BỘ CÔNG THƯƠNG
TRƯỜNG CAO ĐẲNG THƯƠNG MẠI VÀ DU LỊCH

GIÁO TRÌNH
MÔN HỌC: LẬP TRÌNH CƠ BẢN

NGÀNH: CÔNG NGHỆ THÔNG TIN (ỨNG DỤNG PHẦN MỀM)
TRÌNH ĐỘ: TRUNG CẤP

(Ban hành kèm theo Quyết định số: 405/ QĐ CTMDL ngày 5 tháng 7 năm 2022 của Trường Cao đẳng Thương mại và Du lịch)

Thái Nguyên, năm 2022
(Lưu hành nội bộ)

TUYÊN BỐ BẢN QUYỀN

LỜI NÓI ĐẦU

Lập trình cơ bản C++ là một trong những ngôn ngữ lập trình hướng đối tượng mạnh và phổ biến hiện nay do tính mềm dẻo và đa năng của nó. Không chỉ các ứng dụng được viết trên C++ mà cả những chương trình hệ thống lớn đều được viết hầu hết trên C++. C++ là ngôn ngữ lập trình hướng đối tượng được phát triển trên nền tảng của C, không những khắc phục một số nhược điểm của ngôn ngữ C mà quan trọng hơn, C++ cung cấp cho học sinh, sinh viên một phương tiện lập trình theo kỹ thuật mới: lập trình hướng đối tượng. Hiện nay NNLT C++ đã được đưa vào giảng dạy trong hầu hết các trường Đại học, Cao đẳng để thay thế một số NNLT đã cũ như FORTRAN, Pascal … Tập bài giảng này được viết ra với mục đích đó, trang bị kiến thức và kỹ năng thực hành cho sinh viên bắt đầu học vào NNLT C++ tại Khoa Khoa học cơ bản – Trường Cao đẳng Thương mại và Du lịch Thái Nguyên.

Để phù hợp với chương trình, giáo trình này chỉ đề cập tới các thành phần lập trình cơ bản như các khái niệm, các cấu trúc chương trình cơ bản và cách mà một trương trình C++ hoạt động, thực hành một số bài tập liên quan.

Nội dung giáo trình này gồm 3 chương.

Chương 1: Các khái niệm cơ bản trong NNLT C++.

Chương 2: Các kiểu dữ liệu, biểu thức và câu lệnh trong C++.

Chương 3: Cấu trúc dữ liệu và kiểu dữ liệu kiểu mảng.

Bên cạnh đó, giáo trình cũng không thể tránh khỏi những sai sót nhất định. Nhóm tác giả rất mong nhận được những ý kiến đóng góp, phản hồi từ quý đồng nghiệp, các bạn người học và bạn đọc.

Trân trọng cảm ơn./.

MỤC LỤC

GIÁO TRÌNH MÔN HỌC

1. Tên môn học: Lập Trình Cơ Bản

2. Mã môn học: MH11

3. Vị trí, tính chất, ý nghĩa và vai trò của môn học:

3.1. Vị trí: Giáo trình dành cho người học trình độ Trung cấp và Cao đẳng tại trường Cao đẳng Thương mại và Du lịch Thái Nguyên.

3.2. Tính chất: Giáo trình cung cấp kiến thức, kỹ năng và năng lực tự chủ và trách nhiệm cho người học liên quan đến hoạt động lập trình cơ bản, gồm có: tổng quan về khái niệm, kiểu dữ liệu, cấu trúc dữ liệu của C++. Qua đó, người học đang học tập tại trường sẽ: (1) có bộ giáo trình phù hợp với chương trình đào tạo của trường; (2) dễ dàng tiếp thu cũng như vận dụng các kiến thức và kỹ năng được học vào môi trường học tập và thực tế thuộc lĩnh vực CNTT và UDPM.

3.3. Ý nghĩa và vai trò của môn học: Môn học lập trình cơ bản là môn học khoa học magn tính tư duy logic và trừu tượng và dành cho đối tượng là người học thuộc các chuyên ngành phần mềm, lập trình chương trình, vv,... Nội dung chủ yếu của môn học này nhằm cung cấp các kiến thức và kỹ năng về lập trình cơ bản: (1) Nhận biết được các thông tin của lập trình; Giải thích được một số nội dung: kiểu dữ liệu, biểu thức và câu lệnh trong C++, nhận dạng và phân biệt được các chương trình của C++. Qua đó, giáo trình cung cấp các kiến thức cơ bản cho các hoạt động lập trình phần mềm, ứng dụng CNTT và phần mềm.

4. Mục tiêu của môn học:

+ Về kiến thức:

- Hiểu được công dụng của ngôn ngữ lập trình C, hiểu cú pháp, công dụng của các câu lệnh dùng trong ngôn ngữ lập trình C.
- Nắm vững quy tắc xây dựng và sử dụng hàm, cách truyền tham số cho hàm

+ Về kỹ năng:

- Phân tích thiết kế và cài đặt các bài toán.
- Xác định các điều khiển áp dụng cho việc nhập dữ liệu đảm bảo chính xác, có chu trình xử lý dữ liệu.
- Vận dụng thành thạo các phương pháp lặp điều kiện trước hoặc sau, đảm bảo điều kiện kết thúc của vòng lặp

+ Về năng lực tự chủ và trách nhiệm: Sinh viên nhận thức được tầm quan trọng của học phần, từ đó có ý thức học tập, rèn luyện, bổ sung kiến thức cho bản thân.

5. Nội dung của môn học

5.1. Chương trình khung

Mã MH	Tên môn học	Số tín chỉ	Thời gian học tập *(giờ)*			
			Tổng số	Trong đó		
				Lý thuyết	Thực hành/ thực tập/ bài tập/ thảo luận	Thi/Kiểm tra
I	**Các môn học chung**	**12**	**255**	**94**	**148**	**13**
MH01	Chính trị	2	30	15	13	2
MH02	Pháp luật	1	15	9	5	1
MH03	Giáo dục thể chất	1	30	4	24	2
MH04	Giáo dục quốc phòng và an ninh	2	45	21	21	3
MH05	Tin học	2	45	15	29	1
MH06	Ngoại ngữ	4	90	30	56	4
II	**Các môn học chuyên môn**	**64**	**1560**	**504**	**1013**	**43**
II.1	***Môn học cơ sở***	***16***	***240***	***224***	**-**	***13***
MH07	Tin học văn phòng	2	30	12	17	1
MH08	Bảng tính Excel	2	30	12	17	1
MH09	Cấu trúc máy tính	2	30	28	-	2
MH10	Mạng máy tính	2	30	15	14	1
MH11	Lập trình cơ bản	2	30	28	-	2
MH12	Cấu trúc dữ liệu và giải thuật	2	30	28	-	2
MH13	Cơ sở dữ liệu	2	30	28	-	2
MH14	Lắp ráp và bảo trì máy tính	2	30	28	-	2
II.2	***Môn học chuyên môn***	***46***	***1290***	***313***	***948***	***28***
MH15	Ngoại ngữ ch.ngành CNTT	4	60	57	-	3
MH16	Hệ điều hành Windows Server	2	30	28	-	2
MH17	Quản trị CSDL với Access 1	3	45	43	-	2
MH18	Quản trị CSDL với SQL Server	3	45	27	17	1
MH19	Lập trình Windows 1 (VB.NET)	3	45	43	-	2
MH20	Thiết kế và quản trị website	3	45	43	-	2
MH21	Đồ họa ứng dụng	2	30	28	-	2

MH22	An toàn và bảo mật thông tin	2	30	28	-	2
MH23	TH xây dựng phần mềm quản lý	4	120	-	114	6
MH24	TH thiết kế và quản trị website	4	120	-	114	6
MH25	Thực tập tốt nghiệp	16	720	-	720	
II.3	***Môn học tự chọn*** *(chọn 1 trong 2)*	***2***	***30***	***28***	***-***	***2***
MH26	Kỹ năng giao tiếp, phục vụ khách hàng	2	30	28	-	2
MH27	Lập trình mạng	2	30	28	-	2
	Tổng cộng	**76**	**1815**	**598**	**1161**	**56**

5.2. Chương trình chi tiết môn học

Số TT	Tên chương	Thời gian (giờ)			
		Tổng số	Lý thuyết	Thực hành	Kiểm tra
1	**Chương 1: Các khái niệm cơ bản trong NNLT C++**	**8**	**8**	**0**	**0**
2	**Chương 2: Các kiểu dữ liệu, biểu thức và câu lệnh trong C++**	**11**	**10**	**0**	**1**
3	**Chương 3: Cấu trúc dữ liệu và kiểu dữ liệu kiểu mảng**	**11**	**10**	**0**	**1**
	Cộng	**30**	**28**	**0**	**2**

6. Điều kiện thực hiện môn học:

6.1. Phòng học Lý thuyết/Thực hành: Đáp ứng phòng học chuẩn

6.2. Trang thiết bị dạy học: Projetor, máy vi tính, bảng, phấn

6.3. Học liệu, dụng cụ, mô hình, phương tiện: Giáo trình, mô hình học tập,…

6.4. Các điều kiện khác: Người học tìm hiểu thêm về kiến thức tư duy logic.

7. Nội dung và phương pháp đánh giá:

7.1. Nội dung:

- Kiến thức: Đánh giá tất cả nội dung đã nêu trong mục tiêu kiến thức
- Kỹ năng: Đánh giá tất cả nội dung đã nêu trong mục tiêu kỹ năng.
- Năng lực tự chủ và trách nhiệm: Trong quá trình học tập, người học cần:

+ Nghiên cứu bài trước khi đến lớp.

+ Chuẩn bị đầy đủ tài liệu học tập.

+ Tham gia đầy đủ thời lượng môn học.

+ Nghiêm túc trong quá trình học tập.

7.2. Phương pháp:

Người học được đánh giá tích lũy môn học như sau:

7.2.1. Cách đánh giá

- Áp dụng quy chế đào tạo Cao đẳng hệ chính quy ban hành kèm theo Thông tư số 09/2017/TT-LĐTBXH, ngày 13/3/2017 của Bộ trưởng Bộ Lao động – Thương binh và Xã hội.

- Hướng dẫn thực hiện quy chế đào tạo áp dụng tại Trường Cao đẳng Thương mại và Du lịch như sau:

Điểm đánh giá	Trọng số
+ Điểm kiểm tra thường xuyên (Hệ số 1)	40%
+ Điểm kiểm tra định kỳ (Hệ số 2)	
+ Điểm thi kết thúc môn học	60%

7.2.2. Phương pháp đánh giá

Phương pháp đánh giá	Phương pháp tổ chức	Hình thức kiểm tra	Chuẩn đầu ra đánh giá	Số cột	Thời điểm kiểm tra
Thường xuyên	Viết/ Thuyết trình	Tự luận/ Trắc nghiệm/	Kiến thức cơ bản		Sau từng chương
Định kỳ	Viết/ Thuyết trình	Tự luận/ Trắc nghiệm/	Kiến thức, kỹ năng lập trình		Sau từng chương
Kết thúc môn học	Viết/ thực hành	Tự luận và trắc nghiệm	Kiến thức, tư duy logic và kỹ năng lập trình		Sau 28 giờ

7.2.3. Cách tính điểm

- Điểm đánh giá thành phần và điểm thi kết thúc môn học được chấm theo thang điểm 10 (từ 0 đến 10), làm tròn đến một chữ số thập phân.

- Điểm môn học là tổng điểm của tất cả điểm đánh giá thành phần của môn học nhân với trọng số tương ứng. Điểm môn học theo thang điểm 10 làm tròn đến một

chữ số thập phân, sau đó được quy đổi sang điểm chữ và điểm số theo thang điểm 4 theo quy định của Bộ Lao động Thương binh và Xã hội về đào tạo theo tín chỉ.

8. Hướng dẫn thực hiện môn học

8.1. Phạm vi, đối tượng áp dụng: Đối tượng Trung cấp nghề CNTT và UDPM.

8.2. Phương pháp giảng dạy, học tập môn học

8.2.1. Đối với người dạy

*** Lý thuyết:** Áp dụng phương pháp dạy học tích cực bao gồm: thuyết trình ngắn, nêu vấn đề, hướng dẫn đọc tài liệu, bài tập tình huống, câu hỏi thảo luận….

* **Bài tập:** Phân chia nhóm nhỏ thực hiện bài tập theo nội dung đề ra.

*** Thảo luận:** Phân chia nhóm nhỏ thảo luận theo nội dung đề ra.

*** Hướng dẫn tự học theo nhóm (nếu có):** Nhóm trưởng phân công các thành viên trong nhóm tìm hiểu, nghiên cứu theo yêu cầu nội dung trong bài học, cả nhóm thảo luận, trình bày nội dung, ghi chép và viết báo cáo nhóm.

8.2.2. Đối với người học: Người học phải thực hiện các nhiệm vụ như sau:

- Nghiên cứu kỹ bài học tại nhà trước khi đến lớp. Các tài liệu tham khảo sẽ được cung cấp nguồn trước khi người học vào học môn học này (trang web, thư viện, tài liệu...)

- Tham dự tối thiểu 70% các buổi giảng lý thuyết. Nếu người học vắng >30% số tiết lý thuyết phải học lại môn học mới được tham dự kì thi lần sau.

- Tự học và thảo luận nhóm (nếu có): là một phương pháp học tập kết hợp giữa làm việc theo nhóm và làm việc cá nhân. Một nhóm gồm 8-10 người học sẽ được cung cấp chủ đề thảo luận trước khi học lý thuyết, thực hành. Mỗi người học sẽ chịu trách nhiệm về 1 hoặc một số nội dung trong chủ đề mà nhóm đã phân công để phát triển và hoàn thiện tốt nhất toàn bộ chủ đề thảo luận của nhóm.

- Tham dự đủ các bài kiểm tra thường xuyên, định kỳ.

- Tham dự thi kết thúc môn học.

- Chủ động tổ chức thực hiện giờ tự học.

9. Tài liệu tham khảo:

[1]. Phạm Văn Ất – Giáo trình ngôn ngữ lập trình C – Nhà xuất bản Khoa học Kỹ thuật năm 2004

[2]. Tiêu kim Cương – Giáo trình ngôn ngữ lập trình C – Nhà xuất bản Giáo dục năm 2006.

CHƯƠNG 1
CÁC KHÁI NIỆM CƠ BẢN CỦA C++

❖ GIỚI THIỆU CHƯƠNG 1

Chương 1 là chương giới thiệu bức tranh tổng quan về các khái niệm cơ bản của C++ để người học có được kiến thức nền tảng và dễ dàng tiếp cận nội dung môn học ở những chương tiếp theo.

❖ MỤC TIÊU CHƯƠNG 1

Sau khi học xong chương này, người học có khả năng:

➢ ***Về kiến thức:***

- *Trình bày và giải thích được khái niệm cơ bản của C++.*
- *Trình bày và giải thích được khái niệm, vai trò, nội dung, nguyên tắc, môi trường hoạt động của C++.*
- *Vận dụng được các đặc trưng của các khái niệm vào một chương trình C++.*

➢ ***Về kỹ năng:***

- *Nhận diện được các ký tự, từ khóa, tên gọi của các biến, hàm trong chương trình.*
- *Phân tích được những các bước để tạo một chương trình, các dữ liệu đầu vào, đầu ra của chương trình.*

➢ ***Về năng lực tự chủ và trách nhiệm:***

- *Ý thức được tầm quan trọng và ý nghĩa thực tiễn của ngôn ngữ lập trình C++.*
- *Cân nhắc đưa ra quyết định thực hiện chương trình.*
- *Tuân thủ nội quy, quy định nơi làm việc.*

❖ PHƯƠNG PHÁP GIẢNG DẠY VÀ HỌC TẬP CHƯƠNG 1

- *Đối với người dạy: sử dụng phương pháp giảng giảng dạy tích cực (diễn giảng, vấn đáp, dạy học theo vấn đề); yêu cầu người học thực hiện câu hỏi thảo luận và bài tập chương 1 (cá nhân hoặc nhóm).*
- *Đối với người học: chủ động đọc trước giáo trình (chương 1) trước buổi học; hoàn thành đầy đủ câu hỏi thảo luận và bài tập tình huống chương 1 theo cá nhân hoặc nhóm và nộp lại cho người dạy đúng thời gian quy định.*

❖ ĐIỀU KIỆN THỰC HIỆN CHƯƠNG 1

- ***Phòng học chuyên môn hóa/nhà xưởng:*** Phòng thực hành có máy tính.
- ***Trang thiết bị máy móc:*** Máy chiếu và các thiết bị dạy học khác
- ***Học liệu, dụng cụ, nguyên vật liệu:*** Chương trình môn học, giáo trình, tài liệu tham khảo, giáo án, phim ảnh, và các tài liệu liên quan.
- ***Các điều kiện khác:*** Không có

❖ **KIỂM TRA VÀ ĐÁNH GIÁ CHƯƠNG 1**

- **Nội dung:**

✓ *Kiến thức: Kiểm tra và đánh giá tất cả nội dung đã nêu trong mục tiêu kiến thức*

✓ *Kỹ năng: Đánh giá tất cả nội dung đã nêu trong mục tiêu kĩ năng.*

✓ *Năng lực tự chủ và trách nhiệm: Trong quá trình học tập, người học cần:*

+ Nghiên cứu bài trước khi đến lớp

+ Chuẩn bị đầy đủ tài liệu học tập.

+ Tham gia đầy đủ thời lượng môn học.

+ Nghiêm túc trong quá trình học tập.

- **Phương pháp:**

✓ ***Điểm kiểm tra thường xuyên:*** *không có*

✓ ***Kiểm tra định kỳ lý thuyết:*** *không có*

❖ **NỘI DUNG CHƯƠNG 1**

I. CÁC YẾU TỐ CƠ BẢN

Lập trình cơ bản C++ là môn lập trình cho phép học sinh biểu hiện ý tưởng của mình để giải quyết một vấn đề, bài toán bằng cách diễn đạt gần với ngôn ngữ thông thường thay vì phải diễn đạt theo ngôn ngữ máy (dãy các kí hiệu 0,1). Hiển nhiên, các ý tưởng học sinh muốn trình bày phải được viết theo một cấu trúc chặt chẽ thường được gọi là *thuật toán* hoặc *giải thuật* và theo đúng các qui tắc của ngôn ngữ gọi là *cú pháp* hoặc *văn phạm*. Trong giáo trình này chúng ta giới thiệu đến một ngôn ngữ lập trình như vậy, đó là ngôn ngữ lập trình C++ và làm thế nào để thể hiện các ý tưởng giải quyết vấn đề bằng cách viết thành chương trình trong C++.

Trước hết, trong mục này chúng ta sẽ trình bày về các qui định bắt buộc đơn giản và cơ bản nhất. Thông thường các qui định này sẽ được nhớ dần trong quá trình học ngôn ngữ, tuy nhiên để có một vài khái niệm tương đối tổng quát về C++ chúng ta trình bày sơ lược các khái niệm cơ bản đó.

1. Bảng ký tự của C++

Hầu hết các ngôn ngữ lập trình hiện nay đều sử dụng các kí tự tiếng Anh, các kí

hiệu thông dụng và các con số để thể hiện chương trình. Các kí tự của những ngôn ngữ khác không được sử dụng (ví dụ các chữ cái tiếng Việt). Dưới đây là bảng kí tự được phép dùng để tạo nên những câu lệnh của ngôn ngữ C++.

- Các chữ cái la tinh (viết thường và viết hoa): a .. z và A .. Z. Cùng một chữ cái nhưng viết thường phân biệt với viết hoa. Ví dụ chữ cái 'a' là khác với 'A'.
- Dấu gạch dưới: _
- Các chữ số thập phân: 0, 1, . ., 9.
- Các ký hiệu toán học: +, -, *, /, % , &, ||, !, >, <, = ...
- Các ký hiệu đặc biệt khác: , ;: [ ], {}, #, dấu cách, ...

2. Từ khoá

Một từ khoá là một từ được qui định trước trong ngôn ngữ lập trình (NNLT) với một ý nghĩa cố định, thường dùng để chỉ các loại dữ liệu hoặc kết hợp thành câu lệnh. Học sinh có thể tạo ra những từ mới để chỉ các đối tượng của mình nhưng không được phép trùng với từ khóa có sẵn trong NNLT. Dưới đây chúng tôi liệt kê một vài từ khoá thường gặp, ý nghĩa của các từ này, sẽ được trình bày dần trong các đề mục liên quan.

auto, break, case, char, continue, default, do, double, else, externe, float, for, goto, if, int, long, register, return, short, sizeof, static, struct, switch, typedef, union, unsigned, while ....

3. Tên gọi

Để phân biệt các đối tượng với nhau chúng cần có một tên gọi. Hầu hết một đối tượng được viết ra trong chương trình thuộc 2 dạng, một dạng đã có sẵn trong ngôn ngữ (ví dụ các từ khoá, tên các hàm chuẩn ...), một số do học sinh tạo ra dùng để đặt tên cho hằng, biến, kiểu, hàm ... các tên gọi do học sinh tự đặt phải tuân theo một số qui tắc sau:

- Là dãy ký tự liên tiếp (không chứa dấu cách) và phải bắt đầu bằng chữ cái hoặc gạch dưới.
- Phân biệt kí tự in hoa và thường.
- Không được trùng với từ khóa.
- Số lượng chữ cái dùng để phân biệt tên gọi có thể được đặt tuỳ ý.
- Chú ý các tên gọi có sẵn của C++ cũng tuân thủ theo đúng qui tắc trên.

Trong một chương trình nếu đặt tên sai thì trong quá trình xử lý sơ bộ (trước khi chạy chương trình) máy sẽ báo lỗi (gọi là lỗi văn phạm).

Ví dụ 1 :

- Các tên gọi sau đây là đúng (được phép): i, i1, j, tinhoc, tin_hoc, luu_luong
- Các tên gọi sau đây là sai (không được phép): 1i, tin hoc, luu-luong-nuoc
- Các tên gọi sau đây là khác nhau: ha_noi, Ha_noi, HA_Noi, HA_NOI, ...

4. Chú thích trong chương trình

Một chương trình thường được viết một cách ngắn gọn, do vậy thông thường bên cạnh các câu lệnh chính thức của chương trình, học sinh còn được phép viết vào chương trình các câu ghi chú, giải thích để làm rõ nghĩa hơn chương trình. Một chú thích có thể ghi chú về nhiệm vụ, mục đích, cách thức của thành phần đang được chú thích như biến, hằng, hàm hoặc công dụng của một đoạn lệnh ... Các chú thích sẽ làm cho chương trình sáng sủa, dễ đọc, dễ hiểu và vì vậy dễ bảo trì, sửa chữa về sau.

Có 2 cách báo cho chương trình biết một đoạn chú thích:

Nếu chú thích là một đoạn kí tự bất kỳ liên tiếp nhau (trong 1 dòng hoặc trên nhiều dòng) ta đặt đoạn chú thích đó giữa cặp dấu đóng mở chú thích /* (mở) và */ (đóng).

Nếu chú thích bắt đầu từ một vị trí nào đó cho đến hết dòng, thì ta đặt dấu // ở vị trí đó. Như vậy // sử dụng cho các chú thích chỉ trên 1 dòng.

Như đã nhắc ở trên, vai trò của đoạn chú thích là làm cho chương trình dễ hiểu đối với người đọc, vì vậy đối với máy các đoạn chú thích sẽ được bỏ qua. Lợi dụng đặc điểm này của chú thích đôi khi để tạm thời bỏ qua một đoạn lệnh nào đó trong chương trình (nhưng không xoá hẳn để khỏi phải gõ lại khi cần dùng đến) ta có thể đặt các dấu chú thích bao quanh đoạn lệnh này (ví dụ khi chạy thử chương trình, gỡ lỗi ...), khi cần sử dụng lại ta có thể bỏ các dấu chú thích.

Chú ý: Cặp dấu chú thích /* ... */ không được phép viết lồng nhau, ví dụ dòng chú thích sau là không được phép:

```
/* Đây là đoạn chú thích /* chứa đoạn chú thích này */ như đoạn chú thích con */
```

cần phải sửa lại như sau:

- hoặc chỉ giữ lại cặp dấu chú thích ngoài cùng

```
/* Đây là đoạn chú thích chứa đoạn chú thích này như đoạn chú thích con */
```

- hoặc chia thành các đoạn chú thích liên tiếp nhau

```
/* Đây là đoạn chú thích */ /*chứa đoạn chú thích này*/ /*như đoạn chú thích con */
```

II. MÔI TRƯỜNG LÀM VIỆC CỦA C++

1. Khởi động - Thoát khỏi C++

Khởi động C++ cũng như mọi chương trình khác bằng cách nhấp đúp chuột lên biểu tượng của chương trình. Khi chương trình được khởi động sẽ hiện ra giao diện

gồm có menu công việc và một khung cửa sổ bên dưới phục vụ cho soạn thảo. Một con trỏ nhấp nháy trong khung cửa sổ và chúng ta bắt đầu nhập nội dung (văn bản) chương trình vào trong khung cửa sổ soạn thảo này. Mục đích của giáo trình này là trang bị những kiến thức cơ bản của lập trình thông qua NNLT C++ cho các sinh viên mới bắt đầu nên chúng tôi vẫn chọn trình bày giao diện của các trình biên dịch quen thuộc là **Dev-C++.** Về các trình biên dịch khác độc giả có thể tự tham khảo trongcác tài liệu liên quan.

Để kết thúc làm việc với C++ (soạn thảo, chạy chương trình ...) và quay về môi trường Windows chúng ta ấn **Alt-X hoặc chọn close window**.

2. Giao diện và cửa sổ soạn thảo

a. Mô tả chung

Khi gọi chạy C++ trên màn hình sẽ xuất hiện một menu xổ xuống và một cửa sổ soạn thảo. Trên menu gồm có các nhóm chức năng: **File, Edit, Search, Run, Compile, Debug, Project, Options, Window, Help**. Để kích hoạt các nhóm chức năng, có thể ấn **Alt+chữ cái** biểu thị cho menu của chức năng đó (là chữ cái có gạch dưới). Ví dụ để mở nhóm chức năng **File** ấn **Alt+F**, sau đó dịch chuyển hộp sáng đến mục cần chọn rồi ấn Enter. Để thuận tiện cho học sinh, một số các chức năng hay dùng còn được gắn vớimột tổ hợp các phím cho phép người dùng có thể chọn nhanh chức năng này mà không cần thông qua việc mở menu như đã mô tả ở trên. Một số tổ hợp phím cụ thể đó sẽ được trình bày vào cuối phần này.

b. Các chức năng soạn thảo

Giống hầu hết các bộ soạn thảo văn bản, bộ soạn thảo của **Dev-C++** cũng sử dụng các phím sau cho quá trình soạn thảo:

- *Dịch chuyển con trỏ*: các phím mũi tên cho phép dịch chuyển con trỏ sang trái, phải một kí tự hoặc lên trên, xuống dưới 1 dòng. Để dịch chuyển nhanh có các phím như Home (về đầu dòng), End (về cuối dòng), PgUp, PgDn (lên, xuống một trang màn hình). Để dịch chuyển xa hơn có thể kết hợp các phím này cùng phím Control (Ctrl, ^) như ^PgUp: về đầu tệp, ^PgDn: về cuối tệp.

- *Chèn, xoá, sửa*: Phím Insert cho phép chuyển chế độ soạn thảo giữa chèn và đè. Các phím Delete, Backspace cho phép xoá một kí tự tại vị trí con trỏ và trước vị trí con trỏ (xoá lùi).

- *Các thao tác với khối dòng*: Để đánh dấu khối dòng (thực chất là khối kí tự liền nhau bất kỳ) ta đưa con trỏ đến vị trí đầu ấn Ctrl-KB và Ctrl-KK tại vị trí cuối. Cũng có thể thao tác nhanh hơn bằng cách giữ phím Shift và dùng các phím dịch

chuyển con trỏ quét từ vị trí đầu đến vị trí cuối, khi đó khối kí tự được đánh dấu sẽ chuyển mầu nền. Một khối được đánh dấu có thể dùng để cắt, dán vào một nơi khác trong văn bản hoặc xoá khỏi văn bản. Để thực hiện thao tác cắt dán, đầu tiên phải đưa khối đã đánh dấu vào bộ nhớ đệm bằng nhóm phím Shift-Delete (cắt), sau đó dịch chuyển con trỏ đến vị trí mới cần hiện nội dung vừa cắt và ấn tổ hợp phím Shift-Insert. Một đoạn văn bản được ghi vào bộ nhớ đệm có thể được dán nhiều lần vào nhiều vị trí khác nhau bằng cách lặp lại tổ hợp phím Shift-Insert tại các vị trí khác nhau trong văn bản. Để xoá một khối dòng đã đánh dấu mà không ghi vào bộ nhớ đệm, dùng tổ hợp phím Ctrl-Delete. Khi một nội dung mới ghi vào bộ nhớ đệm thì nó sẽ xoá (ghi đè) nội dung cũ đã có, do vậy cần cân nhắc để sử dụng phím Ctrl-Delete (xoá và không lưu lại nội dung vừa xoá vào bộ đệm) và Shift-Delete (xoá và lưu lại nội dung vừa xoá) một cách phù hợp.

c. Chức năng tìm kiếm và thay thế

Chức năng này dùng để dịch chuyển nhanh con trỏ văn bản đến từ cần tìm. Để thực hiện tìm kiếm bấm Ctrl-F, tìm kiếm và thay thế bấm Ctrl-H. Vào từ hoặc nhóm từ cần tìm vào cửa sổ Find, nhóm thay thế (nếu dùng Ctrl-H) vào cửa sổ Replace và đánh dấu vào các tuỳ chọn trong cửa sổ bên dưới sau đó ấn Enter. Các tuỳ chọn gồm: không phân biệt chữ hoa/thường, tìm từ độc lập hay đứng trong từ khác, tìm trong toàn văn bản hay chỉ trong phần được đánh dấu, chiều tìm đi đến cuối hay ngược về đầu văn bản, thay thế có hỏi lại hay không hỏi lại … Để dịch chuyển con trỏ đến các vùng khác nhau trong một menu hay cửa sổ chứa các tuỳ chọn ta sử dụng phím Tab.

d. Các chức năng liên quan đến tệp

- Ghi tệp lên đĩa: Chọn menu File\Save hoặc phím F2. Nếu tên tệp chưa có (mang tên Noname.cpp) máy sẽ yêu cầu cho tên tệp. Phần mở rộng của tên tệp được mặc định là CPP.

- Soạn thảo tệp mới: Chọn menu File\New. Hiện ra cửa sổ soạn thảo trắng và tên file tạm thời lấy là *Noname.cpp*.

- Soạn thảo tệp cũ: Chọn menu File\Open hoặc ấn phím F3, nhập tên tệp hoặc dịch chuyển con trỏ trong vùng danh sách tệp bên dưới đến tên tệp cần soạn rồi ấn Enter. Cũng có thể áp dụng cách này để soạn tệp mới khi không nhập vào tên tệp cụ thể.

- Ghi tệp đang soạn thảo lên đĩa với tên mới: Chọn menu File\Save As và nhập tên tệp mới vào rồi ấn Enter.

e. Chức năng dịch và chạy chương trình

- F9: Khởi động chức năng dịch và biên dịch toàn bộ chương trình.
- F10: Chạy chương trình từ đầu đến dòng lệnh hiện tại (đang chứa con trỏ)
- F11: Biên dịch và chạy toàn bộ chương trình.

Các chức năng liên quan đến dịch chương trình có thể được chọn thông qua menu Compile (Alt-C).

3. Cấu trúc một chương trình trong C++

Một chương trình C++ có thể được đặt trong một hoặc nhiều file văn bản khác nhau. Mỗi file văn bản chứa một số phần nào đó của chương trình. Với những chương trình đơn giản và ngắn thường chỉ cần đặt chúng trên một file.

Một chương trình gồm nhiều hàm, mỗi hàm phụ trách một công việc khác nhau của chương trình. Đặc biệt trong các hàm này có một hàm duy nhất có tên hàm là main(). Khi chạy chương trình, các câu lệnh trong hàm main() sẽ được thực hiện đầu tiên. Trong hàm main() có thể có các câu lệnh gọi đến các hàm khác khi cần thiết, và các hàm này khi chạy lại có thể gọi đến các hàm khác nữa đã được viết trong chương trình (trừ việc gọi quay lại hàm main()). Sau khi chạy đến lệnh cuối cùng của hàm main() chương trình sẽ kết thúc.

Cụ thể, thông thường một chương trình gồm có các nội dung sau:

– *Phần khai báo các tệp nguyên mẫu*: khai báo tên các tệp chứa những thành phần có sẵn (như các hằng chuẩn, kiểu chuẩn và các hàm chuẩn) mà học sinh sẽ dùng trong chương trình.

– *Phần khai báo các kiểu dữ liệu, các biến, hằng* ... do học sinh định nghĩa và được dùng chung trong toàn bộ chương trình.

– *Danh sách các hàm của chương trình* do học sinh viết, bao gồm cả hàm main().

– Dưới đây là một đoạn chương trình đơn giản chỉ gồm 1 hàm chính là hàm main(). Nội dung của chương trình dùng in ra màn hình dòng chữ: *Chào các bạn, bây giờ là 2 giờ.*

```
#include <iostream.h>
void main()
 {
    int h = 2,  // Khai báo và khởi tạo biến h = 2
     cout << “Chào các bạn, bây giờ là ” << h << " giờ" ; // in ra màn hình
 }
```

 # Giải thích câu lệnh trên:

Dòng đầu tiên của chương trình là khai báo tệp nguyên mẫu iostream.h. Đây là khai báo bắt buộc vì trong chương trình có sử dụng phương thức chuẩn "cout <<" (in ra màn hình), phương thức này được khai báo và định nghĩa sẵn trong iostream.h.

Không riêng hàm main(), mọi hàm khác đều phải bắt đầu tập hợp các câu lệnh của mình bởi dấu { và kết thúc bởi dấu }. Tập các lệnh bất kỳ bên trong cặp dấu này được gọi là khối lệnh. Khối lệnh là một cú pháp cần thiết trong các câu lệnh có cấu trúc như ta sẽ thấy trong các chương tiếp theo.

III. CÁC BƯỚC TẠO VÀ THỰC HIỆN CHƯƠNG TRÌNH

1. Qui trình viết và thực hiện chương trình

Trước khi viết và chạy một chương trình thông thường chúng ta cần:

- *Xác định yêu cầu của chương trình.* Nghĩa là xác định dữ liệu đầu vào (input) cung cấp cho chương trình và tập các dữ liệu cần đạt được tức đầu ra (output).Các tập hợp dữ liệu này ngoài các tên gọi còn cần xác định kiểu của nó. Ví dụ để giải một phương trình bậc 2 dạng: $ax^2 + bx + c = 0$, cần báo cho chương trình biết dữ liệu đầu vào là a, b, c và đầu ra là nghiệm x1 và x2 của phương trình. Kiểu của a, b, c, x1, x2 là các số thực.

- *Xác định thuật toán giải.*

- Cụ thể hoá các khai báo kiểu và thuật toán thành dãy các lệnh, tức viết thành chương trình thông thường là trên giấy, sau đó bắt đầu soạn thảo vào trong máy. Quá trình này được gọi là soạn thảo chương trình nguồn.

- Dịch chương trình nguồn để tìm và sửa các lỗi gọi là lỗi cú pháp.

- Chạy chương trình, kiểm tra kết quả in ra trên màn hình. Nếu sai, sửa lại chương trình, dịch và chạy lại để kiểm tra. Quá trình này được thực hiện lặp đi lặp lại cho đến khi chương trình chạy tốt theo yêu cầu đề ra của học sinh.

2. Soạn thảo tệp chương trình nguồn

Soạn thảo chương trình nguồn là một công việc đơn giản: gõ nội dung của chương trình (đã viết ra giấy) vào trong máy và lưu lại nó lên đĩa. Thông thường khi đã lưu lại chương trình lên đĩa lần sau sẽ không cần phải gõ lại. Có thể soạn chương trình nguồn trên các bộ soạn thảo (editor) khác nhưng phải chạy trong môi trường tích hợp C++ (**Dev-C++**). Mục đích của soạn thảo là tạo ra một văn bản chương trình và đưa vào bộ nhớ của máy. Văn bản chương trình cần được trình bày sáng sủa, rõ ràng. Các câu lệnh cần gióng thẳng cột theo cấu trúc của lệnh (các lệnh chứa trong một lệnh cấu trúc được trình bày thụt vào trong so với điểm bắt

đầu của lệnh). Các chú thích nên ghi ngắn gọn, rõ nghĩa và phù hợp.

3. Dịch chương trình

Sau khi đã soạn thảo xong chương trình nguồn, bước tiếp theo thường là dịch (ấn phím F9) để tìm và sửa các lỗi gọi là lỗi cú pháp. Trong khi dịch C++ sẽ đặt con trỏ vào nơi gây lỗi (viết sai cú pháp) trong văn bản. Sau khi sửa xong một lỗi có thể dùng F9. Quá trình sửa lỗi biên dịch được lặp lại cho đến khi văn bản đã được sửa hết lỗi cú pháp.

Sản phẩm sau khi dịch là một tệp mới gọi là chương trình đích có đuôi EXE tức là tệp mã máy để thực hiện. Tệp này có thể lưu tạm thời trong bộ nhớ phục vụ cho quá trình chạy chương trình hoặc lưu lại trên đĩa tuỳ theo tuỳ chọn khi dịch của học sinh. Trong và sau khi dịch, C++ sẽ hiện một cửa sổ chứa thông báo về các lỗi (nếu có), các lỗi này còn lỗi cú pháp, hoặc thông báo chương trình đã được dịch thành công.

Các lỗi này còn lỗi cú pháp. Để dịch chương trình ta chọn menu Compile\Compile hoặc \Compile\Make hoặc nhanh chóng hơn bằng cách ấn tổ hợp phím Alt-F9.

4. Chạy chương trình

Ấn F9 để chạy chương trình, nếu chương trình chưa dịch sang mã máy, máy sẽ tự động dịch lại trước khi chạy. Kết quả của chương trình sẽ hiện ra trong một cửa sổ kết quả để học sinh kiểm tra. Nếu kết quả chưa được như mong muốn, quay lại văn bản để sửa và lại chạy lại chương trình. Quá trình này được lặp lại cho đến khi chương trình chạy đúng như yêu cầu đã đề ra. Khi chương trình chạy, cửa sổ kết quả sẽ hiện ra tạm thời che khuất cửa sổ soạn thảo. Sau khi kết thúc chạy chương trình cửa sổ soạn thảo sẽ tự động hiện ra trở lại và che khuất cửa sổ kết quả.

IV. VÀO/RA TRONG C++

Trong phần này chúng ta làm quen một số lệnh đơn giản cho phép học sinh nhập dữ liệu vào từ bàn phím hoặc in kết quả ra màn hình. Trong phần sau của giáo trình chúng ta sẽ khảo sát các câu lệnh vào/ra phức tạp hơn.

1. Vào dữ liệu từ bàn phím

Để nhập dữ liệu vào cho các biến có tên **biến_1**, **biến_2**, **biến_3** chúng ta sử dụng câu lệnh:

```
cin >> biến_1 ;
cin >> biến_2 ;
cin >> biến_3 ;
```

hoặc: cin >> biến_1 >> biến_2 >> biến_3 ;

biến_1, biến_2, biến_3 là các **biến** được sử dụng để lưu trữ các giá trị học sinh nhập vào từ bàn phím. Khái niệm biến sẽ được mô tả cụ thể hơn trong chương 2, ở đây biến_1, biến_2, biến_3 được hiểu là các tên gọi để chỉ 3 giá trị khác nhau. Hiển nhiên có thể nhập dữ liệu nhiều hơn 3 biến bằng cách tiếp tục viết tên biến vào bên phải sau dấu >> của câu lệnh.

Khi chạy chương trình nếu gặp các câu lệnh trên chương trình sẽ "tạm dừng" để chờ học sinh nhập dữ liệu vào cho các biến. Sau khi học sinh nhập xong dữ liệu, chương trình sẽ tiếp tục chạy từ câu lệnh tiếp theo sau của các câu lệnh trên. Cách thức nhập dữ liệu của học sinh phụ thuộc vào loại giá trị của biến cần nhập mà ta gọi là **kiểu,** ví dụ nhập một số có cách thức khác với nhập một chuỗi kí tự. Giả sử cần nhập độ dài hai cạnh của một hình chữ nhật, trong đó cạnh dài được qui ước bằng tên biến cd và chiều rộng được qui ước bởi tên biến cr. Câu lệnh nhập sẽ như sau:

```
cin >> cd >> cr ;
```

Khi máy dừng chờ nhập dữ liệu học sinh sẽ gõ giá trị cụ thể của các chiều dài, rộng theo đúng thứ tự trong câu lệnh. Các giá trị này cần cách nhau bởi ít nhất một dấu trắng (ta qui ước gọi dấu trắng là một trong 3 loại dấu được nhập bởi các phím sau: phím spacebar (dấu cách), phím tab (dấu tab) hoặc phím Enter (dấu xuống dòng)). Các giá trị học sinh nhập vào cũng được hiển thị trên màn hình để dễ theo dõi.

Ví dụ nếu nhập vào 23 11 thì chương trình sẽ gán giá trị 23 cho biến cd và 11 cho biến cr.

Chú ý: giả sử nhập 2311 (không có dấu cách giữa 23 và 11) thì chương trình sẽ xem 2311 là một giá trị và gán cho cd. Máy sẽ tạm dừng chờ nhập tiếp giá trị cho biến cr.

2. In dữ liệu ra màn hình

Để in giá trị của các **biểu thức** ra màn hình ta dùng câu lệnh sau:

```
cout << bt_1 ;
cout << bt_2 ;
cout << bt_3 ;
```

hoặc: cout << bt_1 << bt_2 << bt_3 ;

Cũng giống câu lệnh nhập ở đây chúng ta cũng có thể mở rộng lệnh in với nhiều hơn 3 biểu thức. Câu lệnh trên cho phép in giá trị của các biểu thức bt_1, bt_2, bt_3. Các giá trị này có thể là tên của biến hoặc các kết hợp tính toán trên biến. Ví dụ để in

câu "Chiều dài là " và số 23 và tiếp theo là chữ "mét", ta có thể sử dụng 3 lệnh sau đây:

```
cout << "Chiều dài là" ;
cout << 23 ;
cout << "mét";
```

hoặc có thể chỉ bằng 1 lệnh: cout << "Chiều dài là 23 mét" ;

Chú ý: để sử dụng các câu lệnh nhập và in trong phần này, đầu chương trình phải có dòng khai báo **#include <iostream.h>.**

Thông thường ta hay sử dụng lệnh in để in câu thông báo nhắc nhập dữ liệu trước khi có câu lệnh nhập. Khi đó trên màn hình sẽ hiện dòng thông báo này rồi mới tạm dừng chờ dữ liệu nhập vào từ bàn phím. Nhờ vào thông báo này học sinh sẽ biết phải nhập dữ liệu, nhập nội dung gì và như thế nào ...

ví dụ:

```
cout << "Hãy nhập chiều dài: "; cin >> cd;
cout << "Và nhập chiều rộng: "; cin >> cr;
```

Khi đó máy sẽ in dòng thông báo "Hãy nhập chiều dài: " và chờ sau khi nhập xong 23, máy sẽ thực hiện câu lệnh tiếp theo tức in dòng thông báo "Và nhập chiều rộng: " và chờ đến khi nhập xong 11 chương trình sẽ tiếp tục thực hiện các câu lệnh tiếp theo.

BÀI TẬP

1. Những tên gọi nào sau đây là hợp lệ:

- x – 123variabe – tin_hoc – toan tin - so-dem
- RADI – one.0 – number# – Radius - nam2000

2. Bạn hãy thử viết một chương trình ngắn nhất có thể được.

3. Tìm các lỗi cú pháp trong chương trình sau:

```
#include (iostream.h)
void main();                    // Giải phương trình bậc 1
{
   cout << 'Day la chương trình: Gptb1.\nXin chao cac ban';
   return 0;
}
```

4. Viết chương trình in nội dung một bài thơ nào đó.

5. Viết chương trình in ra 4 dòng, 2 cột gồm các số sau và gióng cột:

- thẳng theo lề trái 0.63 64.1

– thẳng theo lề phải 12.78 -11.678

– thẳng theo dấu chấm thập phân -124. 6 59.002

6. Hãy viết và chạy các chương trình trong các ví dụ 3, 5.

7. Chương trình sau khai báo 5 biến kí tự a, b, c, d, e và một biến số nam. Hãy điền thêm các câu lệnh vào các dòng … để chương trình thực hiện nhiệm vụ sau:

– Nhập giá trị cho biến nam

– Nhập giá trị cho các biến kí tự a, b, c, d, e.

– In ra màn hình dòng chữ được ghép bởi 5 kí tự đã nhập và chữ "năm" sau đóin số đã nhập (nam). Ví dụ nếu 5 chữ cái đã nhập là 'H', 'A', 'N', 'O', 'I' và namđược nhap là 2000, thì màn hình in ra dòng chữ: HANOI năm 2000.

– Nhập chương trình đã sửa vào máy và chạy để kiểm tra kết quả.

```
#include <iostream.h>
#include <conio.h>
main()
{
    int nam;
    char a, b, c, d, e;
    cin >> nam ;
    cin.get(a); cin.get(b); cin.get(c); … ; … ;
    // in kết quả
    cout << a << … << … << … << … << " nam " << … ;
    getch();
}
```

CHƯƠNG 2
KIỂU DỮ LIỆU, BIỂU THỨC VÀ CÂU LỆNH

❖ GIỚI THIỆU CHƯƠNG 2

Chương 2 là chương mô tả về các kiểu dữ liệu, biểu thức và câu lệnh thực hiện trong một chương trình C++.

❖ MỤC TIÊU CHƯƠNG 2

Sau khi học xong chương này, người học có khả năng:

➢ ***Về kiến thức:***

- *Trình bày và giải thích được một số khái niệm như: kiểu dữ liệu, biến thường dùng trong C++, biểu thức chính và câu lệnh chủ yếu trong C++*
- *Vận dụng được các hằng, biến, phép toán vào trong chương trình C++.*

➢ ***Về kỹ năng:***

- *Nhận diện được kiểu dữ liệu cần dùng, biến cần khai báo và sử dụng trong C++.*
- *Thực hiện được phép toán trong thực tế.*
- *Lựa chọn được phương pháp giải thuật logic trong tổ chức chương trình C++.*

➢ ***Về năng lực tự chủ và trách nhiệm:***

- *Ý thức được tầm quan trọng và ý nghĩa thực tiễn của các kiểu dữ liệu, biểu thức, biến trong C++.*
- *Tuân thủ nội quy, quy định nơi làm việc.*

❖ PHƯƠNG PHÁP GIẢNG DẠY VÀ HỌC TẬP CHƯƠNG 2

- *Đối với người dạy: sử dụng phương pháp giảng giảng dạy tích cực (diễn giảng, vấn đáp, dạy học theo vấn đề); yêu cầu người học thực hiện câu hỏi thảo luận và bài tập chương 2 (cá nhân hoặc nhóm).*
- *Đối với người học: chủ động đọc trước giáo trình (chương 2) trước buổi học; hoàn thành đầy đủ câu hỏi thảo luận và bài tập tình huống chương 2 theo cá nhân hoặc nhóm và nộp lại cho người dạy đúng thời gian quy định.*

❖ ĐIỀU KIỆN THỰC HIỆN BÀI GIẢNG CHƯƠNG 2

- *Đối với người dạy: sử dụng phương pháp giảng giảng dạy tích cực (diễn giảng, vấn đáp, dạy học theo vấn đề); yêu cầu người học thực hiện câu hỏi thảo luận và bài tập chương 2 (cá nhân hoặc nhóm).*
- *Đối với người học: chủ động đọc trước giáo trình (chương 2) trước buổi học; hoàn thành đầy đủ câu hỏi thảo luận và bài tập tình huống chương 2 theo cá nhân hoặc nhóm và nộp lại cho người dạy đúng thời gian quy định.*

❖ **ĐIỀU KIỆN THỰC HIỆN CHƯƠNG 2**

- ***Phòng học chuyên môn hóa/nhà xưởng:*** Phòng thực hành tin học
- ***Trang thiết bị máy móc:*** Máy chiếu và các thiết bị dạy học khác
- ***Học liệu, dụng cụ, nguyên vật liệu:*** Chương trình môn học, giáo trình, tài liệu tham khảo, giáo án, phim ảnh, và các tài liệu liên quan.
- ***Các điều kiện khác:*** Không có

❖ **KIỂM TRA VÀ ĐÁNH GIÁ CHƯƠNG 2**

- **Nội dung:**

✓ *Kiến thức: Kiểm tra và đánh giá tất cả nội dung đã nêu trong mục tiêu kiến thức*

✓ *Kỹ năng: Đánh giá tất cả nội dung đã nêu trong mục tiêu kĩ năng.*

✓ *Năng lực tự chủ và trách nhiệm: Trong quá trình học tập, người học cần:*

+ Nghiên cứu bài trước khi đến lớp

+ Chuẩn bị đầy đủ tài liệu học tập.

+ Tham gia đầy đủ thời lượng môn học.

+ Nghiêm túc trong quá trình học tập.

- **Phương pháp:**

✓ ***Điểm kiểm tra thường xuyên:*** *1 điểm kiểm tra (hình thức: hỏi miệng/ thuyết trình)*

✓ ***Điểm kiểm tra định kỳ: 1 điểm kiểm tra (hình thức: viết 1 chương trình đơn giản)***

❖ **NỘI DUNG CHƯƠNG 2**

I. Khái niệm về kiểu dữ liệu

Thông thường dữ liệu hay dùng là số và chữ. Tuy nhiên việc phân chia chỉ 2 loai dữ liệu là không đủ. Để dễ dàng hơn cho lập trình, hầu hết các NNLT đều phân chia dữ liệu thành nhiều kiểu khác nhau được gọi là các kiểu cơ bản hay chuẩn. Trên cơ sở kết hợp các kiểu dữ liệu chuẩn, học sinh có thể tự đặt ra các kiểu dữ liệu mới để phục vụ cho chương trình giải quyết bài toán của mình. Có nghĩa lúc đó mỗi đối

tượng được quản lý trong chương trình sẽ là một tập hợp nhiều thông tin hơn và được tạo thành từ nhiều loại (kiểu) dữ liệu khác nhau. Dưới đây chúng ta sẽ xét đến một số kiểu dữ liệu chuẩn được qui định sẵn bởi C++.

tên kiểu: là một từ dành riêng để chỉ định kiểu của dữ liệu.

– **số byte:** Trong bộ nhớ để lưu trữ một đơn vị dữ liệu thuộc kiểu này: Thông thường số byte này phụ thuộc vào các trình biên dịch và hệ thống máy khác nhau, ở đây ta chỉ xét đến hệ thống máy PC thông dụng hiện nay.

– **Miền giá trị của kiểu**: Cho biết một đơn vị dữ liệu thuộc kiểu này sẽ có thể lấy giá trị trong miền nào, ví dụ nhỏ nhất và lớn nhất là bao nhiêu. Hiển nhiên các giá trị này phụ thuộc vào số byte mà hệ thống máy qui định cho từng kiểu. Học sinh cần nhớ đến miền giá trị này để khai báo kiểu cho các biến cần sử dụng một cách thích hợp.

Dưới đây là bảng tóm tắt một số kiểu chuẩn đơn giản và các thông số của nó đượcsử dụng trong C++.

Bảng 1. Các loại kiểu đơn giản

Loại dữ liệu	Tên kiểu	Số ô nhớ	Miền giá trị
Kí tự	char	1 byte	- 128 .. 127
	unsigned char	1 byte	0 .. 255
Số nguyên	int	2 byte	- 32768 .. 32767
	unsigned int	2 byte	0 .. 65535
Số thực	short	2 byte	- 32768 .. 32767
	long	4 byte	$-2^{15} .. 2^{15} - 1$
	float	4 byte	$+, -10^{-37} . . +,- 10^{+38}$
	double	8 byte	$+,- 10^{-307} . .+,- 10^{+308}$

Trong chương này chúng ta chỉ xét các loại kiểu đơn giản trên đây. Các loại kiểu có cấu trúc do người dùng định nghĩa sẽ được trình bày trong các chương sau.

1. Kiểu ký tự

Một kí tự là một kí hiệu trong bảng mã ASCII. Như đã biết một số kí tự có mặt chữ trên bàn phím (ví dụ các chữ cái, chữ số) trong khi một số kí tự lại không (ví dụ kí tự biểu diễn việc lùi lại một ô trong văn bản, kí tự chỉ việc kết thúc một dòng hay kết thúc một văn bản). Do vậy để biểu diễn một kí tự người ta dùng chính mã ASCII của kítự đó trong bảng mã ASCII và thường gọi là giá trị của kí tự. Ví dụ phát biểu

"Cho kítự 'A'" là cũng tương đương với phát biểu "Cho kí tự 65" (65 là mã ASCII của kí tự'A'), hoặc "Xoá kí tự xuống dòng" là cũng tương đương với phát biểu "Xoá kí tự 13" vì 13 là mã ASCII của kí tự xuống dòng.

Theo bảng trên ta thấy có 2 loại kí tự là char với miền giá trị từ -128 đến 127 và unsigned char (kí tự không dấu) với miền giá trị từ 0 đến 255.

Ví dụ 1 :

```
char c, d ;                 // c, d được phép gán giá trị từ -128  đến 127
unsigned e ;                // e được phép gán giá trị từ 0 đến 255
c = 65 ; d = 179 ;          // d có giá trị ngoài miền cho phép
e = 179; f = 330 ;          // f có giá trị ngoài miền cho phép
cout << c << int(c) ;       // in ra chữ cái 'A' và giá trị số 65
cout << d << int(d) ;       // in ra là kí tự '|' và giá trị số -77
```

Chú ý: Qua ví dụ trên ta thấy một biến nếu được gán giá trị ngoài miền cho phép sẽ dẫn đến kết quả không theo suy nghĩ thông thường. Do vậy nên tuân thủ qui tắc chỉ gán giá trị cho biến thuộc miền giá trị mà kiểu của biến đó qui định. Ví dụ nếu muốn sửdụng biến có giá trị từ 128 .. 255 ta nên khai báo biến dưới dạng kí tự không dấu (unsigned char), còn nếu giá trị vượt quá 255 ta nên chuyển sang kiểu nguyên (int).

2. Kiểu số nguyên

Các số nguyên được phân chia thành 4 loại kiểu khác nhau với các miền giá trị tương ứng được cho trong bảng 1. Đó là kiểu số nguyên ngắn (short) tương đương với kiểu số nguyên (int) sử dụng 2 byte và số nguyên dài (long int) sử dụng 4 byte. Kiểu số nguyên thường được chia làm 2 loại có dấu (int) và không dấu (unsigned int hoặc có thể viết gọn hơn là unsigned). Qui tắc mã bù cũng được áp dụng nếu giá trị của biến vượt ra ngoài miền giá trị cho phép, vì vậy cần cân nhắc khi khai báo kiểu cho cácbiến. Ta thường sử dụng kiểu int cho các số nguyên trong các bài toán với miền giá trị vừa phải (có giá trị tuyệt đối bé hơn 32767), chẳng hạn các biến đếm trong các vòng lặp, ...

3. Kiểu số thực

Để sử dụng số thực ta cần khai báo kiểu float hoặc double mà miền giá trị của chúng được cho trong bảng 1. Các giá trị số kiểu double được gọi là số thực với độ chính xác gấp đôi vì với kiểu dữ liệu này máy tính có cách biểu diễn khác so với kiểu float để đảm bảo số số lẻ sau một số thực có thể tăng lên đảm bảo tính chính xác cao hơn so với số kiểu float. Tuy nhiên, trong các bài toán thông dụng thường ngày

độ chính xác của số kiểu float là đủ dùng.

Như đã nhắc đến trong phần các lệnh vào/ra ở chương 1, liên quan đến việc in ấn số thực ta có một vài cách thiết đặt dạng in theo ý muốn, ví dụ độ rộng tối thiểu để in một số hay số số lẻ thập phân cần in ...

Ví dụ 2 : Chương trình sau đây sẽ in diện tích và chu vi của một hình tròn có bán kính 2cm với 3 số lẻ.

```
#include <iostream.h>
#include <iomanip.h>
void main()
{
        float r = 2 ;                                   // r là tên biến dùng để chứa bán kính
         cout << "Diện tích = " <<  r * r * 3.1416 ;
         return 0;
}
```

II. HẰNG - KHAI BÁO VÀ SỬ DỤNG HẰNG

Hằng là một giá trị cố định nào đó ví dụ 3 (hằng nguyên), 'A' (hằng kí tự), 5.0 (hằng thực), "Ha noi" (hằng xâu kí tự). Một giá trị có thể được hiểu dưới nhiều kiểu khác nhau, do vậy khi viết hằng ta cũng cần có dạng viết thích hợp.

1. Hằng nguyên

– kiểu short, int: 3, -7, ...

– kiểu unsigned: 3, 123456, ...

– kiểu long, long int: 3L, -7L, 123456L, ... (viết L vào cuối mỗi giá trị)

Các cách viết trên là thể hiện của số nguyên trong hệ thập phân, ngoài ra chúng còn được viết dưới các hệ đếm khác như hệ cơ số 8 hoặc hệ cơ số 16. Một số nguyên trong cơ số 8 luôn luôn được viết với số 0 ở đầu, tương tự với cơ số 16 phải viết với 0x ở đầu. Ví dụ ta biết 65 trong cơ số 8 là 101 và trong cơ số 16 là 41, do đó 3 cách viết 65, 0101, 0x41 là như nhau, cùng biểu diễn giá trị 65.

2. Hằng thực

Một số thực có thể được khai báo dưới dạng kiểu float hoặc double và các giá trị của nó có thể được viết dưới một trong hai dạng.

a. Dạng dấu phẩy tĩnh

Theo cách viết thông thường. Ví dụ: 3.0, -7.0, 3.1416, ...

b. Dạng dấu phẩy động

Tổng quát, một số thực x có thể được viết dưới dạng: men hoặc mEn, trong đó

m được gọi là phần định trị, n gọi là phần bậc (hay mũ). Số men biểu thị giá trị x = m x 10^n. Ví dụ số π = 3.1416 có thể được viết:

π = … = 0.031416e2 = 0.31416e1 = 3.1416e0 = 31.416e-1 = 314.16e-2 = …

vì π = 0.031416 x 10^2 = 0.31416 x 10^1 = 3.1416 x 10^0 = …

Như vậy một số x có thể được viết dưới dạng mEn với nhiều giá trị m, n khác nhau, phụ thuộc vào dấu phảy ngăn cách phần nguyên và phần thập phân của số. Do vậy cách viết này được gọi là dạng dấu phảy động.

3. Hằng kí tự

a. Cách viết hằng

Có 2 cách để viết một hằng kí tự. Đối với các kí tự có mặt chữ thể hiện ta thường sử dụng cách viết thông dụng đó là đặt mặt chữ đó giữa 2 dấu nháy đơn như: 'A', '3', ' ' (dấu cách) ... hoặc sử dụng trực tiếp giá trị số của chúng. Ví dụ các giá trị tương ứng của các kí tự trên là 65, 51 và 32. Với một số kí tự không có mặt chữ ta buộc phải dùnggiá trị (số) của chúng, như viết 27 thay cho kí tự được nhấn bởi phím Escape, 13 thay cho kí tự được nhấn bởi phím Enter ...

Để biểu diễn kí tự bằng giá trị số ta có thể viết trực tiếp (không dùng cặp dấu nháy đơn) giá trị đó dưới dạng hệ số 10 (như trên) hoặc đặt chúng vào cặp dấu nháy đơn, trường hợp này chỉ dùng cho giá trị viết dưới dạng hệ 8 hoặc hệ 16 theo mẫu sau:

– '\kkk': không quá 3 chữ số trong hệ 8. Ví dụ '\11' biểu diễn kí tự có mã 9.

– '\xkk': không quá 2 chữ số trong hệ 16. Ví dụ '\x1B' biểu diễn kí tự có mã 27.

b. Một số hằng thông dụng

Đối với một số hằng kí tự thường dùng nhưng không có mặt chữ tương ứng, hoặc các kí tự được dành riêng với nhiệm vụ khác, khi đó thay vì phải nhớ giá trị của chúng ta có thể viết theo qui ước sau:

'\n': biểu thị kí tự xuống dòng (cũng tương ứng với endl)

'\t' : kí tự tab

'\a' : kí tự chuông (tức thay vì in kí tự, loa sẽ phát ra một tiếng 'bíp')

'\r' : xuống dòng

'\f' : kéo trang

'\\' : dấu \

Ví dụ:

cout << "Hôm nay trời \t nắng \a \a \a \n" ; sẽ in ra màn hình dòng chữ "Hôm nay trời" sau đó bỏ một khoảng cách bằng một tab (khoảng 8 dấu cách) rồi in tiếp chữ

"nắng", tiếp theo phát ra 3 tiếng chuông và cuối cùng con trỏ trên màn hình sẽ nhảy xuống đầu dòng mới.

Do dấu cách (phím spacebar) không có mặt chữ, nên trong một số trường hợp để tránh nhầm lẫn chúng tôi qui ước sử dụng kí hiệu <> để biểu diễn dấu cách. Ví dụ trong giáo trình này dấu cách (có giá trị là 32) được viết ' ' (dấu nháy đơn bao một dấu cách) hoặc rõ ràng hơn bằng cách viết theo qui ước <>.

4. Hằng xâu kí tự

Là dãy kí tự bất kỳ đặt giữa cặp dấu nháy kép. Ví dụ: "Lớp K43*", "12A4", "A", "<>", "" là các hằng xâu kí tự, trong đó "" là xâu không chứa kí tự nào, các xâu "<>", "A" chứa 1 kí tự ... Số các kí tự giữa 2 dấu nháy kép được gọi là độ dài của xâu. Ví dụ xâu "" có độ dài 0, xâu "<>" hoặc "A" có độ dài 1 còn xâu "Lớp K43*" có độ dài 8.

Chú ý phân biệt giữa 2 cách viết 'A' và "A", tuy chúng cùng biểu diễn chữ cái A nhưng chương trình sẽ hiểu 'A' là một kí tự còn "A" là một xâu kí tự (do vậy chúng được bố trí khác nhau trong bộ nhớ cũng như cách sử dụng chúng là khác nhau). Tương tự ta không được viết " (2 dấu nháy đơn liền nhau) vì không có khái niệm kí tự "rỗng". Để chỉ xâu rỗng (không có kí tự nào) ta phải viết "" (2 dấu nháy kép liền nhau).

5. Khai báo hằng

Một giá trị cố định (hằng) được sử dụng nhiều lần trong chương trình đôi khi sẽ thuận lợi hơn nếu ta đặt cho nó một tên gọi, thao tác này được gọi là khai báo hằng. Ví dụ một chương trình quản lý sinh viên với giả thiết số sinh viên tối đa là 50. Nếu sốsinh viên tối đa không thay đổi trong chương trình ta có thể đặt cho nó một tên gọi như sosv chẳng hạn. Trong suốt chương trình bất kỳ chỗ nào xuất hiện giá trị 50 ta đều có thể thay nó bằng sosv. Tương tự C++ cũng có những tên hằng được đặt sẵn, được gọi là các hằng chuẩn và có thể sử dụng khi cần thiết. Ví dụ hằng π được đặt sẵn trong C++ với tên gọi M_PI. Việc sử dụng tên hằng thay cho hằng có nhiều điểm thuận lợi như sau:

– Chương trình dễ đọc hơn, vì thay cho các con số ít có ý nghĩa, một tên gọi sẽ dễ hình dung vai trò, nội dung của nó. Ví dụ, khi gặp tên gọi sosv học sinh sẽ hình dung được chẳng hạn, "đây là số sinh viên tối đa trong một lớp", trong khi số 50 có thể là số sinh viên mà cũng có thể là tuổi của một sinh viên nào đó.

– Chương trình dễ sửa chữa hơn, ví dụ bây giờ nếu muốn thay đổi chương trình sao cho bài toán quản lý được thực hiện với số sinh viên tối đa là 60, khi đó ta cần

tìm và thay thế hàng trăm vị trí xuất hiện của 50 thành 60. Việc thay thế như vậy dễ gây ra lỗi vì có thể không tìm thấy hết các số 50 trong chương trình hoặc thay nhầm số 50 với ý nghĩa khác như tuổi của một sinh viên nào đó chẳng hạn. Nếu trong chương trình sử dụng hằng sosv, bây giờ việc thay thế trở nên chính xác và dễ dàng hơn bằng thao tác khai báo lại giá trị hằng sosv bằng 60. Lúc đó trong chương trình bất kỳ nơi nào gặp tên hằng sosv đều được chương trình hiểu với giá trị 60.

– Để khai báo hằng ta dùng các câu khai báo sau:

#define tên_hằng giá_trị_hằng ;

Hoặc: const tên_hằng = giá_trị_hằng ;

Ví dụ:

```
#define sosv 50 ;
#define MAX 100 ;
  const sosv = 50 ;
```

– Như trên đã chú ý một giá trị hằng chưa nói lên kiểu sử dụng của nó vì vậy ta cần khai báo rõ ràng hơn bằng cách thêm tên kiểu trước tên hằng trong khai báo const, các hằng khai báo như vậy được gọi là hằng có kiểu.

Ví dụ:

```
const int sosv = 50 ;
const float nhiet_do_soi = 100.0 ;
```

III. BIẾN - KHAI BÁO VÀ SỬ DỤNG BIẾN

1. Khai báo biến

Biến là các tên gọi để lưu giá trị khi làm việc trong chương trình. Các giá trị được lưu có thể là các giá trị dữ liệu ban đầu, các giá trị trung gian tạm thời trong quá trình tính toán hoặc các giá trị kết quả cuối cùng. Khác với hằng, giá trị của biến có thể thay đổi trong quá trình làm việc bằng các lệnh đọc vào từ bàn phím hoặc gán. Hình ảnh cụ thể của biến là một số ô nhớ trong bộ nhớ được sử dụng để lưu các giá trị của biến.

Mọi biến phải được khai báo trước khi sử dụng. Một khai báo như vậy sẽ báo cho chương trình biết về một biến mới gồm có: tên của biến, kiểu của biến (tức kiểu của giá trị dữ liệu mà biến sẽ lưu giữ). Thông thường với nhiều NNLT tất cả các biến phải được khai báo ngay từ đầu chương trình hay đầu của hàm, tuy nhiên để thuận tiện C++ cho phép khai báo biến ngay bên trong chương trình hoặc hàm, có nghĩa bất kỳ lúc nào thấy cần thiết sử dụng biến mới, họ có quyền khai báo và sử dụng nó từ đó trở đi.

Cú pháp khai báo biến gồm tên kiểu, tên biến và có thể có hay không khởi tạo giá trị ban đầu cho biến. Để khởi tạo hoặc thay đổi giá trị của biến ta dùng lệnh gán (=).

a. Khai báo không khởi tạo

tên_kiểu tên_biến_1 ;
tên_kiểu tên_biến_2 ;
tên_kiểu tên_biến_3 ;

Nhiều biến cùng kiểu có thể được khai báo trên cùng một dòng:

tên_kiểu tên_biến_1, tên_biến_2, tên_biến_3 ;

Ví dụ:

```
void main()
{
    int i, j ;                      // khai báo 2 biến i, j có kiểu nguyên
    float x ;                       // khai báo biến thực x
    char c, d[100] ;                // biến kí tự c, xâu d chứa tối đa 100 kí tự
   unsigned  int u ;                // biến nguyên không dấu u
   …
}
```

b. Khai báo có khởi tạo

Trong câu lệnh khai báo, các biến có thể được gán ngay giá trị ban đầu bởi phéptoán gán (=) theo cú pháp:

tên_kiểu tên_biến_1 = gt_1,
tên_biến_2 = gt_2,
tên_biến_3 = gt_3 ;

trong đó các giá trị gt_1, gt_2, gt_3 có thể là các hằng, biến hoặc biểu thức.

Ví dụ:

```
const int n = 10 ;
void main()
{
    int i = 2, j , k = n + 5;          // khai báo i và khởi tạo bằng 2, k bằng 15
    float eps = 1.0e-6 ;               // khai báo biến thực epsilon khởi tạo bằng 10^-6
   char c = 'Z';                       // khai báo biến kí tự c và khởi tạo bằng 'A'
  char d[100] = "Tin học";             // khai báo xâu kí tự d chứa dòng chữ "Tin học"
    …
```

}

2. Phạm vi của biến

Như đã biết chương trình là một tập hợp các hàm, các câu lệnh cũng như các khai báo. Phạm vi tác dụng của một biến là nơi mà biến có tác dụng, tức hàm nào, câu lệnh nào được phép sử dụng biến đó. Một biến xuất hiện trong chương trình có thể được sử dụng bởi hàm này nhưng không được bởi hàm khác hoặc bởi cả hai, điều này phụ thuộc chặt chẽ vào vị trí nơi biến được khai báo. Một nguyên tắc đầu tiên là biến sẽ có tác dụng kể từ vị trí nó được khai báo cho đến hết khối lệnh chứa nó.

3. Gán giá trị cho biến (phép gán)

Trong các ví dụ trước chúng ta đã sử dụng phép gán dù nó chưa được trình bày, đơn giản một phép gán mang ý nghĩa tạo giá trị mới cho một biến. Khi biến được gán giá trị mới, giá trị cũ sẽ được tự động xoá đi bất kể trước đó nó chứa giá trị nào (hoặc chưa có giá trị, ví dụ chỉ mới vừa khai báo xong). Cú pháp của phép gán như sau:

tên_biến = biểu thức ;

Khi gặp phép gán chương trình sẽ tính toán giá trị của biểu thức sau đó gán giá trịnày cho biến. Ví dụ:

```
int n, i = 3;                          // khởi tạo i=3
n = 10;                                // gán cho n giá trị 10
cout << n <<", " << i << endl;         // in ra: 10, 3
i = n / 2;                             // gán lại giá trị của i  n/2 = 5
cout << n <<", " << i << endl;         // in ra: 10, 5
```

Trong ví dụ trên n được gán giá trị bằng 10; trong câu lệnh tiếp theo biểu thức n/2 được tính (bằng 5) và sau đó gán kết quả cho biến i, tức i nhận kết quả bằng 5 dù trước đó nó đã có giá trị là 2 (trong trường hợp này việc khởi tạo giá trị 2 cho biến i là không có ý nghĩa).

Một khai báo có khởi tạo cũng tương đương với một khai báo và sau đó thêm lệnh gán cho biến (ví dụ int i = 3 cũng tương đương với 2 câu lệnh int i; i = 3) tuy nhiên về mặt bản chất khởi tạo giá trị cho biến vẫn khác với phép toán gán như ta sẽ thấy trong các phần sau.

4. Một số điểm lưu ý về phép gán

Với ý nghĩa thông thường của phép toán (nghĩa là tính toán và cho lại một giá trị) thì phép toán gán còn một nhiệm vụ nữa là trả lại một giá trị. Giá trị trả lại của phép toán gán chính là giá trị của biểu thức sau dấu bằng. Lợi dụng điều này C++ cho phép chúng ta gán "kép" cho nhiều biến nhận cùng một giá trị bởi cú pháp:

biến_1 = biến_2 = … = biến_n = gt ;

với cách gán này tất cả các biến sẽ nhận cùng giá trị gt.

Ví dụ:

int i, j, k ;

i = j = k = 1;

Biểu thức gán trên có thể được viết lại như (i = (j = (k = 1))), có nghĩa đầu tiên để thực hiện phép toán gán giá trị cho biến i chương trình phải tính biểu thức (j = (k = 1)), tức phải tính k = 1, đây là phép toán gán, gán giá trị 1 cho k và trả lại giá trị 1, giá trị trả lại này sẽ được gán cho j và trả lại giá trị 1 để tiếp tục gán cho i.

IV. PHÉP TOÁN, BIỂU THỨC VÀ CÂU LỆNH

1. Phép toán

C++ có rất nhiều phép toán loại 1 ngôi, 2 ngôi và thậm chí cả 3 ngôi. Để hệ thống, chúng tôi tạm phân chia thành các lớp và trình bày chỉ một số trong chúng. Các thành phần tên gọi tham gia trong phép toán được gọi là hạng thức hoặc toán hạng, các kí hiệu phép toán được gọi là toán tử. Ví dụ trong phép toán a + b; a, b được gọi là toán hạng và + là toán tử. Phép toán 1 ngôi là phép toán chỉ có một toán hạng, ví dụ - a (đổi dấu số a), &x (lấy địa chỉ của biến x) … Một số kí hiệu phép toán cũng được sử dụng chung cho cả 1 ngôi lẫn 2 ngôi (hiển nhiên với ngữ nghĩa khác nhau), ví dụ kí hiệu - được sử dụng cho phép toán trừ 2 ngôi a - b, hoặc phép & còn được sử dụng cho phép toán lấy hội các bit (a & b) của 2 số nguyên a và b …

a. Các phép toán số học: +, -, *, /, %

– Các phép toán + (cộng), - (trừ), * (nhân) được hiểu theo nghĩa thông thường trong số học.

– Phép toán a / b (chia) được thực hiện theo kiểu của các toán hạng, tức nếu cả hai toán hạng là số nguyên thì kết quả của phép chia chỉ lấy phần nguyên, ngược lại nếu 1 trong 2 toán hạng là thực thì kết quả là số thực. Ví dụ:

13/5 = 2 // do 13 và 5 là 2 số nguyên

13.0/5 = 13/5.0 = 13.0/5.0 = 2.6 // do có ít nhất 1 toán hạng là thực

– Phép toán a % b (lấy phần dư) trả lại phần dư của phép chia a/b, trong đó a và b là 2 số nguyên.

Ví dụ:

13%5 = 3 // Phần dư của 13/5

5%13 = 5 // Phần dư của 5/13

b. Các phép toán tự tăng, giảm: i++, ++i, i--, --i

– Phép toán ++i và i++ sẽ cùng tăng i lên 1 đơn vị tức tương đương với câu lệnh i = i+1. Tuy nhiên nếu 2 phép toán này nằm trong câu lệnh hoặc biểu thức thì

++i khác với i++. Cụ thể ++i sẽ tăng i, sau đó i mới được tham gia vào tính toán trong biểu thức. Ngược lại i++ sẽ tăng i sau khi biểu thức được tính toán xong (với giá trị i cũ). Điểm khác biệt này được minh hoạ thông qua ví dụ sau, giả sử i = 3, j = 15.

Phép toán	Tương đương	Kết quả
i = ++j ; // tăng trước	j = j + 1 ; i = j ;	i = 16 , j = 16
i = j++ ; // tăng sau	i = j ; j = j + 1 ;	i = 15 , j = 16
j = ++i + 5 ;	i = i + 1 ; j = i + 5 ;	i = 4, j = 9
j = i++ + 5 ;	j = i + 5; i = i + 1;	i = 4, j = 8

Ghi chú: Việc kết hợp phép toán tự tăng giảm vào trong biểu thức hoặc câu lệnh (như ví dụ trong phần sau) sẽ làm chương trình gọn nhưng khó hiểu hơn.

c. Các phép toán so sánh và lôgic

Đây là các phép toán mà giá trị trả lại là đúng hoặc sai. Nếu giá trị của biểu thứclà đúng thì nó nhận giá trị 1, ngược lại là sai thì biểu thức nhận giá trị 0. Nói cách khác 1 và 0 là giá trị cụ thể của 2 khái niệm "đúng", "sai". Mở rộng hơn C++ quan niệm mộtgiá trị bất kỳ khác 0 là "đúng" và giá trị 0 là "sai".

• Các phép toán so sánh:

== (bằng nhau), != (khác nhau), > (lớn hơn), < (nhỏ hơn), >= (lớn hơn hoặc bằng), <= (nhỏ hơn hoặc bằng).

Hai toán hạng của các phép toán này phải cùng kiểu. Ví dụ:

```
3 == 3 hoặc 3 == (4 -1)          // nhận giá trị 1 vì đúng
3 == 5                           // = 0 vì sai
3 != 5                           // = 1
3 + (5 < 2)                      // = 3 vì 5<2 bằng 0
3 + (5 >= 2)                     // = 4 vì 5>=2 bằng 1
```

Chú ý: cần phân biệt phép toán gán (=) và phép toán so sánh (==). Phép gán vừa gán giá trị cho biến vừa trả lại giá trị bất kỳ (là giá trị của toán hạng bên phải), trongkhi phép so sánh luôn luôn trả lại giá trị 1 hoặc 0.

• Các phép toán lôgic: **&& (và), || (hoặc), ! (không, phủ định)**

Hai toán hạng của loại phép toán này phải có kiểu lôgic tức chỉ nhận một tronghai giá trị "đúng" (được thể hiện bởi các số nguyên khác 0) hoặc "sai" (thể hiện

bởi 0). Khi đó giá trị trả lại của phép toán là 1 hoặc 0 và được cho trong bảng sau:

a	b	a && b	a || b	! a
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

Vậy:

– Phép toán "và" đúng khi và chỉ khi hai toán hạng cùng đúng

– Phép toán "hoặc" sai khi và chỉ khi hai toán hạng cùng sai

– Phép toán "không" (hoặc "phủ định") đúng khi và chỉ khi toán hạng của nó sai.

Ví dụ:

```
3 && (4 > 5)              // = 0 vì có hạng thức (4>5) sai
(3 >= 1) && (7)           // = 1 vì cả hai hạng thức cùng đúng
!1                        // = 0
! (4 + 3 < 7)             // = 1 vì (4+3<7) bằng 0
5 || (4 >= 6)             // = 1 vì có một hạng thức (5) đúng
(5 < !0) || (4 >= 6)      // = 0 vì cả hai hạng thức đều sai
```

Chú ý: việc đánh giá biểu thức được tiến hành từ trái sang phải và sẽ dừng khi biết kết quả mà không chờ đánh giá hết biểu thức. Cách đánh giá này sẽ cho những kết quả phụ khác nhau nếu trong biểu thức ta "tranh thủ" đưa thêm vào các phép toán tự tăng giảm. Ví dụ cho i = 2, j = 3, xét 2 biểu thức sau đây:

```
x = (++i < 4 && ++j > 5)      cho kết quả x = 0 , i = 3 , j = 4
y = (++j > 5 && ++i < 4)      cho kết quả y = 0 , i = 2 , j = 4
```

cách viết hai biểu thức là như nhau (ngoại trừ hoán đổi vị trí 2 toán hạng của phéptoán &&). Với giả thiết i = 2 và j = 3 ta thấy cả hai biểu thức trên cùng nhận giá trị 0. Tuy nhiên các giá trị của i và j sau khi thực hiện xong hai biểu thức này sẽ có kết quả khác nhau. Cụ thể với biểu thức đầu vì ++i < 4 là đúng nên chương trình phải tiếp tục tính tiếp ++j > 5 để đánh giá được biểu thức. Do vậy sau khi đánh giá xong cả i và jđều được tăng 1 (i=3, j=4). Trong khi đó với biểu thức sau do ++j > 5 là sai nên chương trình có thể kết luận được toàn bộ biểu thức là sai mà không cần tính tiếp ++i < 4. Có nghĩa chương trình sau khi đánh giá xong ++j > 5 sẽ dừng và vì

vậy chỉ có biến j được tăng 1, từ đó ta có i = 2, j = 4 khác với kết quả của biểu thức trên.

2. Biểu thức

Biểu thức là dãy kí hiệu kết hợp giữa các toán hạng, phép toán và cặp dấu () theo một qui tắc nhất định. Các toán hạng là hằng, biến, hàm. Biểu thức cung cấp một cách thức để tính giá trị mới dựa trên các toán hạng và toán tử trong biểu thức. Ví dụ:

(x + y) * 2 - 4 ; 3 - x + sqrt(y) ; (-b + sqrt(delta)) / (2*a) ;

a. Thứ tự ưu tiên của các phép toán

Để tính giá trị của một biểu thức cần có một trật tự tính toán cụ thể và thống nhất.

Ví dụ xét biểu thức x = 3 + 4 * 2 + 7

– Nếu tính theo đúng trật tự từ trái sang phải, ta có x = ((3+4) * 2) + 7 = 21,

– Nếu ưu tiên dấu + được thực hiện trước dấu *, x = (3 + 4) * (2 + 7) = 63,

– Nếu ưu tiên dấu * được thực hiện trước dấu +, x = 3 + (4 * 2) + 7 = 18.

Như vậy cùng một biểu thức tính x nhưng cho 3 kết quả khác nhau theo những cách hiểu khác nhau. Vì vậy cần có một cách hiểu thống nhất dựa trên thứ tự ưu tiên của các phép toán, tức những phép toán nào sẽ được ưu tiên tính trước và những phép toán nào được tính sau. C++ qui định trật tự tính toán theo các mức độ ưu tiên như sau:

1. Các biểu thức trong cặp dấu ngoặc ()
2. Các phép toán 1 ngôi (tự tăng, giảm, lấy địa chỉ, lấy nội dung con trỏ …)
3. Các phép toán số học.
4. Các phép toán quan hệ, logic.
5. Các phép gán.

Nếu có nhiều cặp ngoặc lồng nhau thì cặp trong cùng (sâu nhất) được tính trước. Các phép toán trong cùng một lớp có độ ưu tiên theo thứ tự: lớp nhân (*, /, &&), lớp cộng (+, -, ||). Nếu các phép toán có cùng thứ tự ưu tiên thì chương trình sẽ thực hiệntừ trái sang phải. Các phép gán có độ ưu tiên cuối cùng và được thực hiện từ phải sang trái.

Ví dụ: Theo mức ưu tiên đã qui định, biểu thức tính x trong ví dụ trên sẽ được tính như sau: x = 3 + (4 * 2) + 7 = 18.

Phần lớn các trường hợp muốn tính toán theo một trật tự nào đó ta nên sử dụng cụ thể các dấu ngoặc (vì các biểu thức trong dấu ngoặc được tính trước).

Ví dụ: Để tính $\Delta = b^2 - 4ac$ ta viết delta = b * b - 4 * a * c ;

Để tính nghiệm phương trình bậc 2 $x = \frac{-b+\sqrt{\Delta}}{2a}$ viết : x = - b + sqrt(delta) / 2*a; là sai vì theo mức độ ưu tiên x sẽ được tính như -b + ((sqrt(delta)/2) * a) (thứ tự tính sẽ là phép toán 1 ngôi đổi dấu -b, đến phép chia, phép nhân và cuối cùng là phép cộng). Để tính chính xác cần phải viết (-b + sqrt(delta)) / (2*a).

– Cho a = 1, b = 2, c = 3. Biểu thức a += b += c cho giá trị c = 3, b = 5, a = 6. Thứ tự tính sẽ là từ phải sang trái, tức câu lệnh trên tương đương với các câu lệnh sau:

```
a = 1 ; b = 2 ; c = 3 ;
b = b + c ;                    // b = 5
a = a + b ;                    // a = 6
```

Để rõ ràng, tốt nhất nên viết biểu thức cần tính trước trong các dấu ngoặc.

b. Phép chuyển đổi kiểu

Khi tính toán một biểu thức phần lớn các phép toán đều yêu cầu các toán hạng phải cùng kiểu. Ví dụ để phép gán thực hiện được thì giá trị của biểu thức phải có **cùng kiểu** với biến. Trong trường hợp kiểu của giá trị biểu thức khác với kiểu của phép gán thì hoặc là chương trình sẽ tự động chuyển kiểu giá trị biểu thức về thành kiểu của biếnđược gán (nếu được) hoặc sẽ báo lỗi. Do vậy khi cần thiết phải sử dụng các câu lệnh để chuyển kiểu của biểu thức cho phù hợp với kiểu của biến.

– **Chuyển kiểu tự động:** về mặt nguyên tắc, khi cần thiết các kiểu có giá trị thấp sẽ được chương trình tự động chuyển lên kiểu cao hơn cho phù hợp với phép toán. Cụ thể phép chuyển kiểu có thể được thực hiện theo sơ đồ như sau:

char ↔ int ↔ long int ↔ float ↔ double

Ví dụ:

```
int i = 3;
float f ;
f = i + 2;
```

Trong ví dụ trên i có kiểu nguyên và vì vậy i+2 cũng có kiểu nguyên trong khi f có kiểu thực. Tuy vậy phép toán gán này là hợp lệ vì chương trình sẽ tự động chuyển kiểu cuả i+2 (bằng 5) sang kiểu thực (bằng 5.0) rồi mới gán cho f.

– **Ép kiểu:** trong chuyển kiểu tự động, chương trình chuyển các kiểu từ thấp đến cao, tuy nhiên chiều ngược lại không thể thực hiện được vì nó có thể gây mất dữ liệu. Do đó nếu cần thiết phải ra lệnh cho chương trình. Ví dụ:

```
int i;
float f = 3 ;              // tự động chuyển 3 thành 3.0 và gán cho f
i = f + 2 ;                // sai vì mặc dù f + 2 = 5 nhưng không gán được cho i
```

Trong ví dụ trên để câu lệnh i = f+2 thực hiện được ta phải ép kiểu của biểu thức f+2 về thành kiểu nguyên. Cú pháp tổng quát như sau:

(tên_kiểu)biểu_thức // cú pháp cũ trong C

hoặc: **tên_kiểu(biểu_thức)** // cú pháp mới trong C++

Trong đó tên_kiểu là kiểu cần được chuyển sang. Như vậy câu lệnh trên phải được viết lại: i = int(f + 2) ; khi đó f+2 (bằng 5.0) được chuyển thành 5 và gán cho i.

Dưới đây ta sẽ xét một số ví dụ về lợi ích của việc ép kiểu.

- Phép ép kiểu từ một số thực về số nguyên sẽ cắt bỏ tất cả phần thập phân của số thực, chỉ để lại phần nguyên. Như vậy để tính phần nguyên của một số thực x ta chỉ cần ép kiểu của x về thành kiểu nguyên, có nghĩa int(x) là phần nguyên của số thực x bất kỳ.

Ví dụ để kiểm tra một số nguyên n có phải là số chính phương, ta cần tính căn bậc hai của n. Nếu căn bậc hai x của n là số nguyên thì n là số chính phương, tức nếu int(x) = x thì x nguyên và n là chính phương.

Ví dụ:
```
int n = 10 ;
    float x = sqrt(n);   // hàm sqrt(n) trả về giá trị căn bậc 2 của n.
    if (int(x) == x)
       cout << "n chính phương" ;
    else
      cout << "n không chính phương" ;
```

- Để biết mã ASCII của một kí tự ta chỉ cần chuyển kí tự đó sang kiểu nguyên.

```
char c ; cin >> c ;
cout << "Mã của kí tự vừa nhập là " << int(c) ;
```

Ghi chú: Xét ví dụ sau:

```
int i = 3 , j = 5 ;
float x ;
x = i / j * 10;                    // x = 6 ?
```

cout << x ; trong ví dụ này mặc dù x được khai báo là thực nhưng kết quả in ra sẽ là 0 thay vì 6 như mong muốn. Lý do là vì phép chia giữa 2 số nguyên i và j sẽ cho lại số nguyên, tức i/j = 3/5 = 0. Từ đó x = 0*10 = 0. Để phép chia ra kết quả thực ta cần phải ép kiểu hoặc i hoặc j hoặc cả 2 thành số thực, khi đó phép chia sẽ cho kết

quả thực và x được tính đúng giá trị. Cụ thể câu lệnh x = i/j*10 được đổi thành:

```
x = float(i) / j * 10 ;                    // đúng
x = i / float(j) * 10;                     // đúng
x = float(i) / float(j) * 10 ;             //đúng
x = float(i/j) * 10 ;                      //sai
```

Phép ép kiểu: x = float(i/j) * 10 ; vẫn cho kết quả sai vì trong dấu ngoặc phép chia i/j vẫn là phép chia nguyên, kết quả x vẫn là 0.

3. Câu lệnh và khối lệnh

Một **câu lệnh** trong C++ được thiết lập từ các từ khoá và các biểu thức và luôn luôn được kết thúc bằng dấu chấm phẩy. Các ví dụ vào/ra hoặc các phép gán tạo thành những câu lệnh đơn giản như:

```
cin >> x >> y ;
x = 3 + x ;
y = (x = sqrt(x)) + 1 ;
cout << x ;
cout << y ;
```

Các câu lệnh được phép viết trên cùng một hoặc nhiều dòng. Một số câu lệnh được gọi là lệnh có cấu trúc, tức bên trong nó lại chứa dãy lệnh khác. Dãy lệnh này phải được bao giữa cặp dấu ngoặc {} và được gọi là *khối lệnh*. Ví dụ tất cả các lệnh trong một hàm (như hàm main()) luôn luôn là một khối lệnh. Một đặc điểm của khối lệnh là các biến được khai báo trong khối lệnh nào thì chỉ có tác dụng trong khối lệnh đó. Chi tiết hơn về các đặc điểm của lệnh và khối lệnh sẽ được trình bày trong các chương tiếp theo của giáo trình.

V. THƯ VIỆN CÁC HÀM TOÁN HỌC

Trong phần này chúng tôi tóm tắt một số các hàm toán học hay dùng. Các hàm này đều được khai báo trong file nguyên mẫu math.h.

1. Các hàm số học

- abs(x), labs(x), fabs(x) : trả lại giá trị tuyệt đối của một số nguyên, số nguyên dài và số thực.
- pow(x, y) : hàm mũ, trả lại giá trị x lũy thừa y (x^y).
- exp(x) : hàm mũ, trả lại giá trị e mũ x (e^x).
- log(x), log10(x) : trả lại lôgarit cơ số e và lôgarit thập phân của x (lnx, logx) .
- sqrt(x) : trả lại căn bậc 2 của x.
- atof(s_number) : trả lại số thực ứng với số viết dưới dạng xâu kí tự s_number.

2. Các hàm lượng giác

- sin(x), cos(x), tan(x) : trả lại các giá trị sinx, cosx, tgx.

BÀI TẬP

1. Viết câu lệnh khai báo biến để lưu các giá trị sau:

– Tuổi của một người Số lượng cây trong thành phố

– Độ dài cạnh một tam giác Khoảng cách giữa các hành tinh

– Một chữ số Nghiệm x của phương trình bậc 1

– Một chữ cái Biệt thức Δ của phương trình bậc 2

2. Viết câu lệnh nhập vào 4 giá trị lần lượt là số thực, nguyên, nguyên dài và kí tự. Inra màn hình các giá trị này để kiểm tra.

3. Viết câu lệnh in ra màn hình các dòng sau (không kể các số thứ tự và dấu: ở đầumỗi dòng)

1: Bộ Giáo dục và Đào tạo Cộng hoà xã hội chủ nghĩa Việt Nam

2: Sở Giáo dục Hà Nội Độc lập - Tự do - Hạnh phúc

Chú ý: khoảng trống giữa chữ Đào tạo và Cộng hoà (dòng 1) là 2 tab. Dòng 2: để trống.

4. Viết chương trình nhập vào một kí tự. In ra kí tự đó và mã ascii của nó.

5. Viết chương trình nhập vào hai số thực. In ra hai số thực đó với 2 số lẻ và cách nhau 5 cột.

6. Nhập, chạy và giải thích kết quả đạt được của đoạn chương trình sau:

```
#include <iostream.h>
void main() {
char c1 = 200;
unsigned char c2 = 200 ;
cout << "c1 = " << c1 << ", c2 = " << c2 << "\n" ;
cout << "c1+100 = " << c1+100 << ", c2+100 = " << c2+100 ;
}
```

7. In ra tổng, tích, hiệu và thương của 2 số được nhập vào từ bàn phím.

8. In ra trung bình cộng, trung bình nhân của 3 số được nhập vào từ bàn phím.

9. Viết chương trình nhập cạnh, bán kính và in ra diện tích, chu vi của các hình: vuông, chữ nhật, tròn.

CHƯƠNG 3
CẤU TRÚC ĐIỀU KHIỂN TRONG C++

❖ GIỚI THIỆU CHƯƠNG 3

Chương 3 là chương mô tả về các cấu trúc điều khiển cơ bản trong C++, nhằm giúp học sinh hiểu được các câu lệnh rẽ nhánh trong C++ và áp dụng được vào chương trình cụ thể.

❖ MỤC TIÊU CHƯƠNG 3

Sau khi học xong chương này, người học có khả năng:

➢ ***Về kiến thức:***

- Trình bày và giải thích được một số khái niệm về các loại cấu trúc điều khiển trong C++.

- Vận dụng được các cấu trúc này vào các phép toán vào trong chương trình C++.

➢ ***Về kỹ năng:***

- Nhận diện được các loại cấu trúc điều khiển cần khai báo và sử dụng trong C++.

- Thực hiện được phép toán trong thực tế.

- Lựa chọn được phương pháp giải thuật logic trong tổ chức chương trình C++.

➢ ***Về năng lực tự chủ và trách nhiệm:***

- Ý thức được tầm quan trọng và ý nghĩa thực tiễn của các kiểu cấu trúc điều khiển trong C++.

- Tuân thủ nội quy, quy định nơi làm việc.

❖ PHƯƠNG PHÁP GIẢNG DẠY VÀ HỌC TẬP CHƯƠNG 3

- Đối với người dạy: sử dụng phương pháp giảng giảng dạy tích cực (diễn giảng, vấn đáp, dạy học theo vấn đề); yêu cầu người học thực hiện câu hỏi thảo luận và bài tập chương 3 (cá nhân hoặc nhóm).

- Đối với người học: chủ động đọc trước giáo trình (chương 3) trước buổi học; hoàn thành đầy đủ câu hỏi thảo luận và bài tập tình huống chương 3 theo cá nhân hoặc nhóm và nộp lại cho người dạy đúng thời gian quy định.

❖ ĐIỀU KIỆN THỰC HIỆN BÀI GIẢNG CHƯƠNG 3

- *Đối với người dạy: sử dụng phương pháp giảng giảng dạy tích cực (diễn giảng, vấn đáp, dạy học theo vấn đề); yêu cầu người học thực hiện câu hỏi thảo luận và bài tập chương 3 (cá nhân hoặc nhóm).*

- *Đối với người học: chủ động đọc trước giáo trình (chương 3) trước buổi học; hoàn thành đầy đủ câu hỏi thảo luận và bài tập tình huống chương 3 theo cá nhân hoặc nhóm và nộp lại cho người dạy đúng thời gian quy định.*

❖ **ĐIỀU KIỆN THỰC HIỆN CHƯƠNG 3**

- ***Phòng học chuyên môn hóa/nhà xưởng:*** Phòng thực hành tin học
- ***Trang thiết bị máy móc:*** Máy chiếu và các thiết bị dạy học khác
- ***Học liệu, dụng cụ, nguyên vật liệu:*** Chương trình môn học, giáo trình, tài liệu tham khảo, giáo án, phim ảnh, và các tài liệu liên quan.
- ***Các điều kiện khác:*** Không có

❖ **KIỂM TRA VÀ ĐÁNH GIÁ CHƯƠNG 3**

- **Nội dung:**

✓ *Kiến thức: Kiểm tra và đánh giá tất cả nội dung đã nêu trong mục tiêu kiến thức*

✓ *Kỹ năng: Đánh giá tất cả nội dung đã nêu trong mục tiêu kĩ năng.*

✓ *Năng lực tự chủ và trách nhiệm: Trong quá trình học tập, người học cần:*

+ Nghiên cứu bài trước khi đến lớp

+ Chuẩn bị đầy đủ tài liệu học tập.

+ Tham gia đầy đủ thời lượng môn học.

+ Nghiêm túc trong quá trình học tập.

- **Phương pháp:**

✓ ***Điểm kiểm tra thường xuyên:*** *1 điểm kiểm tra (hình thức: hỏi miệng/ thuyết trình)*

✓ ***Điểm kiểm tra định kỳ: 1 điểm kiểm tra (hình thức: viết 1 chương trình giả phương trình bậc 2 đơn giản)***

❖ **NỘI DUNG CHƯƠNG 3**

I. CẤU TRÚC RẼ NHÁNH

Nói chung việc thực hiện chương trình là hoạt động tuần tự, tức thực hiện từng lệnh một từ câu lệnh bắt đầu của chương trình cho đến câu lệnh cuối cùng. Tuy nhiên, để việc lập trình hiệu quả hơn hầu hết các NNLT bậc cao đều có các câu lệnh rẽ nhánh và các câu lệnh lặp cho phép thực hiện các câu lệnh của chương trình không theo trình tự tuần tự như trong văn bản. Phần này chúng tôi sẽ trình bày các câu lệnh cho phép rẽ nhánh như vậy. Để thống nhất mỗi câu lệnh được trình bày về cú pháp (tức cách viết câu lệnh), cách sử dụng, đặc điểm, ví dụ minh hoạ và một vài điều cần chú ý khi sử dụng lệnh.

1. Câu lệnh điều kiện if

a. Cú pháp

if (điều kiện) { khối lệnh 1; }

hoặc:

```
if (điều kiện) {
        khối lệnh 1;
} else {
        khối lệnh 2;
}
```

Trong cú pháp trên câu lệnh if có hai dạng: có else và không có else. Điều kiện là một biểu thức lôgic tức nó có giá trị đúng (khác 0) hoặc sai (bằng 0).

Khi chương trình thực hiện câu lệnh if nó sẽ tính biểu thức điều kiện. Nếu điều kiện đúng chương trình sẽ tiếp tục thực hiện các lệnh trong khối lệnh 1, ngược lại nếu điều kiện sai chương trình sẽ thực hiện khối lệnh 2 (nếu có else) hoặc không làm gì (nếu không có else).

b. Ý nghĩa

Một câu lệnh **if** cho phép chương trình có thể thực hiện khối lệnh này hay khối lệnh khác phụ thuộc vào một điều kiện được viết trong câu lệnh là đúng hay sai. Nói cách khác câu lệnh **if** cho phép chương trình rẽ nhánh (chỉ thực hiện 1 trong 2 nhánh).

c. Đặc điểm

– Đặc điểm chung của các câu lệnh có cấu trúc là bản thân nó chứa các câu lệnh khác. Điều này cho phép các câu lệnh if có thể lồng nhau.

– Nếu nhiều câu lệnh if (có else và không else) lồng nhau việc hiểu if và else nào đi với nhau cần phải chú ý. Qui tắc là else sẽ đi với if gần nó nhất mà chưa được ghép cặp với else khác.

– Ví dụ câu lệnh:

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

là tương đương với

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

d. Ví dụ minh hoạ

Ví dụ 1 : Bằng phép toán gán có điều kiện có thể tìm số lớn nhất max trong 2 số a, b như sau:

```
if (a > b) max = a;
else max = b;
```

Ví dụ 2 : Giải phương trình bậc 2. Cho phương trình $ax^2 + bx + c = 0$ ($a \neq 0$), tìm x?

```
#include <iostream.h>                          // tệp chứa các phương thức vào/ra
#include <math.h>                              // tệp chứa các hàm toán họcvoid
main() {
   float a, b, c;                              // khai báo các hệ sốfloat delta;
   float x1, x2;                               // 2 nghiệm
   cout << “Nhap a, b, c:\n” ; cin >> a >> b >> c ;      // qui ước nhập a ≠ 0
   delta = b*b - 4*a*c ;
   if (delta < 0)
        cout << “ph. trình vô nghiệm\n” ;
   else if (delta==0)
        cout<<“ph. trình có nghiệm kép:" << -b/(2*a) << '\n';
   else{
        x1 = (-b+sqrt(delta))/(2*a);
        x2 = (-b-sqrt(delta))/(2*a);
        cout << “nghiem 1 = " << x1 << " và nghiem 2 = " << x2 ;
          }
}
```

Chú ý: do C++ quan niệm "đúng" là một giá trị khác 0 bất kỳ và "sai" là giá trị 0 nên thay vì viết if (x != 0) hoặc if (x == 0) ta có thể viết gọn thành if (x) hoặc if (!x) vì nếu (x != 0) đúng thì ta có x 0 và vì x ≠ 0 nên (x) cũng đúng. Ngược lại nếu (x) đúng thì x ≠ 0, từ đó (x != 0) cũng đúng. Tương tự ta dễ dàng thấy được (x == 0) là tương đương với (!x).

2. Câu lệnh lựa chọn switch (HS tự tìm hiểu)

a. Ý nghĩa

Câu lệnh if cho ta khả năng được lựa chọn một trong hai nhánh để thực hiện, do đó nếu sử dụng nhiều lệnh if lồng nhau sẽ cung cấp khả năng được rẽ theo nhiềunhánh. Tuy nhiên trong trường hợp như vậy chương trình sẽ rất khó đọc, do vậy C++ còn cung cấp một câu lệnh cấu trúc khác cho phép chương trình có thể chọn một

trong nhiều nhánh để thực hiện, đó là câu lệnh switch.

b. Cú pháp

```
switch (biểu thức điều khiển)
{
        case biểu_thức_1: dãy lệnh 1 ;
        case biểu_thức_2: dãy lệnh 2 ;
        case biểu_thức_n: dãy lệnh n ;
        default: dãy lệnh n+1;
}
```

– biểu thức điều khiển: phải có kiểu nguyên hoặc kí tự,

– các biểu_thức_i: được tạo từ các hằng nguyên hoặc kí tự,

– các dãy lệnh có thể rỗng. Không cần bao dãy lệnh bởi cặp dấu {},

– nhánh default có thể có hoặc không và vị trí của nó có thể nằm bất kỳ trong câu lệnh (giữa các nhánh case), không nhất thiết phải nằm cuối cùng.

c. Cách thực hiện

Để thực hiện câu lệnh **switch** đầu tiên chương trình tính giá trị của biểu thức điều khiển (btđk), sau đó so sánh kết quả của btđk với giá trị của các biểu_thức_i bên dưới lần lượt từ biểu thức đầu tiên (thứ nhất) cho đến biểu thức cuối cùng (thứ n), nếu giá trị của btđk bằng giá trị của biểu thức thứ i đầu tiên nào đó thì chương trình sẽ thực hiện dãy lệnh thứ i và tiếp tục thực hiện tất cả dãy lệnh còn lại (từ dãy lệnh thứ i+1) cho đến hết (gặp dấu ngoặc đóng } của lệnh switch). Nếu quá trình so sánh không gặp biểu thức (nhánh case) nào bằng với giá trị của btđk thì chương trình thực hiện dãy lệnh trong default và tiếp tục cho đến hết (sau default có thể còn những nhánh case khác). Trường hợp câu lệnh switch không có nhánh default và btđk không khớp với bất cứ nhánh case nào thì chương trình không làm gì, coi như đã thực hiện xong lệnh switch.

Nếu muốn lệnh switch chỉ thực hiện nhánh thứ i (khi btđk = biểu_thức_i) mà không phải thực hiện thêm các lệnh còn lại thì cuối dãy lệnh thứ i thông thường ta đặt thêm lệnh **break;** đây là lệnh cho phép thoát ra khỏi một lệnh cấu trúc bất kỳ.

d. Ví dụ minh hoạ

Ví dụ 1 : In số ngày của một tháng bất kỳ nào đó được nhập từ bàn phím.

```
int th;
cout << "Cho biết tháng cần tính: " ; cin >> th ;
switch (th)
{
```

```
case 1: case 3: case 5: case 7: case 8: case 10: case 12:
cout << "tháng này có 31 ngày" ;
break ;
case 2: cout << "tháng này có 28 ngày" ;
break; case 4: case 6: case 9:
case 11: cout << "tháng này có 30 ngày" ;
break;
default: cout << "Bạn đã nhập sai tháng, không có tháng này" ;
}
```

Trong chương trình trên giả sử nhập tháng là 5 thì chương trình bắt đầu thực hiện dãy lệnh sau case 5 (không có lệnh nào) sau đó tiếp tục thực hiện các lệnh còn lại, cụ thể là bắt đầu từ dãy lệnh trong case 7, đến case 12 chương trình gặp lệnh in kết quả "tháng này có 31 ngày", sau đó gặp lệnh break nên chương trình thoát ra khỏi câu lệnh switch (đã thực hiện xong). Việc giải thích cũng tương tự cho các trường hợp khác của tháng. Nếu nhập sai tháng (ví dụ tháng nằm ngoài phạm vi 1..12), chương trình thấy th không khớp với bất kỳ nhánh case nào nên sẽ thực hiện câu lệnh trong default, in ra màn hình dòng chữ "Bạn đã nhập sai tháng, không có tháng này" và kết thúc lệnh.

II. CẤU TRÚC LẶP

Một trong những cấu trúc quan trọng của lập trình cấu trúc là các câu lệnh cho phép lặp nhiều lần một đoạn lệnh nào đó của chương trình. Chẳng hạn trong ví dụ về bài toán nhân theo phương pháp Ấn độ, để lặp lại một đoạn lệnh chúng ta đã sử dụng câu lệnh goto. Tuy nhiên như đã lưu ý việc dùng nhiều câu lệnh này làm chương trình rất khó đọc. Do vậy cần có những câu lệnh khác trực quan hơn và thực hiện các phép lặp một cách trực tiếp. C++ cung cấp cho chúng ta 3 lệnh lặp như vậy. Mỗi bài toán lặp có một đặc trưng riêng, ví dụ lặp cho đến khi đã đủ số lần định trước thì dừng hoặc lặp cho đến khi một điều kiện nào đó không còn thoả mãn nữa thì dừng … việc sử dụng câu lệnh lặp phù hợp sẽ làm cho chương trình dễ đọc và dễ bảo trì hơn. Đây là ý nghĩa chung của các câu lệnh lặp, do vậy trong các trình bày về câu lệnh tiếp theo sau đây chúng ta sẽ không cần phải trình bày lại ý nghĩa của chúng.

1. Lệnh lặp for

a. Cú pháp

for (dãy biểu thức 1 ; điều kiện lặp ; dãy biểu thức 2) { khối lệnh lặp; }

– Các biểu thức trong các dãy biểu thức 1, 2 cách nhau bởi dấu phảy (,). Có thể

có nhiều biểu thức trong các dãy này hoặc dãy biểu thức cũng có thể trống.

– Điều kiện lặp: là biểu thức lôgic (có giá trị đúng, sai).

– Các dãy biểu thức và/hoặc điều kiện có thể trống tuy nhiên vẫn giữ lại các dấu chấm phảy (;) để ngăn cách các thành phần với nhau.

b. Cách thực hiện

Khi gặp câu lệnh **for** trình tự thực hiện của chương trình như sau:

• Thực hiện dãy biểu thức 1 (chủ yếu là các lệnh khởi tạo cho một sốbiến),

• Kiểm tra điều kiện lặp, nếu đúng thì thực hiện khối lệnh lặp → thực hiện dãy biểu thức 2 → quay lai kiểm tra điều kiện lặp và lặp lại quá trình trên cho đến bước nào đó việc kiểm tra điều kiện lặp cho kết quả sai thì dừng.

Tóm lại, biểu thức 1 sẽ được thực hiện 1 lần duy nhất ngay từ đầu quá trình lặp sau đó thực hiện các câu lệnh lặp và dãy biểu thức 2 cho đến khi nào không còn thoả điều kiện lặp nữa thì dừng.

c. Ví dụ minh hoạ

Ví dụ 1 : Nhân 2 số nguyên theo phương pháp Ấn độ

```
void main() {
    long m, n, kq;                          // Các số cần nhân và kết quả kq
    cout << "Nhập m và n: " ; cin >> m >> n ;
    for (kq = 0 ; m ; m >>= 1, n <<= 1)
    if (m%2)
    kq += n ;
    cout << "m nhân n =" << kq ;
}
```

So sánh ví dụ này với ví dụ dùng goto ta thấy chương trình được viết rất gọn. Để bạn đọc dễ hiểu câu lệnh for, một lần nữa chúng ta nhắc lại cách hoạt động của nó thông qua ví dụ này, trong đó các thành phần được viết trong cú pháp là như sau:

– Dãy biểu thức 1: kq = 0,

– Điều kiện lặp: m. Ở đây điều kiện là đúng nếu m ≠ 0 và sai nếu m = 0.

– Dãy biểu thức 2: m >>= 1 và n <<= 1. 2 biểu thức này có nghĩa m = m >> 1 (tương đương với m = m / 2) và n = n << 1 (tương đương với n = n * 2).

– Khối lệnh lặp: chỉ có một lệnh duy nhất if (m%2) kq += n ; (nếu phần dư của m chia 2 là khác 0, tức m lẻ thì cộng thêm n vào kq).

Cách thực hiện của chương trình như sau:

• Đầu tiên thực hiện biểu thức 1 tức gán kq = 0. Chú ý rằng nếu kq đã được

khởi tạo trước bằng 0 trong khi khai báo (giống như trong ví dụ 6) thì thành phần biểu thức 1 ở đây có thể để trống (nhưng vẫn giữ lại dấu ; để phân biệt với các thành phần khác).

- Kiểm tra điều kiện: giả sử m ≠ 0 (tức điều kiện đúng) for sẽ thực hiện lệnh lặp tức kiểm tra nếu m lẻ thì cộng thêm n vào cho kq.
- Quay lại thực hiện các biểu thức 2 tức chia đôi m và nhân đôi n và vòng lặp được tiếp tục lại bắt đầu bằng việc kiểm tra m …
- Đến một bước lặp nào đó m sẽ bằng 0 (vì bị chia đôi liên tiếp), điều kiện không thoả, vòng lặp dừng và cho ta kết quả là kq.

Ví dụ 2 : Tính tổng của dãy các số từ 1 đến 100.

Chương trình dùng một biến đếm i được khởi tạo từ 1, và một biến kq để chứa tổng. Mỗi bước lặp chương trình cộng i vào kq và sau đó tăng i lên 1 đơn vị. Chương trình còn lặp khi nào i còn chưa vượt qua 100. Khi i lớn hơn 100 chương trình dừng. Sau đây là văn bản chương trình.

```
void main() {
int i, kq = 0;
for (i = 1 ; i <= 100 ; i ++)
kq += i ;
cout << "Tổng = " << kq;
}
```

d. Đặc điểm

Thông qua phần giải thích cách hoạt động của câu lệnh for ta có thể thấy các thành phần của for có thể để trống, tuy nhiên các dấu chấm phẩy vẫn giữ lại để ngăn cách các thành phần với nhau.

Do đó, việc sử dụng dạng viết nào của for phụ thuộc vào thói quen của học sinh, tuy nhiên việc viết đầy đủ các thành phần của for làm cho việc đọc chương trình trở nên dễ dàng hơn.

e. Lệnh for lồng nhau

Trong dãy lệnh lặp có thể chứa cả lệnh for, tức các lệnh for cũng được phép lồng nhau như các câu lệnh có cấu trúc khác.

Ví dụ 4 : Bài toán cổ: vừa gà vừa chó bó lại cho tròn đếm đủ 100 chân. Hỏi có mấy gà và mấy con chó, biết tổng số con là 36.

Để giải bài toán này ta gọi g là số gà và c là số chó. Theo điều kiện bài toán ta thấy g có thể đi từ 0 (không có con nào) và đến tối đa là 50 (vì chỉ có 100 chân), tương tự c có thể đi từ 0 đến 25. Như vậy ta có thể cho g chạy từ 0 đến 50 và với

mỗi giá trịcụ thể của g lại cho c chạy từ 0 đến 25, lần lượt với mỗi cặp (g, c) cụ thể đó ta kiểm tra 2 điều kiện: g + c == 36 ? (số con) và 2g + 4c == 100 ? (số chân). Nếu cả 2 điều kiện đều thoả thì cặp (g, c) cụ thể đó chính là nghiệm cần tìm. Từ đó ta có chương trình với2 vòng for lồng nhau, một vòng for cho g và một vòng cho c.

```
void main() {
int g, c ;
for (g = 0 ; g <= 50 ; g++)
for (c = 0 ; c <= 25 ; C++)
if (g+c == 36 && 2*g+4*c == 100)
cout << "gà=" << g << ", chó=" << c ;
}
```

Chương trình trên có thể được giải thích một cách ngắn gọn như sau: Đầu tiên cho g = 0, thực hiện lệnh for bên trong tức lần lượt cho c = 0, 1, …, 25, với c=0 và g=0 kiểm tra điều kiện, nếu thoả thì in kết quả nếu không thì bỏ qua, quay lại tăng c, cho đến khi nào c>25 thì kết thúc vòng lặp trong quay về vòng lặp ngoài tăng g lên 1, lại thực hiện vòng lặp trong với g=1 này (tức lại cho c chạy từ 0 đến 25). Khi g của vòng lặp ngoài vượt quá 50 thì dừng. Từ đó ta thấy số vòng lặp của chương trình là 50 x 25 = 1000 lần lặp.

Chú ý: Có thể giảm bớt số lần lặp bằng nhận xét số gà không thể vượt quá 36 (vì tổng số con là 36). Một vài nhận xét khác cũng có thể làm giảm số vòng lặp, tiết kiệm thời gian chạy của chương trình. Bạn đọc tự nghĩ thêm các phương án giải khác để giảm số vòng lặp đến ít nhất.

2. Lệnh lặp while (Tự tìm hiểu)

a. Cú pháp

while (điều kiện) { khối lệnh lặp ; }

b. Thực hiện

Khi gặp lệnh while chương trình thực hiện như sau: đầu tiên chương trình sẽ kiểm tra điều kiện, nếu đúng thì thực hiện khối lệnh lặp, sau đó quay lại kiểm tra điều kiện và tiếp tục. Nếu điều kiện sai thì dừng vòng lặp. Tóm lại có thể mô tả một cách ngắn gọn về câu lệnh while như sau: lặp lại các lệnh trong khi điều kiện vẫn còn đúng.

***c.* Đặc điểm**

Khối lệnh lặp có thể không được thực hiện nếu điều kiện sai ngay từ đầu.

Để vòng lặp không lặp vô hạn thì trong khối lệnh thông thường phải có ít nhất một câu lệnh nào đó gây ảnh hưởng đến kết quả của điều kiện, ví dụ làm cho điều

kiện đang đúng trở thành sai.

Nếu điều kiện luôn luôn nhận giá trị đúng (ví dụ biểu thức điều kiện là 1) thì trong khối lệnh lặp phải có câu lệnh kiểm tra dừng và lệnh break.

d. Ví dụ minh hoạ

Ví dụ 1 : Nhân 2 số nguyên theo phương pháp Ấn độ

```
void main() {
     long m, n, kq;                          // Các số cần nhân và kết quả kq
    cout << "Nhập m và n: " ; cin >> m >> n ;
    kq = 0 ;
while (m) {
            if (m%2) kq += n ;
            m >>= 1;
            n <<= 1;
           }
        cout << "m nhân n =" << kq ;
}
```

Trong chương trình trên câu lệnh while (m) … được đọc là "trong khi m còn khác0 thực hiện …", ta thấy trong khối lệnh lặp có lệnh m >>= 1, lệnh này sẽ ảnh hưởngđến điều kiện (m), đến lúc nào đó m bằng 0 tức (m) là sai và chương trình sẽ dừng lặp.

Ví dụ 2 : Tìm ước chung lớn nhất (UCLN) của 2 số nguyên m và n.

Áp dụng thuật toán Euclide bằng cách liên tiếp lấy số lớn trừ đi số nhỏ khi nào 2 số bằng nhau thì đó là UCLN. Trong chương trình ta qui ước m là số lớn và n là sốnhỏ. Thêm biến phụ r để tính hiệu của 2 số. Sau đó đặt lại m hoặc n bằng r sao cho m >n và lặp lại. Vòng lặp dừng khi m = n.

```
void main(){
int m, n, r;
cout << "Nhập m, n: " ; cin >> m >> n ;
if (m < n) {  // nếu m < n thì đổi vai trò hai số
   int t = m;
   m = n;
    n = t;
  }
while (m != n) {
r = m - n ;
```

```
if (r > n)
        m = r;
else {
         m = n ; n = r ;
       }
    }
cout << "UCLN = " << m ;
}
```

BÀI TẬP

Lệnh rẽ nhánh

1. Nhập một kí tự. Cho biết kí tự đó có phải là chữ cái hay không.

2. Nhập vào một số nguyên. Trả lời số nguyên đó: âm hay dương, chẵn hay lẻ ?

3. Cho n = x = y và bằng: a. 1 b. 2 c. 3 d. 4

4. Hãy cho biết giá trị của x, y sau khi chạy xong câu lệnh:

```
if (n % 2 == 0)
if (x > 3) x = 0;
else y = 0;
```

5. Nhập 2 số a, b. In ra max, min của 2 số đó. Mở rộng với 3 số, 4 số ?

6. Nhập 3 số a, b, c. Hãy cho biết 3 số trên có thể là độ dài 3 cạnh của một tam giác. Nếu là một tam giác thì đó là tam giác gì: vuông, đều, cân, vuông cân hay tam giácthường ?

7. Nhập vào một số, in ra thứ tương ứng với số đó (qui ước 2 là thứ hai, …, 8 là chủnhật).

8. Nhập 2 số biểu thị tháng và năm. In ra số ngày của tháng năm đó (có kiểm tra nămnhuận).

9. Lấy ngày tháng hiện tại làm chuẩn. Hãy nhập một ngày bất kỳ trong tháng. Cho biết thứ của ngày vừa nhập ?

Lệnh lặp

10. Giá trị của i bằng bao nhiêu sau khi thực hiện cấu trúc for sau:

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

11. Giá trị của x bằng bao nhiêu sau khi thực hiện cấu trúc for sau:

```
for (x = 2; i < 10; x+=3) ;
```

12. Bạn bổ sung gì vào lệnh for sau:

```
for ( ; nam < 1997 ; ) ;
```

để khi kết thúc nam có giá trị 2000.

13. Bao nhiêu kí tự ‘X’ được in ra màn hình khi thực hiện đoạn chương trình sau:

```
for (x = 0; x < 10; x ++)
for (y = 5; y > 0; y --)
cout << ‘X’;
```

14. Nhập vào tuổi cha và tuổi con hiện nay sao cho tuổi cha lớn hơn 2 lần tuổi con. Tìm xem bao nhiêu năm nữa tuổi cha sẽ bằng đúng 2 lần tuổi con (ví dụ 30 và 12, sau 6 năm nữa tuổi cha là 36 gấp đôi tuổi con là 18).

15. Nhập số tự nhiên n. In ra màn hình biểu diễn của n ở dạng nhị phân.

16. In ra màn hình các số có 2 chữ số sao cho tích của 2 chữ số này bằng 2 lần tổngcủa 2 chữ số đó (ví dụ số 36 có tích 3*6 = 18 gấp 2 lần tổng của nó là 3 + 6 = 9).

17. Số *hoàn chỉnh* là số bằng tổng mọi ước của nó (không kể chính nó). Ví dụ 6 = 1 +2 + 3 là một số hoàn chỉnh. Hãy in ra màn hình tất cả các số hoàn chỉnh < 1000.

18. Các số *sinh đôi* là các số nguyên tố mà khoảng cách giữa chúng là 2. Hãy in tất cảcặp số sinh đôi < 1000.