int *px, *py;
```

Ta có thể gán trị của px, py như sau:

```
px = &x;
```

```
py = &y;
```

Các biến x, y có thể được truy xuất gián tiếp như sau:

```
*px = 2; // tương đương với câu lệnh gán x = 2;
```

```
*py = 3; // tương đương với câu lệnh gán y = 3;
```

Chú ý:

Cần phân biệt

```
*px = *py; //gán trị của vùng nhớ mà py đang trỏ đến cho vùng  
nhớ mà px đang trỏ đến.
```

với

```
px = py; //sau lệnh này px và py cùng trỏ đến cùng 1 vùng nhớ.
```

Con trỏ NULL là con trỏ không chứa địa chỉ của bất kỳ vùng nhớ nào. Nên khởi tạo giá trị NULL hoặc địa chỉ của vùng nhớ nào đó cho biến trỏ lúc khai báo. Cần lưu ý, việc truy xuất vùng nhớ thông qua con trỏ NULL là lỗi về cú pháp.

```
int *py = NULL; // py không trỏ đến bất kỳ vùng nhớ nào.
```

Con trỏ void * là con trỏ đa năng, và tương thích với mọi kiểu dữ liệu mà 1 biến trỏ trỏ đến, i.e. ta có thể gán giá trị của con trỏ thuộc một kiểu bất kỳ nào đó cho con trỏ void *. Không được phép thực hiện các phép tính số học trên con trỏ void*.

4. Số học con trỏ

Ngoài **phép gán trị** của 1 biến trỏ cho biến trỏ khác cùng kiểu với nó, ta có thể thực hiện các phép toán số học sau trên biến trỏ:

Phép cộng con trỏ ptr với một số nguyên N sẽ cho kết quả địa chỉ vùng nhớ cách con trỏ ptr N vị trí như sau: (vẽ hình)

Phép trừ 2 biến trỏ cùng kiểu ptr1 và ptr2 sẽ cho kết quả khoảng cách (số phần tử) giữa 2 biến trỏ trên như sau: (vẽ hình)

Phép so sánh 2 con trỏ cùng kiểu với nhau được thực hiện dựa trên vị trí vùng nhớ tương ứng với 2 biến trỏ (và kết quả trả về là trị 0 hoặc 1).

5. Liên hệ giữa con trỏ và mảng

Do tên biến mảng là 1 giá trị hằng địa chỉ của phần tử đầu tiên của mảng, nên ta có thể gán giá trị địa chỉ này cho con trỏ có kiểu nền cùng kiểu với kiểu cơ sở của biến mảng.

Giả sử có các khai báo sau:

```
int a[5];
```

```
int *pa=a;
```

Khi đó, ta có thể truy xuất các phần tử mảng và địa chỉ của chúng như sau:

$a[0] \Leftrightarrow *(a+0) \Leftrightarrow *(pa+0) \Leftrightarrow pa[0]$ $\&a[0] \Leftrightarrow a+0 \Leftrightarrow pa+0 \Leftrightarrow \&pa[0]$

$a[1] \Leftrightarrow *(a+1) \Leftrightarrow *(pa+1) \Leftrightarrow pa[1]$ $\&a[1] \Leftrightarrow a+1 \Leftrightarrow pa+1 \Leftrightarrow \&pa[1]$

$a[2] \Leftrightarrow *(a+2) \Leftrightarrow *(pa+2) \Leftrightarrow pa[2]$ $\&a[2] \Leftrightarrow a+2 \Leftrightarrow pa+2 \Leftrightarrow \&pa[2]$

$a[3] \Leftrightarrow *(a+3) \Leftrightarrow *(pa+3) \Leftrightarrow pa[3] \quad \&a[3] \Leftrightarrow a+3 \Leftrightarrow pa+3 \Leftrightarrow \&pa[3]$

$a[4] \Leftrightarrow *(a+4) \Leftrightarrow *(pa+4) \Leftrightarrow pa[4] \quad \&a[4] \Leftrightarrow a+4 \Leftrightarrow pa+4 \Leftrightarrow \&pa[4]$

6. Con trỏ đa cấp

Bản thân biến trỏ cũng có địa chỉ, do đó ta có thể chứa địa chỉ của nó trong 1 biến trỏ khác. Ta gọi biến trỏ này là con trỏ trỏ đến con trỏ, hay con trỏ 2 cấp. Số lượng dấu ‘*’ xác định cấp của 1 biến trỏ. Ta có con trỏ 2 cấp, con trỏ 3 cấp, . . .

Con trỏ 2 cấp có liên quan mật thiết với mảng 2 chiều.

Giả sử có các khai báo sau:

```
int a[2][3];

int **ppa = new int*[2];

ppa[0] = a[0];

ppa[1] = a[1];
```

Khi đó, ta có thể truy xuất các phần tử mảng như sau:

$a[0][0] \Leftrightarrow *((*(a+0)+0) \Leftrightarrow *((*(ppa+0)+0) \Leftrightarrow ppa[0][0]$

$a[0][1] \Leftrightarrow *((*(a+0)+1) \Leftrightarrow *((*(ppa+0)+1) \Leftrightarrow ppa[0][1]$

$a[0][2] \Leftrightarrow *((*(a+0)+2) \Leftrightarrow *((*(ppa+0)+2) \Leftrightarrow ppa[0][2]$

$a[1][0] \Leftrightarrow *((*(a+1)+0) \Leftrightarrow *((*(ppa+1)+0) \Leftrightarrow ppa[1][0]$

$a[1][1] \Leftrightarrow *((a+1)+1) \Leftrightarrow *((ppa+1)+1) \Leftrightarrow ppa[1][1]$

$a[1][2] \Leftrightarrow *((a+1)+2) \Leftrightarrow *((ppa+1)+2) \Leftrightarrow ppa[1][2]$

Để truy xuất địa chỉ các phần tử mảng:

$\&a[0][0] \Leftrightarrow \&(a+0) \Leftrightarrow \&(ppa+0)$

$\&a[0][1] \Leftrightarrow \&(a+0) \Leftrightarrow \&(ppa+1)$

$\&a[0][2] \Leftrightarrow \&(a+0) \Leftrightarrow \&(ppa+2)$

$\&a[1][0] \Leftrightarrow \&(a+1) \Leftrightarrow \&(ppa+0)$

$\&a[1][1] \Leftrightarrow \&(a+1) \Leftrightarrow \&(ppa+1)$

$\&a[1][2] \Leftrightarrow \&(a+1) \Leftrightarrow \&(ppa+2)$

7. Truyền tham số con trỏ cho hàm

Trong phần khai báo và định nghĩa hàm, ta khai báo kiểu dữ liệu con trỏ là `<type> *`.

Còn trong lời gọi hàm, ta phải cung cấp biểu thức có trị là địa chỉ của vùng nhớ cùng kiểu với kiểu của tham số biến trỏ tương ứng.

Ví dụ: hàm `Swap(int*, int*);` // có 2 tham số là biến trỏ

8. Mảng các con trỏ

Kiểu phần tử của biến mảng có thể là kiểu con trỏ. Khi đó ta sẽ có một mảng các con trỏ, và ta có thể xem các biến có địa chỉ chứa trong các phần tử mảng con trỏ là một mảng, nhưng có vùng nhớ không liên tục. (Vẽ hình)

9. Từ khóa const với con trỏ

Ta đã biết một công dụng của từ khóa const trong việc định nghĩa một biến hằng. Khi ta khai báo `const int MAX = . . . ;`, thì TBD sẽ cấp phát vùng nhớ cho hằng MAX (ở đây là 2 bytes) và không cho phép USER thay đổi giá trị của MAX trong chương trình.

Tương tự, các khai báo sau:

```
// px và *px có thể thay đổi giá trị.
```

```
<datatype>* px;
```

```
// px là con trỏ trỏ đến vùng nhớ có giá trị không đổi, i.e. px có thể thay đổi, *px thì không được phép thay đổi.
```

```
const <datatype>* px;
```

```
// px là con trỏ hằng, i.e. *px có thể thay đổi, px thì không được phép thay đổi.
```

```
<datatype>* const px;
```

```
// px là con trỏ hằng trỏ đến vùng nhớ có giá trị không đổi.
```

```
const <datatype>* const px;
```

10. Cấp phát động

Cấp phát động là cấp phát vùng nhớ lúc thực hiện chương trình. Còn cấp phát vùng nhớ lúc biên dịch được gọi là cấp phát tĩnh.

Vùng nhớ của các đối tượng (biến) cấp phát động sẽ được đặt tại HEAP. Việc cấp phát động được thực hiện nhờ vào các hàm cấp phát bộ nhớ sau:

Trong C, dùng các hàm `malloc(. . .)`, `calloc(. . .)`, `realloc(. . .)`, . . . được khai báo trong `<alloc.h>`, `<stdlib.h>`

// `size_t` là kiểu dữ liệu định nghĩa trong `<stdlib.h>` và tương đương với một `unsigned int`.

```
void* malloc(size_t size);  
  
void* calloc(size_t nitems, size_t size);  
  
void* realloc(void * ptr, size_t size);
```

Trong C++, dùng toán tử `new` :

// xin cấp phát vùng nhớ trên HEAP có kích thước = `sizeof(<type>)`

```
<biến> = new <type>;  
  
// xin cấp phát vùng nhớ trên HEAP kích thước =  
sizeof(<type>)*n  
  
<biến> = new <type>[n];
```


Khi không còn sử dụng các vùng nhớ đã cấp phát động, ta phải thu hồi chúng, để có thể sử dụng vào việc khác. Nếu không làm như vậy thì bộ nhớ sẽ nhanh chóng cạn kiệt. Việc thu hồi các vùng nhớ đã cấp phát động được thực hiện nhờ vào hàm sau:

Trong C, dùng hàm `free(ptr)`, với `ptr` là biến trỏ chỉ đến vùng nhớ đã được cấp phát động bằng các hàm `malloc(...)`, `calloc(...)`, `realloc(...)`

Trong C++, dùng toán tử `delete`:

```
delete <biến>;
```

```
delete [] <biến>;
```

// Chương trình cấp phát động mảng một chiều

```
#include <iostream.h>

#include <iomanip.h>

#include <stdlib.h>

#include <alloc.h>

void randomInit( int* a, int n );

void output( const int* a, int n );

void main() {

    int* a;

    int n;
```

```

    cout << "Cho biết số phần tử ? : "; cin >> n;

    // a = (int* ) calloc( n, sizeof( int ) );

    a = new int [ n ];

    randomInit( a, n );

    output( a, n );

    // free( a );

    delete [] a;

}

void randomInit( int* a, int n ) {

    for ( int i = 0; i < n; i++ )

        a[ i ] = rand()% 100;

}

void output( const int* a, int n ) {

    for ( int i = 0; i < n; i++ )

        cout << setw( 4 ) << a[ i ];

    cout << endl;

```

```

// Chương trình cấp phát động mảng 2 chiều

```

```

#include <iostream.h>

```

```

#include <iomanip.h>

#include <stdlib.h>

#include <alloc.h>

void randomInit( int** a, int m, int n );

void output( const int** a, int m, int n );

void main() {

    int** a;

    int m, n;

    cout << "Cho biết số dòng ? : "; cin >> m;

    cout << "Cho biết số cột ? : "; cin >> n;

    a = new int [ m ];

    for ( int i = 0; i < m; i++ )

        a[ i ] = new int [ n ];

    randomInit( a, m, n );

    output( a, m, n );

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

        delete [] a[ i ];

    delete [] a;

```

```

}

void randomInit( int** a, int m, int n )
{
    for ( int i = 0; i < m; i++ )
        for ( int j = 0; j < n; j++ )
            a[ i ][ j ] = rand()% 100;
}

void output( const int** a, int m, int n )
{
    for ( int i = 0; i < m; i++ )
    {
        for ( int j = 0; j < n; j++ )
            cout << setw( 4 ) << a[ i ][ j ];

        cout << endl;
    }
}

```

11. Con trỏ hàm

Trong NNLT “C/C++”, tên hàm là địa chỉ vùng nhớ của chỉ thị đầu tiên của hàm và do đó ta có thể gán giá trị địa chỉ này cho 1 biến

trở có kiểu nền cùng kiểu với kiểu giá trị trả về của hàm. Ta gọi con trở này là con trở hàm. Ta có thể gọi thực hiện một cách gián tiếp một hàm nào đó thông qua con trở hàm. Mặt khác, một hàm nào đó có thể được dùng làm tham số cho một hàm khác nhờ vào con trở hàm.

Con trở hàm được khai báo như sau:

```
<type> (* <tên con trở hàm>) ([Danh sách các tham số]);
```

Giả sử có các khai báo sau:

```
int a, b;
```

```
void swap( int* px, int* py );
```

```
void (* pf) ( int *, int* );
```

Khi đó ta có thể gọi thực hiện hàm swap một cách gián tiếp như sau:

```
pf = swap;
```

```
pf( &a, &b );
```

12. Con trở và chuỗi kí tự

Chuỗi (string) là một dãy các kí tự liên tiếp trong bộ nhớ được kết thúc bằng kí tự NUL (‘\0’). Như vậy để sử dụng biến chuỗi chứa MAX kí tự, ta khai báo như sau: `char s[MAX+1];`

Ta cũng có thể khai báo biến chuỗi như sau:

```
char* s;
```

Có thể khởi tạo biến chuỗi như sau:

```
char s[] = "Hello C++";
```

Có thể nhập/xuất chuỗi kí tự bằng lệnh `cin` (dùng kèm với `>>`) và `cout` (dùng kèm với `<<`). Tuy nhiên cần lưu ý, lệnh `cin >> s`; chỉ cho phép nhập vào chuỗi kí tự không có khoảng trắng.

Hàm nhập chuỗi của đối tượng `cin`:

// đọc các kí tự từ `cin` vào `s`, kể cả kí tự khoảng trắng.

```
getline( char* s, int size, char delim='\\n' );
```

```
read( char* s, int size ); // cin.read( s, 5 );
```

// cho phép đọc từng kí tự từ `cin` vào trong `ch` và trả về trị 1. Hàm trả về trị 0 nếu gặp kí tự `'\\n'`.

```
get( char ch );
```

Hàm xuất chuỗi của đối tượng `cout`:

```
put( char ); // cout.put( ch ).put( ch );
```

```
write( const char* s, int size ); // cout.write( s, 1 ).write( s+1, 2 );
```

Một số hàm thông dụng khai báo trong `<string.h>` cho phép thao tác, xử lý chuỗi kí tự:

```
size_t strlen( const char* s );
```

```
int strcmp( const char* s1, const char* s2 );

int strcmpi( const char* s1, const char* s2 );

char* strcpy( char* dest, const char* src );

char* strcat( char* dest, const char* src );

...
```

13. Ứng dụng con trỏ

- Sắp xếp mảng các con trỏ
- Danh sách liên kết
- Cấp phát mảng động

14. Sơ lược về kiểu tham chiếu (Reference) - Chỉ có trong C++.

NNLT C++ cung cấp khả năng tham chiếu đến địa chỉ vùng nhớ của biến đã tồn tại trước đó. Về bản chất, tham chiếu là bí danh của một đối tượng (biến) xác định trong chương trình.

Sau khi khai báo 1 biến, ta có thể khai báo biến tham chiếu đến biến đó như sau:

```
int x;

int &rx = x; // rx là biến tham chiếu đến biến x.
```

Sau câu lệnh trên, ta có thể xem rx là tên gọi khác (bí danh) của biến x.

Chú ý, biến kiểu tham chiếu phải tham chiếu đến một biến đã tồn tại, i.e. biến đã khai báo trước. Không thể có khai báo biến tham chiếu như sau:

```
int &rx; // sai !! rx là bí danh của biến nào ???
```

Ta thường dùng kiểu tham chiếu trong việc truyền tham số cho hàm. Các tham số thực tương ứng (theo vị trí) có thể bị thay đổi giá trị ngay bên trong hàm.

Ví dụ: hàm `swap(int &, int &);`

Hằng tham chiếu khác có thể tham chiếu đến một biến hay một hằng trực tiếp nào đó. Ví dụ, ta có các khai báo sau:

```
int x = 5, y = 7;

const int &rx = 123; // ok

const int &ry = y;

ry++; // sai
```

Tuy nhiên hằng tham chiếu khác với biến tham chiếu ở chỗ: ta không thể dùng hằng tham chiếu để làm thay đổi vùng nhớ mà nó tham chiếu đến.

15. Bài tập

1) Giả sử có các khai báo sau:

```
int i1 = 11,
```



```
i2 = 22;
```

```
double d1 = 3.45,
```

```
d2 = 6.78;
```

- Viết các khai báo sao cho biến p1 và p2 có giá trị của nó là địa chỉ trong bộ nhớ, nơi mà một giá trị double có thể chứa.
 - Viết câu lệnh để gán địa chỉ của d1 cho p1, hoặc giải thích tại sao điều này không thể thực hiện.
 - Viết câu lệnh để gán địa chỉ của i2 cho p2, hoặc giải thích tại sao điều này không thể thực hiện.
 - Viết các khai báo để khởi tạo biến ptr1 và ptr2 trỏ đến i1 và i2 tương ứng.
 - Viết câu lệnh để p1 và p2 trỏ đến cùng một địa chỉ.
 - Viết câu lệnh để chép giá trị được chứa tại địa chỉ mà ptr2 trỏ đến vào địa chỉ mà ptr1 trỏ đến.
 - Viết các câu lệnh sử dụng p1 và p2 để hoán đổi giá trị của d1 và d2.
- 2) Viết chương trình C++ thực hiện từng bước các yêu cầu sau:
- Khai báo biến con trỏ kiểu char có tên là charPtr.
- 3) Cấp phát một vùng nhớ nặc danh và cho charPtr trỏ đến đó.
- 4) Nhập một ký tự và chứa nó vào vùng nhớ nặc danh.
- 5) Hiển thị nội dung trong vùng nhớ nặc danh.
- 6) Nếu nội dung của vùng nhớ nặc danh là ký tự hoa thì chuyển thành ký tự thường và hiển thị kết quả ra màn hình.

- 7) Viết chương trình C++ thực hiện từng bước các yêu cầu sau:
- Khai báo biến con trỏ kiểu double có tên là doublePtr.
 - Cấp phát vùng nhớ nặc danh cho một mảng có n (nhập từ bàn phím) phần tử kiểu double và chứa địa chỉ của nó vào doublePtr.
 - Nhập giá trị cho tất cả các phần tử trong mảng.
 - Tính và hiển thị trung bình cộng các giá trị trong mảng.
 - Giải phóng vùng nhớ đã cấp phát cho mảng.
 - Hiển thị địa chỉ và giá trị của 3 phần tử đầu tiên trong mảng có vùng nhớ vừa được giải phóng.
- 8) Viết chương trình khởi tạo một con trỏ trỏ đến mảng unsigned int có 20 phần tử. Sau đó gán giá trị cho các phần tử trong mảng là những số chẵn bắt đầu từ 2, hiển thị các giá trị này ra màn hình theo nhiều cách khác nhau (nếu có thể) thành 4 dòng, mỗi dòng có 5 phần tử.
- 9) Viết chương trình có tên là binary để khi thực hiện tại dấu nhắc lệnh
- 10) C:\>binary DecimalValue.↵
- 11) chương trình sẽ tính và hiển thị giá trị nhị phân ứng với giá trị thập phân đã nhập. Trong đó, DecimalValue là giá trị thập phân 2 byte (−32768 đến 32767).
- 12) Giá trị trung bình của một dãy có n số là một số thực và được định nghĩa là giá trị mà nó có n/2 giá trị lớn hơn nó, và n/2 giá trị nhỏ hơn nó. Viết chương trình có tên là median để khi thực hiện tại dấu nhắc lệnh
- 13) C:\>median FileName.↵

- 14) chương trình sẽ tính và hiển thị giá trị trung bình của các giá trị trong tập tin FileName, nhưng nếu gõ lệnh
- 15) C:\>median↵
- 16) chương trình sẽ tính và hiển thị giá trị trung bình của n ($2 \leq n \leq 10$) giá trị được nhập từ bàn phím.
- 17) Viết chương trình để thực hiện sao chép tập tin File1 thành tập tin File2:
- 18) C:\>copy File1 File2↵
- 19) Viết chương trình để khi thực hiện lệnh
- 20) C:\>page File↵
- 21) thì nội dung tập tin được chỉ định sẽ hiển thị lên màn hình theo từng trang (23 dòng), người sử dụng có thể ấn phím bất kỳ để xem trang kế tiếp.
- 22) Viết chương trình xử lý chuỗi kí tự bao gồm các chức năng sau: (Chú ý: dùng con trỏ để cài đặt và không được dùng hàm thư viện)
 - Tính chiều dài của chuỗi nhập.
 - Sao chép 2 chuỗi với nhau.
 - So sánh 2 chuỗi với nhau.
 - Tìm một kí tự trong chuỗi nhập.
 - Tìm chuỗi con trong chuỗi nhập.
 - Thêm chuỗi con vào trong chuỗi nhập tại vị trí k.
 - Xoá chuỗi con trong chuỗi nhập.
 - Loại bỏ các khoảng trắng thừa (kí tự Space, Tab) trong chuỗi nhập.
 - Chuẩn hóa chuỗi nhập.
 - Đảo ngược chuỗi nhập.

- Kiểm tra 2 chuỗi nhập có gồm cùng các kí tự hay không ?
- Kiểm tra chuỗi nhập có đối xứng hay không ?
- Kiểm tra chuỗi nhập có tuần hoàn hay không ?
- Đếm tần số xuất hiện của các kí tự trong chuỗi nhập.
- Đếm số từ trong chuỗi nhập.
- Đếm số kí tự, số từ và số dòng trong chuỗi nhập.
- Chuyển từ cuối cùng thành từ đầu tiên trong chuỗi nhập.

23) Viết chương trình khai báo chuỗi có tên là last_first có nội dung là “Smith, Bill”, sau đó tách tên và họ của chuỗi này rồi kết hợp chúng lại để thành “Bill Smith” và gán cho chuỗi first_last. Hiển thị hai chuỗi ra màn hình.

24) Định nghĩa một hàm có ba tham số, mỗi tham số là một chuỗi ký tự gồm: tên, tên lót, và họ. Hàm này trả về một chuỗi chứa ba tham số trên theo thứ tự họ, tên, và ký tự đầu của tên lót. Ví dụ, nếu ba tham số lần lượt có nội dung là “John”, “Quincy”, và “Doe” thì hàm trả về chuỗi “Doe, John Q.”. Viết chương trình áp dụng.

25) Tương tự như câu 2, nhưng định nghĩa hàm chỉ có một tham số. Ví dụ, nếu tham số có nội dung là “John Quincy Doe” thì hàm trả về chuỗi “Doe, John Q.”.

26) Định nghĩa một hàm nhận vào số thứ tự của một tháng và trả về tên của tháng đó. Viết chương trình áp dụng.

27) Định nghĩa một hàm nhận vào tên của một tháng và trả về số thứ tự của tháng đó. Viết chương trình áp dụng.

28) Định nghĩa hai hàm: hàm thứ nhất chuyển chuỗi ký tự bất kỳ thành chuỗi ký tự thường, hàm thứ hai chuyển thành chuỗi ký tự hoa. Viết chương trình áp dụng.

- 29) Viết chương trình có định nghĩa một hàm với tên là `replace_all`, biểu thức `replace_all(str, substring, newSubstring)` trả về một chuỗi mà tất cả những xuất hiện của `substring` trong `str` được thay thế bằng `newSubstring`.
- 30) Một chuỗi được gọi là đối xứng (palindrome) nếu nó không thay đổi khi thứ tự của các ký tự được đảo ngược. Ví dụ: `madam`, `463364`, `ABLE WAS I ERE I SAW ELBA` là các chuỗi đối xứng. Viết hàm nhận vào một chuỗi và cho biết chuỗi này có đối xứng không. Viết chương trình áp dụng.
- 31) Viết chương trình có định nghĩa một hàm nhận vào hai chuỗi và cho biết chuỗi thứ nhất có phải là một hoán vị của các ký tự trong chuỗi thứ hai hay không. Ví dụ chuỗi `“dear“` là một hoán vị của chuỗi `“read“` hay `“dare“`.
- 32) Không sử dụng các hàm thư viện xử lý chuỗi của C++. Giả sử chiều dài chuỗi không lớn hơn 80 ký tự và chuỗi được nhập từ bàn phím. Hãy viết chương trình có định nghĩa các hàm cho mỗi yêu cầu sau:
- Sao chép chuỗi `s1` sang chuỗi `s2`. Hiện thị hai chuỗi lên màn hình.
 - Lấy `n` (giá trị nhập) ký tự đầu tiên của chuỗi nguồn, hiện thị chuỗi nguồn và chuỗi kết quả ra màn hình.
 - Lấy `n` (giá trị nhập) ký tự cuối cùng của chuỗi nguồn, hiện thị chuỗi nguồn và chuỗi kết quả ra màn hình.
 - Chèn một chuỗi vào chuỗi khác tại vị trí được chỉ định (nếu vị trí chỉ định không hợp lệ thì chương trình sẽ không thực hiện thao tác này). Hiện thị chuỗi kết quả ra màn hình.

- Xoá một số ký tự trong một chuỗi tại vị trí bắt đầu và số ký tự cần xoá được chỉ định (cần kiểm tra tính hợp lệ của hai tham số này). Hiển thị chuỗi còn lại ra màn hình.

33) Viết chương trình đọc vào một chuỗi tối đa 80 ký tự và định nghĩa các hàm thực hiện các yêu cầu sau:

- Hiển thị số lần mà ký tự đó xuất hiện trong chuỗi. Các ký tự được sắp theo thứ tự alphabet. Ví dụ nhập Le Van B, thì hiển thị ký tự 'a' có một lần, 'b' có một lần, 'c' không có, ...
- Hiển thị số lần mà các từ một ký tự, hai ký tự, ba ký tự, ... xuất hiện trong chuỗi. Giả sử chiều dài tối đa của một từ là 10 ký tự.

CHƯƠNG 7. KIỂU STRUCT

1. Khái niệm

Khác với kiểu mảng, struct là kiểu dữ liệu có cấu trúc bao gồm nhiều thành phần có thể thuộc nhiều kiểu dữ liệu khác nhau.

Kiểu struct cho phép lưu trữ thông tin về một đối tượng với đầy đủ các thuộc tính của nó trong một biến. Mỗi thuộc tính là một biến thành phần. Kiểu dữ liệu của các biến này có thể là kiểu cơ sở hay kiểu mảng, thậm chí một kiểu struct khác.

2. Khai báo và truy xuất biến struct trong 'C'

Cách 1:

```
struct <tên cấu trúc>{  
...  
<kiểu> <tên biến thành phần>;  
...  
};  
  
struct <tên cấu trúc> <danh sách biến cấu trúc>;
```

Cách 2:

```
struct <tên cấu trúc>{  
...  
<kiểu> <tên biến thành phần>;
```

```
...  
} <danh sách biến kiểu struct>;
```

Cách 3:

```
struct {  
...  
<kiểu> <tên biến thành phần>;  
...  
} <danh sách biến kiểu struct>;
```

Mỗi khai báo struct là sự mô tả một kiểu dữ liệu mới, và chỉ là sự mô tả cấu trúc khung chung cho mọi biến thể hiện của nó. Mỗi biến kiểu struct sau khi đã khai báo đều bao gồm bên trong nó các thành phần thuộc tính được mô tả trong phần khai báo kiểu struct. Các thành phần thuộc tính của mỗi biến thể hiện có thể được truy xuất nhờ vào toán tử dấu chấm ‘.’ như sau:

```
<tên biến struct>.<tên biến thành phần>
```

Dấu chấm ‘.’ được gọi là toán tử truy nhập thành phần cấu trúc. Nếu bản thân thành phần cấu trúc cũng lại là một kiểu struct khác thì vẫn áp dụng toán tử này để xác định thành phần sâu hơn trong các cấu trúc thành phần.

```
<tên biến struct>.<tên biến struct thành phần>.<tên biến tp>
```


3. Lệnh typedef

Cho phép đặt tên mới cho một kiểu dữ liệu đã khai báo và định nghĩa trước:

```
typedef <tên kiểu đã có> <tên mới>;
```

Ví dụ:

```
typedef unsigned char byte;
```

Sau câu lệnh này, byte là tên gọi khác của kiểu unsigned char.

Khi đó, 2 khai báo sau đây là tương đương nhau:

```
byte c;
```

và

```
unsigned char c;
```

Lệnh typedef thường được sử dụng để định nghĩa các kiểu dữ liệu phức hợp thành một tên duy nhất. Ví dụ: lệnh

```
typedef int * PTR_INT;
```

định nghĩa kiểu dữ liệu con trỏ nguyên. Sau này khi cần khai báo một con trỏ nguyên, ta chỉ cần viết:

```
PTR_INT px; // tương đương với khai báo int * px;
```

Ta có thể đơn giản cách viết khai báo biến cấu trúc bằng cách sử dụng lệnh typedef như sau:

```
typedef struct {  
  
    ...  
  
    <kiểu> <tên biến thành phần>;  
  
    ...  
  
} <tên mới>;
```

Ví dụ: ta định nghĩa phân số như là kiểu cấu trúc

```
typedef struct {  
  
    int tuso, mauso;  
  
} PHANSO;
```

Sau đó, ta định nghĩa các biến kiểu PHANSO

```
PHANSO ps1, ps2;
```

4. Kiểu enum

- Cho phép dùng danh hiệu để đặt tên các giá trị nguyên một cách gợi nhớ. Các trị nguyên trong enum được bắt đầu từ trị 0, trừ khi xác định rõ trị khác. Các trị nguyên kế tiếp có giá trị bằng trị nguyên trước đó cộng thêm một.

Khai báo

```
enum <tên kiểu enum> {Danh sách các trị nguyên};
```

Ví dụ:

```
enum Weekdays {Sunday, Monday, Tuesday, Wednesday,  
Thursday, Friday, Saturday};
```

Khi đó: Sunday có trị = 0, Monday có trị = 1, Tuesday có trị = 2,

...

5. Mảng các struct

```
const MAX = 20;  
  
const MONHOC = 3;  
  
typedef struct {  
  
int d, m, y;  
  
} NGAYSINH;  
  
typedef struct {  
  
char * ho;  
  
char * ten;  
  
NGAYSINH ngsinh;  
  
float diem[MONHOC];  
  
// . . . . .  
  
} SINHVIEN;
```

```
SINHVIEN sv[MAX];
```

6. Con trỏ trỏ đến biến kiểu struct

```
typedef struct {  
    ...  
    <kiểu> <tên biến thành phần>;  
    ...  
} <tên mới>;  
  
<tên mới> * <tên biến trỏ>;
```

Ví dụ:

```
typedef struct {  
    int tuso, mauso;  
} PHANSO;  
  
PHANSO *pps;
```

Ta truy xuất các biến thành phần của pps như sau:

```
(*pps).tuso  
  
(*pps).mauso
```

hay

```
pps->tuso
```

```
pps->mauso
```

7. Truyền tham số kiểu struct cho hàm

Truyền bằng tham trị: chương trình sẽ chậm đi do phải tốn thời gian sao chép giá trị của biến cấu trúc dùng làm tham số thực cho tham số hình thức.

Truyền bằng tham biến hay tham trở: dù kích thước của biến cấu trúc có lớn thì địa chỉ của biến vẫn chỉ là kích thước của địa chỉ vùng nhớ.

8. Kiểu union

Một biến kiểu union cũng bao gồm nhiều thành phần giống như một biến cấu trúc, nhưng khác nhau ở chỗ: các trường thuộc tính trong biến cấu trúc được cấp phát các vùng nhớ khác nhau, còn các trường của biến union được cấp phát chung vùng nhớ. Độ dài của biến union bằng độ dài của thành phần dài nhất trong biến.

Khai báo biến union giống như khai báo biến struct

Ví dụ:

```
typedef union  
{  
  
    unsigned int n;  
  
    unsigned char ch[2];
```

```
}VAL;
```

```
VAL x;
```

```
x.n=0x1B1A;
```

khi đó:

```
ch[0] = 1A và ch[1]=1B
```

9. Bài tập

- 1) Cho trước một thời điểm giờ, phút, giây. Viết chương trình tính thời điểm 1 giây sau, 1 giây trước.
- 2) Viết chương trình tính khoảng cách (số giây) giữa 2 thời điểm cho trước.
- 3) Cho trước một bộ ngày, tháng, năm. Viết chương trình tính ngày hôm sau, ngày hôm trước.
- 4) Viết chương trình tính khoảng cách (số ngày) giữa 2 bộ ngày, tháng, năm cho trước.
- 5) Viết chương trình thực hiện các phép tính (+, -, *, /) trên phân số.
- 6) Viết chương trình thực hiện các phép tính (+, -, *, /) trên số phức.
- 7) Viết chương trình thực hiện phép cộng, và phép nhân 2 ma trận.

CHƯƠNG 8. File

1. Giới thiệu chung

File là cơ chế cho phép lưu trữ dữ liệu một cách lâu dài, với số lượng lớn (về mặt lý thuyết là không hạn chế).

C++ xem File chỉ đơn thuần là một dãy các byte. File được đánh dấu kết thúc bằng kí hiệu đặc biệt (CTRL+Z cho DOS và CTRL+D cho UNIX, . . .)

Để có thể làm việc với File đĩa, trước hết ta phải liên kết file với dòng (stream) nhập/xuất bằng cách tạo ra đối tượng File thuộc lớp: `fstream` (dùng cho việc Vào/Ra File), `ofstream` (dùng cho Ghi ra File), `ifstream` (dùng cho Đọc File). Các lớp này được khai báo trong tập tin tiêu đề `<fstream.h>`.

Có 2 loại Vào/Ra file đĩa cơ bản trong C++: **Vào/Ra file định dạng** (formatted file I/O), và **Vào/Ra file nhị phân** (binary file I/O)

Quy trình thao tác file: mở file – kiểm tra – xử lý file – đóng file

2. Thao tác trên kiểu file

2.1. Mở File (liên kết kênh nhập/xuất với file đĩa)

Dùng hàm thiết lập: `<fstream> filevar(<filename>);`

Hoặc dùng hàm thành phần `open` của đối tượng luồng nhập/xuất:

```
<fstream> filevar;
```

```
filevar.open( <filename>, <i/o mode> );
```

Các chế độ mở file:

Chế độ	Mô tả
ios::in	Mở để đọc (mặc định cho ifstream)
ios::out	Mở để ghi (mặc định cho ofstream)
ios::ate	Bắt đầu đọc/ghi ở cuối file
ios::app	Bắt đầu ghi ở cuối file
ios::trunc	Cắt bỏ file (TRUNCate)
ios::nocreate	Lỗi nếu file không tồn tại
ios::noreplace	Lỗi khi mở để ghi ra nếu file đã tồn tại, trừ khi ate hoặc app được thiết lập
ios::binary	Mở file trong chế độ nhị phân (mặc định là văn bản)

2.2.Xử lý File

Trước khi đọc/ghi file, cần kiểm tra file đã được mở thành công hay không: `if (!file) ... else ...`

Trong quá trình làm việc với file, cần dùng hàm kiểm tra kết thúc tập tin trong vòng lặp: `while(!file.eof()) ...`

Vào/Ra kí tự: hàm `get()` và `put()` làm các hàm thành phần của `ofstream` và `ifstream` cho phép đọc kí tự từ file, và ghi kí tự ra file.

Vào/Ra file nhị phân: hàm `read()` và `write()` cho phép thực hiện việc đọc/ghi một số lượng lớn các byte dữ liệu từ bộ đệm sang file. Các tham số cho hàm `read()` và `write()` là địa chỉ của bộ đệm dữ liệu và kích thước của chúng. Địa chỉ phải ép thành kiểu `char` và kích thước được tính bằng `byte`.

Vào/Ra đối tượng (objects I/O): dùng chế độ mở file nhị phân, và các hàm read(), write().

Con trỏ file: mỗi đối tượng file kết hợp với nó 2 giá trị nguyên gọi là get pointer (con trỏ đọc) và put pointer (con trỏ ghi). Hai giá trị này cho biết vị trí đọc và vị trí ghi hiện tại của con trỏ trong quá trình làm việc với file. Để điều khiển con trỏ file phục vụ cho việc đọc/ghi file tại vị trí tùy ý, ta dùng hàm seekg(), seekp() (để di chuyển con trỏ file đến vị trí mong muốn), tellg(), tellp() (để đọc vị trí hiện hành của con trỏ file). Hàm seekg() và tellg() cho phép thiết lập và kiểm tra con trỏ đọc, còn các hàm seekp() và tellp() thực hiện tương tự đối với con trỏ ghi. Hàm seekg() và seekp() có 2 tham số: tham số thứ 1 biểu diễn độ lệch (offset), tính từ vị trí cụ thể xác định bởi tham số thứ 2 (ios::beg, ios::cur, ios:end)

2.3.Đóng File

Dùng hàm close() của đối tượng file.

2.4.Ví dụ Vào/Ra File định dạng

Hàm ghi file số nguyên

```
void WriteIntFile() {  
  
    ofstream outf( "test.dat", ios::out );  
  
    if ( ! outf ) {  
  
        cerr << "Cannot open file test.dat . . \n";  
  
        exit( 1 );  
    }  
}
```

```

    }

    for ( int i = 0; i < 100; i++ )

        outf << i + 1 << ' ';

    outf.close();

}

```

Hàm đọc file số nguyên

```

void ReadIntFile() {

    ifstream inf( "test.dat", ios::in );

    if ( ! inf ) {

        cerr << "Cannot open file test.dat . . .\n";

        exit( 1 );

    }

    int i;

    while ( ! inf.eof() ) {

        inf >> i;

        cout << setw( 4 ) << i;

    }

    inf.close();
}

```

```
}
```

Hàm đọc/ghi file số nguyên

```
void IntFile() {  
  
    fstream file;  
  
    file.open( "test.dat", ios::out | ios::in );  
  
    if ( ! file ) {  
  
        cerr << "Cannot open file test.dat . . .\n";  
  
        exit( 1 );  
  
    }  
  
    for ( int i = 0; i < 100; i++ )  
  
        outf << i + 1 << ' ';  
  
    file.seekg( 0, ios::beg ); // ve dau file  
  
    while ( ! file.eof() ) {  
  
        inf >> i;  
  
        cout << setw( 4 ) << i;  
  
    }  
  
    file.close();  
  
}
```

2.5. Ví dụ Vào/Ra kí tự

Hàm ghi kí tự ra file

```
void ReadCharFile() {  
  
    ofstream outf( "test.txt", ios::out );  
  
    if ( ! outf ) {  
  
        cerr << "Cannot open file test.txt ... \n";  
  
        exit( 1 );  
  
    }  
  
    char str[] = "KHOA TIN HOC";  
  
    int i = 0;  
  
    while ( s[ i ] ) {  
  
        outf.put( s[ i ] ); // outf << s[ i ]  
  
        i++;  
  
    }  
  
    outf.close();  
  
}
```

Hàm đọc kí tự từ file

```
void ReadCharFile() {
```

```

ifstream inf( "test.txt", ios::in );

if ( ! inf ) {

    cerr << "Cannot open file test.txt . . .\n";

    exit( 1 );

}

char ch;

while ( ! inf.eof() ) {

    inf.get( ch );

    cout << ch;

}

inf.close();

}

```

2.6. Ví dụ Vào/Ra File nhị phân

```

#include <iostream.h>

#include <fstream.h>

const int MAX = 100;

void main() {

    int buffer[ MAX ] = { 0 };

```

```

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

    buffer[ i ] = i;

ofstream outf( "test.dat", ios::binary );

if ( ! outf ) {

    cout << "Cannot open file test.dat . . .\n";

    exit( 1 );

}

outf.write( ( char* ) buffer, MAX * sizeof( int ) );

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

    buffer[ i ] = 0;

ifstream inf( "test.dat", ios::binary );

if ( ! inf ) {

    cout << "Cannot open file test.dat . . .\n";

    exit( 1 );

}

inf.read( ( char* ) buffer, MAX * sizeof( int ) );

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

    if ( buffer[ i ] = i ) {

```

```
cerr << "Du lieu khong dung . . \n";

return;

}

cout << "Du lieu dung . . \n";

}
```

3. Bài tập

- 1) Viết chương trình nối hai tập tin văn bản, tập tin thứ hai được nối vào cuối tập tin thứ nhất. Hiển thị nội dung của tập tin sau khi nối ra màn hình. Tên hai tập tin được nhập từ bàn phím.
- 2) Viết chương trình đọc một tập tin văn bản, tên tập tin được nhập từ bàn phím. Sau đó đếm số từ trong tập tin và hiển thị ra màn hình.
- 3) Viết chương trình đọc một tập tin văn bản, đếm số lần xuất hiện của một chuỗi (nếu có) trong tập tin và hiển thị ra màn hình. Tên tập tin và chuỗi cần tìm được nhập từ bàn phím.
- 4) Viết chương trình sao chép một tập tin văn bản thành một tập tin văn bản khác và mỗi dòng trong tập tin đích phải được đánh số thứ tự phía bên trái. Tên hai tập tin được nhập từ bàn phím.
- 5) Viết chương trình đọc một tập tin văn bản có tên được nhập từ bàn phím, đếm số ký tự trên mỗi dòng. Chương trình phải hiển thị số dòng, chiều dài của dòng ngắn nhất, dài nhất, và giá trị trung bình của các ký tự trên mỗi dòng.

- 6) Viết chương trình chuẩn hoá một tập tin văn bản thành tập tin khác, tên hai tập tin được nhập từ bàn phím. Chương trình sẽ bỏ những dòng trống và khoảng trắng dư thừa với qui ước như sau: Chỉ giữ lại một khoảng trắng sau dấu phẩy hay dấu chấm và có 5 khoảng trắng cho mỗi đầu dòng. Giả sử tập tin chỉ chứa các ký tự chữ cái, khoảng trắng, dấu phẩy, và dấu chấm.
- 7) Viết chương trình đọc tập tin văn bản, đếm số ký tự khác trắng, số dòng khác trống, số từ, và số câu. Sau đó tính trung bình số ký tự trên mỗi từ và số từ trên mỗi câu. Giả sử tập tin chỉ chứa các ký tự chữ cái, khoảng trắng, dấu phẩy, dấu chấm, dấu chấm phẩy, và dấu hai chấm. Một từ là một dãy các ký tự chữ cái liên tiếp nhau, một câu là một dãy các từ được phân cách bằng khoảng trắng, dấu phẩy, dấu chấm phẩy, hay dấu hai chấm và kết thúc bằng dấu chấm.
- 8) Viết chương trình đọc tập tin văn bản và hiển thị nội dung của nó trong một khối có tối đa 20 dòng. Nếu sau khi hiển thị mà nội dung tập tin vẫn chưa được đọc hết thì cần thông báo cho người sử dụng biết có muốn hiển thị khối kế tiếp không. Ngược lại, kết thúc chương trình.
- 9) Giả sử mỗi dòng trong tập tin văn bản chứa thông tin về một sinh viên, bao gồm: Số thứ tự (từ 1 đến 100), họ tên, và điểm (số thực). Viết chương trình để nhập một số thứ tự từ bàn phím và hiển thị thông tin về sinh viên có số thứ tự tương ứng.

TÀI LIỆU THAM KHẢO

1. **Nguyễn xuân Huy**, *Thuật toán*, Nhà xuất bản Khoa Học và Kỹ Thuật.
2. **Hoàng Kiêm**, *Giải một bài toán trên máy tính như thế nào*, Nhà xuất bản Giáo dục.
3. **Nguyễn Thanh Thủy** (chủ biên), *Nhập môn lập trình Ngôn ngữ C*, Nhà xuất bản Khoa học và Kỹ thuật.
4. **Trần Văn Lăng**, *Lập trình hướng đối tượng sử dụng C++*, Nhà xuất bản Thống Kê.
5. **GS. Phạm Văn Át**, *C++ và lập trình hướng đối tượng*, Nhà xuất bản Khoa học và Kỹ thuật.
6. **Nguyễn Thanh Thủy**, *Lập trình hướng đối tượng*, Nhà xuất bản Khoa học và Kỹ thuật.
7. **Tô Oai Hùng**, *Giáo trình Cơ sở lập trình sử dụng ngôn ngữ C++*, Tài liệu lưu hành nội bộ (Trường Đại Học Mở TP. HCM)
8. **Joel Adams & Larry Nyhoff**, *C++ An Introduction to Computing*, Prentice Hall 2002, Third Edition.
9. **H.M. Deitel & P.J. Deitel**, *C++ How to Program*, Prentice Hall, New Jersey, 2003, Fourth Edition.
10. **Bjarne Stroustrup**, *The C++ Programming Language*, Addison Wesley Longman, 1997, Third Edition.

Biên soạn LÊ PHÚ HIẾU