

MỤC LỤC

Chương 1. Tổng quan.....	5
1.1 Khái niệm về chương trình và ngôn ngữ lập trình	5
1.2 Ngôn ngữ lập trình C++	8
1.3 Môi trường lập trình của C++	9
1.4 Phần mềm và phần cứng.....	10
<i>1.4.1 Phần mềm</i>	<i>10</i>
<i>1.4.2 Phần cứng.....</i>	<i>11</i>
1.5 Thuật toán	12
Bài tập chương 1	17
Chương 2. Các khái niệm cơ bản trong C++	19
2.1 Các thành phần cơ bản của C++	19
<i>2.1.1 Bộ ký tự.....</i>	<i>19</i>
<i>2.1.2 Định danh và từ khóa</i>	<i>19</i>
<i>2.1.3 Câu lệnh.....</i>	<i>20</i>
2.2 Cấu trúc của một chương trình trong C++	21
2.3 Các kiểu dữ liệu và cách sử dụng	24
<i>2.3.1 Khái niệm về kiểu dữ liệu</i>	<i>24</i>
<i>2.3.2 Kiểu dữ liệu cơ sở.....</i>	<i>24</i>
2.4 Biến và cách khai báo biến.....	26
<i>2.4.1 Cách khai báo biến.....</i>	<i>27</i>
<i>2.4.2 Phạm vi hoạt động của các biến.....</i>	<i>28</i>
2.5 Biểu thức và các phép toán	30
<i>2.5.1 Các phép toán cơ bản trong C++</i>	<i>30</i>
<i>2.5.2 Biểu thức.....</i>	<i>33</i>
<i>2.5.3 Một số hàm toán học trong C++.....</i>	<i>33</i>
Bài tập chương 2	36

Chương 3. Các câu lệnh điều kiện	41
3.1 Giới thiệu.....	41
3.2 Câu lệnh if.....	42
3.3 Một số ví dụ về câu lệnh if	45
3.4 Cấu trúc switch.....	49
Bài tập chương 3.....	52
Chương 4. Các câu lệnh lặp	58
4.1 Giới thiệu.....	58
4.2 Câu lệnh while	58
4.3 Câu lệnh for	63
4.4 Câu lệnh do-while	67
4.5 Sự khác nhau giữa các câu lệnh lặp.....	71
Bài tập chương 4.....	77
Chương 5. Hàm trong C++	80
5.1. Giới thiệu.....	80
5.2 Khai báo và cách sử dụng hàm	81
5.3 Hàm đệ quy	94
Bài tập chương 5.....	98
Chương 6. Kiểu mảng.....	106
6.1 Khái niệm mảng	106
6.2 Mảng một chiều	107
<i>6.2.1 Khai báo mảng một chiều</i>	<i>107</i>
<i>6.2.2 Nhập xuất dữ liệu cho mảng một chiều.....</i>	<i>108</i>
<i>6.2.3 Sắp xếp và tìm kiếm trên mảng một chiều.....</i>	<i>114</i>
<i>6.2.4 Một số ví dụ khác</i>	<i>119</i>
6.3 Mảng hai chiều	125

6.4 Sử dụng mảng làm tham số trong hàm.....	135
Bài tập chương 6	136
Chương 7. Xâu kí tự.....	141
7.1 Khái niệm xâu và cách khai báo.....	141
7.1.1 Khái niệm xâu kí tự.....	141
7.1.2 Khai báo xâu kí tự	141
7.2 Nhập và xuất xâu ký tự	143
7.3 Một số hàm sử dụng trên xâu kí tự.....	148
Bài tập chương 7	159
Chương 8. Lập trình hướng đối tượng với C++	161
8.1 Giới thiệu	161
8.2 Hàm tạo (constructors)	163
8.3 Phép gán	164
8.4 Hàm toán tử	168
8.5 Sự chuyển đổi kiểu dữ liệu trong lớp.....	170
8.5.1. Hàm toán tử chuyển đổi từ kiểu cơ sở sang kiểu lớp	170
8.5.2 Hàm toán tử chuyển đổi từ kiểu lớp sang kiểu cơ sở	172
8.5.3 Hàm toán tử chuyển đổi từ kiểu lớp sang kiểu lớp.....	174
8.6 Thừa kế và sự tương tác giữa các lớp	176
8.6.1 Thừa kế	176
8.6.2 Cách sử dụng các từ khóa public, private và protected trong thừa kế lớp.....	178
8.7 Tính đa hình.....	178
Bài tập chương 8.....	181
Chương 9. Kiểu con trỏ và kiểu cấu trúc	184
9.1 Kiểu con trỏ	184
9.1.1 Khái niệm kiểu con trỏ	184
9.1.2 Mối liên hệ giữa mảng và con trỏ	185

9.1.3 Truyền tham số là con trỏ cho hàm.....	187
9.1.4 Cấp phát bộ nhớ động.....	189
9.2 Kiểu cấu trúc	191
9.2.1 Giới thiệu kiểu cấu trúc	191
9.2.2 Mảng với các phần tử có kiểu cấu trúc.....	193
9.2.3 Danh sách liên kết trên cấu trúc	194
Bài tập chương 9.....	206
Tài liệu tham khảo	210
2.5.4 Định dạng giá trị khi in ra màn hình	210

Chương 1. Tổng quan

1.1 Khái niệm về chương trình và ngôn ngữ lập trình

Ngày nay, rất nhiều hệ thống máy móc hiện đại hoạt động được đều cần có một hệ thống xử lý thông tin để điều khiển hoặc trợ giúp quá trình điều khiển. Hệ thống này gọi là **chương trình máy tính**. Một số ví dụ sử dụng chương trình máy tính để thực hiện việc điều khiển hệ thống có thể kể ra như người máy, robot, các hệ thống tự động hóa trong công nghiệp, các sản phẩm như tủ lạnh, máy giặt v.v. Vậy chương trình máy tính là gì? Chương trình máy tính có thể hiểu là tập hợp hữu hạn các câu lệnh được bố trí theo một trình tự xác định nhằm giải quyết yêu cầu của bài toán đặt ra. Khái niệm máy tính có thể là một máy vi tính đơn chiếc, một hệ thống máy tính kết nối với nhau, một máy tính bỏ túi đơn giản, một hệ thống vi điều khiển, hay một hệ thống siêu máy tính v.v.. Chương trình máy tính trên thực tế còn gọi là phần mềm. Chương trình máy tính được viết bởi một hoặc một vài *ngôn ngữ lập trình* cụ thể nào đó. Trải qua quá trình phát triển gần một thế kỷ bắt đầu từ những năm 40 của thế kỷ XX, đã có rất nhiều ngôn ngữ lập trình ra đời nhằm mục đích viết chương trình cho các ứng dụng khác nhau. Có những ngôn ngữ lập trình đã không còn sử dụng nữa tuy nhiên sự ra đời của ngôn ngữ lập trình sau chính là sự kế thừa của các ngôn ngữ lập trình trước đó. Ngôn ngữ lập trình có thể chia thành hai loại: ngôn ngữ lập trình bậc thấp (ngôn ngữ máy, ngôn ngữ assembly) và ngôn ngữ lập trình bậc cao (C++, Java, Visual Basic, ASP,...).

Ngôn ngữ máy

Chương trình trong ngôn ngữ máy bao gồm dãy các lệnh máy mà CPU có thể thực hiện trực tiếp. Đó là ngôn ngữ lập trình duy nhất mà máy tính có thể hiểu được. Tùy theo thiết kế về phần cứng, mỗi loại máy tính có một tập lệnh cho ngôn ngữ máy khác nhau. Các lệnh viết bằng ngôn ngữ máy nói chung ở dạng nhị phân hoặc biến thể của chúng trong hệ đếm 16. Ví dụ về các lệnh trong ngôn ngữ máy:

```
11000000 0000000000001 0000000000010
```

```
11110000 0000000000010 0000000000011
```

Mỗi lệnh của ngôn ngữ máy gồm hai phần: phần chỉ dẫn và phần địa chỉ. Phần chỉ dẫn là các số nằm bên trái, phần bên phải là địa chỉ sử dụng. Chẳng hạn, dòng lệnh đầu trên ví dụ trên là câu lệnh cộng, thì giá trị của hai địa chỉ trong phần còn lại của câu lệnh trên sẽ được cộng vào với nhau. Ngôn ngữ máy có nhược điểm là khó học vì phải hiểu cấu trúc của hệ thống máy tính cũng như các câu lệnh ở dạng nhị phân.

Ngôn ngữ Assembly

Để khắc phục nhược điểm của ngôn ngữ máy, người ta đề xuất một ngôn ngữ giao tiếp với máy ở mức độ thân thiện với con người hơn gọi là hợp ngữ. Về cơ bản, các câu lệnh của hợp ngữ có cấu trúc rất giống với ngôn ngữ máy, điểm khác là trong hợp ngữ có thể viết lệnh dưới dạng mã chữ. Mã chữ thể hiện mã lệnh hoặc các đối tượng trong lệnh (trong ngôn ngữ máy nó là mã lệnh và địa chỉ của đối tượng). Mã lệnh ở dạng chữ thường chính là những từ trong tiếng Anh có ý nghĩa rõ ràng, còn đối tượng do ta tự đặt tên phù hợp với ý niệm về đối tượng đó. Ví dụ, nếu đoạn chương trình trên dùng để cộng chiều dài và chiều rộng của hình chữ nhật cho việc tính nửa chu vi thì trong hợp ngữ ta chỉ cần viết:

```
ADD 2, 5  
MUL 2, 3
```

Như vậy ngôn ngữ assembly cần một bộ chuyển đổi để chuyển các mã lệnh trên về dạng mã máy. Bộ chuyển đổi này trên thực tế gọi là **assembler**. Hạn chế của ngôn ngữ assembly là nó cũng phụ thuộc vào cấu trúc của các dòng máy tính, mỗi loại vi xử lý khác nhau sẽ có bộ lệnh khác nhau.

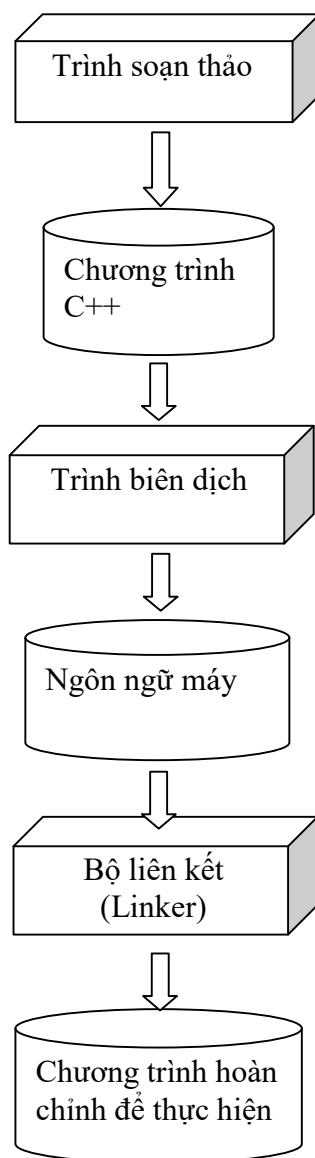
Ngôn ngữ lập trình bậc cao

Ngôn ngữ máy và ngôn ngữ assembly gọi là ngôn ngữ bậc thấp, các ngôn ngữ này có hạn chế là chỉ sử dụng được cho một loại máy hoặc một kiểu máy xác định. Ngược lại, ngôn ngữ lập trình bậc cao sử dụng các câu lệnh giống với ngôn ngữ thông thường, chẳng hạn như tiếng Anh, và có thể chạy trên nhiều loại máy tính khác nhau. Hơn nữa, do các câu lệnh gần giống với ngôn ngữ thông thường nên chương trình sẽ dễ viết, dễ đọc, dễ sửa lỗi. Hiện nay, các phần mềm ứng dụng đa số đều được viết bởi ngôn ngữ lập trình bậc cao; các phần mềm dùng cho các mục đích chuyên biệt trong điều khiển, trong các hệ thống như robot, người máy có thể được viết bởi ngôn ngữ lập trình bậc thấp. Một số

ngôn ngữ lập trình bậc cao đang được sử dụng như C++, Java, Visual Basic, C#, ASP...

Chẳng hạn, sử dụng ngôn ngữ C++, câu lệnh tính tổng và tích hai số nguyên a và b là:

```
tong = a + b;  
tich = a * b;
```



Hình 1-1: Quá trình viết và thực hiện chương trình trên máy tính

Sau khi viết chương trình để thực hiện công việc, chúng ta cần dịch chương trình đó sang ngôn ngữ máy. Việc thực hiện dịch chương trình sang ngôn ngữ máy có thể thực hiện bằng hai cách: thực hiện dịch và thi hành từng câu lệnh riêng rẽ gọi là trình thông dịch, hoặc dịch tất cả các câu lệnh sang ngôn ngữ máy rồi mới thực hiện – gọi là trình biên dịch. Hình 1-1 minh họa việc tạo lập và thi hành chương trình trên C++. Trước tiên,

chúng ta cần một trình soạn thảo để soạn chương trình theo cú pháp của C++. Sau đó chương trình dịch (với C++ là trình biên dịch) sẽ chuyển chương trình này sang ngôn ngữ máy, tiếp theo bộ liên kết (Linker) sẽ làm nhiệm vụ liên kết chương trình với các tác nhân khác (mà chương trình yêu cầu) để được một chương trình hoàn chỉnh thực hiện trên máy tính. Chương trình này sẽ được nạp vào bộ nhớ của máy tính và thực hiện.

Lập trình hướng thủ tục và lập trình hướng đối tượng

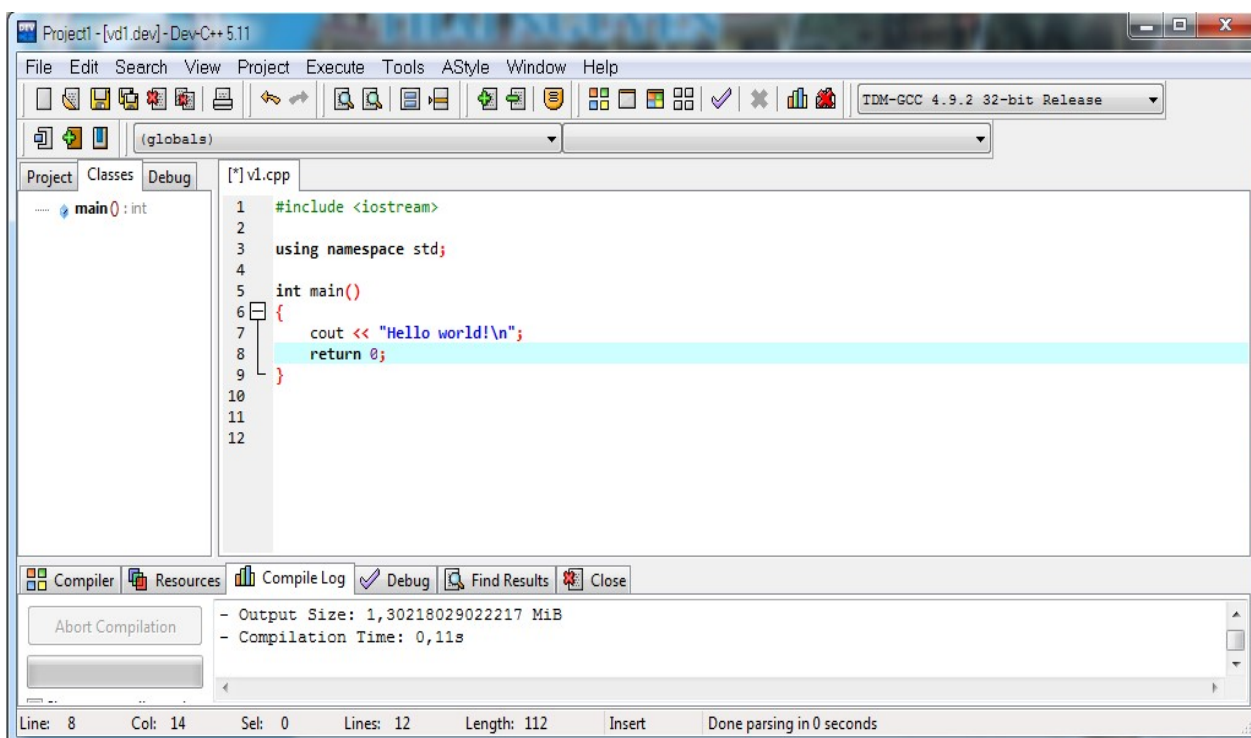
Hiện nay, để viết chương trình, chúng ta có thể thiết kế các chương trình theo hai phương pháp chính là phương pháp *hướng thủ tục* và phương pháp *hướng đối tượng*. Hiểu một cách đơn giản phương pháp lập trình hướng thủ tục là việc chia bài toán lớn thành các bài toán nhỏ hơn và giải quyết từng bài toán con một, trong khi phương pháp hướng đối tượng giải quyết bài toán bằng cách chia bài toán thành các đối tượng trong đó chứa cả dữ liệu và các phương thức để xử lý dữ liệu đó. Ưu điểm của lập trình hướng đối tượng là tính kế thừa, tức là các đoạn mã có thể được kế thừa nhiều lần trong khi viết chương trình. Ngày nay, trên thực tế vẫn tồn tại hai chiến lược thiết kế chương trình như trên. Tùy theo mục đích của bài toán đối với các ứng dụng cụ thể, người thiết kế và xây dựng chương trình sẽ có những lựa chọn phù hợp.

1.2 Ngôn ngữ lập trình C++

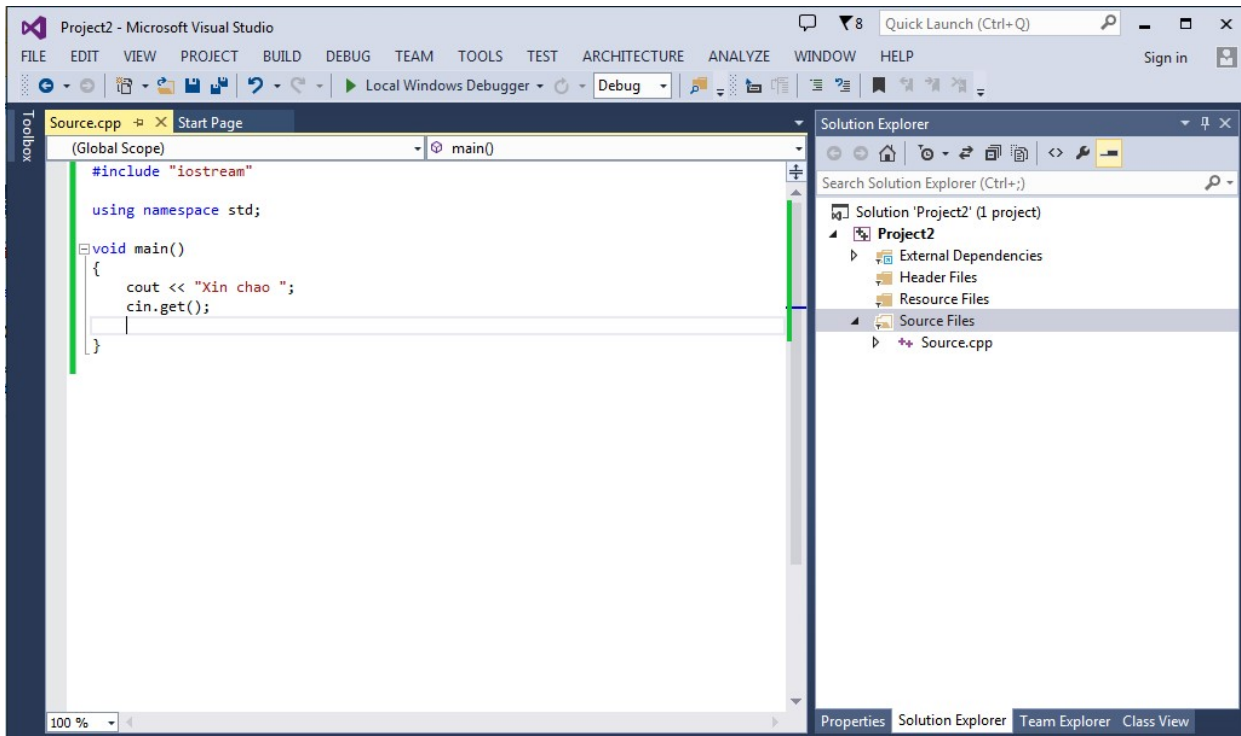
Ngôn ngữ hướng thủ tục đầu tiên được giới thiệu là ngôn ngữ FORTAN. Ngôn ngữ này được giới thiệu vào năm 1957 và được sử dụng trong những năm 60 và 70 của thế kỷ XX. Mục đích của ngôn ngữ này là dùng cho các bài toán trong khoa học và kỹ thuật. Một ngôn ngữ khác là ngôn ngữ COBOL, được giới thiệu năm 1960. Ngôn ngữ COBOL được dùng cho các bài toán về quản lý thông tin như toán và lập các báo, thống kê cho mục đích thương mại. Năm 1970, ngôn ngữ C được giới thiệu bởi Ken Thomson, Dennis Ritchie, và Brian Kernighan và nó trở thành ngôn ngữ phù hợp cho các bài toán **khoa** học kỹ thuật. Ngôn ngữ C++ được giới thiệu vào đầu những năm 80 của thế **kỷ** XX bởi Bjarne Stroustrup, đây là ngôn ngữ phát triển từ ngôn ngữ C và là ngôn ngữ lập trình hướng đối tượng. Trải qua nhiều năm phát triển và hoàn thiện, hiện nay C++ là ngôn ngữ phổ biến trên thế giới, nó được dùng trong hầu khắp các trường đại học và là ngôn ngữ phục vụ cho các bài toán khoa học kỹ thuật.

1.3 Môi trường lập trình của C++

Hiện nay, có một số môi trường cho phép ta viết, dịch và thực hiện chương trình C++ như: Dev C++, Borland C++, Visual Studio... Hình 1-2 là giao diện của phần mềm Dev C++, trong đó hệ thống giao diện bao gồm các chức năng như File, Edit... dùng để hỗ trợ các thao tác trong quá trình soạn thảo chương trình. Sau khi viết chương trình, chúng ta phải lưu lại tệp lên đĩa, dịch, kiểm tra lỗi sau đó thi hành chương trình. Hình 1-3 là một giao diện khi sử dụng Visual Studio C++, hệ thống phần mềm của hãng Microsoft cho chúng ta công cụ để lập trình với C++. Giao diện của Visual Studio C++ cũng tương tự như Dev C++.



Hình 1-2: Giao diện của môi trường lập trình với Dev C++



Hình 1-3. Giao diện của Visual Studio 2012

1.4 Phần mềm và phần cứng

1.4.1 Phần mềm

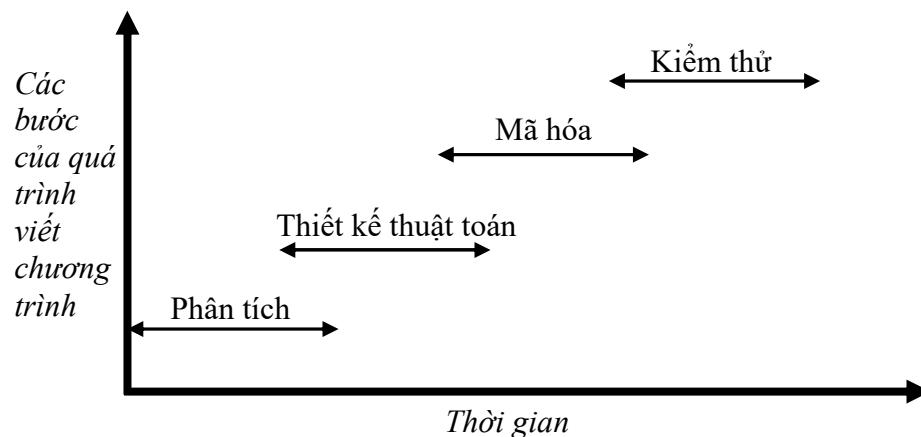
Trong phần 1.1, chúng ta đã thấy chương trình hay phần mềm có thể hiểu là đồng nhất. Phần mềm là một tập hợp các câu lệnh được viết bởi một hay một vài ngôn ngữ lập trình nhằm mục đích giải quyết một bài toán cụ thể nào đó. Phần mềm hiện nay có ứng dụng rất rộng rãi và có thể sử dụng ở rất nhiều các lĩnh vực khác nhau:

- Phần mềm ứng dụng: quản lý hệ thống thông tin, tài nguyên, nguồn nhân lực, website, quản trị văn phòng, trò chơi...
- Phần mềm trợ giúp: phục vụ quá trình học tập, giảng dạy, nghiên cứu, mô phỏng, giả lập tình huống...
- Phần mềm điều khiển: các phần mềm ứng dụng trong hệ thống tự động hóa, điều khiển trong công nghiệp, các thiết bị như máy giặt, tủ lạnh, ô tô, vũ trụ,...
- Phần mềm thông minh: nhận dạng, tìm kiếm, phân loại, dự đoán, phân tích, khai phá dữ liệu, Robot và người máy thông minh.

Việc xây dựng phần mềm được thực hiện qua các bước cơ bản sau:

- Phân tích để hiểu rõ yêu cầu bài toán: Hiểu chính xác yêu cầu bài toán là nhiệm vụ đầu tiên của quá trình xây dựng chương trình. Trong bước này, chúng ta cần xác định rõ *input* và *output* của bài toán, các ràng buộc về kích thước dữ liệu, kiểu dữ liệu,...
- Xây dựng và thiết kế giải thuật: Từ pha phân tích yêu cầu, chúng ta cần phải xác định chiến thuật để giải quyết bài toán, cần phải chỉ ra các bước xác định để biến *input* thành *output*.
- Mã hóa: Giải thuật sẽ được chuyển sang mã lệnh bằng ngôn ngữ lập trình cụ thể để thực hiện.
- Kiểm thử: Sau khi viết xong chương trình, chúng ta cần phải kiểm tra (testing) chương trình viết ra có đúng với yêu cầu của bài toán hay không. Một chương trình cần tiến hành hàng loạt các bộ dữ liệu kiểm thử để kiểm tra xem đầu ra có như mong muốn hay không.

Một cách trực quan, chúng ta có thể hình dung các bước xây dựng chương trình như trong hình 1-4.



Hình 1-4: Các bước của quá trình xây dựng chương trình trên máy tính

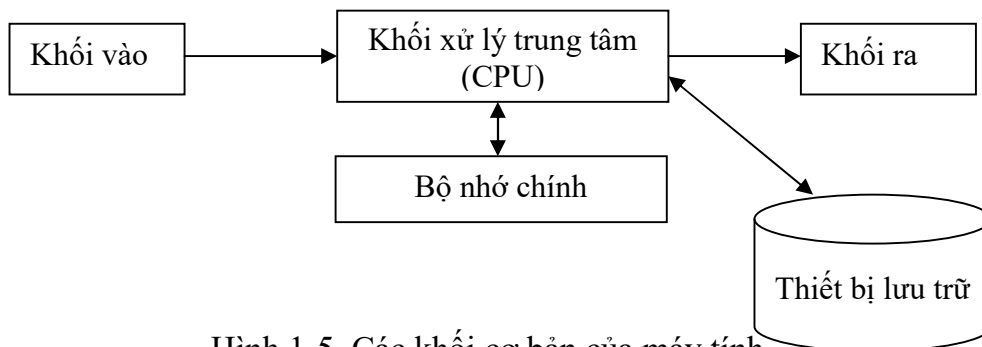
1.4.2 Phần cứng

Phần cứng được hiểu là toàn bộ các thiết bị máy tính cấu thành nên hệ thống máy tính để thực thi các phần mềm. Mỗi phần mềm phải được thực thi trên một môi trường phần cứng cụ thể. Nếu như phần mềm được ví như phần hồn thì phần cứng được ví như

phần xác của một hệ thống. Nếu chỉ có phần cứng mà không có phần mềm thì không thực hiện được và nói đến phần mềm thì cần có phần cứng đi kèm. Các phần mềm đều chạy trên một hệ thống gọi là hệ thống máy tính. Hệ thống này có các tính chất sau:

- Phải có thiết bị nhập xuất dữ liệu
- Phải lưu trữ được thông tin
- Phải thực hiện được các thao tác toán học và logic
- Phải quản lý, điều khiển và ra lệnh cho toàn bộ hệ thống hoạt động.

Hình 1-5 minh họa các khối cơ bản của một máy tính. Các máy tính bắt đầu được nghiên cứu, sản xuất và đưa vào sử dụng những năm 40 của thế kỷ XX. Ngày nay, với sự phát triển mạnh mẽ của khoa học công nghệ, máy tính không còn xa lạ với con người ở khắp nơi trên thế giới. Các thiết bị như máy tính để bàn, máy tính xách tay, hay điện thoại thế hệ mới đều được coi là các hệ thống máy tính.



Hình 1-5. Các khối cơ bản của máy tính

1.5 Thuật toán

Trước khi một chương trình được viết, người lập trình (nhóm lập trình) phải hiểu rõ vấn đề cần giải quyết: dữ liệu đầu vào là gì, kết quả mong muốn đạt được những gì và phải có trình tự để biến dữ liệu đầu vào thành kết quả ra mong muốn. Trình tự hay lời giải dùng để giải quyết bài toán trong trường hợp này gọi là *thuật toán* (algorithm). Về cơ bản, thuật toán được định nghĩa là một dãy hữu hạn các bước nhằm diễn tả quá trình xử lý dữ liệu đầu vào nhằm đưa ra kết quả của bài toán như mong muốn.

Tính chất của thuật toán:

- *Đầu vào:* Đầu vào của thuật toán là dữ liệu bài toán cung cấp cho chương trình.
- *Đầu ra:* Đầu ra của thuật toán là kết quả của yêu cầu cho bài toán đó.

- *Tính xác định*: Mỗi bước của thuật toán phải rõ ràng.
- *Tính chính xác*: Mỗi thuật toán phải đưa ra kết quả chính xác với mỗi đầu vào tương ứng.
- *Tính hữu hạn*: Một thuật toán cần đưa ra kết quả sau một số hữu hạn bước.
- *Tính phổ dụng*: Thuật toán phải giải quyết được một lớp các bài toán có cùng dạng, không đơn thuần chỉ là một bài toán đặc biệt.

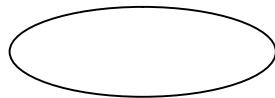
Các phương pháp biểu diễn thuật toán cơ bản

- Phương pháp liệt kê từng bước

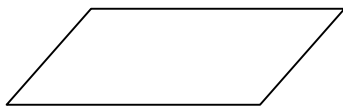
Trong phương pháp này các bước thực hiện ý tưởng giải quyết bài toán sẽ được liệt kê theo trình tự từng bước từ đầu đến cuối. Phương pháp này thực tế chỉ phù hợp cho các dự án nhỏ, với các dự án lớn sẽ rất khó áp dụng vì số lượng các bước nhiều, gây khó hiểu cho các công đoạn tiếp theo.

- Phương pháp sử dụng sơ đồ khối

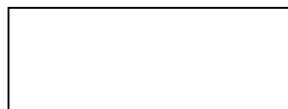
Với phương pháp sử dụng sơ đồ khối, mỗi thao tác của thuật toán sẽ sử dụng các khối để biểu diễn và như vậy làm cho các công đoạn được trực quan, dễ hiểu cho người sử dụng ở các công đoạn sau. Hình 1-6 mô tả các khối được sử dụng trong khi xây dựng sơ đồ khối.



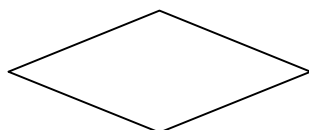
Khối bắt đầu / kết thúc



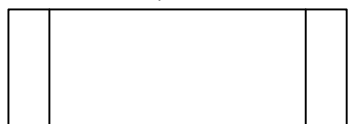
Khối nhập xuất dữ liệu



Khối thao tác



Khối điều kiện



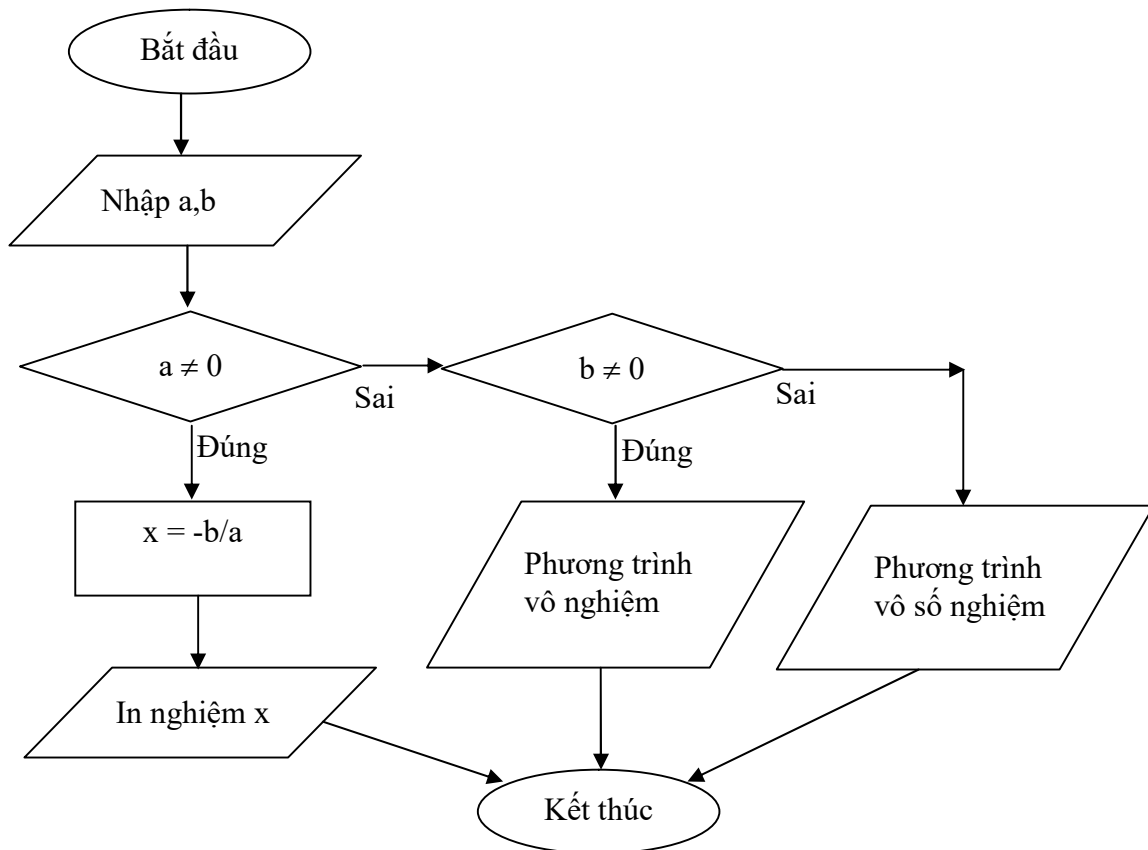
Sử dụng các thao tác đã xây dựng trước

—————→ Kết nối giữa các khối

Hình 1-6. Các kí hiệu dùng trong sơ đồ khối

Ví dụ 1.1 Vẽ sơ đồ khối minh họa giải thuật giải phương trình bậc nhất $ax + b = 0$.

Giải: Đầu vào của thuật toán là hệ số a và b ; đầu ra của thuật toán là nghiệm của phương trình: có thể có nghiệm, có vô số nghiệm, hoặc vô nghiệm. Chúng ta đã biết cách giải trong toán học của phương trình này, sơ đồ khối biểu diễn giải thuật được trình bày trong hình 1-7.



Hình 1-7. Sơ đồ khối biểu diễn thuật toán giải phương trình bậc nhất $ax + b = 0$

Ví dụ 1.2 Mô tả thuật toán tìm giá trị lớn nhất của một dãy hữu hạn các số nguyên.

Giải: Chúng ta thường gặp bài toán này trong thực tế chẳng hạn như tìm người có điểm cao nhất của kỳ thi tuyển sinh Đại học, hay tìm kiếm mặt hàng có khách hàng mua nhiều nhất của một siêu thị... Thuật toán trên được minh họa theo phương pháp liệt kê từng bước như sau:

Bước 1: Giả định đặt giá trị lớn nhất max bằng giá trị của phần tử đầu tiên của dãy

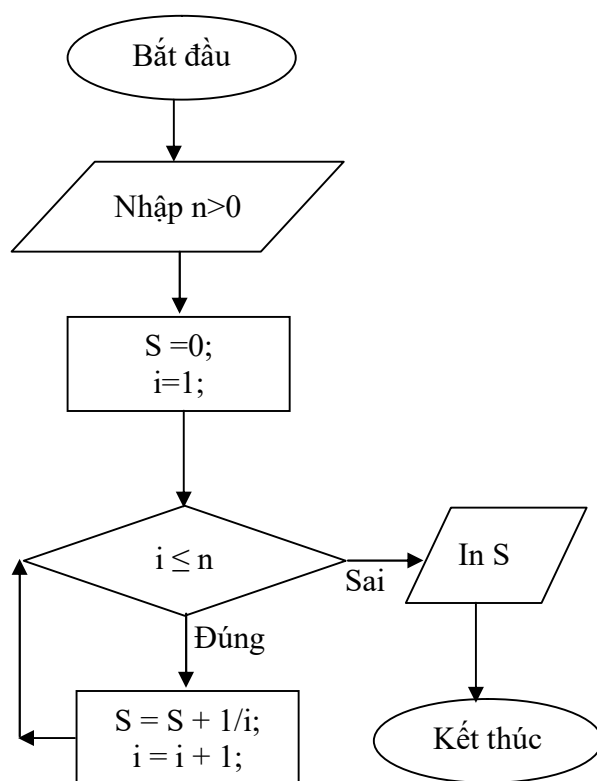
Bước 2: So sánh phần tử thứ hai với giá trị max , nếu giá trị max nhỏ hơn thì đặt giá trị max bằng giá trị thứ hai

Bước 3: Nếu vẫn còn các phần tử tiếp theo thì thực hiện lặp lại giống như bước hai

Bước 4: Thuật toán sẽ dừng lại nếu không còn phần tử nào chưa được xét. Phần tử max cuối cùng sẽ là giá trị lớn nhất của dãy.

Ví dụ 1.3 Vẽ sơ đồ khối minh họa giải thuật tính tổng $S = 1 + 1/2 + 1/3 + \dots + 1/n$ với n nguyên dương.

Giải: Sơ đồ khối được trình bày trong hình 1-8:

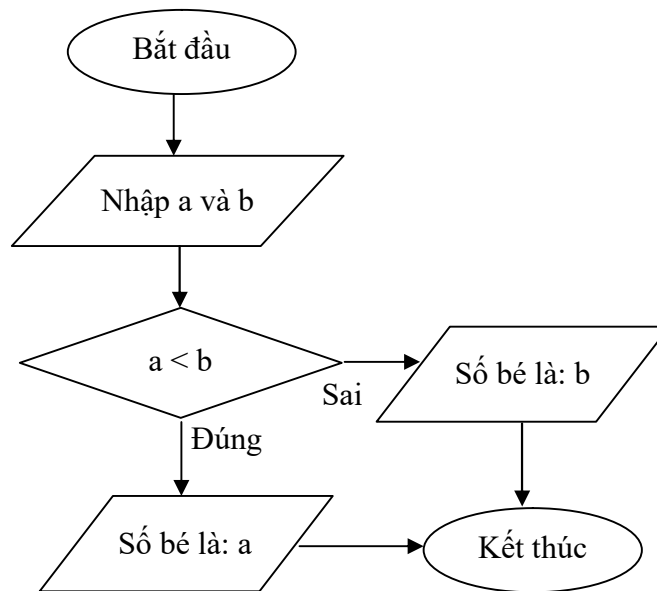


Hình 1-8. Sơ đồ khối của ví dụ 1.3

Trong sơ đồ trên, trước tiên chúng ta phải có thao tác nhận giá trị của n . Sau khi có n thì tổng S mới được xác định. Quá trình tính tổng S sẽ được lặp đi lặp lại: tại bước thứ i giá trị của S sẽ được cộng thêm một lượng là $1/i$; chu trình lặp sẽ dừng lại khi i có giá trị vượt quá n . Lúc đó ta sẽ có giá trị của tổng S .

Ví dụ 1.4 Cho hai số a và b , vẽ sơ đồ khối minh họa giải thuật tìm số bé nhất trong hai số trên.

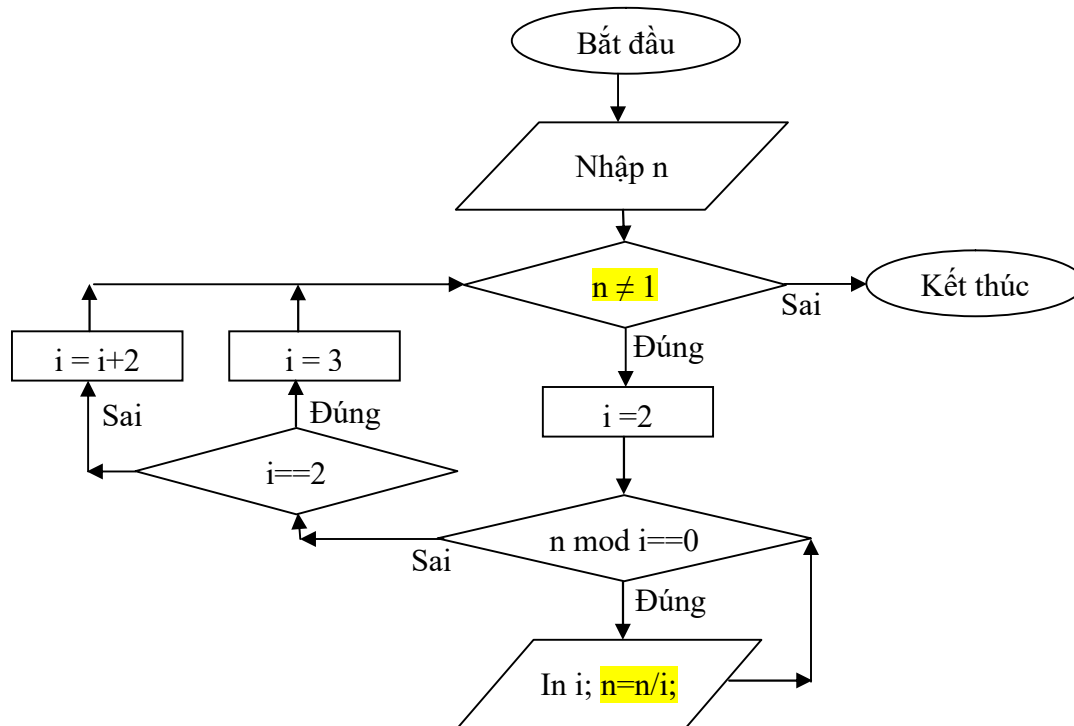
Giải: Sơ đồ khối này khá đơn giản (hình 1-9), chúng ta sẽ sử dụng khối điều kiện để xác định a hay b nhỏ hơn và thông báo kết quả sau khi so sánh.



Hình 1-9: Sơ đồ khối của ví dụ 1.4

Ví dụ 1.5 Viết sơ đồ khối minh họa giải thuật phân tích một số nguyên thành tích các số nguyên tố. Ví dụ: $8 = 2 \cdot 2 \cdot 2$, $30 = 2 \cdot 3 \cdot 5$.

Giải:



Hình 1-10. Sơ đồ khối của ví dụ 1.5

Bài tập chương 1

Bài 1. Vẽ sơ đồ khối nhập vào 5 số a, b, c, d, và e. Tìm số lớn nhất và nhỏ nhất trong 5 số trên.

Bài 2. Vẽ sơ đồ khối tính giá trị của hàm f, với x là tham số.

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x^2 - x & 0 < x \leq 2 \\ x^2 - \sin \pi x^2 & x > 2 \end{cases}$$

Bài 3. Vẽ sơ đồ khối cho bài toán tính tiền điện với chỉ số mới và chỉ số cũ được cung cấp để tính toán. Quy tắc tính như sau: 100 kWh đầu giá 550 đồng, từ kWh 101 - 150 giá 1.110 đồng, từ kWh 151 - 200 giá 1.470 đồng, từ kWh 201 - 300 giá 1.600 đồng, từ kWh 301 - 400 giá 1.720 đồng, từ kWh 401 trở lên giá 1.780 đồng.

Bài 4. Cho số nguyên a, vẽ sơ đồ khối in ra tất cả các ước số của số đó.

Bài 5. Cho số nguyên a, vẽ sơ đồ khối kiểm tra xem số đó có phải là số nguyên tố hay không?

Bài 6. Vẽ sơ đồ khối giải hệ phương trình sau:

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$$

Bài 7. Vẽ sơ đồ khối giải phương trình bậc bốn sau:

$$ax^4 + bx^2 + c = 0$$

Bài 8. Cho 3 số thực a, b, c. Vẽ sơ đồ khối kiểm tra xem a, b, c có phải là 3 cạnh của tam giác không? Nếu là 3 cạnh của tam giác thì tính diện tích của tam giác theo công thức sau:

$$s = \sqrt{p * (p - a) * (p - b) * p - c)}, \text{ với } p = (a + b + c) / 2$$

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

Bài 9. Cho số nguyên a biểu diễn một năm nào đó. Vẽ sơ đồ khối kiểm tra xem năm đó có phải là năm nhuận hay không.

Bài 10. Vẽ sơ đồ khối tính tổng sau:

$$S = 1 + \left(\frac{1}{1+2^3} \right)^2 + \left(\frac{1}{2+3^4} \right)^2 + \dots + \left(\frac{1}{(n-1)+n^{n+1}} \right)^2$$

Bài 11. Vẽ sơ đồ khối tính tổng sau:

$$S = \left(\frac{1}{1+2} \right)^2 + \left(\frac{1+2}{2+3} \right)^2 + \dots + \left(\frac{(n-2)+(n-1)}{(n-1)+n} \right)^2$$

Bài 12. Vẽ sơ đồ khối tính tổng sau:

$$S = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \dots + \frac{1}{1+2+\dots+n}$$

Bài 13. Vẽ sơ đồ khối tính tổng sau:

$$S = 1 + \frac{1}{1^2+2^2} + \frac{1}{1^2+2^2+3^2} + \dots + \frac{1}{1^2+2^2+\dots+n^2}$$

Bài 14. Vẽ sơ đồ khối tính tổng sau:

$$S = \frac{1}{1^3+2^3} + \frac{1}{1^3+2^3+3^3} + \dots + \frac{1}{1^3+2^3+\dots+n^3}$$

Chương 2. Các khái niệm cơ bản trong C++

2.1 Các thành phần cơ bản của C++

2.1.1 Bộ ký tự

Để viết chương trình trên C++, chúng ta cần biết bộ ký tự để thực hiện nó. Giống như các ngôn ngữ lập trình khác, C++ sử dụng các chữ cái la tinh (a..z, A..Z), các chữ số (0..9), các phép toán (+, -, *, /), các ký tự đặc biệt (% , \$, #, &, ||,...) và dấu cách để viết các đoạn mã cho chương trình. Trong khi viết chương trình, chúng ta sử dụng một số thành phần sau để tạo nên chương trình: *định danh, biến, hằng, hàm, câu lệnh...* Các khái niệm này sẽ được đề cập ở các phần tiếp sau.

2.1.2 Định danh và từ khóa

Định danh dùng để phân biệt các thành phần khác nhau trong một chương trình. Định danh có thể là: *tên hằng, tên biến, tên mảng, tên hàm, tên con trỏ, tên cấu trúc, tên nhãn...*

Định danh được đặt theo qui tắc sau:

- Ký tự đầu tiên của định danh phải là một chữ cái hoặc ký tự gạch dưới (_)
- Định danh chỉ chứa ký tự số, ký tự chữ cái la tinh hoặc ký tự gạch dưới
- Không được trùng với từ khóa (bảng 2.1)
- Chiều dài tối đa là 1024

Chú ý rằng, C++ có phân biệt chữ hoa và chữ thường nên khi viết chương trình chúng ta phải chú ý về cách phân biệt này. Tất cả các từ khóa do C++ định nghĩa đều phải được viết bởi chữ thường.

Từ khóa là các từ được ngôn ngữ lập trình quy định sẵn và sử dụng vào một mục đích cụ thể trong quá trình viết chương trình. Giống như các ngôn ngữ nói chung, từ khóa được tạo nên bởi bộ ký tự, các từ khóa đa phần xuất phát từ tiếng Anh. Bảng 2-1 liệt kê một số từ khóa cơ bản của C++.

auto	delete	friend	private
break	do	if	protected
case	double	inline	public
char	else	int	return
const	float	long	short

continue	for	new	sizeof
static	switch	void	while

Bảng 2-1 Một số từ khóa cơ bản của C++

2.1.3 Câu lệnh

Giống như các ngôn ngữ lập trình khác, mỗi câu lệnh trong C++ nhằm thực hiện một mục đích cụ thể nào đó do C++ quy định. Thông thường các câu lệnh được viết dựa trên ngôn ngữ tiếng Anh, câu lệnh được tạo nên bởi các từ khóa. Một câu lệnh có thể được viết trên một hoặc nhiều dòng. Trong C++, mỗi câu lệnh sẽ kết thúc bằng dấu chấm phẩy (;). Viết chương trình chính là viết các câu lệnh để nhằm mục đích giải quyết bài toán nào đó. Mỗi ngôn ngữ lập trình sẽ có một bộ câu lệnh phục vụ cho quá trình viết các chương trình, chẳng hạn như câu lệnh nhập xuất dữ liệu, câu lệnh rẽ nhánh, câu lệnh lặp... Trong phần tiếp theo, chúng ta sẽ tìm hiểu một số câu lệnh đơn giản.

2.1.3.1 Câu lệnh gán

Câu lệnh gán là câu lệnh đơn giản nhưng tần suất sử dụng nhiều nhất trong khi viết chương trình; cú pháp lệnh như sau:

$$v = \langle \text{biểu thức} \rangle;$$

trong đó v là một biến, giá trị của biểu thức phải có cùng kiểu với v . Khi gặp câu lệnh trên trình biên dịch sẽ tính giá trị của biểu thức và sau đó sẽ gán giá trị đó cho biến v . Ví dụ về một vài câu lệnh gán:

```
x = 3 + x;
y = sqrt(x*x) + 1;
chuvi = 3.14*r*r;
```

2.1.3.2 Câu lệnh nhập dữ liệu từ bàn phím

Cú pháp của câu lệnh như sau:

$$\text{cin} >> b_1 >> b_2 >> \dots >> b_n;$$

trong đó b_1, b_2, \dots, b_n là tên các biến. Câu lệnh trên sẽ cho phép nhập giá trị từ bàn phím cho lần lượt các biến b_1, b_2, \dots, b_n .

Khi chuẩn bị viết chương trình, người lập trình cần phải xác định đầu vào (input) cho mỗi bài toán. Và một trong những cách nhập các giá trị đầu vào là nhập từ bàn phím.

Ví dụ, để viết chương trình giải phương trình **bậc** hai $ax^2 + bx + c = 0$, chúng ta phải nhập dữ liệu đầu vào cho 3 số a, b, c từ bàn phím, khi đó câu lệnh nhập sẽ là :

```
cin>>a ; cin>>b ; cin>>c ;
```

Để tính diện tích và chu vi của một hình chữ nhật, chúng ta phải nhập vào hai giá trị là chiều dài (kí hiệu là a) và chiều rộng (kí hiệu là b), câu lệnh như sau :

```
cin>>a ; cin>>b ;
```

2.1.3.3 Câu lệnh in dữ liệu lên màn hình

Cú pháp lệnh như sau:

```
cout<<v1<<v2<<...<<vm;
```

Trong đó, v_1, v_2, \dots, v_m có thể là biến, hằng, hàm, biểu thức. Câu lệnh này sẽ in giá trị của các v_1, v_2, \dots, v_m lên màn hình.

Một số ví dụ về câu lệnh cout:

```
cout<<"Day la chuong trinh C++";  
cout<<sqrt(5);
```

trong đó câu lệnh thứ nhất sẽ in dòng chữ Day la chuong trinh C++ lên màn hình; câu lệnh thứ hai sẽ in căn bậc hai của 5 lên màn hình.

2.2 Cấu trúc của một chương trình trong C++

Trên thực tế, việc viết một chương trình trên máy tính được thực hiện theo chiến lược hướng thủ tục hoặc hướng đối tượng. Với ngôn ngữ lập trình C++ thì mỗi thủ tục sẽ tương ứng với một *hàm* (function) hoặc một *lớp* (class). Mỗi hàm nhằm mục đích giải quyết một bài toán con cụ thể. Một lớp thì phức tạp hơn một hàm vì trong nó sẽ chứa dữ liệu và các hàm để xử lý dữ liệu. Một lớp sẽ đóng gói cả dữ liệu và các thao tác để xử lý các dữ liệu đó. Như vậy, mỗi lớp sẽ chứa các thành phần đầu vào, đầu ra và một tập các hàm để xử lý nó.

Cấu trúc chung của một chương trình viết trên C++ gồm 2 phần như sau:

Phần khai báo:

- Khai báo các tệp đầu chương trình (tệp header)
- Khai các biến, các hằng, các kiểu dữ liệu,...
- Khai báo các hàm

Phần thân chương:

```
int main()
{
    câu lệnh 1;
    câu lệnh 2;
    ...
    return 0;
}
```

Nhìn vào cấu trúc chương trình trên, chúng ta có thể thấy một chương trình cơ bản gồm hai phần: phần khai báo và phần hàm `main()`. Mỗi chương trình có thể có nhiều hàm tuy nhiên chỉ có một và chỉ một hàm `main()`. Hàm `main()` là điểm mà tất cả các chương trình C++ bắt đầu thực hiện; đối với Dev C++ và Visual Studio C++ hàm `main()` có kiểu trả ra là `int`. Để hình dung về một chương trình của C++, chúng ta xét ví dụ đơn giản sau đây:

Ví dụ 2.1 Chương trình in dòng chữ Ví dụ đầu tiên về C++ lên màn hình.

Giải:

```
#include <iostream> //sử dụng tệp iostream
using namespace std;
int main()
{
    cout<<"Ví dụ đầu tiên về C++";
    return 0;
}
```

Chương trình trên đây là chương trình đầu tiên mà hầu hết những người học về lập trình viết và kết quả của nó là viết câu "Ví dụ đầu tiên về C++" lên màn hình. Đây là một trong những chương trình đơn giản nhất có thể viết bằng C++ nhưng nó đã bao gồm những phần cơ bản mà mọi chương trình C++ có. Chúng ta hãy xem xét từng dòng một:

Dòng đầu tiên là câu lệnh `#include<iostream>` báo cho trình dịch biết cần phải sử dụng tệp `iostream`. Trong tệp `iostream` chứa hai lớp là `istream` và `ostream`, hai lớp này cung cấp các phương thức vào, ra tương ứng phục vụ cho việc viết chương trình. Các tệp khai báo sau `#include` được gọi là các tệp header. Dòng tiếp theo cho chúng ta biết sử dụng namespace, đây là một không gian tên `std`. Câu lệnh này chỉ dẫn cho chương trình biết chỗ để lưu trữ các tệp header. Chỉ dẫn namespace là một cơ chế trong C++ cho phép chúng ta phân nhóm các thực thể như lớp (`class`), đối tượng (`object`), hàm (`function`) thành những nhóm riêng biệt, mỗi nhóm đó được đặt một tên, gọi là không gian tên (`namespace`).

Theo sau hàm `main()` là một cặp ngoặc đơn bởi vì nó là một hàm. Trong C++, tất cả các hàm mà sau đó là một cặp ngoặc đơn `()` có nghĩa là nó có thể có hoặc không có tham số. Nội dung của hàm `main()` tiếp ngay sau phần khai báo chính thức được bao trong các ngoặc móc `{ }`.

```
cout<<"Vi du dau tien ve C++";
```

Câu lệnh `cout` được định nghĩa trong thư viện `iostream` dùng để in chuỗi kí tự "Vi du dau tien ve C++" ra màn hình. Với C++, nhiều câu lệnh có thể viết trên cùng một dòng. Giữa các câu lệnh cách nhau bởi dấu ";". Câu lệnh `return 0;` đặt trước dấu ngoặc móc cuối cùng của hàm `main()` báo hiệu kết thúc chương trình.

Cách chú thích trong chương trình C++

Trong chương trình, người viết đôi khi có các câu chú thích cho một đoạn câu lệnh nào đó để ghi nhớ hoặc để giải thích cho đoạn lệnh, phục vụ cho người đọc chương trình hoặc các lần chỉnh sửa chương trình lần sau. Trong C++ có hai cách để chú thích gồm: chú thích trên một dòng và chú thích trên nhiều dòng:

Cách 1: Chú thích trên một dòng

```
// Dòng chú thích
```

Cách 2: Chú thích trên nhiều dòng

```
/*  
Dòng chú thích 1  
Dòng chú thích 2  
...
```

* /

Chú thích **trên một dòng** bắt đầu từ cặp dấu xô (//) cho đến cuối dòng. Chú thích **trên nhiều dòng** bắt đầu bằng /* và kết thúc bằng */

2.3 Các kiểu dữ liệu và cách sử dụng

2.3.1 Khái niệm về kiểu dữ liệu

Mục tiêu của việc viết các chương trình là xử lý dữ liệu, tức là biến dữ liệu đầu vào thành kết quả ra mong muốn. Ví dụ, đầu vào có thể là các dữ liệu số (điểm thi, bảng số liệu về thuế,...), dữ liệu chữ hay kí tự, dữ liệu âm thanh, hình ảnh, ... Để biểu diễn và xử lý các dữ liệu này trên máy tính, chúng ta phải biết chúng sẽ được khai báo ra sao và kèm với các thao tác gì để xử lý chúng. C++ chia thành hai tập hợp kiểu dữ liệu chính: kiểu dữ liệu cơ sở (built - in) đây là kiểu mà ngôn ngữ cung cấp cho người lập trình và kiểu do người dùng định nghĩa (user - defined).

2.3.2 Kiểu dữ liệu cơ sở

Kiểu số nguyên

Trong C++, có 9 kiểu số nguyên được sử dụng bao gồm các kiểu: `bool`, `char`, `short int`, `int`, `long int`, `unsigned char`, `unsigned short int`, `unsigned int` và `unsigned long int`. Sự khác nhau của các kiểu này phụ thuộc vào khả năng biểu diễn giá trị các số của nó. Bảng 2-2 mô tả phạm vi biểu diễn của từng loại kiểu dữ liệu. Khi viết chương trình chúng ta phải xác định các kiểu dữ liệu phù hợp cho từng loại dữ liệu cần lưu trữ và xử lý đảm bảo tối ưu về mặt không gian nhớ trong máy tính.

Kiểu	Phạm vi biểu diễn	Số byte
<code>char</code>	256 kí tự	1
<code>bool</code>	true hoặc false	1
<code>short int</code>	-32.768...32.767	2
<code>unsigned short int</code>	0 ... 65.535	2
<code>int</code>	-2.147.483.648.. -2.147.483.647	4
<code>unsigned int</code>	0 ... 4.294.967.295	4
<code>long int</code>	-2.147.483.648.. -2.147.483.647	4
<code>unsigned long int</code>	0 ... 4.294.967.295	4

Bảng 2-2. Thông tin về các kiểu dữ liệu nguyên trong C++

Dữ liệu kiểu char

Kiểu char dùng để lưu trữ các kí tự riêng rẽ. Kí tự bao gồm: các ký tự chữ, kí tự số, và các kí tự đặc biệt (tất cả có 256 kí tự). Một kí tự đơn có thể là kí tự chữ (thường hay hoa), kí tự số, hoặc kí tự đặc biệt. Ví dụ các kí tự có thể là: ‘A’, ‘b’, ‘8’, ‘%’, ‘@’, ‘O’, ‘{’. Giá trị tương ứng của các kí tự được biểu diễn trong máy tính dạng mã ASCII (American Standard Code for Information Interchange). Bảng 2.3 trình bày mã ASCII tương ứng với các kí tự hoa.

Kí tự	Mã ASCII	Ký tự	Mã ASCII
A	01000001	N	01001111
B	01000010	O	01001110
C	01000011	P	01010000
D	01000100	Q	01010001
E	01000101	R	01010010
F	01000110	S	01010011
G	01000111	T	01010100
H	01001000	U	01010101
I	01001001	V	01010110
J	01001010	W	01010111
K	01001011	X	01011000
L	01001100	Y	01011001
M	01001101	Z	01011010

Bảng 2-3 Mã ASCII của các ký tự hoa

Dữ liệu kiểu bool

Trong C++, kiểu bool được dùng để diễn tả giá trị logic đúng và sai và được định giá trị tương ứng là 1 và 0. Kiểu dữ liệu này thường được dùng trong các lệnh điều kiện hay biểu thức điều kiện mà kết quả trả về là đúng hoặc sai. Ngoài hai kiểu bool và char, các kiểu còn lại được phân biệt bởi khoảng giá trị của chúng.

Chú ý: Chúng ta có thể sử dụng hàm `sizeof(tên_kiểu_dữ_liệu)` để biết số byte lưu trữ của kiểu dữ liệu đó trên chương trình dịch của C++ đang dùng.

Ví dụ 2.2. Ví dụ về sử dụng hàm `sizeof()`

Giải:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "\n So byte lưu kiểu int: " << sizeof(int);
    cout << "\n So byte lưu kiểu char: " << sizeof(char);
    return 0;
}
```

Kiểu số thực

Trong C++ cho phép sử dụng ba loại dữ liệu thực, đó là `float`, `double` và `long double` (bảng 2-4). Sự khác nhau giữa các kiểu trên chỉ phụ thuộc vào số byte sử dụng để biểu diễn chúng. Máy tính có thể lưu trữ được các số kiểu `float` có giá trị tuyệt đối từ $1.4e^{-45}$ đến $3.4e^{+38}$. Các số có giá trị tuyệt đối nhỏ hơn $1.4E^{-45}$ được xem bằng 0. Phạm vi biểu diễn của số `double` được hiểu theo nghĩa tương tự.

Kiểu dữ liệu	Số byte	Giá trị lưu trữ (+ và -)
<code>float</code>	4	1.40129846432481707e-45 đến 3.40282346638528860e+38
<code>double</code> và <code>long double</code>	8	4.94065645841246544e-324 đến 1.79769313486231570e+308

Bảng 2-4 Phạm vi biểu diễn của kiểu số thực

2.4 Biến và cách khai báo biến

Trong khi viết chương trình để lưu trữ các giá trị khi xử lý, chúng ta phải sử dụng đến *biến*. Các giá trị cần lưu trữ ở đây gồm các giá trị đầu vào và các giá trị trung gian trong quá trình tính toán. Chẳng hạn, để giải phương trình bậc hai chúng ta phải có các biến lưu giữ giá trị cho các hệ số và các nghiệm của phương trình này. Để có thể sử dụng một biến trong C++, đầu tiên chúng ta phải khai báo nó, ghi rõ nó là kiểu dữ liệu nào. Các biến sẽ được lưu trữ trong bộ nhớ của máy tính. Bộ nhớ của máy tính được bố trí thành từng dãy các ô nhớ, mỗi ô nhớ đều có địa chỉ riêng biệt. Vậy biến là một sự thể hiện của các giá trị thật được lưu trong bộ nhớ, có nghĩa là để thao tác chúng ta chỉ cần thao tác với các biến.

Trong C++, quy tắc đặt tên biến giống như cách đặt tên các định danh trong C++. Thực tế, nên đặt tên biến có tính chất gợi nhớ để dễ sử dụng trong quá trình viết chương trình, nhất là các chương trình có số lượng lớn các biến.

2.4.1 Cách khai báo biến

Việc khai báo biến trong C++ là quá trình đặt tên và xác định kiểu cho biến đó. Cấu trúc khai báo là như sau:

<kiểu của biến> tên_biến;

trong đó *<kiểu của biến>* có thể là các kiểu có sẵn trong C++ hoặc có thể là các kiểu do người dùng định nghĩa. Khi viết chương trình, việc đầu tiên cần phải xác định các biến và kiểu của nó để phù hợp với bài toán. Ví dụ về một số khai báo biến sau đây:

```
float a, b, c;  
double x, y, s;  
char a, b;
```

Ví dụ 2.3 Viết chương trình nhập vào hai số a và b tương ứng là chiều dài và chiều rộng của một hình chữ nhật. Tính diện tích và chu vi của hình chữ nhật đó rồi in kết quả lên màn hình.

Giải: Trước tiên, chúng ta phải xác định số lượng biến cần khai báo ở đây sẽ có: biến chứa chiều dài (a), biến chứa chiều rộng (b), biến chứa diện tích (dt) và chu vi (cv).

```
#include <iostream>  
using namespace std;  
int main ()  
{  
    double a, b, dt, cv;  
    cout<<"Chương trình tính diện tích, chu vi hình chữ nhật: \n";  
    cout<<"Nhập chiều dài a = "; cin>>a;  
    cout<<"Nhập chiều rộng b = "; cin>>b;  
    dt = a*b;  
    cv= (a +b)*2;  
    cout<<"\n";  
    cout<<"Diện tích hình chữ nhật = "<<dt<<"\n";  
    cout<<"Chu vi hình chữ nhật = "<<cv;  
    return 0;  
}
```

Ví dụ về kết quả thực hiện chương trình như sau:

```

Chương trình tính diện tích, chu vi hình chu nhật:
Nhập chiều dài a = 3
Nhập chiều rộng b = 4

Diện tích hình chu nhật = 12
Chu vi hình chu nhật = 14
-----

```

2.4.2 Phạm vi hoạt động của các biến

Trong khi viết chương trình, tất cả các biến mà chúng ta sẽ sử dụng đều phải được khai báo trước. Trong C++, có hai loại biến đó là biến toàn cục và biến địa phương. *Biến toàn cục* có thể được sử dụng ở bất kỳ đâu trong chương trình, ngay sau khi nó được khai báo. Biến địa phương có phạm vi hoạt động giới hạn trong phần mã mà nó được khai báo. Nếu chúng được khai báo ở đầu một hàm (như hàm `main()`) thì tầm hoạt động sẽ là toàn bộ hàm `main()`. Trong hàm `main()`, chúng ta cũng có thể khai báo các biến tiếp theo ở bất kỳ chỗ nào, tuy nhiên phạm vi hoạt động của nó sẽ phụ thuộc vào chỗ nó được khai báo. Chúng ta sẽ tìm hiểu kỹ hơn vấn đề này ở các chương sau.

Ví dụ 2.4 Viết chương trình nhập vào bán kính của một hình cầu. Tính diện tích **hình** cầu, thể tích hình cầu rồi in các kết quả lên màn hình.

Giải: Để giải quyết bài toán trên chúng ta sử dụng các câu lệnh gán, câu lệnh nhập dữ liệu và câu lệnh in dữ liệu lên màn hình.

```

#include <iostream>
using namespace std;
int main()
{
    float r;
    float PI = 3.14;
    cout<<"Tính diện tích và thể tích hình cầu: "<<endl;
    cout<<"Nhập bán kính r = "; cin>>r;
    float s;
    s = 4*PI*r*r;
    cout<<"Diện tích mặt cầu = "<<s<<endl;
    float v;
    v = (4*PI*r*r*r)/3;
    cout<<"Thể tích mặt cầu = "<<v<<endl;
    return 0;
}

```

Trong chương trình trên chúng ta thấy rằng các biến địa phương s và v có thể khai báo xen vào các dòng lệnh miễn là trước khi nó được sử dụng. Từ khóa `endl` được sử dụng để xuống dòng mới trong câu lệnh `cout`. Ví dụ về kết quả thực hiện chương trình như sau:

```
Tính diện tích và thể tích hình cầu:
Nhập bán kính r = 2
Diện tích mặt cầu = 50.24
Thể tích mặt cầu = 33.4933
-----
```

2.5 Khai báo hằng trong C++

Khi viết chương trình đôi khi chúng ta sử dụng một giá trị nào đó không thay đổi trong toàn bộ chương trình, khi đó chúng ta có thể khai báo hoặc định nghĩa chúng dưới dạng hằng số. Để sử dụng hằng trong chương trình ta có **hai** cách như sau:

- 1) `#define tên_hằng [giá trị]`
- 2) `const kiểu_dữ_liệu tên_hằng = [giá trị];`

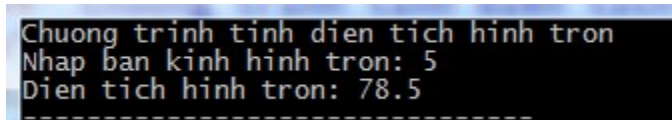
chú ý rằng với `#define` chúng ta không có dấu chấm phẩy ở cuối.

Ví dụ 2.5. Viết chương trình nhập vào bán kính hình tròn bất kỳ từ bàn phím. Sử dụng cách định nghĩa hằng để định nghĩa hằng π có giá trị bằng 3.14. Tính và hiển thị diện tích của hình tròn lên màn hình.

Giải :

```
#include <iostream>
using namespace std;
int main ()
{
    // Định nghĩa hằng
    #define PI 3.14
    float ban_kinh, dien_tich;
    cout << "Chương trình tính diện tích hình tròn"<<endl;
    cout<<"Nhập bán kính hình tròn: "; cin>>ban_kinh;
    dien_tich=PI*ban_kinh*ban_kinh;
    cout<<"Diện tích hình tròn: "<<dien_tich;
    return 0;
}
```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Chương trình tính diện tích hình tròn
Nhập bán kính hình tròn: 5
Diện tích hình tròn: 78.5

```

Ví dụ 2.6 Viết chương trình nhập vào bán kính hình tròn. Sử dụng cách khai báo hằng để khai báo hằng PI có giá trị bằng 3.14. Tính và hiển thị diện tích của hình tròn lên màn hình.

Giải:

```

#include <iostream>
using namespace std;
int main ()
{
    // Khai bao hang
    const float PI = 3.14;
    float ban_kinh, dien_tich;
    cout << "Chương trình tính diện tích hình tròn"<<endl;
    cout<<"Nhập bán kính hình tròn: "; cin>>ban_kinh;
    dien_tich=PI*ban_kinh*ban_kinh;
    cout<<"Diện tích hình tròn: "<<dien_tich;
    return 0;
}

```

Trong chương trình trên hằng PI được khai báo và sử dụng câu lệnh gán tính diện tích hình tròn rồi in kết quả lên màn hình bằng câu lệnh `cout`.

2.5 Biểu thức và các phép toán

2.5.1 Các phép toán cơ bản trong C++

Khi sử dụng các kiểu dữ liệu cơ sở, chúng ta có thể sử dụng các phép toán để thao tác trên các toán hạng của nó. Các phép toán tương ứng với hai kiểu dữ liệu số nguyên và số thực được trình bày trong bảng 2-5.

Kiểu dữ liệu	Phép toán
Kiểu số nguyên	<code>+</code> , <code>-</code> , <code>*</code> , <code>/</code> , <code>%</code> , <code>=</code> , <code>==</code> , <code>!=</code> , <code><=</code> , <code>>=</code> , <code>sizeof()</code> , các thao tác trên bit
Kiểu số thực	<code>+</code> , <code>-</code> , <code>*</code> , <code>/</code> , <code>=</code> , <code>==</code> , <code>!=</code> , <code><=</code> , <code>>=</code> , <code>sizeof()</code>

Bảng 2-5. Các phép toán trên kiểu dữ liệu cơ sở

Các phép toán số học bao gồm: cộng (+), trừ (-), nhân (*), chia (/), chia lấy phần dư (%). Các phép toán được gọi là phép toán hai ngôi vì cần phải có hai toán hạng để thực hiện phép toán. Chẳng hạn một số phép toán đơn giản như sau:

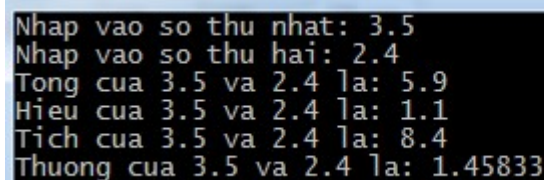
```
14 + 25
23.5 - 35.1
12.5 % 4
```

Ví dụ 2.7. Viết chương trình nhập vào hai số a và b kiểu thực từ bàn phím. Thực hiện việc tính rồi hiển thị kết quả tổng, hiệu, tích, thương của a và b lên màn hình

Giải:

```
#include <iostream>
using namespace std;
int main ()
{
    float a, b;
    cout<<"Nhập vào số thứ nhất: "; cin>>a;
    cout<<"Nhập vào số thứ hai: "; cin>>b;
    cout<<"Tổng của "<< a <<" và "<<b<<" là: "<<a+b<<endl;
    cout<<"Hiệu của "<< a <<" và "<<b<<" là: "<<a-b<<endl;
    cout<<"Tích của "<< a <<" và "<<b<<" là: "<<a*b<<endl;
    cout<<"Thương của "<< a <<" và "<<b<<" là: "<<a/b;
    return 0;
}
```

Ví dụ về kết quả thực hiện chương trình như sau:



```
Nhập vào số thứ nhất: 3.5
Nhập vào số thứ hai: 2.4
Tổng của 3.5 và 2.4 là: 5.9
Hiệu của 3.5 và 2.4 là: 1.1
Tích của 3.5 và 2.4 là: 8.4
Thương của 3.5 và 2.4 là: 1.45833
-----
```

Phép chia với số nguyên

Kết quả của phép toán chia có kiểu cùng với kiểu của các toán hạng, tức là nếu hai toán hạng cùng là số nguyên thì kết quả phép chia là số nguyên, nếu một trong hai toán hạng là số thực thì kết quả sẽ là số thực. Để kết quả của thương phép chia hai số nguyên có kiểu thực ta cần thực hiện việc ép kiểu dữ liệu. Cú pháp ép kiểu dữ liệu như sau:

(tên_kiểu) biểu_thức hoặc tên_kiểu (biểu_thức)

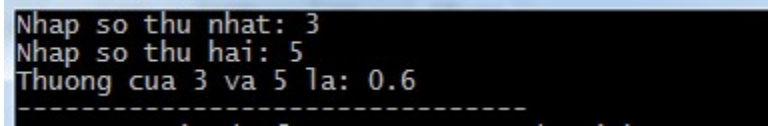
trong đó tên_kiểu là kiểu dữ liệu chúng ta mong muốn định dạng cho biểu_thức. Chẳng hạn kết quả của phép chia $7/3$ sẽ cho giá trị là 2 tuy nhiên nếu chúng ta muốn kết quả là giá trị số thực đúng chúng ta phải ghi là `(double)7/3`; hoặc `double(7)/3`;

Ví dụ 2.8 Viết chương trình nhập vào từ bàn phím hai số a, b kiểu nguyên. Hiển thị kết quả của phép chia a cho b lên màn hình.

Giải:

```
#include <iostream>
using namespace std;
int main ()
{
    int a, b;
    cout<<"Nhập số thứ nhất: "; cin>>a;
    cout<<"Nhập số thứ hai: "; cin>>b;
    cout<<"Thương của "<< a <<" và "<<b<<" là: "<<float(a)/b;
    return 0;
}
```

Ví dụ về kết quả thực hiện chương trình như sau:



```
Nhap so thu nhât: 3
Nhap so thu hai: 5
Thuong cua 3 va 5 la: 0.6
-----
```

Trong ví dụ 2.8b, các cách ép kiểu sau là đúng:

```
float(a)/b;
a/float(b);
float(a)/float(b);
```

Còn cách ép kiểu sau vẫn không cho kết quả là thực:

```
float(a/b);
```

Phép chia dư

Phép chia lấy phần dư được ký hiệu là `%`. Phép toán này chỉ được thực hiện khi cả hai toán hạng có kiểu nguyên. Ví dụ: $3 \% 2 = 1$; $13 \% 10 = 3$;

Các phép toán logic

C++ sử dụng các phép toán trong quá trình viết các biểu thức logic như `<`, `<=`, `>`, `>=`, `!=` (phép so sánh khác), `==` (phép so sánh bằng), `&&` (phép và), `||` (phép hoặc),... Kết quả của phép toán logic cho ta một trong hai giá trị đúng hoặc sai.

2.5.2 Biểu thức

Biểu thức được tạo thành bằng cách sử dụng các phép toán và các toán hạng để diễn đạt một công thức toán học nào đó. Chú ý rằng tên hằng, biến, phần tử mảng (chương 6) và hàm (chương 5) có vai trò như các toán hạng khi viết trong biểu thức. Việc xác định giá trị của biểu thức được thực hiện như quy ước trong toán học. Khi viết biểu thức chúng ta chỉ sử dụng một kiểu dấu ngoặc duy nhất là ngoặc tròn. Ví dụ về một số biểu thức như sau:

```
a+2*b;  
b*b - 4*a*c;  
(x-y) / (2*z -5) ;  
(( a > 0) && (b>0) && (c>))
```

2.5.3 Một số hàm toán học trong C++

C++ xây dựng sẵn một số hàm toán học thông dụng. Trong quá trình viết chương trình, chúng ta chỉ việc sử dụng các hàm này trong quá trình tính toán. Để sử dụng các hàm toán học, chúng ta phải khai báo tệp nguyên mẫu `math.h` (hoặc `cmath`). Một số hàm toán học được liệt kê trong bảng 2-6.

Tên hàm	Ý nghĩa
<code>cos (x)</code>	Hàm tính cos (theo radial)
<code>sin (x)</code>	Hàm tính sin (theo radial)
<code>tan (x)</code>	Hàm tính tan
<code>acos (x)</code>	Hàm tính arc cos
<code>asin (x)</code>	Hàm tính arc sin
<code>atan (x)</code>	Hàm tính arc tan
<code>exp (x)</code>	Hàm số mũ, trả lại giá trị e^x
<code>log (x)</code>	Hàm tính logarithm
<code>pow (x, n)</code>	Hàm tính x^n
<code>sqrt (x)</code>	Hàm căn bậc hai của x

<code>ceil(x)</code>	Hàm làm tròn lên của số thực x
<code>floor(x)</code>	Hàm làm tròn xuống của số thực x
<code>round(x)</code>	Hàm làm tròn số thực x
<code>trunc(x)</code>	Hàm tính phần nguyên của số thực x

Bảng 2-6. Một số hàm toán học trong C++

Ví dụ 2.9. Viết chương trình nhập vào số nguyên a, tính a^5 , căn bậc hai của a^6 , và in lên màn hình các kết quả đó.

Giải:

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int a,b;
    cout<<"a= "; cin>>a;
    cout<<"a^5 = "<<pow(a,5)<<endl;
    cout<<"Can bac hai cua a^6 = "<<sqrt(pow(a,6))<<endl;
    return 0;
}
```

Ví dụ 2.9 sử dụng lệnh `cin` để a vào từ bàn phím. Tiếp đó chúng ta sử dụng hàm `pow()` để tính a^5 và a^6 , `sqrt()` để tính căn bậc hai của a^6 . Chú ý rằng, hai hàm này nằm trong tệp header tên là `cmath` nên chúng ta phải khai báo `cmath` sau `include` ở phần đầu chương trình. Cuối cùng chương trình sử dụng lệnh `cout` để hiển thị dữ liệu.

Ví dụ 2.10. Viết chương trình nhập vào độ dài hai cạnh của một tam giác vuông. Tính độ dài cạnh huyền của tam giác vuông đó bằng hai cách: sử dụng công thức toán học và dùng hàm `hypot()`. Hiển thị kết quả ra màn hình để kiểm tra kết quả.

Giải:

```
#include <iostream>
#include <cmath>
using namespace std;
int main ()
{
```

```

float a, b, c, d;
cout<<"Nhap do dai canh vuong 1: "; cin>>a;
cout<<"Nhap do dai canh vuong 2: "; cin>>b;
c = hypot(a,b);
d = sqrt(a*a+b*b);
cout<<"Do dai canh huyen (su dung ham): "<<c<<endl;
cout<<"Do dai canh huyen (khong su dung ham): "<<d<<endl;
return 0;
}

```

Ví dụ 2.11. Viết chương trình nhập vào hai số thực a và b. Sử dụng các hàm làm tròn `ceil()`, `floor()`, `round()` để hiển thị kết quả của a/b.

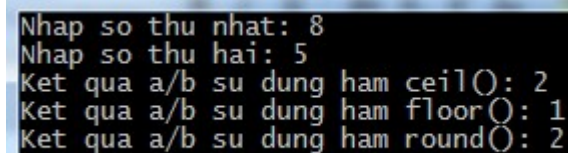
Giải:

```

#include <iostream>
#include <cmath>
using namespace std;
int main ()
{
    float a, b, c, d, e;
    cout<<"Nhap so thu nhat: "; cin>>a;
    cout<<"Nhap so thu hai: "; cin>>b;
    c = ceil(a/b);
    d = floor(a/b);
    e = round(a/b);
    cout<<"Ket qua a/b su dung ham ceil(): "<<c<<endl;
    cout<<"Ket qua a/b su dung ham floor(): "<<d<<endl;
    cout<<"Ket qua a/b su dung ham round(): "<<e<<endl;
    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Nhap so thu nhat: 8
Nhap so thu hai: 5
Ket qua a/b su dung ham ceil(): 2
Ket qua a/b su dung ham floor(): 1
Ket qua a/b su dung ham round(): 2

```

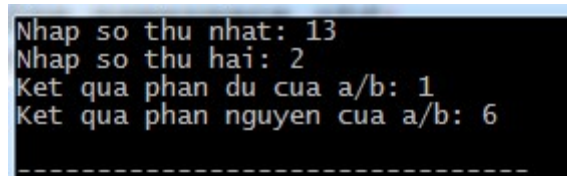
Trong ví dụ 2.11, sau khi thực hiện chương trình với $a = 8$, $b = 5$ thì $a/b = 1.6$; Khi đó, hàm `ceil()` là hàm tròn lên, nên kết quả $c = 2$, hàm `floor()` là hàm làm tròn xuống nên $d = 1$; hàm `round()` là hàm làm tròn theo đúng quy tắc toán học, nên $d = 2$.

Ví dụ 2.12. Viết chương trình nhập vào hai số thực a và b. Hiển thị ra màn hình phần dư và phần nguyên của phép chia a/b.

Giải:

```
#include <iostream>
#include <cmath>
using namespace std;
int main ()
{
    float a, b, c, d, e;
    cout<<"Nhập số thứ nhất: "; cin>>a;
    cout<<"Nhập số thứ hai: "; cin>>b;
    c = fmod(a,b);
    d = trunc(a/b);
    cout<<"Kết quả phần dư của a/b: "<<c<<endl;
    cout<<"Kết quả phần nguyên của a/b: "<<d<<endl;
    return 0;
}
```

Ví dụ về kết quả thực hiện chương trình như sau:



Nhập số thứ nhất: 13
Nhập số thứ hai: 2
Kết quả phần dư của a/b: 1
Kết quả phần nguyên của a/b: 6

Quan sát kết quả trên ta thấy, khi $a = 13$, $b = 2$, $a/b = 6$ dư 1 nên hàm `fmod()` trả về kết quả là 1; hàm `trunc()` trả về kết quả là 6.

Bài tập chương 2

Bài 1. Tìm các lỗi trong cách viết biểu thức của C++ tương ứng với biểu diễn dạng trong biểu thức đại số của nó trong bảng sau:

Biểu thức đại số	Biểu thức C++
$(3)(7) + (7)(2)$	$(3)(7) + (7)(2)$
$\frac{3+6}{2}$	$3+6/2$
$\frac{13.5}{2.4-1.2}$	$13.5/2.4-1.2$

$3.8(2.5 + 6.3)$	$3.8(2.5 + 6.3)$
------------------	------------------

Bài 2. Xác định giá trị các biến trong vế trái của các câu lệnh gán sau đây:

- a) $a = 3 + 6 * 9$
- b) $b = 100 - 5 / 10$
- c) $c = 10 * (2 + 9 * 6)$
- d) $d = (100 + 7) \% 8$

Bài 3. Xác định kết quả của chương trình sau đây:

```
#include<iostream>
using namespace std;
int main()
{
    cout<<" Ket qua cua phep chia 13 cho 4 la "<<13 % 4;
    cout<<"\n Ket qua cua phep chia 8 cho 3 la: "<<8 % 3;
    return 0;
}
```

Bài 4. Xác định giá trị các biểu thức sau:

- a) $3.0 + 2.0 * 4.0;$
- b) $6.0 + 2.0 / 3.0 + 9.0;$
- c) $10.0 * (1.0 + 3.0 * 7.0);$
- d) $(20.0 - 7.0) * (3.0 - 1.0);$

Bài 5. Xác định giá trị các biểu thức hỗn hợp sau đây.

- a) $10.0 + 15 / 2 + 4.8;$
- b) $10.0 - 2 / 6 + 8;$
- c) $3.9 * 4 \% 6 + 9;$
- c) $11 + 18 / 5 + 100$

Bài 6. Giả sử a lưu giữ giá trị 1, m lưu giữ giá trị 50, n lưu giữ giá trị 10, và p lưu giữ giá trị nguyên 5. Tính giá trị của các biểu thức sau:

- a) $n / p + 3;$
- b) $m / p + n - 10 * a;$
- c) $m - 3 * n + 4 * a;$
- d) $(m + n) / (p + a);$
- e) $m + n / p + a;$

Bài 7. Các tên biến sau đây là đúng hay sai, tại sao?

- a) chieu_dai
- b) abc123
- c) abcd
- d) %354dg
- e) abc\$de
- f) do
- g) abc def

Bài 8. Viết các khai báo biến cho các yêu cầu sau đây:

- a) a và b dùng để lưu giữ các số nguyên
- b) c, d, và e dùng để lưu giữ các số thực
- c) f dùng để lưu trữ giá trị ký tự

Bài 9. Viết lại các khai báo sau đây bằng ba câu lệnh riêng rẽ cho mỗi ý.

- a) int thang=12, ngay=30, nam=2016;
- b) double gio=24, phut, giay;
- c) double diem_toan, diem_ly, diem_hoa;

Bài 10. Viết chương trình C++ hiển thị các kết quả của biểu thức $3.0*5.0$, $7.1*8.9 - 5.5$.

Tính toán giá trị bằng máy tính cầm tay để đối chiếu với đầu ra của chương trình.

Bài 11. Viết chương trình nhập vào từ bàn phím hai biến x và y kiểu nguyên. Sử dụng hàm abs() và pow(x,y) để hiển thị trị tuyệt đối của y và hiển thị $x^{|y|}$

Bài 12. Viết chương trình sử dụng hàm exp().

Bài 13. Viết chương trình sử dụng hàm log().

Bài 14. Viết chương trình sử dụng hàm sin(), cos(), tan()

Bài 15. Viết chương trình tính chu vi, diện tích hình tam giác có ba cạnh a, b, c nhập từ bàn phím

Gợi ý: Diện tích tam giác được tính theo công thức: $s = \sqrt{p*(p-a)*(p-b)*(p-c)}$, trong đó $p = (a + b + c)/2$.

Bài 17. Cho tam giác ABC, viết chương trình để thực hiện việc tính chu vi, diện tích và bán kính đường tròn nội tiếp, ngoại tiếp của nó.

Bài 18. Viết chương trình tính thể tích của một hình cầu bán kính r nhập từ bàn phím.

Biết công thức tính thể tích là: $V = \frac{4\pi r^3}{3}$

Bài 19. Viết chương trình tính thời gian đi tới đích của một chiếc ô tô biết độ dài quãng đường s và vận tốc trung bình của xe là v được nhập vào từ bàn phím.

Công thức tính thời gian: $t = s/v$

Bài 20. Viết chương trình tổng các số nguyên từ a tới b biết:

$$s = (n/2) * (2*a + (n-1)* d)$$

Với n, a, d kiểu số nguyên, được nhập vào từ bàn phím.

- s: tổng từ a tới b
- n: số lượng số nguyên được thực hiện tính tổng
- a: giá trị của số nguyên đầu tiên
- d: độ lệch giữa hai số liên tiếp (cách đều nhau)

Bài 21. Công thức Newton tính trọng lượng của một vật như sau:

$$F = M * A_e$$

F: trọng lượng của vật

M: khối lượng của vật

A_e : Gia tốc gây ra bởi lực hấp dẫn của trái đất (9.82 m/s^2)

Từ các thông tin trên, hãy viết chương trình tính trọng lượng của một vật với khối lượng của vật được nhập vào từ bàn phím.

Bài 22. Viết chương trình chuyển đổi nhiệt độ từ độ F sang độ C rồi hiển thị kết quả ra màn hình với nhiệt độ F được nhập vào từ bàn phím. Công thức chuyển đổi giữa hai thang nhiệt độ như sau: $C = 5.0/9.0(F-32.0)$.

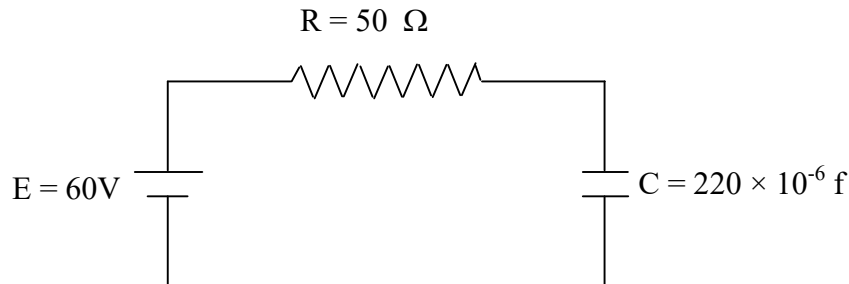
Bài 23. Viết chương trình để tính động năng của một vật khi nó di chuyển biết khối lượng của vật và vận tốc di chuyển được nhập vào từ bàn phím.

Công thức Newton để tính động năng như sau:

$$E = \frac{1}{2} * m * v^2$$

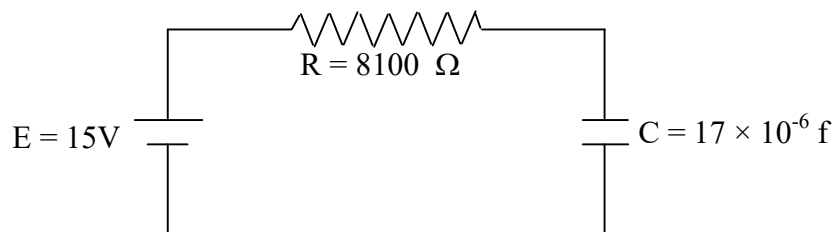
Với: E là động năng của vật, m là khối lượng của vật và v là vận tốc di chuyển của vật.

Bài 24. Cho mạch điện RC như hình 2-3; trong đó hiệu điện thế V được cho bởi công thức $V = E(1 - e^{-t/RC})/R$, t là thời gian (tính theo giây) sau khi công tắc bật. Sử dụng công thức trên tính hiệu điện thế qua tụ điện khi t có thời gian là 0.55 giây.



Hình 2-3. Mạch RC

Bài 25. Cho mạch điện như hình 2-4, trong đó cường độ dòng điện i được tính theo công thức $i = (E)e^{-t/RC}/R$. Sử dụng công thức này, hãy viết chương trình tính hiệu điện thế của tụ điện khi $t = 0.32$ giây.



Hình 2-4. Mạch RC

Bài 26. Định luật làm lạnh của Newton áp dụng cho một vật thể với nhiệt độ ban đầu là T được đặt trong môi trường có nhiệt độ là A , vật đó sẽ có nhiệt độ là T_{FIN} sau t phút được theo công thức $T_{FIN} = (T - A)e^{-kt} + A$. Trong đó $e = 2.71828$, k là hệ số nhiệt dựa vào chất liệu làm vật thể đó. Sử dụng công thức trên viết chương trình xác định nhiệt độ đạt tới của một vật thể sau 20 phút khi chúng được đặt trong nước với nhiệt độ 60 độ C. Giả sử rằng nhiệt độ của vật thể lúc đầu là 150 độ C và hệ số nhiệt k là 0.0367.

Chương 3. Các câu lệnh điều kiện

3.1 Giới thiệu

Câu lệnh có điều kiện là một trong những câu lệnh phổ biến trong khi lập trình. Rất nhiều đoạn lệnh chỉ được thực hiện khi điều kiện nào đó xảy ra. Trong câu lệnh điều kiện có các biểu thức logic tức là các biểu thức trả về một trong hai giá trị đúng hoặc sai. Để định giá và biểu diễn các biểu thức này chúng ta cần sử dụng các toán tử logic. Bảng 3-1 trình bày một số phép toán logic trong C++.

Toán tử quan hệ	Ngữ nghĩa	Ví dụ
<	Nhỏ hơn	diem < 7.0
>	Lớn hơn	chieudai > 100.0
<=	Nhỏ hơn hoặc bằng	tuoi <=30
>=	Lớn hơn hoặc bằng	namsinh>=1980
==	Bằng	a == 0
!=	Khác	x !=0

Bảng 3-1. Các toán tử quan hệ trong C++

Với các biểu thức logic phức tạp chứa các toán tử “và”, “hoặc” hay “phủ định”, trong C++ sẽ được dùng bởi &&, ||, ! tương ứng. Giống như các biểu thức thông thường, biểu thức logic trong C++ trả về giá trị 0 hoặc 1 trong đó 0 tương ứng với biểu thức có giá trị sai và 1 tương ứng với biểu thức có giá trị đúng. Ví dụ một số biểu thức logic như sau:

```
((a+b >c) && (b==0))
!(a + b>c)
((a == 0) || (b ==0) || (c == 0))
```

Ví dụ 3.1. Viết chương trình in các giá trị của biểu thức sau lên màn hình:

- 5 > 9
- 2 !=0
- (2 >3) & (9 >8)

Chương trình như sau:

```
#include<iostream>
```

```
using namespace std;
int main()
{
    cout<<"Gia tri cua bieu thuc 5>9 la: "<<(5>9)<<endl;
    cout<<"Gia tri cua bieu thuc 2!=0 la: "<<(2!=0)<<endl;
    cout<<"Gia tri cua bieu thuc (2>3) & (9>8) la: "<<((2>3) &
(9 > 8));
    return 0;
}
```

Kết quả thực hiện chương trình như sau:

```
Gia tri cua bieu thuc 5>9 la: 0
Gia tri cua bieu thuc 2!=0 la: 1
Gia tri cua bieu thuc (2>3) & (9>8) la: 0
```

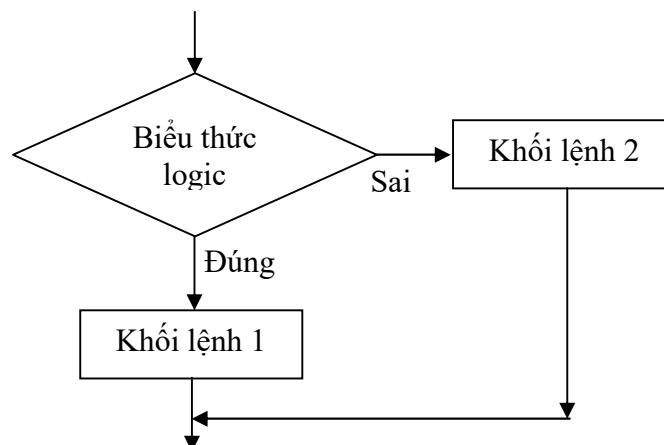
3.2 Câu lệnh if

Dạng 1: Cấu trúc if dạng đầy đủ:

Cấu trúc như sau:

```
if <biểu thức logic>
    <khối lệnh 1>;
else
    <khối lệnh 2>;
```

Cách thực hiện câu lệnh if: khi gặp câu lệnh này máy tính sẽ xác định giá trị của biểu thức logic, nếu giá trị này là đúng thì khối lệnh 1 sẽ được thực hiện, nếu sai khối lệnh 2 sẽ được thực hiện. Sơ đồ thực hiện khối lệnh minh họa ở hình 3-1.



Hình 3-1 Sơ đồ khối mô tả câu lệnh if

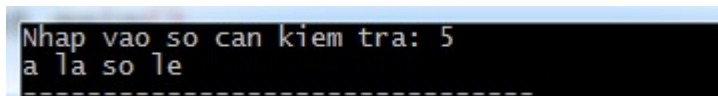
Chú ý rằng, sau `else` không có dấu chấm phẩy, khối lệnh bao gồm một hoặc nhiều câu lệnh của C++ và khi khối lệnh có từ hai câu lệnh trở lên thì phải được đặt trong cặp `{}`.

Ví dụ 3.2. Viết chương trình nhập vào số nguyên `a`, kiểm tra xem `a` có phải là số chẵn hay không rồi in kết quả lên màn hình.

Giải:

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    cout<<"Nhập vào số cần kiểm tra: "; cin>>a;
    if (a % 2 == 0)
        cout<<"a là số chẵn" ;
    else
        cout<<"a là số lẻ";
    return 0;
}
```

Ví dụ về kết quả thực hiện chương trình như sau:



```
Nhập vào số cần kiểm tra: 5
a là số lẻ
```

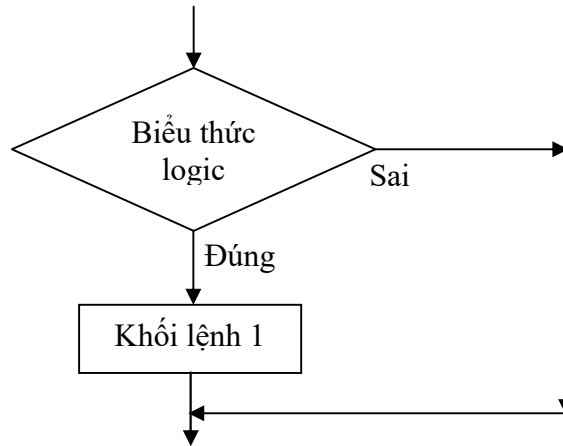
Trong ví dụ 3.1, số nguyên `a` được nhập vào từ bàn phím. Chương trình sẽ kiểm tra xem `a` chia dư cho 2 có bằng 0 hay không, nếu bằng 0 thì hiển thị ra màn hình “a là số chẵn”, ngược lại hiển thị “a là số lẻ”.

Dạng 2: Cấu trúc if dạng khuyết

Cú pháp:

```
if <biểu thức logic>
    <khối lệnh 1>;
```

Khối lệnh 1 được thực hiện khi biểu thức logic có giá trị đúng. Sơ đồ khối minh họa như trong hình 3-2.



Hình 3-2. Cấu trúc lệnh `if` dạng khuyết

Ví dụ 3.3. Viết chương trình nhập hai số nguyên `a`, `b` từ bàn phím. In lên màn hình số lớn nhất của hai số đó.

Giải: Chúng ta sẽ sử dụng câu lệnh `if` để giải bài toán trên. Chương trình như sau:

```
#include<iostream>
using namespace std;
int main()
{
    int a,b,max;
    cout<<"Nhập số thứ nhất: "; cin>>a;
    cout<<"Nhập số thứ hai: "; cin>>b;
    max=a;
    if (max<b)
        max=b;
    cout<<"Số lớn nhất Max = " <<max;
    return 0;
}
```

Ví dụ về kết quả thực hiện chương trình như sau:

```
Nhập số thứ nhất: 6
Nhập số thứ hai: 9
Số lớn nhất Max = 9
-----
```

Trong ví dụ 3.2, lệnh `if` kiểm tra điều kiện `max` nhỏ hơn `b`, nếu điều kiện này đúng, `max` được gán bằng `b`. Ở đây không có lệnh `else`, vậy khi điều kiện sai thì chương trình tiếp tục thực hiện câu lệnh tiếp theo sau `if` tức là sẽ hiển thị giá trị `max` ra màn hình.

3.3 Một số ví dụ về câu lệnh `if`

Ví dụ 3.5: Viết chương trình nhập vào số nguyên `b`, kiểm tra xem `b` có phải là số chính phương hay không rồi in kết quả lên màn hình.

Giải:

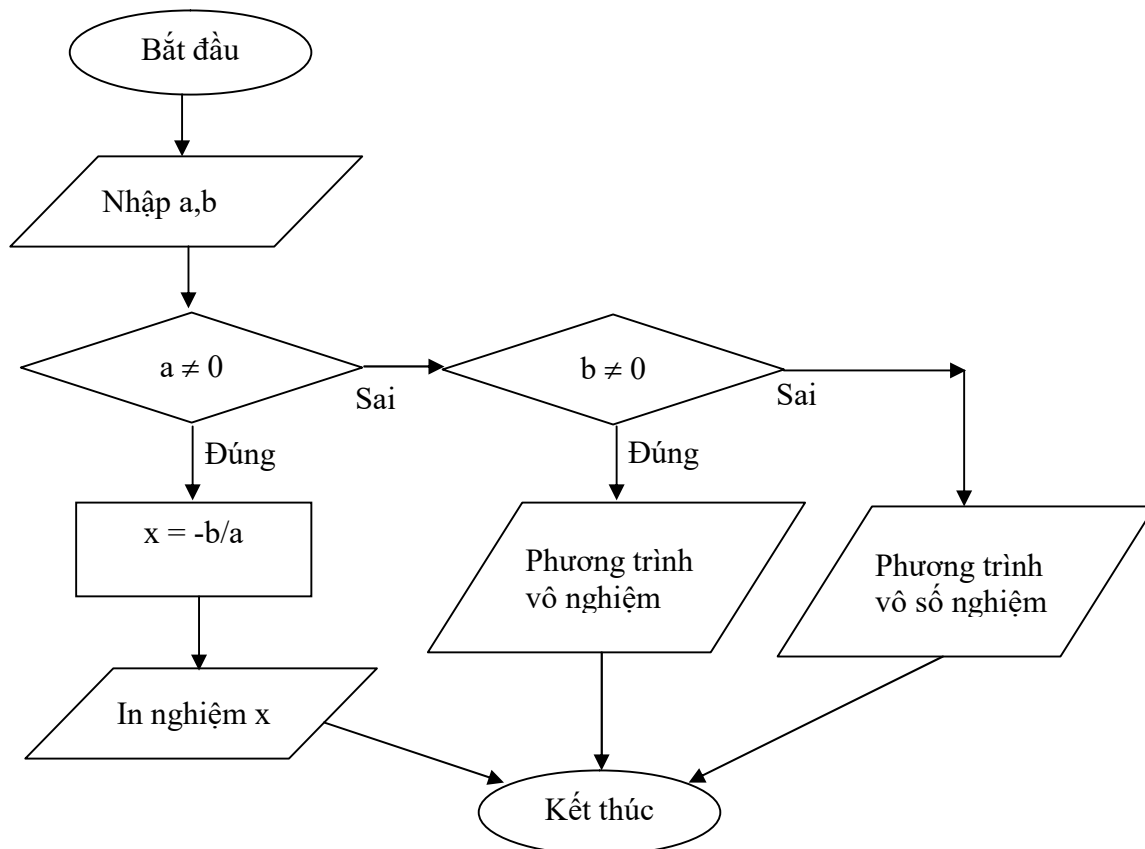
```
#include <iostream>
#include<cmath>
using namespace std;
int main()
{
    int b;
    cout<<"nhap vao so b ="; cin>>b;
    if (b>0)
        if (pow(floor(sqrt(b)),2) == b)
            cout<<"b la so chinh phuong" ;
        else
            cout<<"b khong la so chinh phuong";
    return 0;
}
```

Số chính phương là số có căn bậc hai là một số nguyên. Câu lệnh `if` thứ nhất kiểm tra xem `b` có lớn hơn 0 hay không. Nếu điều kiện này đúng, câu lệnh `if` thứ hai sử dụng hàm lũy thừa `pow()`, hàm lấy phần nguyên `floor()`, hàm căn bậc hai `sqrt()` để kiểm tra xem `b` có phải là số chính phương hay không. Chú ý rằng để sử dụng được các hàm toán học này ta phải sử dụng thư viện toán học `cmath`.

Trong ví dụ trên, chương trình có hai lệnh `if` và chỉ có một lệnh `esle`, khi đó trình biên dịch sẽ nhận biết lệnh `if` gần nhất so với `esle` là `if` của `esle` đó, còn lệnh `if` đầu tiên là cấu trúc `if` khuyết (không có `esle`).

Ví dụ 3.6. Viết chương trình nhập vào hai số a và b tương ứng là hệ số của phương trình bậc nhất $ax + b = 0$, tính và in nghiệm của phương trình lên màn hình.

Giải: Chúng ta đã biết sơ đồ khối minh họa giải thuật của bài toán như hình 3-3.



Hình 3-3. Sơ đồ khối minh họa thuật toán giải phương trình bậc nhất

Chương trình như sau:

```

#include <iostream>
using namespace std;
int main()
{
    float a, b, x;
    cout<<"Giai phuong trinh bac nhat ax + b = 0"<<endl;
    cout<<"Nhap he so a = "; cin>>a;
    cout<<"Nhap he so b = "; cin>>b;
    if (a != 0)
    {

```

```

        x = -b/a;
        cout<<"Nghiem cua pt: x =" << x;
    }
else
    if (b==0)
        cout<<"Phuong trinh vo so nghiem";
    else
        cout<<"Phuong trinh vo nghiem";
return 0;
}

```

Ví dụ 3.6 thực hiện chương trình giải phương trình bậc nhất. Câu lệnh `if` thứ nhất cần thực hiện hai câu lệnh nên chúng được đưa vào trong khối lệnh `{}`. Trong phần `else` chúng ta cần tiếp tục kiểm tra điều kiện thứ hai nên sử dụng cấu trúc `if - else` thứ hai. Cấu trúc `if - else` thứ hai được coi là một câu lệnh nên không cần sử dụng dấu mở khối và đóng khối lệnh ở đây. Ví dụ về kết quả thực hiện chương trình như sau:

```

Giai phuong trinh bac nhat ax + b = 0
Nhap he so a = 4
Nhap he so b = 5
Nghiem cua pt: x =-1.25
-----

```

Ví dụ 3.7. Nhập vào 3 số thực a , b , và c tương ứng là các hệ số của phương trình bậc hai $ax^2 + bx + c = 0$. Viết chương trình giải phương trình này.

Giải: Việc giải phương trình bậc hai được thực hiện từng bước một như trong toán học. Chúng ta sử dụng câu lệnh `if` để chia các trường hợp tương ứng để tính nghiệm. Chương trình như sau:

```

#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    float a, b, c, x1, x2, d;
    cout<<"nhap a ="; cin>>a;
    cout<<"nhap b ="; cin>>b;

```

```

cout<<"nhap c ="; cin>>c;
d=b*b-4*a*c;
if (d > 0)
{
    x1 = (-b+ sqrt(d))/(2*a);
    x2 = (-b- sqrt(d))/(2*a);
    cout<<"Nghiem x1 =" <<x1<<endl;
    cout<<"Nghiem x2 =" <<x2;
}
else
    if (d==0)
        cout<<"Phuong trinh co mot nghiem x= "<<-b/(2*a);
    else
        cout<<"Phuong trinh vo nghiem";
return 0;
}

```

Ví dụ 3.8. Nhập vào điểm toán, điểm lý, điểm hóa của một sinh viên. Tính điểm trung bình của sinh viên đó.

- Nếu điểm trung bình < 5: Hiển thị “*Loai yeu*”
- Nếu điểm trung bình < 7: Hiển thị “*Loai trung binh*”
- Nếu điểm trung bình < 8.5: Hiển thị “*Loai kha*”
- Trái lại: hiển thị “*Loai gioi*”

Giải: Chúng ta sẽ sử dụng câu lệnh `if` dạng đầy đủ thực hiện việc chia các trường hợp của bài toán tương ứng. Chương trình như sau:

```

#include<iostream>
using namespace std;
int main()
{
    float diem_toan, diem_ly, diem_hoa, diem_tb;
    cout<<"Diem toan = "; cin>>diem_toan;
    cout<<"Diem hoa = "; cin>>diem_hoa;

```



```

cout<<"Diem ly = "; cin>>diem_ly;
diem_tb=(diem_toan+diem_ly+diem_hoa)/3;
if(diem_tb>=0&&diem_tb<=10)
    if(diem_tb<5)
        cout<<"Loai Yeu";
    else if(diem_tb<7)
        cout<<"Loai trung binh";
    else if(diem_tb<8.5)
        cout<<"Loai kha";
    else
        cout<<"Loai gioi";
else cout<<"Nhap lai diem";
return 0;
}

```

3.4 Cấu trúc switch

Câu lệnh `switch` cung cấp một sự lựa chọn cho việc viết các đoạn lệnh có nhiều lựa chọn mà giá trị điều kiện của nó là một tập hợp các số nguyên. Cấu trúc câu lệnh `switch` như sau:

```

switch (biểu thức nguyên)
{
    case giá_trị_1:
        <Khối lệnh 1>; break;
    case giá_trị_2:
        <Khối lệnh 2>; break;
    ...
    case giá_trị_n:
        <Khối lệnh n>; break;
    default:
        <Khối lệnh n+1>;
}

```

Câu lệnh trên có bốn từ khóa là: `switch`, `case`, `break`, và `default`. Khi gặp câu lệnh trên giá trị của biểu thức sẽ được xác định, nếu giá trị của biểu thức trùng với `giá_trị_i` sau từ khóa `case` thì <khối lệnh i> sẽ được thực hiện. Nếu không <khối lệnh n+1> sẽ được thực hiện. Máy sẽ thoát khỏi lệnh `switch` khi nó gặp câu lệnh `break` hoặc dấu ngoặc nhọn đóng cuối cùng của thân `switch`. Ta cũng có thể dùng câu lệnh `goto` trong thân của toán tử `switch` để nhảy tới một câu lệnh bất kỳ bên ngoài `switch`.

Ví dụ 3.9: Viết chương trình nhập vào một tháng của năm (tính theo dương lịch), tính xem tháng đó có bao nhiêu ngày rồi in kết quả lên màn hình

Giải:

```
#include<iostream>
using namespace std;
int main()
{
    int t, n;
    cout<<"nhap thang t ="; cin>>t;
    cout<<"nhap nam ="; cin>>n;
    switch (t)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
            cout<<"Thang "<<t<<" nam " <<n <<" co 31 ngay";
            break;
        case 4:
        case 6:
        case 9:
        case 11:
            cout<<"Thang "<<t<<" nam " <<n <<" co 30 ngay";
            break;
        case 2:
            if(((n%4==0)&&(n % 100 !=0))||(n%400 == 0))
                cout<<"Thang 2 nam " <<n <<" co 29 ngay";
```

```

        else
            cout<<"Thang 2 nam co 28 ngay";
            break;
    }
    return 0;
}

```

Chú ý: Việc xác định số ngày của tháng 2 của năm nào đó theo hai trường hợp sau:

- Nếu năm đó chia hết cho 4 và không chia hết cho 100 thì tháng 2 có 29 ngày
- Nếu năm đó chia hết cho 100 thì nếu năm đó cũng chia hết cho 400 thì tháng 2 của năm đó có 29 ngày
- Các trường hợp khác tháng 2 sẽ có 28 ngày.

Ví dụ 3.10. Viết chương trình nhập vào một ký tự. Nếu ký tự đó là a/e/o/u/i thì thông báo là nguyên âm, trái lại thông báo là phụ âm.

Giải: Đối với biến kiểu kí tự luôn được chuyển đổi sang dạng nguyên trong biểu thức nên chúng ta hoàn toàn có thể áp dụng câu lệnh `switch`. Chương trình như sau:

```

#include<iostream>
using namespace std;
int main()
{
    char ky_tu;
    cout<<"Nhap ky = "; cin>>ky_tu;
    if(ky_tu>='a'&&ky_tu<='z')
        switch (ky_tu)
        {
            case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u':
                cout<<"Nguyen am";
                break;
            default:
                cout<<"Phu am";
        }
    else
        cout<<"Nhap lai ky tu";
    return 0;
}

```

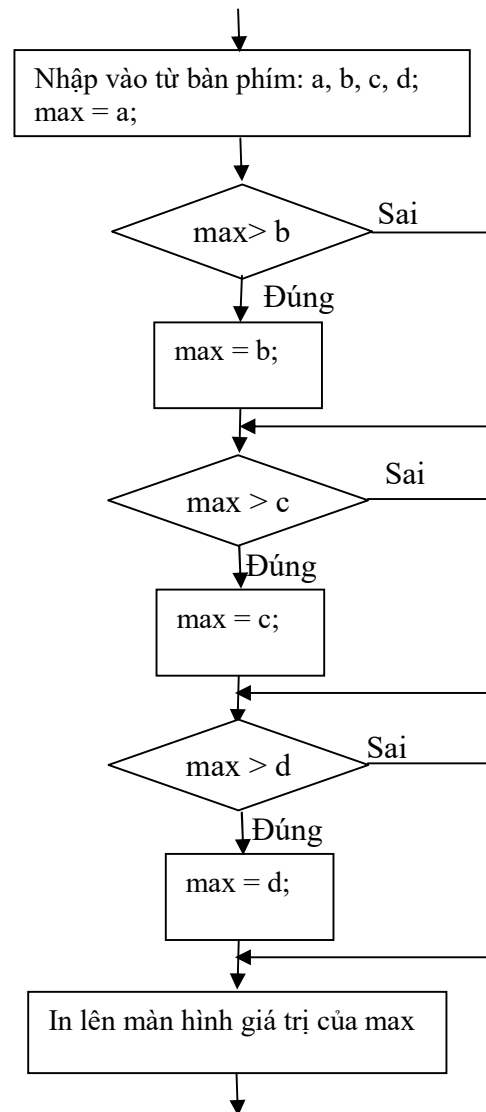
}

Bài tập chương 3

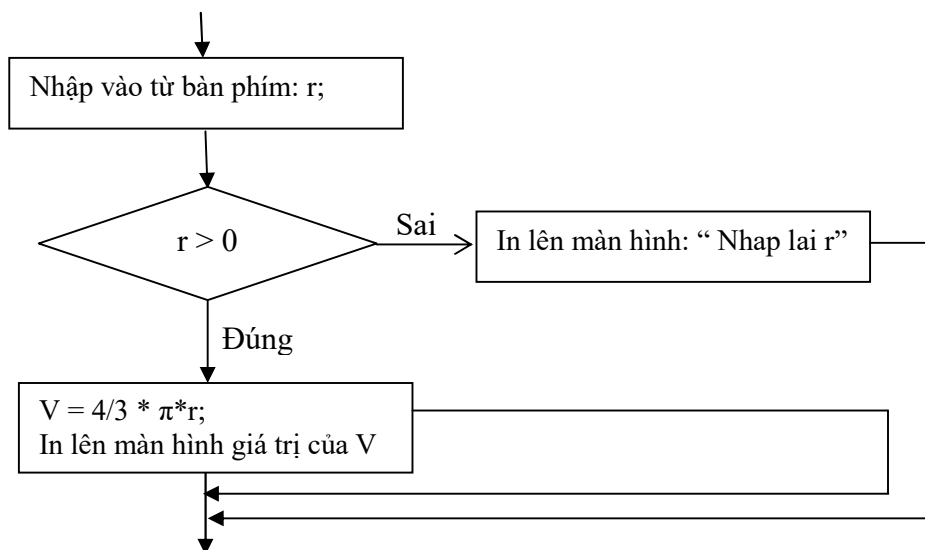
Bài 1. Viết chương trình nhập vào số a kiểu nguyên. Kiểm tra xem a chẵn hay lẻ. Nếu a chẵn hiển thị “Số chẵn”, trái lại hiển thị “Số lẻ”.

Bài 2. Viết đoạn lệnh tương ứng với các sơ đồ khối sau:

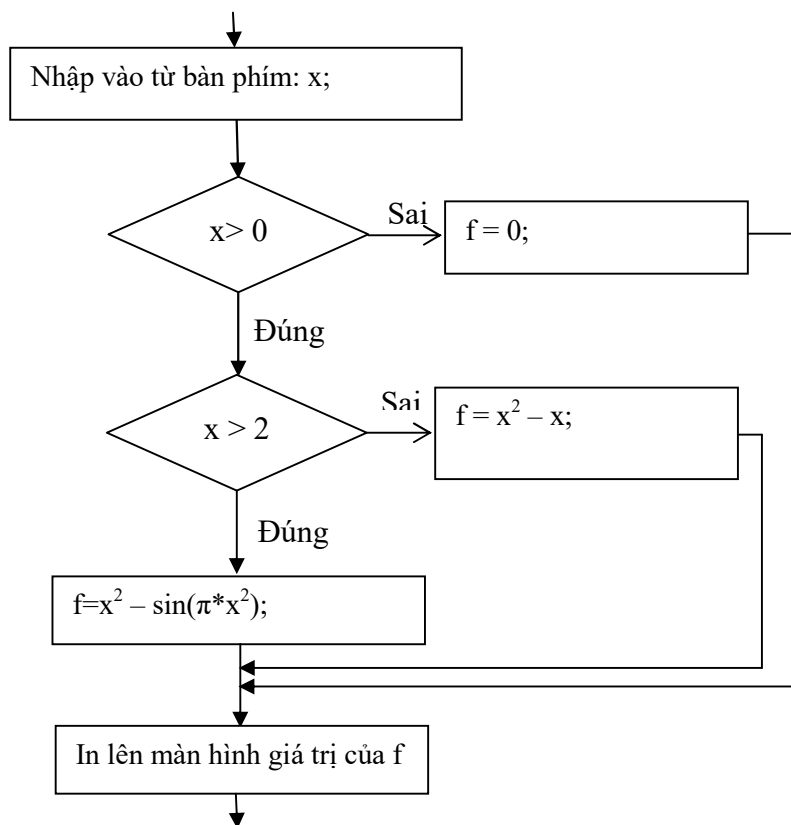
a.



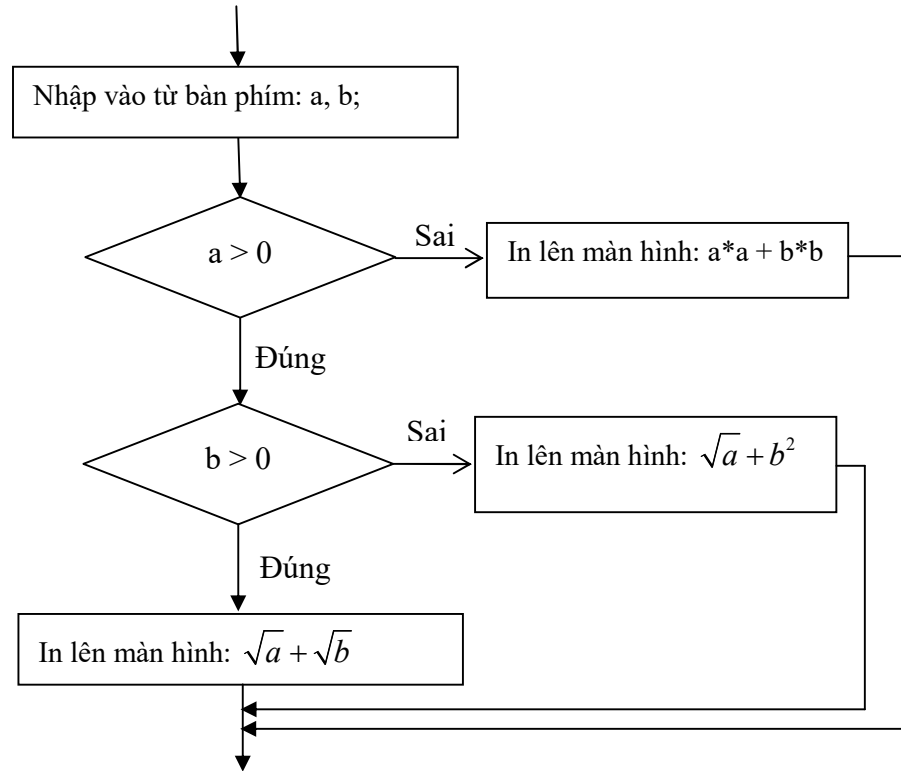
b.



c.



d.



Bài 3. Thang điểm quy định việc xếp loại sinh viên được cho bởi bảng sau:

Điểm trung bình	Xếp loại
Nhỏ hơn 5	Kém
Từ 5 tới nhỏ hơn 7	Trung bình
Từ 7 tới nhỏ hơn 8.5	Khá
Từ 8.5 tới 10	Giỏi

Sử dụng thông tin của bảng trên, viết chương trình bằng ngôn ngữ C++ thực hiện việc nhập vào điểm trung bình của một sinh viên bất kỳ, từ đó xác định xếp loại học lực của sinh viên đó và hiển thị kết quả xếp loại ra màn hình.

Bài 4. Việc quy đổi điểm số từ thang điểm 10 sang thang điểm chữ với sinh viên học theo tín chỉ được thực hiện theo bảng sau:

Điểm số	Điểm chữ
Nhỏ hơn 4	F
Từ 4 tới 5.4	D
Từ 5.5 tới 6.9	C
Từ 7 tới 8.4	B
Từ 8.5 tới 10	A

Sử dụng thông tin trong bảng trên, viết chương trình bằng ngôn ngữ C++ thực hiện việc nhập vào điểm số của một sinh viên bất kỳ, hiển thị kết quả điểm chữ của sinh viên đó ra màn hình.

Bài 5. Viết chương trình nhập vào một biến nhiệt độ. Nếu nhiệt độ nhỏ hơn 100 thông báo “Nho hon nhiet do soi cua nuoc”, nếu bằng 100 thông báo “Bang nhiet do soi cua nuoc”, trái lại thông báo “Lon hon nhiet do soi cua nuoc”.

Bài 6. Viết chương trình nhập vào một số. Nếu số đó âm hiển thị thông báo “So am”, nếu bằng 0 hiển thị thông báo “So 0”, trái lại hiển thị “So duong”.

Bài 7. Viết chương trình nhập vào hai số x, y. Nếu x lớn hơn y và y khác 0 thì tính và hiển thị x/y, trái lại hiển thị x*y.

Bài 8. Viết chương trình nhập số x từ bàn phím. Kiểm tra nếu x chẵn thì hiển thị x^2 , trái lại hiển thị x^3

Bài 9. Viết chương trình nhập vào hai số x, y. Nếu $|x-y| < 0.00005$ thì hiển thị sai số bằng 0, trái lại hiển thị sai số bằng $|x-y|/2$.

Bài 10. Viết chương trình nhập vào hai số a và b. Nếu $a > b$ thì yêu cầu nhập c rồi hiển thị $(a-b)*c$, trái lại hiển thị $a*b$.

Bài 11. Viết chương trình nhập vào hai cạnh của hình chữ nhật a, b. Kiểm tra, nếu a, b dương thì tính diện tích hình chữ nhật, trái lại, thông báo “Du lieu nhap khong dung”.

Bài 12. Viết chương trình nhập vào hai số a và b. Thông báo lên màn hình số nào lớn nhất trong hai số trên.

Bài 13. Một góc là nhọn nếu nó nhỏ hơn 90 độ, là vuông nếu nó là 90 độ và là tù nếu nó lớn hơn 90 độ. Viết chương trình nhập vào số đo của một góc, hiển thị lên màn hình góc đó là góc loại gì?.

Bài 14. Viết chương trình nhập vào một số và theo sau là một dấu cách và tiếp theo là một kí tự. Nếu kí tự đó là f thì số đã nhập là nhiệt độ tính theo Fahrenheit và yêu cầu chuyển sang thang nhiệt độ Celsius, ngược lại nếu kí tự nhập vào là c thì ta sẽ chuyển từ Celsius sang Fahrenheit. Nếu kí tự không phải là f hoặc c thì thông báo lên màn hình nhập sai dữ liệu và kết thúc chương trình. Công thức chuyển đổi như sau:

$$\text{Celsius} = (5.0/9.0) * (\text{Fahrenheit} - 32.0)$$

$$\text{Fahrenheit} = (9.0/5.0) * \text{Celsius} + 32$$

Bài 15. Viết lại đoạn câu lệnh sau sử dụng switch

```
if (factor == 1)
    p = 25.0;
else
    if (factor == 2)
        p = 6.0;
    else
        if (factor == 3)
            p = 100;
        else
            if ((factor == 4) || (factor == 6))
                p = 200;
```

Bài 16. Phân biệt sự khác nhau giữa if và switch, cho ví dụ minh họa.

Bài 17. Viết chương trình nhập vào ba số a, b, c tương ứng là ngày, tháng năm nào đó. In lên màn hình xem đó có phải là ba số hợp lệ hay không.

Bài 18. Viết chương trình nhập vào một năm nào đó. In lên màn hình năm đó có phải là năm nhuận hay không.

Bài 19. Viết chương trình nhập vào 4 số thực a, b, c, d. Tìm và in ra số lớn nhất, số nhỏ nhất trong 4 số đó

Bài 20. Viết chương trình nhập vào 2 số x, y và 1 trong 4 toán tử +, -, *, /. Nếu là + thì in ra kết quả x + y, nếu là - thì in ra x - y, nếu là * thì in ra x * y, nếu là / thì in ra x / y (nếu y = 0 thì thông báo không chia được).

Bài 21. Viết chương trình nhập vào 3 số thực dương a, b, c. Kiểm tra xem a, b, c có phải là 3 cạnh của tam giác không? Nếu là 3 cạnh của tam giác thì tính diện tích của tam giác theo công thức sau:

$$s = \sqrt{p * (p - a) * (p - b) * p - c)}, \text{ với } p = (a + b + c) / 2$$

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

Bài 22. Viết chương trình tính giá trị của hàm f, với x là số thực được nhập từ bàn phím.

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x^2 - x & 0 < x \leq 2 \\ x^2 - \sin \pi x^2 & x > 2 \end{cases}$$

Bài 23. Viết chương trình tính tiền điện với chỉ số mới và chỉ số cũ được nhập vào từ bàn phím. In ra màn hình chỉ số cũ, chỉ số mới, và số tiền phải trả. Biết rằng 100 kWh đầu giá 550, từ kWh 101 - 150 giá 1.110, từ kWh 151 - 200 giá 1.470, từ kWh 201 - 300 giá 1.600, từ kWh 301 - 400 giá 1.720, từ kWh 401 trở lên giá 1.780.

Bài 24. Viết chương trình nhập vào các hệ số $a_1, b_1, c_1, a_2, b_2, c_2$ để giải phương trình bậc nhất như sau:

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$$

Chương 4. Các câu lệnh lặp

4.1 Giới thiệu

Trong khi thiết kế và xây dựng chương trình chúng ta thường gặp các đoạn công việc phải thực hiện lặp đi lặp lại nhiều lần theo một tiêu chuẩn nào đó. Chẳng hạn việc lặp đi lặp lại việc tính tổng một dãy số, lặp đi lặp lại việc tìm kiếm các giá trị thỏa mãn tiêu chuẩn nào đó lên màn hình v.v. Các bài toán như vậy tương ứng với một chu trình lặp trong sơ đồ khối đã xét ở chương 1. Để viết mã cho các chu trình lặp, ngôn ngữ C++ cung cấp cho chúng ta ba loại câu lệnh có thể sử dụng bao gồm:

- `while`
- `for`
- `do while`

Về cơ bản ba câu lệnh này là tương đương, tuy nhiên chúng có một số điểm khác nhau trong quá trình thực hiện. Khi viết chương trình, người lập trình cần phải vận dụng linh hoạt để chọn câu lệnh nào là phù hợp nhất với bài toán của mình. Chúng ta sẽ tìm hiểu từng loại câu lệnh trong các phần tiếp theo.

4.2 Câu lệnh `while`

Cấu trúc của câu lệnh `while` như sau:

```
while (biểu thức logic)
    <khối lệnh>;
```

trong đó `while` là từ khóa, quá trình thực hiện cấu trúc `while` được mô tả như hình 4-1.

Các bước thực hiện như sau:

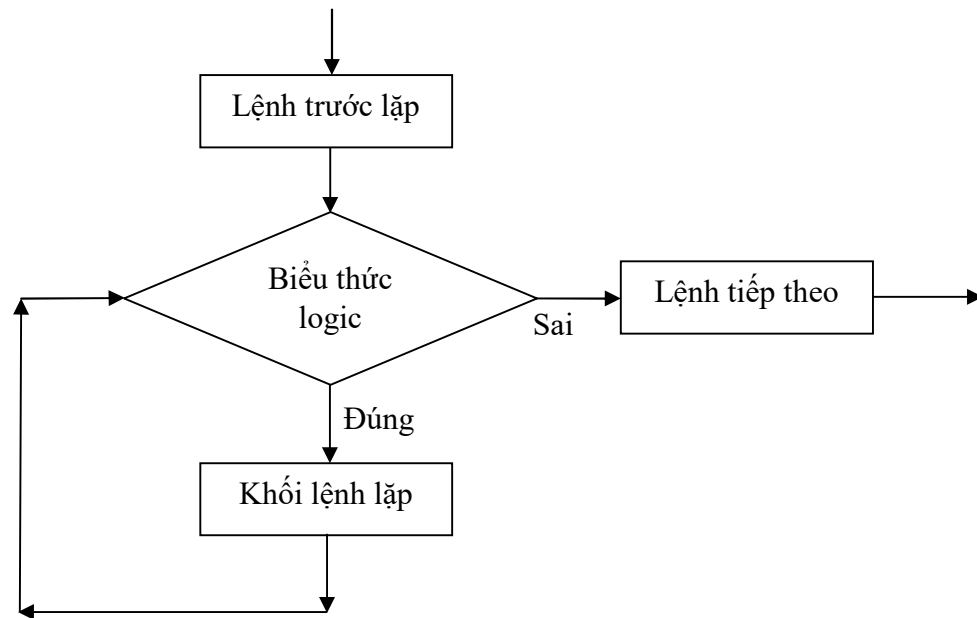
Bước 1: <biểu thức logic> sẽ được định giá trị

Bước 2:

- Nếu <biểu thức logic> đúng thì sẽ thực thi khối lệnh và quay lại bước 1
- Nếu <biểu thức logic> sai, thoát khỏi vòng lặp.

Trong khối lệnh sẽ có một câu lệnh điều khiển giá trị của biểu thức logic sau mỗi lần lặp. Khi viết chương trình chúng ta phải chú ý xác định chính xác điều kiện lặp của

biểu thức logic để tránh trường hợp vòng lặp của chúng ta thực hiện mãi mà không thoát khỏi để thực hiện các câu lệnh khác.



Hình 4-1. Sơ đồ khối biểu diễn câu lệnh `while`

Ví dụ 4.1: Viết chương trình hiển thị các số từ 1 đến n lên màn hình. Với n nhập từ bàn phím.

Giải:

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Nhập giá trị lớn nhất cần hiển thị: "; cin>>n;
    int i = 1; // Lệnh khởi tạo
    while (i <= n)
    {
        cout << i << " ";
        i++; // Thay đổi biến điều kiện
    }
    return 0;
}
```

Chương trình trong ví dụ 4.1 thực thi việc hiển thị các số từ 1 tới n . Giả sử người sử dụng nhập $n = 5$. Ban đầu i được gán giá trị bằng 1, điều kiện trong `while` thỏa mãn

nên nó hiển thị giá trị 1 ra màn hình. Lệnh `i++` giúp tăng giá trị `i` lên 2, khi đó điều kiện vẫn thỏa mãn nên số 2 được hiển thị. Cứ như vậy cho tới khi `i` lớn hơn 5, điều kiện không thỏa mãn nên vòng lặp dừng lại.

Vòng lặp như trong ví dụ trên được gọi là vòng lặp tiến. Chúng ta có thể tạo ra vòng lặp lùi như trong ví dụ sau:

Ví dụ 4.2. Viết chương trình hiển thị các số nguyên theo thứ tự từ $n \rightarrow 1$. Với `n` nhập vào từ bàn phím.

Giải:

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Nhập giá trị lớn nhất cần hiển thị: "; cin>>n;
    int i=n; // Lệnh khởi tạo
    while (i >= 1)
    {
        cout << i << " ";
        i= i -1; // Thay đổi biến điều kiện
    }
    return 0;
}
```

Trong cả hai ví dụ trên, biến `i` khi thay đổi điều kiện đều thay đổi 1 đơn vị, trong ví dụ tiếp theo, `i` sẽ thay đổi 2 đơn vị.

Ví dụ 4.3. Viết chương trình tính và hiển thị tổng các số chẵn từ 1 tới 100.

Giải:

```
#include <iostream>
using namespace std;
int main()
{
    int i=2, tong = 0; // Lệnh khởi tạo
    while (i <= 100)
    {
        tong = tong + i;
        i = i+ 2; // Thay đổi biến điều kiện
    } //Kết thúc lặp
}
```

```

    cout<<"Tong = "<<tong; // Lenh sau lap
    return 0;
}

```

Ví dụ 4.4. Viết chương trình tính tổng sau. Hiển thị kết quả ra màn hình.

$$S = 1 + \left(\frac{1}{1+2^3}\right)^2 + \left(\frac{1}{2+3^4}\right)^2 + \dots + \left(\frac{1}{(n-1)+n^{n+1}}\right)^2$$

Giải: Để tính tổng S thì quá trình sẽ lặp đi lặp lại n - 1 lần, từ công thức tính S theo toán học ta có tại bước thứ i giá trị của S sẽ được cộng thêm một lượng $(1/1+2^{i+1})^i$. Giá trị của i sẽ đi từ 2 đến n. Biểu thức logic trong trường hợp này là $i \leq n$. Chương trình như sau:

```

#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int i;
    float s,n;
    cout<<"Nhap n ="; cin>>n;
    s=1; i=2;
    while (i <=n)
    {
        s= s + 1/(pow(((i-1) + pow(i,i+1)),2));
        i=i+1;
    }
    cout<<"Ket qua tong S = "<<s;
    return 0;
}

```

Ví dụ 4.5. Viết chương trình tính tổng sau và hiển thị kết quả ra màn hình, với n nhập từ bàn phím:

$$S = \left(\frac{1}{1+2}\right)^2 + \left(\frac{1+2}{2+3}\right)^2 + \dots + \left(\frac{(n-2)+(n-1)}{(n-1)+n}\right)^2$$

Giải: Tương tự như ví dụ 4.4 quá trình tính tổng S cũng là lặp đi lặp lại với i đi từ 2 đến n.

```

#include<iostream>
#include<cmath>
using namespace std;
int main()

```

```

{
    float s,i,n;
    cout<<"Nhap n ="; cin>>n;
    s=0; i=2;
    while (i <=n)
    {
        s= s + pow((2*i-3),2)/pow((2*i -1),2);
        i=i+1;
    }
    cout<<"Ket qua tong S = "<<s;
    return 0;
}

```

Trong các ví dụ từ đầu tới giờ chúng ta đều dùng vòng lặp có số lần lặp xác định. Tuy nhiên, vòng lặp `while` còn cho phép xây dựng số lần lặp không biết trước. Ví dụ 4.6 minh họa điều này. Trong ví dụ này, số lần lặp phụ thuộc vào số điểm nhập vào, nếu `diem > 100` thì vòng lặp sẽ dừng lại.

Ví dụ 4.6 Viết chương trình tính và hiển thị tổng điểm của một sinh viên với điểm số các môn học của sinh viên được nhập vào từ bàn phím. Việc nhập dữ liệu điểm dừng lại khi `điểm > 100`.

Giải:

```

#include <iostream>
using namespace std;
int main()
{
    const int DIEM_MAX = 100;
    double diem, tong;
    diem = 0;
    tong = 0;
    cout << "Nhap n > 100 de dung"<<endl;
    while (diem <= DIEM_MAX)
    {
        tong = tong + diem;
        cout << "Diem = "<<endl;
        cin >> diem;
    }
    cout << "Tong diem = " << tong << endl;
    return 0;
}

```

4.3 Câu lệnh `for`

Cấu trúc của câu lệnh `for` như sau:

`for` (khởi tạo biến đếm; biểu thức điều kiện; câu lệnh thay đổi biến đếm)
 <khối lệnh>

Trong đó `for` là từ khóa, câu lệnh `for` được thực hiện theo các bước như sau:

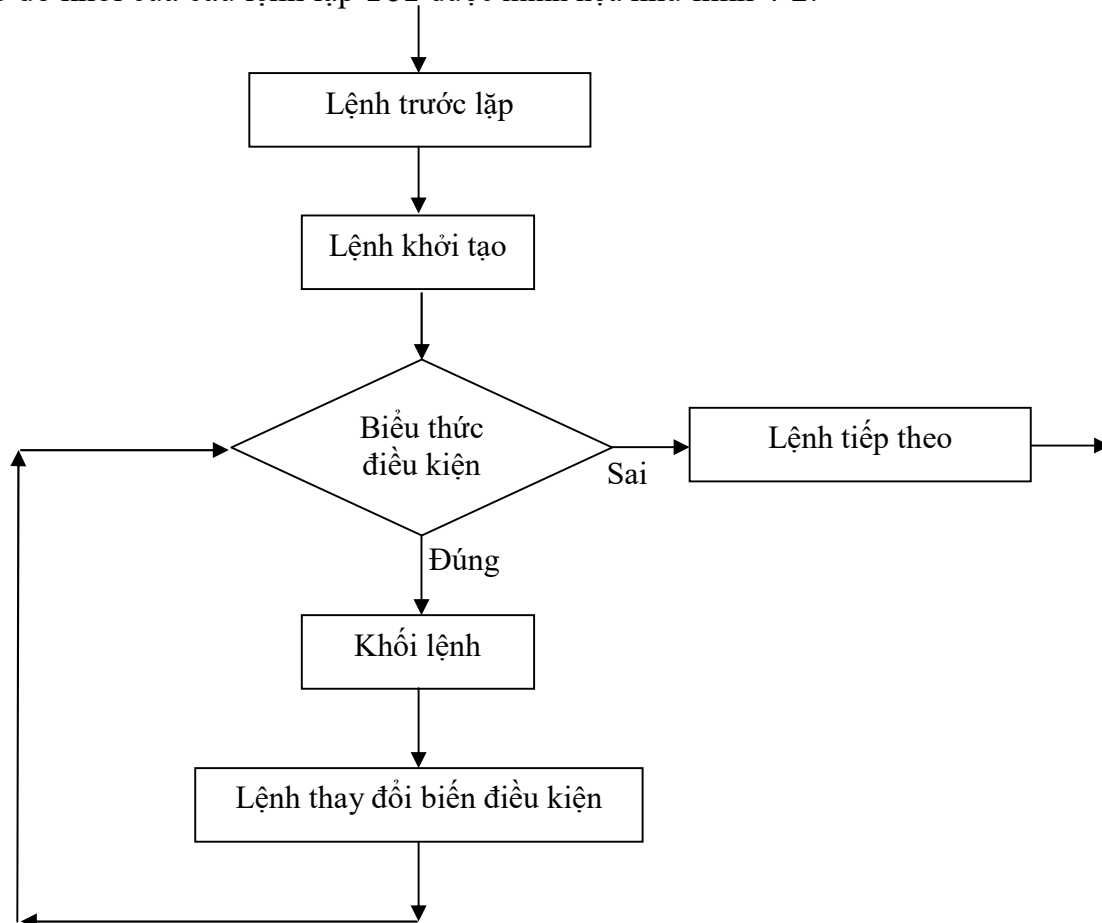
Bước 1: <Khởi tạo biến đếm> nhằm thiết lập giá trị ban đầu cho biến đếm

Bước 2: Kiểm tra <biểu thức điều kiện>

Bước 3: Nếu biểu thức điều kiện có giá trị sai thì thoát khỏi câu lệnh `for`, nếu không:

- <khối lệnh> sẽ được thực hiện
- Thực thi lệnh thay đổi biến điều kiện
- Quay lại thực hiện bước 2

Sơ đồ khối của câu lệnh lặp `for` được minh họa như hình 4-2:



Hình 4-2. Sơ đồ khối biểu diễn câu lệnh `for`

Câu lệnh `for` thực hiện các bước giống câu lệnh `while`, chỉ khác về dạng cú pháp. Khi số lần lặp là cố định, việc sử dụng `for` đơn giản và thường được sử dụng hơn `while`.

Ví dụ 4.7. Sử dụng cấu trúc `for` viết chương trình nhập vào một số `n` từ bàn phím. Hiển thị các giá trị nguyên từ 1 đến `n` ra màn hình.

Giải:

```
#include <iostream>
using namespace std;
int main()
{
    int i, n;
    cout<<"Nhập giá trị lớn nhất cần hiển thị: "; cin>>n;
    for(i = 1; i<=n; i++)
        cout<<i<<" ";
    return 0;
}
```

Trong ví dụ 4.7, giả sử người dùng nhập giá trị `n=5`. Trước tiên biến `i` được khởi tạo giá trị bằng 1, sau đó lệnh lặp kiểm tra xem `i` có nhỏ hơn 5 không (giá trị lớn nhất của vòng lặp), vì `i` nhỏ hơn 5 nên lệnh hiển thị `i` được thực hiện giúp hiển thị số 1 ra màn hình, tiếp theo, `i++` được thực hiện, `i` tăng lên bằng 2 vẫn nhỏ hơn 5 nên số 2 được hiển thị ra màn hình, tương tự như vậy, vòng lặp được thực hiện tới khi `i` tăng lên và lớn hơn 5 thì vòng lặp được dừng lại.

Ví dụ 4.8 Viết chương trình tính và hiển thị giá trị trung bình cộng của các số lẻ từ 1 tới 10.

Giải:

```
#include <iostream>
using namespace std;
int main()
{
    int i, tong=0, dem=0;
    float tbc;
    for(i = 1; i<=10; i=i+2)
    {
        tong = tong + i;
        dem = dem +1;
    }
}
```



```

    tbc=float(tong)/dem;
    cout<<"Trung binh cong cac so le tu 1 - 10 la: "<<tbc;
    return 0;
}

```

Chương trình trong ví dụ 4.8 thực hiện việc tính tổng các số lẻ từ 1 tới 10, đếm các số lẻ và sau đó tính trung bình của các số lẻ. Vòng lặp được khởi tạo với giá trị $i = 1$, lệnh tăng i được thực hiện với $i = i + 2$ để tính giá trị lẻ tiếp theo.

Ví dụ 4.9. Viết chương trình tính và hiển thị tổng sau ra màn hình. Với n nhập từ bàn phím.

$$S = 1 + \left(\frac{1}{1+2^3}\right)^2 + \left(\frac{1}{2+3^4}\right)^2 + \dots + \left(\frac{1}{(n-1)+n^{n+1}}\right)^2$$

Giải: Mấu chốt của việc sử dụng câu lệnh `for` là xác định giá trị đầu, giá trị cuối và điều kiện kết thúc của câu lệnh. Chương trình như sau:

```

#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    float s,i,n;
    cout<<"nhap n ="; cin>>n;
    s=1;
    for (i=2; i<=n;i=i+1)
        s= s + 1/(pow(i-1 + pow(i,i+1),2));
    cout<<"Tong s = "<<s;
    return 0;
}

```

Ví dụ 4.10. Viết chương trình nhập n nguyên dương và số thực x từ bàn phím, tính và in lên màn hình tổng S được xác định bởi công thức sau:

$$S = x - x^{1/2} + x^{1/3} + \dots (-1)^{n+1} x^{1/n}$$

Giải: Trong ví dụ này, chúng ta thấy việc tính tổng được lặp đi lặp lại, chỉ khác là dãy đơn dấu. Sử dụng vòng lặp `for`, chương trình như sau:

```

#include<iostream>
#include<cmath>
using namespace std;
int main()

```

```

{
    double n;
    double i, x, s;
    cout<<"nhap n ="; cin>>n;
    cout<<"Nhap x = "; cin>>x;
    s=0;
    for (i=1; i<=n;i=i+1)
        s= s + pow(-1,i+1)*pow(x,1/i);
    cout<<"Tong s = "<<s;
    return 0;
}

```

Trong nhiều trường hợp ta cần sử dụng một vòng lặp nằm trong một vòng lặp khác hay còn gọi là vòng lặp lồng nhau. Ví dụ sau minh họa điều này:

Ví dụ 4.11. Viết chương trình nhập vào số lượng sinh viên và số môn học của một sinh viên. Tính và hiển thị điểm trung bình của từng sinh viên.

Giải:

```

#include <iostream>
using namespace std;
int main()
{
    int i, j, so_mon_hoc, so_sv;
    double diem, tong_diem, diem_tb;
    cout<<"Nhap vao so sinh vien: "; cin>>so_sv;
    cout<<"Nhap vao so mon hoc: "; cin>>so_mon_hoc;
    for (i = 1; i <= so_sv; i++)
    {
        tong_diem = 0; // Xoa tong_diem ve 0 de tinh cho tung sv
        for (j = 1; j <= so_mon_hoc; j++)
        {
            cout << "Nhap diem mon hoc "<<j <<" cua sinh vien: ";
            cin >> diem;
            tong_diem=tong_diem + diem;
        }
        diem_tb = tong_diem / so_mon_hoc;
        cout << "\nDiem trung binh cua sinh vien " << i<< " la "
        << diem_tb << "\n\n";
    }
    return 0;
}

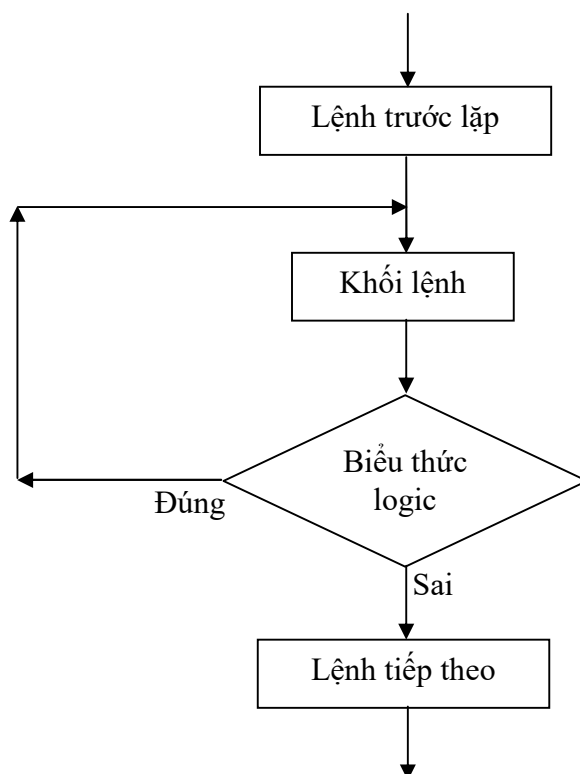
```

4.4 Câu lệnh do-while

Cấu trúc của câu lệnh do - while như sau:

```
do  
    <khối lệnh>  
while (biểu thức logic) ;
```

Trong đó do và while là từ khóa. Khi gặp câu lệnh trên <khối lệnh> sẽ được thực hiện chừng nào (biểu thức logic) còn có giá trị đúng. Sơ đồ khối minh họa cấu trúc do -while được mô tả trong hình 4-3.



Hình 4-3. Sơ đồ khối biểu diễn câu lệnh do-while

Ví dụ 4.12. Viết lại ví dụ 4.1 (while) và 4.7 (for) bằng câu lệnh do-while: Viết chương trình nhập một số nguyên n vào từ bàn phím. Yêu cầu hiển thị các giá trị nguyên từ 1 tới n.

Giải: Sử dụng câu lệnh do-while, chương trình như sau:

```
#include <iostream>  
using namespace std;  
int main()
```

```

{
    int i,n;
    cout<<"Nhap gia tri lon nhat can hien thi: "; cin>>n;
    i=1;
    do
    {
        cout<<i<<" ";
        i++;
    }
    while (i<=n);
    return 0;
}

```

Trong ví dụ 4.12, giả sử ta nhập $n = 5$. Ban đầu i được khởi gán giá trị bằng 1, tiếp theo khối lệnh của vòng lặp `do-while` thực hiện việc hiển thị giá trị của i , rồi tăng i lên 1 bằng 2 rồi mới kiểm tra xem i có nhỏ hơn 5 không. Vì i bằng 2 nhỏ hơn 5 nên i tiếp tục được hiển thị và tăng lên 3. Tương tự tới khi i tăng lên 6, điều kiện trở thành sai và vòng lặp dừng lại.

Ví dụ 4.13. Viết chương trình tính tổng sau, với n nhập từ bàn phím:

$$S = \left(\frac{1}{1+2}\right)^2 + \left(\frac{1+2}{2+3}\right)^2 + \dots + \left(\frac{(n-2)+(n-1)}{(n-1)+n}\right)^2$$

Chúng ta sẽ sử dụng câu lệnh `do - while` để viết chương trình sau.

Giải:

```

#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int n;
    float s,i;
    cout<<"nhap n ="; cin>>n;
    s=0; i=2;
    do
    {

```

```

        s= s + pow((2*i-3),2)/pow((2*i -1),2);
        i++;
    }
    while (i<=n);
    cout<<"Tong s = "<<s;
    return 0;
}

```

Sử dụng câu lệnh do-while để kiểm tra tính hợp lệ của dữ liệu

Lệnh do-while rất hữu ích khi dùng để kiểm tra dữ liệu do người dùng nhập vào có hợp lệ hay không. Ví dụ khi nhập vào số tháng của một năm thì tháng nằm trong khoảng 1 tới 12 là hợp lệ, ngoài khoảng này yêu cầu người dùng nhập lại.

Ví dụ 4.14a Viết chương trình nhập vào số tháng, số năm rồi hiển thị số ngày của năm đó. Yêu cầu kiểm tra tháng từ tháng 1 tới 12, năm từ năm 1900 tới 3000.

Giải:

```

#include<iostream>
using namespace std;
int main()
{
    int t, n, i;
    do
    {
        cout<<"nhap thang t ="; cin>>t;
        cout<<"nhap nam ="; cin>>n;
    }
    while ((t<1)|| (t>12)|| (n<1900)|| (n>3000));
    switch (t)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
            cout<<"Thang " << t << " nam " << n << " co 31 ngay";
            break;
        case 4:
        case 6:

```

```

case 9:
case 11:
    cout<<"Thang " << t << " nam " << n << " co 30 ngay";
    break;
case 2:
    if ((n % 4 == 0) && (n % 100 !=0)) || (n % 400 ==0))
        cout<<"Thang 2 nam " << n << " co 29 ngay";
    else
        cout<<"Thang 2 nam co 28 ngay";
    break;
}
return 0;
}

```

Trong ví dụ này, người sử dụng cần nhập đúng tháng nằm trong khoảng từ tháng 1 đến tháng 12 và năm nằm trong khoảng từ 1900 tới 3000 thì chương trình mới tính toán và hiển thị số ngày của tháng và năm đó. Trái lại, nếu tháng hoặc năm nhập không chính xác thì chương trình sẽ yêu cầu người sử dụng nhập lại cho đúng. Tuy nhiên chương trình trên có một nhược điểm là không báo cho người sử dụng chương trình có lỗi gì. Để khắc phục điều này chúng ta xem ví dụ sau:

Ví dụ 4.14b

Giải:

```

#include<iostream>
using namespace std;
int main()
{
    int t, n, i;
    do
    {
        cout<<"nhap thang t ="; cin>>t;
        cout<<"nhap nam ="; cin>>n;
        if((t<1)|| (t>12)|| (n<1900)|| (n>3000))
            cout<<"Du lieu sai, xin nhap lai!"<<endl;
        else
            break;
    }
    while (1);
    switch (t)
    {

```

```

case 1:
case 3:
case 5:
case 7:
case 8:
case 10:
case 12:
    cout<<"Thang " << t << " nam " << n << " co 31 ngay";
    break;
case 4:
case 6:
case 9:
case 11:
    cout<<"Thang " << t << " nam " << n << " co 30 ngay";
    break;
case 2:
    if ((n % 4 == 0) && (n % 100 !=0)) || (n % 400 ==0))
        cout<<"Thang 2 nam " << n << " co 29 ngay";
    else
        cout<<"Thang 2 nam co 28 ngay";
    break;
}
return 0;
}

```

Trong ví dụ 4.14b, chúng ta tạo ra một vòng lặp luôn luôn đúng. Trong vòng lặp này, trước tiên sử dụng lệnh `cin` để nhập dữ liệu vào. Sau đó sử dụng câu lệnh `if` để kiểm tra điều kiện, nếu điều kiện không đúng yêu cầu đề bài, yêu cầu nhập lại, trái lại ta dùng lệnh `break` để thoát khỏi vòng lặp.

4.5 Sự khác nhau giữa các câu lệnh lặp

Cấu trúc `while` và `for` thuộc dạng kiểm tra điều kiện trước rồi thực hiện khối lệnh còn cấu trúc `do-while` thuộc dạng thực hiện câu lệnh trước rồi mới kiểm tra điều kiện. Trong các ngôn ngữ lập trình khác `for` được dùng khi số lần lặp biết trước còn `do-while` và `while` được dùng khi số lần lặp không biết trước. Còn trong C++ cả ba vòng lặp này đều có thể thay thế được cho nhau. Việc sử dụng vòng lặp nào là tùy vào kinh nghiệm của lập trình viên. Tuy nhiên, thông thường các lập trình viên thường lựa chọn cách sử dụng phổ biến nhất. Tức là, khi số lần lặp biết trước ta thường sử dụng cấu trúc

for còn khi số lần lặp không biết trước ta thường sử dụng câu lệnh while hoặc do - while. Tiếp theo chúng ta sẽ tìm hiểu thêm một số ví dụ về các loại câu lệnh này.

Ví dụ 4.15 Viết chương trình nhập vào số nguyên dương n. Phân tích n thành tích các số nguyên tố rồi in kết quả lên màn hình.

Ví dụ: n = 6 thì kết quả là 2*3

n = 28 thì kết quả là 2*2*7

Giải: Để viết chương trình giải bài toán trên chúng ta có nhận xét sau: Nếu số a nào đó chia hết cho số b thì chúng cũng sẽ chia hết cho ước số của b. Vì vậy chúng ta sẽ tiến hành chia liên tiếp n cho 2 và tăng dần lên các số lẻ đến khi n còn lại 1 thì kết thúc quá trình lặp. Rõ ràng trong trường hợp này câu lệnh **while** là phù hợp nhất cho việc xây dựng vòng lặp để giải quyết bài toán. Chương trình như sau:

```
#include<iostream>
using namespace std;
int main()
{
    int i,n;
    cout<<"nhap n ="; cin>>n;
    cout<<"n se phan tich thanh tich cac so sau:";
    i=2;
    while (n != 1)
    {
        while (n % i ==0)
        {
            cout<<i<<" ";
            n = n /i;
        }
        if (i==2)
            i=3;
        else
            i=i+2;
    }
    return 0;
}
```

Ví dụ về kết quả thực hiện chương trình như sau:

```
nhap n =20
n se phan tich thanh tich cac so sau:2 2 5
-----
```

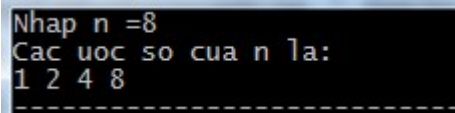

Ví dụ 4.16. Nhập vào số nguyên dương n . In lên màn hình các ước số của n . Yêu cầu cần có quá trình kiểm tra để đảm bảo n phải là số nguyên dương.

Giải:

```
#include<iostream>
using namespace std;
int main()
{
    int i,n;
    do
    {
        cout<<"Nhap n =";
        cin>>n;
        if(n<=0)
            cout<<"Nhap lai n!"<<endl;
        else break;
    }
    while (1);
    cout<<"Cac uoc so cua n la:"<<"\n";
    for (i=1; i<=n; i++)
        if (n % i ==0)
            cout<<i<<" ";
    return 0;
}
```

Trong chương trình trên, `while(1)` là điều kiện luôn luôn đúng, do đó vòng lặp sẽ dừng lại khi $n > 0$, điều kiện `if` sẽ sai và câu lệnh `break` được thực hiện.

Ví dụ về kết quả thực hiện chương trình như sau:



```
Nhap n =8
Cac uoc so cua n la:
1 2 4 8
```

Ví dụ 4.17. Nhập vào số nguyên dương n . Kiểm tra xem n có phải là số hoàn thiện hay không. Một số nguyên n được gọi là số hoàn thiện nếu nó **bằng** tổng các ước của nó không kể chính nó. Ví dụ: $6 = 1 + 2 + 3$.

Giải: Chúng ta sẽ đi tìm tất cả các ước số của n và tính tổng lại, nếu tổng đó bằng n thì kết luận đây là số hoàn thiện và in thông báo lên màn hình. Chương trình như sau:

```
#include<iostream>
using namespace std;
```

```

int main()
{
    int i,n, dem;
    do
    {
        cout<<"Nhap n =";
        cin>>n;
        if(n<=0) cout<<"Nhap lai n!"<<endl;
        else break;
    }
    while (1);
    dem=0;
    for (i=1; i<n; i++)
        if (n % i ==0)
            dem = dem +i;
    if (dem ==n)
        cout<<"So vua nhap la so hoan thien ";
    else
        cout<<"So vua nhap khong la so hoan thien ";
    return 0;
}

```

Ví dụ 4.18. Nhập vào số nguyên dương n . In lên màn hình các số nguyên tố không lớn hơn n . Số nguyên tố là số chỉ có hai ước duy nhất là một và chính nó. Ví dụ về một vài số nguyên tố là: 2, 3, 5, 7, 11.

Giải: Chúng ta sẽ kiểm tra lần lượt từ 2 đến n , gặp số nguyên tố nào chúng ta sẽ in lên màn hình số đó. Để kiểm tra một số có phải là số nguyên tố hay không chúng ta chỉ việc xem nó có chia hết cho số nào ngoài một và chính nó hay không?

```

#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int i,n, dem;
    do
    {
        cout<<"Nhap n =";
        cin>>n;
    }
    while (n<=0);
}

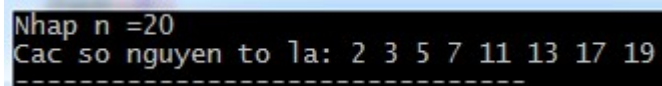
```

```

    dem=0;
    cout<<"Cac so nguyen to la: ";
    for (i=2; i<=n; i++)
    {
        if (i<=3)
            cout<<i<<" ";
        else
        {
            dem =0;
            for (int j=2; j<=sqrt(i); j++)
                if (i % j ==0)
                    {dem = dem +1; break;}
            if (dem ==0)
                cout<<i<<" ";
        }
    }
    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Nhap n =20
Cac so nguyen to la: 2 3 5 7 11 13 17 19

```

Ví dụ 4.19: Viết chương trình in lên màn hình số nguyên tố lớn nhất có 3 chữ số.

Giải: Số nguyên tố lớn nhất sẽ nằm trong đoạn từ 100 đến 999; ta sẽ duyệt lần lượt các số từ 999 ngược lại gặp số nào là số nguyên tố đó chính là số lớn nhất cần tìm. Chương trình như sau:

```

#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    for (int i=999; i>100; i--)
    {
        int dem=0;
        for (int j=2; j<=sqrt(i); j++)
            if (i % j==0)
                dem=dem+1;
        if (dem==0)
        {
            cout<<"So nguyen to lon nhat co 3 chu so la: "<<i;

```

```

        break;
    }
}
return 0;
}

```

Ví dụ 4.20. Viết chương trình in lên màn hình số chính phương lớn nhất có 4 chữ số.

Giải: Số chính phương lớn nhất có 4 chữ số sẽ nằm trong đoạn từ 1000 đến 9999. Như vậy chúng ta sử dụng vòng lặp đi từ 9999 ngược lại gặp số nào là chính phương đó chính là số lớn nhất. Chương trình như sau:

```

#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    for (int i=9999; i>=1000; i--)
    {
        if (sqrt(i) == int(sqrt(i)))
        {
            cout<<"Số chính phương lớn nhất có 4 chữ số: "<<i;
            break;
        }
    }
    return 0;
}

```

Số chính phương là số có căn bậc hai của nó là số nguyên, vì vậy ngoài cách kiểm tra số chính phương như đã trình bày trong chương 3 chúng ta còn có thể dùng cú pháp ép kiểu dữ liệu về kiểu nguyên như trong ví dụ này.

Ví dụ 4.21. Vừa gà vừa chó, bó lại cho tròn có ba mươi sáu con và một trăm chân chẵn. Hỏi có bao nhiêu con gà, bao nhiêu con chó? Viết chương trình để hiển thị kết quả của bài toán này.

Giải: Gọi i là số gà thì $36 - i$ sẽ là số chó. Điều kiện nghiệm sẽ là $2*i + (36 - i)*4 = 100$.

Với bài này chúng ta không cần sử dụng câu lệnh nhập. Chương trình như sau:

```

#include<iostream>
using namespace std;
int main()
{

```

```

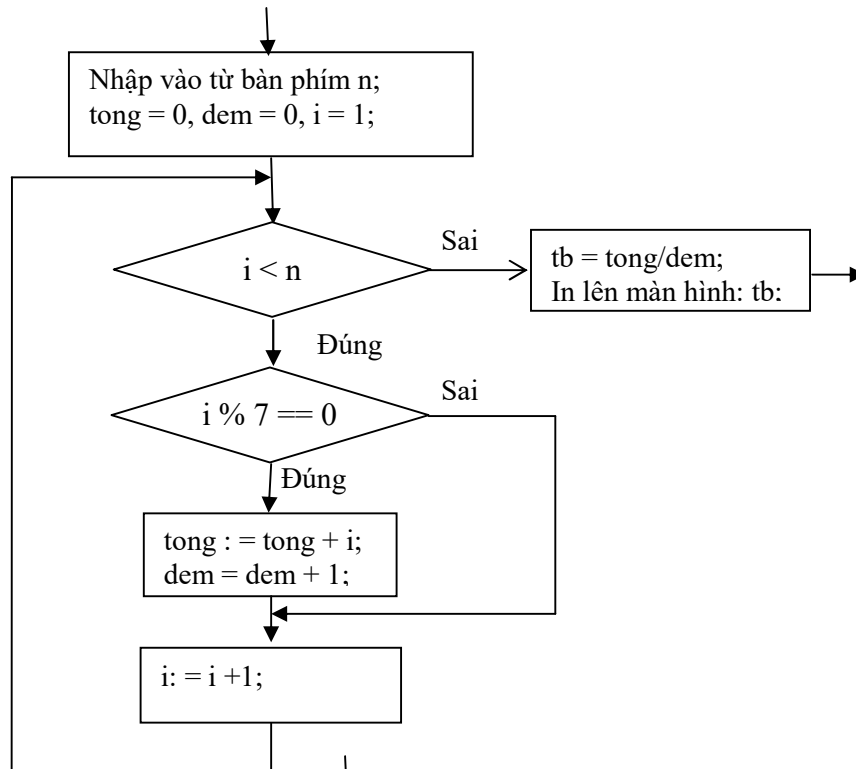
cout << "Ket qua bai toan vua ga vua cho: \n";
for (int i = 9; i < 25; i++)
    if ((i * 2 + (36 - i) * 4) == 100)
    {
        cout << "So con ga la: " << i << endl;
        cout << "So con cho la: " << (36 - i) ;
    }
return 0;
}

```

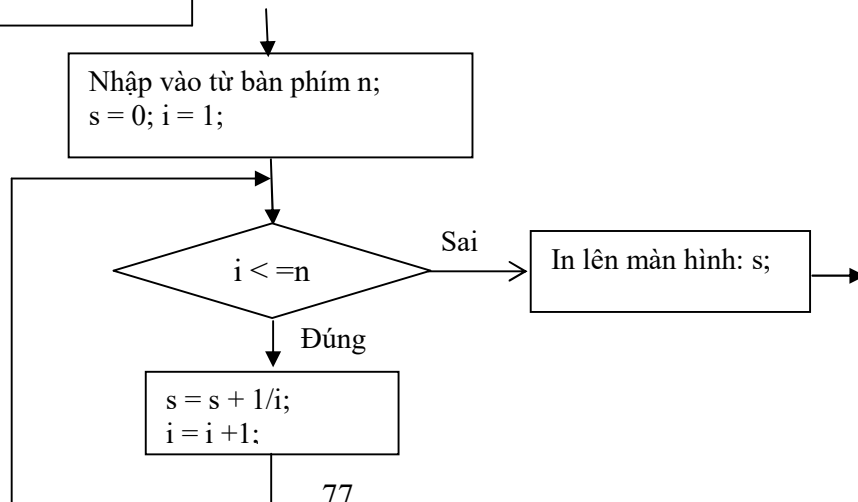
Bài tập chương 4

Bài 1. Viết đoạn lệnh tương ứng với các sơ đồ khối sau:

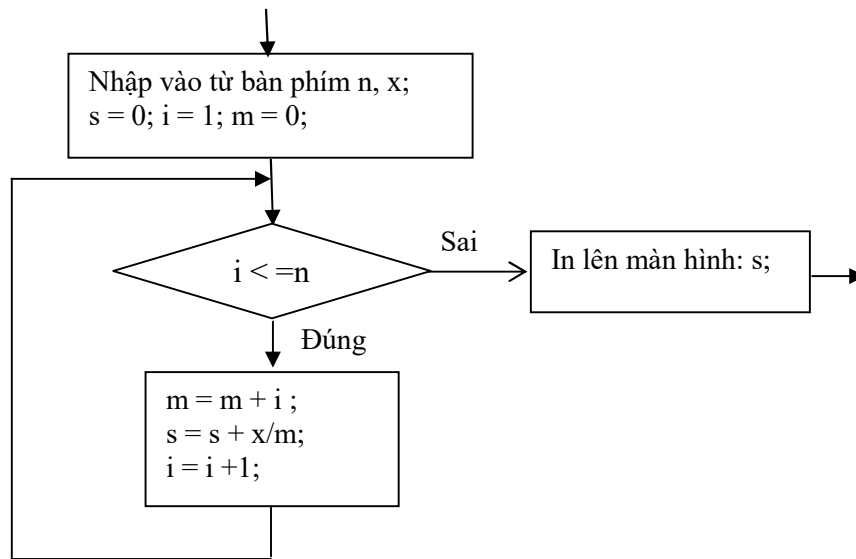
a.



b.



c.



Bài 2. Viết chương trình tính tổng sau với n nguyên dương nhập vào từ bàn phím

$$S = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \dots + \frac{1}{1+2+\dots+n}$$

Bài 3. Viết chương trình tính tổng sau với n nguyên dương nhập vào từ bàn phím

$$S = 1 + \frac{1}{1^2+2^2} + \frac{1}{1^2+2^2+3^2} + \dots + \frac{1}{1^2+2^2+\dots+n^2}$$

Bài 4. Viết chương trình tính tổng sau với n nguyên dương nhập vào từ bàn phím

$$S = \frac{1}{1^3+2^3} + \frac{1}{1^3+2^3+3^3} + \dots + \frac{1}{1^3+2^3+\dots+n^3}$$

Bài 5. Viết chương trình tính tổng sau với n nguyên dương nhập vào từ bàn phím

$$S = 1 - \frac{1}{1^2+2^2} + \frac{1}{1^2+2^2+3^2} - \dots + \frac{(-1)^{n+1}}{1^2+2^2+3^2+\dots+n^2}$$

Bài 6. Viết chương trình tính tổng sau với n nguyên dương nhập vào từ bàn phím.

$$S = 1 - \frac{2}{1!} + \frac{2^2}{2!} - \frac{2^3}{3!} + \dots + \frac{(-2)^n}{n!}$$

Bài 7. Viết chương trình tính tổng sau với n nguyên dương nhập vào từ bàn phím

$$S = x - (1+x)^{1/2} + (1+x^2)^{1/3} - \dots (-1)^{n+1} (1+x^{n-1})^{1/n}$$

Bài 8. Viết chương trình tính tổng sau với n nguyên dương nhập vào từ bàn phím

$$S = \frac{1}{2} - \frac{1}{4} + \frac{1}{6} - \frac{1}{8} + \dots + (-1)^{n+1} \frac{1}{2n}$$

Bài 9. Viết chương trình tính tổng sau với n nguyên dương nhập vào từ bàn phím

$$S = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots + (-1)^{n+1} \frac{1}{2n-1}$$

Bài 10. Viết chương trình tính tổng sau với n nguyên dương nhập vào từ bàn phím

$$S = 1 + \frac{2}{1!} + \frac{2^2}{2!} + \frac{2^3}{3!} + \dots + \frac{2^n}{n!}$$

Bài 11. Viết chương trình tính tổng sau với n nguyên dương nhập vào từ bàn phím

$$S = \frac{1}{1!} - \frac{1}{2!} + \frac{1}{3!} - \frac{1}{4!} + \dots + (-1)^{n+1} \frac{1}{n!}$$

Bài 12. Số thứ n của dãy số Fibonacci được xác định theo công thức $F_n = F_{n-1} + F_{n-2}$ với $F_0 = 1, F_1 = 1$. Viết chương trình tính số Fibonacci thứ n của dãy với n nhập từ bàn phím.

Bài 13. Viết chương trình nhập vào hai số nguyên dương a và b. Tính và in lên màn hình ước số chung lớn nhất của hai số đó.

Bài 14. Viết chương trình nhập vào hai số nguyên dương a và b. Tính và in lên màn hình bội số chung lớn nhất của hai số đó.

Bài 15. Viết chương trình nhập vào số nguyên dương n, hãy phân tích n thành tích của các số nguyên tố, hiển thị tích đó.

Bài 16. Viết chương trình nhập vào số bắt đầu, công sai. Tính và in lên màn hình số thứ n của cấp số cộng trên.

Bài 17. Viết chương trình nhập vào số bắt đầu và công bội. Tính và in lên màn hình số thứ n của cấp số nhân trên.

Bài 19. Viết chương trình in lên màn hình n số nguyên tố đầu tiên.

Bài 20. Viết chương trình in lên màn hình các số hoàn thiện nhỏ hơn một số nguyên n cho trước. Một số được gọi là hoàn thiện nếu giá trị của tổng các ước số của nó (không kể chính nó) bằng chính số đó.

Chương 5. Hàm trong C++

5.1. Giới thiệu

Trong các chương trước chúng ta đã tìm hiểu việc viết các chương trình trong đó chỉ có một hàm `main()`, các câu lệnh sẽ được viết ngay trong hàm `main()` đó. Trong chương này chúng ta sẽ học cách thiết kế và xây dựng chương trình bằng phương pháp sử dụng các hàm. Việc sử dụng hàm có các ưu điểm sau đây:

- Phù hợp với quy trình thiết kế bài toán theo kiểu mô module hóa (top down) nghĩa là để giải quyết một bài toán lớn, chúng ta chia bài toán lớn thành các bài toán nhỏ hơn để giải quyết; mỗi bài toán con sẽ do một vài hàm xử lý.
- Việc sử dụng hàm làm cho chương trình dễ viết, dễ sửa lỗi, dễ thực hiện, cũng như trong sáng và dễ hiểu; một hàm được viết có thể sử dụng nhiều lần và ở các nơi khác nhau.
- Cho phép thiết kế chương trình giải quyết các bài toán có tính chất đệ quy, tức là sử dụng các hàm trong đó có lời gọi đến chính nó. Thực tế có nhiều bài toán đến nay chỉ có phương pháp giải bằng đệ quy là hữu dụng nhất.

Để hiểu ý nghĩa cũng như sự cần thiết của việc sử dụng hàm chúng ta sẽ xem vị trí của hàm trong cấu trúc chương trình C++ như sau:

Phần khai báo:

- Khai báo các tệp *header*
- Khai các hằng, biến, kiểu dữ liệu,...
- *Khai báo hàm 1*
- *Khai báo hàm 2*
- ...
- *Khai báo hàm m*

Phần thân chương:

```
int main()  
{  
    Câu lệnh 1;
```


Câu lệnh 2;

....

return 0;

}

Như vậy, muốn sử dụng hàm chúng ta phải khai báo, nói cách khác phải xây dựng chúng để giải quyết các công việc cụ thể.

5.2 Khai báo và cách sử dụng hàm

Mỗi hàm được định nghĩa một lần trong một chương trình và sau đó có thể được sử dụng bởi bất kỳ hàm nào khác trong chương trình mà có khai báo phù hợp. Giống như hàm `main()`, mọi hàm C++ bao gồm hai phần: tiêu đề hàm (function header) và thân hàm (function body), như minh họa trong hình 5-1. Mục đích của tiêu đề hàm là để xác định kiểu dữ liệu của giá trị mà hàm trả về, đặt tên hàm, và xác định số lượng, thứ tự, các tham số của hàm. Trong đó tham số của hàm có thể là tham biến trị, tham số biến hoặc con trỏ; chúng ta sẽ tìm hiểu các chủ đề này ở các phần tiếp theo. Mục đích của thân hàm là để thao tác trên dữ liệu của chương trình và trả về các giá trị nếu có yêu cầu. Tiêu đề hàm luôn luôn nằm ở dòng đầu tiên của hàm, chứa kiểu giá trị trả về, tên hàm và tên, kiểu dữ liệu của các tham số. Phần tên của hàm được quy ước đặt giống như tên biến.

<kiểu trả về của hàm> tên_hàm(danh sách tham số)

{

- Các khai báo biến, hằng,...

- Các câu lệnh

- [return <biểu thức>]

}

Hình 5-1 Cấu trúc chung của một hàm

Ví dụ 5.1a. Xây dựng hàm tính tổng của ba số a, b, c kiểu nguyên.

```
int tong(int a, int b, int c)
```

```
{
```

```
    int tong;
```

```
    tong = a + b + c;
```

```
    return tong;
}
```

Hàm trên có tên là `tong`, mục đích là nhận ba tham số nguyên `a`, `b` và `c`, tính tổng rồi trả về giá trị kết quả cho hàm qua câu lệnh `return`. Như vậy kiểu trả về cho hàm là `int`, tham số truyền vào cho hàm là `a`, `b` và `c`.

Để sử dụng (gọi hàm) hàm trong khi viết chương trình chúng ta cần quan tâm đến *<kiểu trả về của hàm>* và *danh sách tham số* cần sử dụng cho hàm đó. Sau đây chúng ta sẽ tìm hiểu một số trường hợp khác của việc khai báo hàm.

Hàm không trả lại giá trị:

Trong một số trường hợp chúng ta xây dựng hàm mà không cần trả ra bất kỳ giá trị gì sau câu lệnh `return` – đơn giản trong thân hàm chỉ có các nhiệm vụ tính toán, thao tác mà không phải trả ra giá trị. Với hàm này, có thể khai báo tiêu đề hàm bằng cách sau:

`void tên_hàm` (danh sách tham số)

Chú ý rằng trong trường hợp này không cần sử dụng câu lệnh `return` trong thân của hàm.

Ví dụ 5.1b Xây dựng hàm tính tổng của ba số `a`, `b`, `c` kiểu nguyên, hiển thị giá trị tổng trong hàm.

Giải:

```
void tong(int a, int b, int c)
{
    int tong;
    tong = a + b + c;
    cout<<"Tong = "<<tong;
}
```

Để sử dụng hàm chúng ta chỉ cần ghi tên hàm kèm theo các tham số thực sự của nó. Cách gọi hàm này cũng tương tự như cách gọi các hàm toán học đã xây dựng sẵn trong C++ như `sqrt()`, `pow()`, hay `abs()`.

Hàm không chứa tham số:

Chúng ta có thể xây dựng hàm mà không cần có tham số truyền vào. Với hàm này có thể khai báo tiêu đề hàm bằng một trong hai cách sau:

- Cách 1: *<kiểu trả về của hàm>* `tên_hàm()`
- Cách 2: *<kiểu trả về của hàm>* `tên_hàm(void)`

<kiểu trả về của hàm> có thể là một kiểu dữ liệu hoặc `void`.

Ví dụ 5.2. Xây dựng hàm `nhapdl()` để nhập dữ liệu cho các tham số trong chương trình của ví dụ 5.1.

Giải:

```
void nhapdl()
{
    cout<<"x = "; cin>>x;
    cout<<"y = "; cin>>y;
    cout<<"z = "; cin>>z;
}
```

Từ ví dụ 5.1a và ví dụ 5.2 ta có thể xây dựng một chương trình hoàn chỉnh sử dụng hai hàm `nhapdl()` và hàm `tong()` như sau:

Ví dụ 5.3 Viết chương trình xây dựng hàm nhập dữ liệu để nhập vào ba số `a`, `b`, `c` vào từ bàn phím và hàm tính tổng có ba tham số đầu vào là `a`, `b` và `c`. Gọi hai hàm trên trong hàm `main()` để tính và hiển thị tổng của ba số `a`, `b`, `c`.

Giải:

```
#include<iostream>
using namespace std;
int x, y, z;
void nhapdl()
{
    cout<<"x = "; cin>>x;
    cout<<"y = "; cin>>y;
    cout<<"z = "; cin>>z;
}
int tong(int a, int b, int c)
{
    int tong;
    tong = a + b + c;
    return tong;
}
int main()
{
    int sum;
    nhapdl();
    sum=tong(x,y,z);
    cout<<"Tong cua ba so la: "<<sum;
```

```

    return 0;
}

```

Hàm có đối số mặc định:

Để tăng độ linh hoạt cho hàm, C++ cung cấp phương thức sử dụng hàm với đối số mặc định. Các giá trị của đối số mặc định được liệt kê trong phần tiêu đề hàm, được truyền tự động khi có lời gọi hàm mà giá trị đối số truyền vào bị bỏ qua.

Chương trình 5.3 được viết lại với hàm sử dụng tham số mặc định như sau:

Ví dụ 5.4

```

#include<iostream>
using namespace std;
int x, y, z;
void nhapdl()
{
    cout<<"x = "; cin>>x;
    cout<<"y = "; cin>>y;
    cout<<"z = "; cin>>z;
}

int Tong(int a, int b=5, int c=2)
{
    int tong;
    tong = a + b + c;
    return tong;
}

int main()
{
    nhapdl();
    int tong1, tong2, tong3;
    tong1= Tong(x,y,z);
    tong2= Tong(x,y);
    tong3= Tong(x);
    cout<<"Tong 1= "<<tong1<<endl;
    cout<<"Tong 2= "<<tong2<<endl;
    cout<<"Tong 3= "<<tong3<<endl;
    return 0;
}

```

Kết quả của chương trình 5.4 như sau:

```
x = 3
y = 4
z = 5
Tong 1= 12
Tong 2= 9
Tong 3= 10
-----
```

Bốn quy tắc khi sử dụng tham số mặc định cần tuân theo là:

- Các giá trị mặc định được đặt trong tiêu đề hàm
- Nếu bất kỳ tham số nào được nhận tham số mặc định thì các tham số sau nó cũng cần phải có tham số mặc định.
- Khi gọi hàm nếu một đối số được bỏ qua thì tất cả các đối số bên phải nó cũng phải được bỏ qua. Quy tắc thứ 2 và thứ 3 giúp trình biên dịch C++ cung cấp chính xác các giá trị mặc định cho các đối số bị thiếu
- Giá trị mặc định có thể là một biểu thức bao gồm cả hằng và biến đã được khai báo.

Trong thực tế tùy vào mục đích của từng bài toán người xây dựng chương trình sẽ phải xác định chọn kiểu hàm cho phù hợp.

Sử dụng tham biến trị trong hàm:

Trong các ví dụ trên, các tham số truyền vào cho hàm được gọi là các tham biến trị. Trong khi xây dựng hàm chúng ta sử dụng các tham số này vào các công việc cụ thể, khi sử dụng hàm chúng ta phải truyền các giá trị cụ thể tương ứng với các tham số trong hàm. Sau khi hàm được gọi thì các tham biến trị không thay đổi giá trị. Như vậy khi xây dựng hàm có tham số trị chúng ta phải lưu ý:

- Xác định số lượng tham số trị là bao nhiêu, kiểu của từng loại tham số
- Các tham số trị khi truyền cho hàm thì phải được xác định giá trị cụ thể

Ví dụ 5.4. Xây dựng hàm tính diện tích của hình chữ nhật. Gọi hàm này trong hàm `main()` rồi hiển thị giá trị diện tích vừa tính được với chiều dài, chiều rộng hình chữ nhật được nhập vào từ bàn phím.

Giải:

```
#include <iostream>
using namespace std;
```

```

float s_cn(float chieu_dai, float chieu_rong)
{
    return chieu_dai*chieu_rong;
}

int main ()
{
    float a, b, s;
    cout<<"Chương trình tính diện tích hình chu nhật: \n";
    cout<<"Nhập chiều dài a = "; cin>>a;
    cout<<"Nhập chiều rộng b = "; cin>>b;
    s=s_cn(a,b);
    cout<<"Diện tích hình chu nhật = "<<s<<"\n";
    return 0;
}

```

Trong chương trình trên, chúng ta sử dụng lời gọi hàm `s_cn(a,b)` với `a, b` được gọi là tham số thực sự. Giá trị của `a` được truyền cho tham số hình thức `chieu_dai`, giá trị của `b` được truyền cho tham số hình thức `chieu_rong`, hai tham số này đều đã xác định sau khi được nhập từ bàn phím. Dù trong hàm `s_cn(a,b)` có thực hiện lệnh gì thì nó cũng không làm thay đổi được giá trị của biến `a` và `b`. Vì vậy nó được gọi là tham số trị.

Lời gọi hàm `s_cn(a,b)` cũng tương tự như việc sử dụng các hàm đã có sẵn của C++ như `sqrt()`, `pow()`.

Ví dụ 5.5. Viết chương trình kiểm tra một số `x` nguyên dương có phải là số nguyên tố hay không? Yêu cầu viết hàm kiểm tra số nguyên tố.

Giải: Chúng ta thấy phần nhập dữ liệu chỉ cần nhập `x` nên có thể để ngay trong hàm `main()`. Việc kiểm tra xem `x` có phải là số nguyên tố hay không (thuật toán đã có ở chương trước) chúng ta sẽ sử dụng hàm. Hàm này sẽ nhận tham biến trị vào và trả ra giá trị kiểu nguyên theo quy ước: nếu trả ra giá trị 0 thì `x` không phải là số nguyên tố, nếu trả ra giá trị 1 thì `x` là nguyên tố. Chương trình như sau:

```

#include<iostream>

```

```

#include<cmath>
using namespace std;
int kt_nguyen_to(int a)
{
    int i;
    if (a <=1)
        return 0;
    for (i=2; i<=sqrt(a); i++)
        if (a % i ==0)
        {
            return 0;
            break;
        }
    return 1;
}
int main()
{
    int x, y;
    cout<<"x = "; cin>>x;
    if (kt_nguyen_to(x) ==1)
        cout<<"x la so nguyen to ";
    else
        cout<<"x khong la so nguyen to ";
    return 0;
}

```

Ví dụ 5.6. Viết chương trình nhập vào hai số nguyên m và n. Tính và in lên màn hình ước số chung lớn nhất của hai số đó.

Giải: Chúng ta sẽ xây dựng hàm tìm ước số chung lớn nhất nhận hai tham số trị a và b và trả ra giá trị là ước số chung lớn nhất của hai số đó. Thuật toán tìm ước số chung lớn nhất của hai số nguyên được thực hiện theo phương pháp Ôcôlit. Chương trình như sau:

```

#include<iostream>
using namespace std;
int ucln(int a, int b)
{
    int r,x,y;
    x=a;
    y=b;
    r= x % y;
    while (r!=0)

```

```

    {
        x = y;
        y=r;
        r = x % y;
    }
    return y;
}
int main()
{
    int m, n;
    cout<<"m = "; cin>>m;
    cout<<"n = "; cin>>n;
    cout<<"Uoc chung lon nhat cua m va n la: "<<ucln(m,n);
    return 0;
}

```

Hàm trả về nhiều giá trị sử dụng biến tham chiếu:

Trong tất cả các ví dụ từ đầu chương tới giờ, chúng ta đều sử dụng cách truyền tham số theo tham trị. Truyền tham số theo cách này sẽ không làm thay đổi giá trị của biến trước và sau khi sử dụng nó truyền cho hàm. Vấn đề đặt ra là làm sao chúng ta có thể tạo ra đầu ra của hàm với nhiều giá trị trả về? Để giải quyết vấn đề này C++ cung cấp một loại biến mới gọi là *biến tham chiếu*. Hiểu một cách đơn giản, khi truyền tham số cho hàm bằng biến tham chiếu chúng ta có thể thay đổi giá trị các biến đó sau khi hàm đó được gọi, điều này là khác với cách sử dụng tham biến trị như đã nói ở trên. Cách khai báo biến tham chiếu như sau:

<kiểu dữ liệu> tên_biến_tham_chiếu

trong đó sử dụng kí hiệu & báo cho chương trình biết đây là biến tham chiếu. Chúng ta sẽ sử dụng biến tham chiếu trong trường hợp cần xây dựng các hàm trả về nhiều giá trị.

Ví dụ 5.7. Xây dựng hàm sử dụng biến tham chiếu để tính tổng và tích của ba số nguyên.

Giải:

```

void tong_tich(float a, float b, float c, float& tong, float&
tich)
{
    tong = a + b +c;
    tich = a * b * c;
}

```


Trong hàm `tong_tich()` trên chứa 5 tham số trong đó có 3 tham biến trị và 2 biến tham chiếu. Sau khi gọi hàm trên giá trị tổng và tích sẽ được trả ra hai biến tương ứng. Sử dụng hàm `tong_tich()` ở trên chúng ta viết thành chương trình hoàn chỉnh như sau:

```
#include<iostream>
using namespace std;
float x, y, z;
void nhap_dl()
{
    cout<<"x= "; cin>>x;
    cout<<"y= "; cin>>y;
    cout<<"z= "; cin>>z;
}
void tong_tich(float a, float b, float c, float& tong, float&
tich)
{
    tong = a + b + c;
    tich = a * b * c;
}
int main()
{
    float tg1, tg2;
    nhap_dl();
    tong_tich(x, y, z, tg1, tg2);
    cout<<"\n Tong ba so la: "<<tg1<<endl;
    cout<<"\n Tich ba so la: "<<tg2<<endl;
    return 0;
}
```

Sau khi gọi hàm `tong_tich()` với ba tham biến trị và hai biến tham chiếu chúng ta sẽ có giá trị của **tổng** và tích trong **tg1** và **tg2**.

Ví dụ 5.8 Viết chương trình nhập hai số nguyên x, y. Sử dụng biến tham chiếu xây dựng hàm và viết chương trình đổi chỗ giá trị của x và y cho nhau.

Giải:

```
#include<iostream>
using namespace std;
int doi_cho(int& a, int& b)
{
```

```

    int c;
    c=a;
    a=b;
    b=c;
}
int main()
{
    int x,y;
    cout<<"x= "; cin>>x;
    cout<<"y= "; cin>>y;
    cout<<"Gia tri x truoc khi hoan doi = "<<x<<endl;
    cout<<"Gia tri y truoc khi hoan doi = "<<y<<endl;
    doi_cho(x,y);
    cout<<"Gia tri x sau khi hoan doi = "<<x<<endl;
    cout<<"Gia tri y sau khi hoan doi = "<<y<<endl;
    return 0;
}

```

Chú ý: Trong quá trình viết chương trình có sử dụng hàm, chúng ta thường viết theo thứ tự các hàm thành phần trước rồi mới đến hàm `main()`. Chúng ta cũng có thể viết hàm `main()` trước khi viết các hàm con tuy nhiên chúng ta cần phải khai báo các nguyên mẫu hàm thành phần trước hàm `main()`. Các nguyên mẫu hàm chính là dòng tiêu đề của hàm nhưng chúng ta không cần viết các tham số vào đó mà chỉ cần báo cho chương trình biết có các biến kiểu gì và số lượng bao nhiêu. Chẳng hạn hàm `int tinh tong(int a, int b, int c)` thì nguyên mẫu hàm sẽ là `int tinh tong(int, int, int);` hàm `void tong_tich(int a, int b, int& tong, int& tich)` thì nguyên mẫu hàm sẽ là `void tong_tich(int, int, int&, int&);`.

Nạp chồng hàm:

C++ cung cấp khả năng tạo ra nhiều hàm có cùng tên trong khi viết chương trình. Trình biên dịch sẽ quyết định sử dụng hàm nào dựa vào kiểu dữ liệu và số lượng tham số của các biến được truyền vào trong lời gọi hàm.

Ví dụ 5.9 Viết chương trình gồm hai hàm `tong()`, hàm thứ nhất để cộng ba giá trị kiểu nguyên, hàm thứ hai để cộng ba giá trị kiểu thực. Viết hàm `nhapdl()` để nhập vào ba số kiểu nguyên, ba số kiểu thực. Gọi ba hàm trên trong hàm `main()` để kiểm nghiệm kết quả hai hàm `tong()`.

Giải:

```
#include<iostream>
using namespace std;
int x, y, z;
float n, m, p;
void nhapdl()
{
    cout<<"x = "; cin>>x;
    cout<<"y = "; cin>>y;
    cout<<"z = "; cin>>z;
    cout<<"m = "; cin>>m;
    cout<<"n = "; cin>>n;
    cout<<"p = "; cin>>p;
}
int tong(int a, int b, int c)
{
    int tong;
    tong = a + b + c;
    return tong;
}
float tong(float a, float b, float c)
{
    float tong;
    tong = a + b + c;
    return tong;
}
int main()
{
    thapdl();
    cout<<"Tong nguyen = " <<tong(x,y,z)<<endl;
    cout<<"Tong thuc = " <<tong(m,n,p)<<endl;
    return 0;
}
```

Trong chương trình này, lời gọi hàm thứ nhất `tong(x,y,z)`; đã sử dụng hàm cộng ba giá trị kiểu nguyên vì các tham số `x`, `y`, và `z` có kiểu nguyên, còn lời gọi hàm `tong(m,n,p)`; sử dụng hàm cộng ba giá trị kiểu thực vì các tham số `m`, `n`, `p` kiểu thực.

Phạm vi hoạt động của biến:

Đến đây, chúng ta tìm hiểu sâu hơn về phạm vi hoạt động của biến trong khi viết chương trình. C++ cung cấp khả năng khai báo mềm dẻo trong việc sử dụng biến. Nói

chung để sử dụng một biến nào đó thì điều kiện tiên quyết là nó phải được khai báo, một biến sử dụng trong toàn chương trình được gọi là biến toàn cục, nếu chỉ sử dụng trong một đoạn chương trình nào đó thì gọi là biến địa phương. Việc sử dụng biến tuân theo quy tắc sau:

- Biến khai báo trong khối lệnh nào thì chỉ khối lệnh đó được phép sử dụng
- Biến khai báo trong hàm nào thì hàm đó được quyền sử dụng và chỉ tính từ lúc nào được khai báo
- Biến khai bên ngoài các hàm thì phạm vi hoạt động của nó là từ các hàm sau khi nó được khai báo trở đi sẽ được sử dụng nó.

Ví dụ hàm `tong()` trong ví dụ 5.1 ở trên

```
int tong(int a, int b, int c)
{
    int tong;
    tong = a + b + c;
    return tong;
}
```

Trong thân hàm `tong()` có lệnh khai báo biến `tong` kiểu nguyên, vậy `tong` chính là một biến cục bộ. Biến `tong` được khai báo trong hàm `tong()` và chỉ được sử dụng trong phạm vi hàm `tong()`. Sau khi lời gọi hàm kết thúc, biến `tong` sẽ bị hủy đi, không được sử dụng nữa.

Ví dụ 5.10 Xây dựng hàm trả về trị tuyệt đối của một số

Giải:

```
#include<iostream>
using namespace std;
float tri_tuyet_doi(float a)
{
    float c;
    if(a>0)
        c=a;
    else
        c=-a;
    return c;
}
int main()
```

```

{
    float x,z;
    cout<<"x= "    ; cin>>x;
    z=tri_tuyet_doi(x);
    cout<<"tri tuyet doi cua "<<x<<" la: "<<z;
}

```

Trong ví dụ 5.10, hàm `tri_tuyet_doi()` sử dụng biến `c` là biến cục bộ của hàm, hàm `main()` có hai biến cục bộ là `x` và `z`. Hàm `tri_tuyet_doi()` không được phép sử dụng biến `x`, `z` của hàm `main()` và hàm `main()` cũng không được phép sử dụng biến `c` của hàm `tri_tuyet_doi()`.

Ví dụ 5.11 Minh họa về phạm vi hoạt động của biến

```

#include <iostream>
using namespace std;
int a;
void ham_pham_vi()
{
    int b;
    b = 20;
    cout<<"Gia tri cac bien goi tai ham_pham_vi()   la: "<<endl;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    a = 40;
    return;
}
int main()
{
    int b;
    a = 5;
    b = 10;
    cout<<"Gia tri cac bien goi tai ham main()"<<endl;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    ham_pham_vi();
    cout<<"Sau khi goi ham_pham_vi(), gia tri cac bien la:
"<<endl;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    return 0;
}

```

Với biến toàn cục a, lệnh hiển thị giá trị lần đầu tiên và lần thứ hai sẽ hiển thị giá trị được lưu trong biến toàn cục a. Do trong hàm `ham_pham_vi()` có lệnh thay đổi biến toàn cục nên trong lệnh hiển thị giá trị a lần thứ ba nó bị thay đổi thành 40. Với biến cục bộ b, lần hiển thị đầu tiên sẽ hiển thị giá trị được lưu trong hàm `main()`, lần hiển thị thứ 2 sẽ hiển thị giá trị lưu trong hàm `ham_pham_vi()`, lần hiển thị thứ ba lại được gọi trong hàm `main()` nên nó vẫn hiển thị giá trị lưu trong hàm `main()` là `b=10`.

5.3 Hàm đệ quy

Trong các phần trên chúng ta đã thấy có thể sử dụng hàm trong hàm khác hoặc trong hàm chính (hàm `main()`). Tuy nhiên trong C++ còn cho phép trong thân của hàm có lời gọi đến chính hàm đó. Những hàm như thế chúng ta gọi là hàm đệ quy. Hiểu một cách đơn giản quá trình gọi hàm đến chính nó sẽ dẫn đến hàm đang viết được lặp đi lặp lại một số lần tùy theo điều kiện cụ thể. Chúng ta sẽ nghiên cứu các hàm này qua một số ví dụ sau đây.

Ví dụ 5.12. Viết chương trình tính $S_n = 1 + 1/2 + 1/3 + \dots + 1/n$, n nhập từ bàn phím.

Giải: Để tính tổng S_n chúng ta sẽ đi tính $S_{n-1}, S_{n-2}, \dots, S_1$, ta có công thức đệ quy như sau: $S_n = 1/n + S_{n-1}$. Chương trình như sau:

```
#include<iostream>
#include<cmath>
using namespace std;
double n;
void nhap()
{
    int i=1;
    cout<<"n = "; cin>>n;
}
double tinh(double n)
{
    if (n==1)
        return 1;
    else
        return (1/n) + tinh(n-1);
}
int main()
{
```

```

    nhap();
    cout<<"Ket qua: "<< tinh(n);
    return 0;
}

```

Nếu chúng ta muốn tính tổng chẳng hạn $S = 1 + 1/2^2 + 1/3^2 + \dots + 1/n^2$ thì chúng ta chỉ cần thay $1/n$ bằng $1/(n*n)$ trong hàm `tinh()` ở trên. Việc xây dựng hàm cho các bài toán cùng loại cũng được thực hiện tương tự.

Ví dụ 5.13. Viết chương trình tính $S = 1 + 1/(1^2+2^2) + 1/(2^3 + 3^2) + \dots + 1/((n-1)^2 + n^2)$.

Giải: Cách giải tương tự như ví dụ 5.12. Chương trình như sau:

```

#include<iostream>
#include<cmath>
using namespace std;
double n;
void nhap()
{
    int i=1;
    cout<<"n = "; cin>>n;
}
double tinh(double n)
{
    if (n==1)
        return 1;
    else
        return (1/(n*(n-1) +n*n)) + tinh(n-1);
}
int main()
{
    nhap();
    cout<<"Ket qua: "<< tinh(n);
    return 0;
}

```

Ví dụ 5.14. Viết chương trình tính $n!$ Theo công thức $n! = 1.2.3\dots n$

Giải:

```

#include<iostream>
using namespace std;
int giaithua(int m)
{
    if (m==1)

```

```

        return 1;
    else
        return m*giaithua(m-1);
}
int main()
{
    double n;
    cout<<"Nhap vao gia tri cua n: ";
    cin>>n;
    cout<<"Gia tri cu n! la: "<<giaithua(n);
    return 0;
}

```

Ví dụ 5.15. Tính các số Fibonacci thứ n (kí hiệu là F_n), trong đó $F_0 = 1$; $F_1 = 1$; $F_n = F_{n-1} + F_{n-2}$.

Giải: Chúng ta thấy rằng F_n chính là công thức truy hồi đệ quy, như vậy hàm đệ quy sẽ sử dụng trực tiếp công thức này.

```

#include<iostream>
using namespace std;
int fibo(int m)
{
    if ((m==0) || (m==1))
        return 1;
    else
        return fibo(m-1) + fibo(m-2);
}
int main()
{
    double n;
    cout<<"Nhap vao gia tri cua n: ";
    cin>>n;
    cout<<"Gia tri cua so Fibonacci thu n la "<<fibo(n);
    return 0;
}

```

Ví dụ 5.16. *Bài toán tháp Hà nội:* Có n đĩa đặt tại cọc 1, kích thước các đĩa là khác nhau và đĩa bé nằm trên đĩa to, cần chuyển n đĩa từ cọc 1 sang cọc 3 và sử dụng cọc 2 làm trung gian theo điều kiện sau:

- Mỗi lần chuyển chỉ được 1 đĩa
- Các đĩa trên mỗi cọc phải thỏa mãn điều kiện đĩa nhỏ luôn nằm trên đĩa to

Giải: Ý tưởng đệ quy chuyển n đĩa từ cọc 1 sang cọc 3 nhờ cọc 2 làm trung gian như sau:

- Chuyển n-1 đĩa từ cọc 1 sang cọc 2 nhờ cọc 3 làm trung gian,
- Chuyển 1 đĩa từ cọc 1 sang cọc 3,
- Chuyển n-1 đĩa từ cọc 2 sang cọc 3 nhờ cọc 1 làm trung gian.

Chương trình như sau:

```
#include<iostream>
using namespace std;
void chuyen(int n, int a, int b, int c)
{ if (n==1)
    cout<<"Chuyen mot dia tu coc "<<a<<" sang coc "<<c<<"\n";
  else
    { chuyen(n-1, a,c,b);
      chuyen(1, a,b,c);
      chuyen(n-1, b,a,c);}
}
int main()
{
  int n;
  cout<<"Chuong trinh de quy cho bai toan Thap Ha Noi "<<endl;
  cout<<"Nhap vao so dia can chuyen n = ";
  cin>>n;
  cout<<endl<<"Thu tu chuyen dia nhu sau: "<<"\n";
  chuyen(n,1,2,3);
  return 0;
}
```

Kết quả chạy thử nghiệm chương trình như sau:

```
Chuong trinh de quy cho bai toan Thap Ha Noi
Nhap vao so dia can chuyen n = 4

Thu tu chuyen dia nhu sau:
Chuyen mot dia tu coc 1 sang coc 2
Chuyen mot dia tu coc 1 sang coc 3
Chuyen mot dia tu coc 2 sang coc 3
Chuyen mot dia tu coc 1 sang coc 2
Chuyen mot dia tu coc 3 sang coc 1
Chuyen mot dia tu coc 3 sang coc 2
Chuyen mot dia tu coc 1 sang coc 2
Chuyen mot dia tu coc 1 sang coc 3
Chuyen mot dia tu coc 2 sang coc 3
Chuyen mot dia tu coc 2 sang coc 1
Chuyen mot dia tu coc 3 sang coc 1
Chuyen mot dia tu coc 2 sang coc 3
Chuyen mot dia tu coc 1 sang coc 2
Chuyen mot dia tu coc 1 sang coc 3
Chuyen mot dia tu coc 2 sang coc 3
```

Bài tập chương 5

Bài 1. Với mỗi khai báo hàm sau đây xác định số lượng, kiểu và thứ tự các giá trị được truyền vào khi hàm đó được gọi.

- a) `float dien_tich_hinh_chu_nhat(float a, float b)`
- b) `int dien_tich_hinh_chu_nhat(float a, float b, float &s)`
- c) `float can_bac_hai(int a)`

Bài 2. Chọn câu sai trong các câu sau đây:

- a) Hàm không trả lại giá trị thì không cần khai báo kiểu giá trị của hàm.
- b) Các biến được khai báo trong hàm là cục bộ, tự xoá khi hàm thực hiện xong
- c) Hàm không trả lại giá trị sẽ có kiểu giá trị ngầm định là `void`.
- d) Hàm là đơn vị độc lập, không được khai báo hàm lồng nhau.

Bài 3. Chọn câu đúng nhất trong các câu sau đây:

- a) Hàm phải được kết thúc với 1 câu lệnh `return`: Phải có ít nhất 1 câu lệnh `return` cho hàm
- b) Các câu lệnh `return` được phép nằm ở vị trí bất kỳ trong thân hàm
- c) Không cần khai báo kiểu giá trị trả lại của hàm nếu hàm không có lệnh `return`

Bài 4. Chọn câu sai trong các câu sau đây:

- a) Số tham số thực sự phải bằng số tham số hình thức trong lời gọi hàm
- b) Các biến cục bộ trong thân hàm được chương trình dịch cấp phát bộ nhớ
- c) Các tham số hình thức sẽ được cấp phát bộ nhớ tạm thời khi hàm được gọi
- d) Kiểu của tham số thực sự phải bằng kiểu của tham số hình thức tương ứng với nó trong lời gọi hàm

Bài 5. Để thay đổi giá trị của tham biến, các đối của hàm cần khai báo dưới dạng:

- a) Biến bình thường và tham số được truyền theo giá trị
- b) Biến bình thường và tham số được truyền theo địa chỉ
- c) Biến tham chiếu và tham số được truyền theo giá trị

Bài 6. Xác định kiểu dữ liệu và phạm vi hoạt động của các biến trong đoạn mã sau đây:

```
#include<iostream>
using namespace std;
int volts;
```

```

long int resistance;
double current;
int main()
{ int power;
  double factor, time;
  . . .
  return 0;
}
double roi(int mat1, int mat2)
{
  int count;
  . . .
  double weight;
  return weight;
}
int step(double first, double last)
  int h; double frac;
  . . .
  return 10*h;

```

sử dụng bảng để xác định như hình sau:

Tên biến	Kiểu của biến	Phạm vi
volts	int	Hàm main(), roi(), và step()
...		

- Đánh dấu các đoạn mã tương ứng với các phạm vi của mỗi biến ở trên.

- Xác định kiểu và phạm vi hoạt động của các tham số trong các hàm.

Bài 7. Viết các khai báo hàm cho các trường hợp sau đây:

- Hàm `check()` có 3 tham số trong đó tham số đầu là nguyên, 2 tham số sau là thực, hàm không trả về giá trị nào.

- Hàm `findAbs()` có một tham số thực và trả về giá trị tuyệt đối của tham số đó.

- Hàm `powfun()` có hai tham số nguyên a và b , hàm trả về giá trị của a^b .

Bài 8. a. Định nghĩa một hàm có tên là `hien_thi()` có ba tham số. Tham số thứ nhất có kiểu nguyên, tham số thứ hai và thứ ba có kiểu thực. Thân của hàm có nhiệm vụ hiển thị các giá trị của ba tham số lên màn hình.

b. Viết hàm `main()` nhập vào ba biến `a` kiểu nguyên, `b` và `c` có kiểu thực; gọi hàm `hien_thi()` với ba tham số thực sự `a`, `b`, `c` để kiểm tra kết quả.

Bài 9. a. Viết hàm có tên `findAbs()` có một tham số nguyên, tính giá trị tuyệt đối của nó và in kết quả lên màn hình.

b. Viết chương trình gọi hàm `findAbs()` trong hàm `main()` để hiển thị giá trị tuyệt đối của một số `a` kiểu nhập vào từ bàn phím.

Bài 10. a. Viết hàm `bscnn()` có hai tham số `a` và `b` để tính bội số chung nhỏ nhất của số `a` và `b`. Hàm này được tính bằng cách sử dụng hàm `ucln()` đã được định nghĩa trong ví dụ 5.6.

b. Gọi hàm `bscnn()` trong hàm `main()` để hiển thị giá trị bội số chung nhỏ nhất của hai số `x`, `y` nhập vào từ bàn phím.

Bài 11. Viết chương trình với các hàm, cộng, trừ nhân, và chia hai số thực với nhau.

Bài 12. a. Thể tích của một hình trụ `V` được cho bởi công thức sau:

$$V = \pi r^2 l$$

trong đó `r` là bán kính hình trụ, và `l` là chiều dài của nó. Sử dụng công thức trên viết hàm `V_hinh_tru()` có hai tham số là `r` và `l`, hàm trả về giá trị thể tích của hình trụ tương ứng.

b. Viết chương trình gọi hàm `V_hinh_tru()` trong hàm `main()` để trả về diện tích của một hình trụ có bán kính `r` và chiều dài `l` nhập từ bàn phím.

Bài 13. a. Diện tích hình trụ `S` được cho bởi công thức:

$$S = 2\pi r l$$

Với `r` là bán kính, `l` là chiều dài hình trụ.

Sử dụng công thức trên để định nghĩa hàm `S_hinh_tru()` có hai tham số `r` và `l`.

b. Viết chương trình gọi hàm `S_hinh_tru()` trong hàm `main()` để hiển thị diện tích của hình trụ với `r` và `l` nhập từ bàn phím.

Bài 14. a. Định nghĩa hàm `S_v_hinh_tron()` có 1 tham số tham trị `r`, 2 tham số tham chiếu để nhận về diện tích và chu vi hình tròn.

b. Gọi hàm `S_V_hinh_tron()` trong hàm `main()` để hiển thị diện tích và chu vi hình tròn với bán kính `r` nhập vào từ bàn phím.

Bài 15. a. Định nghĩa hàm `r_hinh_tron()` để tính bán kính `r` khi biết chu vi của hình tròn.

b. Định nghĩa hàm `S_hinh_tron()` để tính diện tích hình tròn bằng cách sử dụng hàm `r_hinh_tron()` đã định nghĩa ở trên.

c. Gọi hàm `S_hinh_tron()` trong hàm `main()` để hiển thị diện tích hình tròn khi chu vi được nhập vào từ bàn phím

Bài 16. a. Định nghĩa hàm `S_tam_giac()` để tính diện tích của một tam giác với ba tham số tham trị là độ dài ba cạnh tam giác, một tham số tham chiếu là diện tích của tam giác đó.

b. Gọi hàm `S_tam_giac()` trong hàm `main()` để hiển thị diện tích của tam giác có độ dài ba cạnh nhập vào từ bàn phím

Gợi ý: sử dụng công thức Heron:

$$S = \sqrt{p(p-a)(p-b)(p-c)} \quad \text{Với } p = \frac{(a+b+c)}{2}$$

Bài 17. a. Định nghĩa hàm `dem_giay()` có các tham số tham trị là giờ, phút, giây và một tham số tham chiếu là `tong_so_giay`. Hàm có nhiệm vụ tính toán tổng số giây được truyền cho hàm.

b. Gọi hàm `dem_giay()` trong hàm `main()` và hiển thị giá trị tổng số giây từ số giờ, phút, giây được nhập vào từ bàn phím.

Bài 18. a. Sử dụng thuật toán Newton_Raphson để định nghĩa hàm `Sqrt()` để tính căn bậc hai của một giá trị được truyền vào cho hàm đó. (Không sử dụng hàm `sqrt()`).

b. Gọi hàm `Sqrt()` trong hàm `main()` để tính căn bậc hai của một số thực nhập vào từ bàn phím.

Gợi ý: Thuật toán Newton_Raphson:

$$x_{(n+1)} = (x_n + a / x_n) / 2$$

Với `n` càng lớn thì độ chính xác càng lớn. Nếu ta muốn có độ chính xác là `e` thì ta tính cho tới khi $|x_n - x_{n-1}| < e$

Bài 19. a. Định nghĩa hàm `luy_thua()` có hai tham số x và y để tính giá trị x lũy thừa y .

b. Viết chương trình gọi hàm `luy_thua()` trong hàm `main()` để tính a lũy thừa b với hai biến a, b được nhập vào từ bàn phím (yêu cầu không sử dụng hàm `pow()`).

Bài 20. a. Định nghĩa hàm `chan_le()` có 1 tham số x để kiểm tra x là số chẵn hay lẻ.

b. Viết chương trình gọi hàm `chan_le()` trong hàm `main()` để cho biết số a nhập vào từ bàn phím là chẵn hay lẻ.

Bài 21. Cho công thức: $\text{Pi} = 2.0 * \text{asin}(1.0)$; hàm `asin()` nằm trong thư viện `cmath`.

a. Định nghĩa hàm `pi()` để tính toán giá trị của π theo công thức trên.

b. Viết chương trình gọi hàm `pi()` trong hàm `main()`.

Bài 22. a. Định nghĩa hàm `max()` có 3 tham số kiểu nguyên, hàm trả về giá trị lớn nhất của 3 tham số đó.

b. Viết chương trình gọi hàm `max()` trong hàm `main()` để hiển thị giá trị lớn nhất của ba biến a, b, c kiểu nguyên được nhập giá trị từ bàn phím.

Bài 23. a. Định nghĩa hàm `canh_huyen()` có hai tham số là độ dài hai cạnh của một tam giác vuông, hàm này trả về giá trị của cạnh huyền của tam giác vuông đó.

b. Viết chương trình gọi hàm `canh_huyen()` trong hàm `main()` để hiển thị kết quả của hàm với hai cạnh của tam giác vuông được nhập vào từ bàn phím.

Bài 24. a. Định nghĩa hàm `phuong_trinh_bac_1()` có 2 tham số kiểu nguyên, hàm này thực hiện việc giải phương trình bậc nhất: $a * x + b = 0$.

b. Gọi hàm `phuong_trinh_bac_1()` trong hàm `main()` để kiểm tra hoạt động của hàm.

Bài 25. a. Định nghĩa hàm `pt_bac_hai()` để giải phương trình bậc hai: $a * x^2 + b * x + c = 0$

b. Gọi hàm `pt_bac_hai()` trong hàm `main()` để kiểm tra hoạt động của hàm.

Bài 26. Viết hàm `yearcal()` với ba tham số y, m, d ($d \geq 1900$) tương ứng là ngày, tháng và năm. Hàm trả về số ngày tính từ mốc 1/1/1900.

Bài 27. Viết chương trình gồm các yêu cầu sau:

a. Viết hàm `thap_phan()` trả về giá trị thập phân của một số thực được truyền vào cho hàm

b. Viết hàm `main()` để nhập dữ liệu rồi truyền giá trị vừa nhập vào cho hàm `thap_phan()`, hiển thị giá trị đó.

Ví dụ: Nhập vào giá trị 9.12345 thì sẽ in lên màn hình 0.12345

Bài 28. Thuật toán làm tròn một số thập phân đến n chữ số sau dấu phẩy thực hiện như sau:

Bước 1. Nhân số đó với 10^n .

Bước 2. Cộng thêm 0.5

Bước 3. Xóa phần thập phân

Bước 4. Chia số còn lại cho 10^n

a. Viết hàm `lam_tron()` có hai tham số, sử dụng thuật toán trên và làm tròn một số thập phân a về số thập phân với n chữ số sau dấu phẩy.

b. Viết hàm `main()` để nhập a (số cần làm tròn) và n (số chữ số sau dấu phẩy) vào từ bàn phím, truyền tham số a và n cho hàm `lam_tron()`, hiển thị kết quả sau khi thực hiện hàm `lam_tron()`

Bài 29. Một năm là năm nhuận nếu nó chia hết cho 400 hoặc chia hết cho 4 và không chia hết cho 100.

a. Viết hàm `nam_nhuan()` để kiểm tra một năm có phải là năm nhuận hay không.

b. Viết hàm `main()` để nhập một số từ bàn phím, dùng hàm `nam_nhuan()` để kiểm tra số nhập vào có phải là năm nhuận hay không rồi hiển thị kết quả lên màn hình.

Bài 30.

a. Viết hàm `phan_nguyen()` để trả về phần nguyên của một số thập phân

b. Viết hàm `main()` để nhập vào một số thập phân bất kỳ từ bàn phím. Dùng hàm `phan_nguyen()` để hiển thị phần nguyên của số thập phân đó.

Bài 31.a. Định nghĩa hàm đổi tiền `change()` gồm 7 tham số như sau: Tham số đầu tiên là tham trị kiểu số nguyên (integer) thể hiện tổng số tiền cần đổi, 6 tham số tiếp theo là tham biến kiểu số nguyên gồm hundreds, fifies, twenties, tens, fives, ones tương ứng với số tờ tiền có mệnh giá tương ứng là 100.000VNĐ, 50.000VNĐ, 20.000VNĐ,

10.000VNĐ, 5000VNĐ, 1000VNĐ sau khi đã được đổi ra. Thân hàm thực hiện việc đổi số tiền từ tham số đầu tiên thành số tờ tiền tương ứng với các mệnh giá trên sao cho tổng giá trị sau khi đổi phải bằng tổng giá trị nhập vào, ưu tiên đổi thành tiền có mệnh giá cao nhất có thể.

b. Gọi hàm `change()` trong hàm `main()` để kiểm tra hoạt động của hàm.

Bài 32. a. Định nghĩa hàm `time()` có một tham trị nguyên (integer) tên là `seconds` và 3 tham chiếu nguyên (integer) có tên là `hours`, `mins`, and `secs`. Hàm này chuyển số giây truyền vào thành một số tương đương của giờ, phút và giây.

b. Gọi hàm `time()` trong hàm `main()` để kiểm tra sự hoạt động của hàm.

Bài 33. a. Định nghĩa hàm `khoang_cach()` nhận tọa độ vuông góc của hai điểm: (x_1, y_1) và (x_2, y_2) . Tính khoảng cách giữa hai điểm.

Khoảng cách d giữa hai điểm được tính bằng công thức: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

b. Gọi hàm `khoang_cach()` trong hàm `main()` để kiểm tra sự hoạt động của hàm.

Bài 34. Độ võng của một rầm thép phụ thuộc vào chức năng của nó. Với rầm nằm ngang tối đa là $L/240$ (inch), trong khi với rầm đặt chéo là $L/180$ với L là chiều dài của xà.

a. Viết hàm `do_vong()` để tính độ võng tối đa với tham số đầu vào là L và chức năng (nằm ngang hay chéo) và trả về độ võng tối đa cho phép của nó.

b. Gọi hàm `do_vong()` trong hàm `main()` để hiển thị độ võng tối đa cho phép biết chiều dài L được nhập vào từ bàn phím

Bài 35. Chất lỏng chảy trong một ống sẽ chia thành hai dạng hoặc là dạng laminar hoặc là dạng turbulent. Tốc độ của mỗi loại được xác định bằng công thức sau: $v_{lam} = 2100\mu/\rho d$ và $v_{tur} = 4000\mu/\rho d$, trong đó v_{lam} là tốc độ của chất lỏng dạng laminar, μ là độ dẻo của chất lỏng, ρ là mật độ của chất lỏng, và d là đường kính trong của ống.

a. Viết chương trình trong đó có hàm `flow()` trả về vận tốc v_{lam} và v_{tur} .

b. Viết hàm `main()` để nhập các tham số μ , ρ , d vào từ bàn phím, truyền các tham số này cho hàm `flow()` rồi hiển thị giá trị v_{lam} và v_{tur}

Bài 36 Thuật toán sinh số ngẫu nhiên *power residue method* được thực hiện như sau:

Phương pháp này bắt đầu với một số m nguyên (có n con số), m gọi là hạt giống. Tiếp theo nhân m với $10^{n/2} - 3$ rồi lấy 6 số cuối của kết quả làm hạt giống mới. Tiếp tục quá trình như vậy ta thu được một dãy các số ngẫu nhiên, trong đó số trước làm hạt giống cho số sau. Nếu hạt giống đầu tiên là số lẻ, không chia hết cho 5, và có số chữ số ít nhất là 4 thì sẽ có $5 \cdot 10^{(n-2)}$ số ngẫu nhiên khác nhau được sinh ra trong mỗi chu kỳ của nó.

Yêu cầu: Viết chương trình có sử dụng hàm sinh các số ngẫu nhiên theo phương pháp *power residue method* ở trên.

Bài 37. Viết chương trình kiểm tra tính hiệu quả của hàm sinh số ngẫu nhiên `rand()` ở bài 36, sử dụng các biến *demkhong*, *demmot*, ..., *demtam*, *demchin* để đếm số lượng các số ngẫu nhiên sinh ra trong đoạn từ 0 đến 9. Thực hiện lặp lại khoảng 1000 lần sinh các số ngẫu nhiên rồi in kết quả của các biến lên màn hình.

Chương 6. Kiểu mảng

6.1 Khái niệm mảng

Để xử lý dữ liệu giải quyết bài toán nhằm biến đầu vào thành đầu ra theo một yêu cầu nào đó khi viết chương trình chúng ta phải biết cách biểu diễn các dữ liệu đó trong máy tính. Trong các chương trước chúng ta đã có các khai báo chẳng hạn như: `int num1; double num2;`... các khai báo này chỉ cho phép lưu trữ các giá trị đơn lẻ. Trên thực tế chúng ta gặp nhiều bài toán cần biểu diễn ở dạng các bảng chẳng hạn dãy các số nguyên tố (bảng 6-1), dãy các số về nhiệt độ trung bình (bảng 6-2), hay bảng số liệu về điểm của sinh viên (bảng 6-3),... Khi gặp các bài toán như trên chúng ta phải cần đến kiểu dữ liệu cho phép biểu diễn các giá trị theo dạng bảng, C++ cung cấp kiểu dữ liệu mảng. Một biến có kiểu mảng trong C++ cho phép lưu trữ một dãy hữu hạn các phần tử có cùng một kiểu dữ liệu liên tục trong bộ nhớ. Có các loại mảng như mảng một chiều, mảng hai chiều, mảng nhiều chiều,... trên thực tế chúng ta chỉ gặp các bài toán sử dụng mảng một chiều, hai chiều và ba chiều.

2	3	5	7	11	13	17	19
---	---	---	---	----	----	----	----

Bảng 6-1. Dãy các số nguyên tố

30	35.2	33	32	32	29	37	28
----	------	----	----	----	----	----	----

Bảng 6-2. Chỉ số nhiệt độ trung bình

	To	Li	Ho	Va	Su	Di	TB
SV1	5	7	6	4	8	9	6.50
SV2	7	8	8	5	9	8	7.50
SV3	6	7	8	6	8	6	6.83
SV4	8	8	8	9	8	9	8.33
SV5	8	9	4	5	8	6	6.67
SV6	7	8	9	0	0	9	5.50
SV7	7	8	9	0	6	8	6.33

Bảng 6-3. Bảng số liệu về điểm của thí sinh

6.2 Mảng một chiều

Mảng một chiều rất hay gặp trong các bài toán thực tế như sắp xếp dãy số, liệt kê các dãy số theo tiêu chuẩn nào đó, tìm giá trị lớn nhất, nhỏ nhất của một dãy số. Sau đây chúng ta sẽ tìm hiểu cách khai báo cũng như việc sử dụng nó trong khi viết chương trình.

6.2.1 Khai báo mảng một chiều

Để khai báo mảng một chiều ta có thể sử dụng các cách sau:

`<tên kiểu dữ liệu> <tên mảng>[số thành phần];` (i)

`<tên kiểu dữ liệu> <tên mảng>[số thành phần] = {dãy giá trị};` (ii)

`<tên kiểu dữ liệu> <tên mảng>[] = {dãy giá trị};` (iii)

Khi chúng ta khai báo mảng một chiều thì mỗi phần tử sẽ được đặc trưng bởi một chỉ số để xác định phần tử đó trong mảng. Các chỉ số ở đây là những số nguyên được đánh bắt đầu từ 0, 1, 2, Muốn truy xuất đến phần tử nào thì chúng ta chỉ cần ghi tên mảng kèm theo chỉ số của phần tử đó.

Trong các cách khai báo (i), (ii), và (iii) ở trên ta có các giải thích như sau:

- `<tên kiểu dữ liệu>` là kiểu dữ liệu của các thành phần, các thành phần này có kiểu giống nhau
- `<tên mảng>`: Là tên do người sử dụng đặt, phải tuân theo quy tắc đặt tên biến trong C++
- `[số thành phần]`: là số lượng các phần tử lớn nhất có thể có trong mảng
- Trong khai báo dạng (ii), chúng ta có thể khởi tạo các giá trị cho mảng, có thể số lượng giá trị không bằng số thành phần, các giá trị còn lại được chương trình khởi bằng 0. Đối với dạng (iii), không cần số thành phần, tuy nhiên giá trị này sẽ được tính bằng số lượng các phần tử trong dãy giá trị khởi tạo.

Một số ví dụ về khai báo mảng như sau:

- Khai báo một mảng số nguyên chứa tối đa 7 số ta viết như sau:

```
int a[7];
```

- Khai báo mảng với các số thực ta viết:

```
double b[100];
```

- Khai báo và khởi tạo giá trị chẳng hạn:

```
int a[10] = {10, 100, 1000};
```

- Khai báo không cần số lượng các phần tử:

```
float c[] = {2, 3, 5, 7, 11}.
```

6.2.2 Nhập xuất dữ liệu cho mảng một chiều

Để nhập giá trị cho các phần tử của mảng chúng ta phải tiến hành nhập từng phần tử một. Chẳng hạn nếu chúng ta muốn nhập giá trị cho một biến x thông thường chúng ta ghi `cin>>x`, tương tự với mảng để nhập giá trị cho phần tử thứ nhất cho mảng a chúng ta ghi `cin>>a[0]`; phần tử thứ hai sẽ là `cin>>a[1]`;

Để nhập dữ liệu cho một mảng a nào đó chứa n phần tử chúng ta phải sử dụng vòng lặp để điều khiển các chỉ số của nó, chẳng hạn đoạn lệnh sau sẽ dùng để nhập dữ liệu cho các phần tử của mảng a.

```
int a[5];
for (i=0; i<=4; i++)
{ cout<<"Nhập vào giá trị cho mảng: ";
  cin>>a[i];
}
```

Để in mảng hay in giá trị các phần tử của mảng chúng ta sử dụng câu lệnh `cout` và tiến hành in từng phần tử một theo yêu cầu của bài toán. Chẳng hạn để in phần tử thứ 2 của mảng a ta viết `cout<<a[1]`, in phần tử thứ 5 của mảng a ta viết `cout<<a[4]`.

Ví dụ 6.1. Cho mảng a có n phần tử, trong đó các phần tử là những số nguyên. Yêu cầu:

- Nhập n và các phần tử vào từ bàn phím
- Tính và in lên màn hình trung bình cộng các phần tử của mảng
- In lên màn hình giá trị lớn nhất và nhỏ nhất của mảng

Giải: Để viết chương trình cho bài này, chúng ta sẽ sử dụng các hàm, mỗi hàm tương ứng với một công việc trong chương trình. Chúng ta sẽ viết ba hàm tương ứng để nhập dữ liệu, tính trung bình cộng và tìm giá trị lớn nhất và nhỏ nhất của mảng.

```
#include<iostream>
using namespace std;
int n, a[100];
void nhap()
{
    int i=1;
```

```

    cout<<"So phan tu cua mang = "; cin>>n;
    for (i=0; i<n; i++)
    {
        cout<<"a["<<i<<" ] = ";
        cin>>a[i];
    }
}
void tbc()
{
    float s=0;
    for (int i=0; i<n; i++)
        s = s+a[i];
    cout<<"Trung binh cong la: "<<s/n<<endl;
}
void max_min()
{
    int max, min;
    max = a[0];
    for (int i=1; i<n; i++)
        if (max<a[i])
            max =a[i];
    cout<<"Gia tri lon nhat cua mang la: "<<max<<endl;
    min = a[0];
    for (int i=1; i<n; i++)
        if (min>a[i])
            min =a[i];
    cout<<"Gia tri nho nhat cua mang la: "<<min<<endl;
}
int main()
{
    nhap();
    tbc();
    max_min();
    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:

```

Số phần tử của mảng = 4
a[0] = 5
a[1] = 6
a[2] = 7
a[3] = 3
Trung bình cộng là: 5.25
Giá trị lớn nhất của mảng là: 7
Giá trị nhỏ nhất của mảng là: 3

```

Ví dụ 6.2. Cho mảng a có n phần tử, trong đó các phần tử là những số nguyên. Yêu cầu:

- Nhập n và các phần tử vào từ bàn phím
- In lên màn hình các số chẵn của mảng
- In lên màn hình các số chính phương của mảng

Giải: Để in các số chẵn, chúng ta duyệt mảng và kiểm tra các phần tử thỏa mãn điều kiện và in lên màn hình. Để kiểm tra một số x là số chính phương hay không chúng ta lấy căn bậc hai của nó và kiểm tra xem đó có phải là số nguyên hay không? Nếu kết quả là số nguyên thì x sẽ là số chính phương.

```

#include<iostream>
#include<cmath>
using namespace std;
const int SOPT=100;
int n, a[SOPT];
void nhap()
{
    int i;
    cout<<"Nhập số phần tử của mảng : "; cin>>n;
    for (i=0; i<n; i++)
    {
        cout<<"a["<<i<<" ] = ";
        cin>>a[i];
    }
}
void insocp()
{
    int i;
    cout<<"\nCác số chính phương của mảng: ";
    for (i=0; i<n; i++)
        if (a[i]>0)
            if (sqrt(a[i])==int(sqrt(a[i])))
                cout<<a[i]<<" ";
}

```

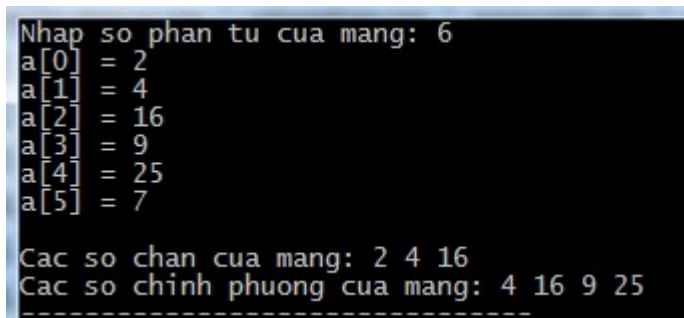
```

    }
void insochan()
{
    int i;
    cout<<"\nCac so chan cua mang: ";
    for (i=0; i<n; i++)
        if (a[i] % 2 ==0)
            cout<<a[i]<<" ";
}

int main()
{
    nhap();
    insochan();
    insocp();
    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Nhap so phan tu cua mang: 6
a[0] = 2
a[1] = 4
a[2] = 16
a[3] = 9
a[4] = 25
a[5] = 7

Cac so chan cua mang: 2 4 16
Cac so chinh phuong cua mang: 4 16 9 25
-----

```

Ví dụ 6.3. Khai báo ba mảng một chiều có tên *current*, *resistance*, và *volts*. Mỗi mảng chứa *n* phần tử (*n* nhập từ bàn phím). Nhập các giá trị cho các mảng *current* và *resistance*. Giá trị mảng *volts* được tính như sau: $volts[i] = current[i] * resistance[i]$. In kết quả lên màn hình theo ba cột lần lượt là giá trị của *current*, *resistance* và *volts* tương ứng.

Giải: Chúng ta sẽ viết một hàm nhập các phần tử cho hai mảng và một hàm tính và in kết quả của mảng volt lên màn hình. Chương trình như sau:

```

#include<iostream>
#include<cmath>
using namespace std;
float current[100], resistance[100], volts[100];
int n;
void nhap()

```

```

{
    int i;
    cout<<"Nhap so phan tu: "; cin>>n;
    cout<<"Nhap gia tri cho mang current"<<endl;
    for (i=0; i<n; i++)
    {
        cout<<"current["<<i<<" ] = ";
        cin>>current[i];
    }
    cout<<"Nhap gia tri cho mang volts: "<<"\n";
    for (i=0; i<n; i++)
    {
        cout<<"volts["<<i<<" ] = ";
        cin>>volts[i];
    }
}

void tinh_toan()
{
    int i;
    for (i=0; i<n; i++)
        resistance[i] = volts[i]*current[i];
    cout<<"Hien thi ba mang theo cot: "<<endl;
    cout<<"volts current resistance "<<endl;
    for (i=0; i<n; i++)
        cout<<"    "<<volts[i]<<"          "<<current[i]
            <<"          "<<resistance[i]<<" "<<endl;
}

int main()
{
    nhap();
    tinh_toan();
    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:


```
Nhap so phan tu: 4
Nhap gia tri cho mang current
current[0] = 2
current[1] = 3
current[2] = 4
current[3] = 5
Nhap gia tri cho mang volts:
volts[0] = 2
volts[1] = 4
volts[2] = 6
volts[3] = 7
Hien thi ba mang theo cot:
volts current resistance
2      2      4
4      3      12
6      4      24
7      5      35
```

Ví dụ 6.4. Cho dãy số a_1, a_2, \dots, a_n là các số nguyên. Yêu cầu viết chương trình thực hiện:

- Nhập dãy số trên từ bàn phím
- In lên màn hình các số hoàn thiện của dãy trên. Một số gọi là hoàn thiện nếu giá trị của nó bằng tổng giá trị các ước số của nó không kể chính nó.

Giải: Để xác định xem một số x nguyên có phải là số hoàn thiện hay không chúng ta phải đi tìm tất cả các ước số của x (trừ chính x) rồi tính tổng lại, nếu kết quả bằng x thì x sẽ là số hoàn thiện. Chương trình như sau:

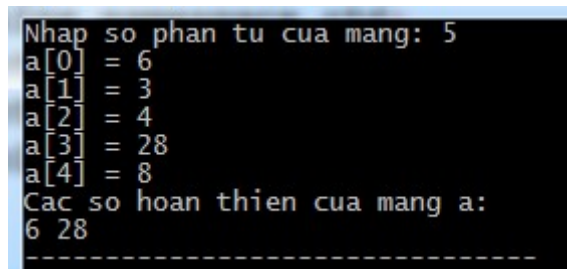
```
#include<iostream>
#include<cmath>
using namespace std;
int n;
int a[100];
void nhap()
{
    int i;
    cout<<"Nhap so phan tu cua mang: "; cin>>n;
    for (i=0; i<n; i++)
    {
        cout<<"a["<<i<<"] = ";
        cin>>a[i];
    }
}
int kt_hoanthien(int x)
{
    int i, tong;
```

```

    tong=0;
    for (i=1;i<x; i++)
        if (x % i ==0)
            tong=tong+i;
    if (tong ==x)
        return 1;
    else
        return 0;
}
//In cac so hoan thien
void inkq()
{
    int i;
    cout<<"Cac so hoan thien cua mang a: "<<endl;
    for(i=0; i<n; i++)
        if (kt_hoanthien(a[i])==1)
            cout<<a[i]<<" ";
}
int main()
{
    nhap();
    inkq();
    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Nhap so phan tu cua mang: 5
a[0] = 6
a[1] = 3
a[2] = 4
a[3] = 28
a[4] = 8
Cac so hoan thien cua mang a:
6 28
-----

```

6.2.3 Sắp xếp và tìm kiếm trên mảng một chiều

Bài toán sắp xếp là quá trình bố trí lại một dãy các phần tử theo một tiêu chuẩn nào đó. Chẳng hạn như sắp xếp danh sách sinh viên theo điểm, sắp xếp kết quả thi học sinh giỏi, sắp xếp các mặt hàng theo đơn giá ... Trên thực tế yêu cầu sắp xếp xuất hiện thường xuyên trong các công việc hàng ngày. Mục đích của phần này tập trung vào sắp xếp trên mảng một chiều dựa trên giá trị các phần tử đó. Trên thực tế có 6 phương pháp sắp xếp

tăng dần (giảm dần) của một dãy số bao gồm các phương pháp: lựa chọn (*selection sort*), chèn trực tiếp (*insertion sort*), nổi bọt (*bubble sort*), sắp xếp nhanh (*quicksort*), và sử dụng cây nhị phân (*heap sort*). Chúng ta sẽ tìm hiểu hai phương pháp là sắp xếp bằng phương pháp lựa chọn và sắp xếp bằng phương pháp chèn trực tiếp thông qua các ví dụ sau.

Ví dụ 6.5 Cho mảng a có n phần tử, trong đó các phần tử là những số nguyên. Yêu cầu:

- Nhập n và các phần tử vào từ bàn phím
- Sắp xếp mảng theo giá trị tăng dần bằng thuật toán sắp xếp kiểu lựa chọn.

Giải: Cho dãy a_1, a_2, \dots, a_n , thuật toán sắp xếp bằng phương pháp lựa chọn thực hiện như sau: tại bước thứ i ($1 \leq i \leq n-1$) ta đi tìm giá trị nhỏ nhất trong đoạn $[a_i, a_{i+1}, \dots, a_n]$ và đặt vào vị trí thứ i .

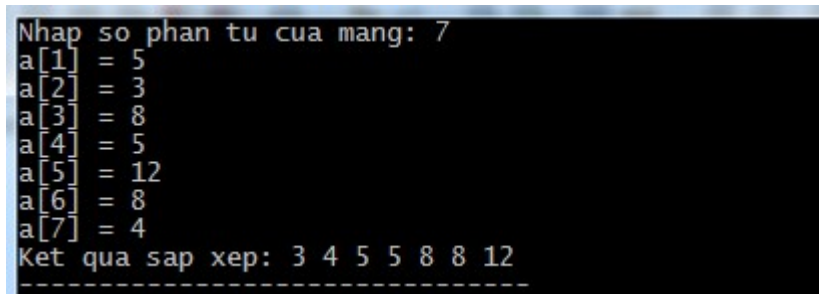
```
#include<iostream>
#include<cmath>
using namespace std;
int n;
int a[100];
void nhap()
{
    int i=0;
    cout<<"Nhập số phần tử của mảng: "; cin>>n;
    for (i=0; i<n; i++)
    {
        cout<<"a["<<i<<"] = ";
        cin>>a[i];
    }
}
void sapxep()
{
    int i,j,m,tg;
    for (i=0; i<n-1; i++)
    {
        m= i;
        for (j=i+1; j<n; j++)
            if (a[j]<a[m])
                m =j;
        if ( m !=i)
            {tg = a[i];a[i] = a[m]; a[m] =tg;}
```

```

    }
    cout<<"Ket qua sap xep: ";
    for (i=0; i<n; i++)
        cout<<a[i]<<" ";
}
int main()
{
    nhap();
    sapxep();
    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Nhap so phan tu cua mang: 7
a[1] = 5
a[2] = 3
a[3] = 8
a[4] = 5
a[5] = 12
a[6] = 8
a[7] = 4
Ket qua sap xep: 3 4 5 5 8 8 12
-----

```

Ví dụ 6.6 Cho mảng a có n phần tử, trong đó các phần tử là những số nguyên. Yêu cầu:

- Nhập n và các phần tử vào từ bàn phím
- Sắp xếp mảng theo giá trị tăng dần bằng thuật toán sắp xếp chèn trực tiếp.

Giải: Ý tưởng cơ bản của phương pháp sắp xếp bằng phương pháp chèn trực tiếp giống như ý tưởng của người chơi bài sắp xếp các cây bài theo thứ tự tăng dần. Đầu tiên giả sử dãy chỉ có một phần tử, khi phần tử tiếp theo được đưa vào ta sẽ tìm chỗ cho nó sao cho tạo thành dãy hai phần tử tăng dần, tương tự đối với phần tử thứ ba, thứ tư... thứ n được thêm vào. Chương trình như sau:

```

#include<iostream>
#include<cmath>
using namespace std;
int n;
int a[100];
void nhap()
{ int i=0;
  cout<<"n = "; cin>>n;

```

```

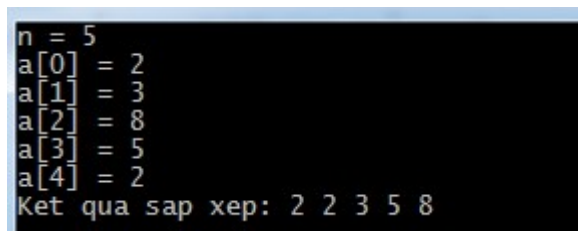
    for (i=0; i<n; i++)
    { cout<<"a["<<i<<" ] = ";
      cin>>a[i];
    }
}

void sapxep()
{ int i,j,m,x;
  for (i=1; i<n; i++)
  { x= a[i]; j=i-1;
    while ((j>0) && (x<a[j]))
      {a[j+1] = a[j]; j=j-1;}
    a[j+1]=x;
  }
  cout<<"Ket qua sap xep: ";
  for (i=0; i<n; i++)
    cout<<a[i]<<" ";
}

int main() {
  nhap();
  sapxep();
  return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

n = 5
a[0] = 2
a[1] = 3
a[2] = 8
a[3] = 5
a[4] = 2
Ket qua sap xep: 2 2 3 5 8

```

Bài toán tìm kiếm là việc tìm xem sự có mặt của một phần tử trong một dãy đã cho hay không. Có một số cách tìm kiếm như tìm kiếm tuần tự, tìm kiếm nhị phân (áp dụng cho dãy đã sắp xếp), tìm kiếm trên cây nhị phân... Sau đây chúng ta tìm hiểu phương pháp tìm kiếm tuần tự.

Ví dụ 6.7. Cho mảng a có n phần tử, trong đó các phần tử là những số nguyên. Yêu cầu:

- Nhập n và các phần tử vào từ bàn phím
- Tìm và đếm xem có bao nhiêu phần tử bằng một số k cho trước.

Giải: Sử dụng một biến đếm số lượng các phần tử bằng k. Chúng ta duyệt mảng từ phần tử đầu tiên đến phần tử cuối cùng và khi có phần tử của mảng bằng k thì biến đếm số lượng sẽ được tăng lên một đơn vị.

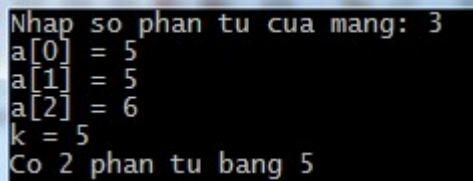
```
#include<iostream>
#include<cmath>
using namespace std;
int n, k;
int a[100];
void nhap()
{
    int i;
    cout<<"Nhập số phần tử của mảng: "; cin>>n;
    for (i=0; i<n; i++)
    {
        cout<<"a["<<i<<"] = ";
        cin>>a[i];
    }
    cout<<"k = "; cin>>k;
}
//Tìm kiếm tuần tự
void timkiem()
{
    int i, dem;
    dem=0;
    for (i=0; i<n; i++)
    if (a[i] == k)
    dem = dem +1;
    if (dem>0)
        cout<<"Có "<<dem<<" phần tử bằng "<<k;
    else
        cout<<"Trong mảng không có phần tử bằng "<<k;
}
int main()
{
    nhap();
    timkiem();
}
```

```

    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Nhap so phan tu cua mang: 3
a[0] = 5
a[1] = 5
a[2] = 6
k = 5
Co 2 phan tu bang 5

```

6.2.4 Một số ví dụ khác

Ví dụ 6.8. Cho mảng a có n phần tử, trong đó các phần tử là những số nguyên. Yêu cầu:

- Nhập n và các phần tử vào từ bàn phím
- Sắp xếp mảng sao cho phần tử chẵn nằm về đầu dãy, phần tử lẻ nằm về cuối dãy.

Giải: Để sắp xếp phần tử chẵn lên đầu dãy và phần tử cuối đứng cuối dãy chúng ta chỉ việc duyệt dãy nếu có cặp phần tử $(a[i], a[j])$ sao cho $a[i]$ lẻ và $a[j]$ chẵn trong khi $i < j$ thì chúng ta sẽ đổi chỗ của $a[i]$ và $a[j]$ cho nhau. Chương trình như sau:

```

#include<iostream>
#include<cmath>
using namespace std;
int n;
int a[100];
void nhap()
{
    int i;
    cout<<"Nhap so phan tu cua mang n = "; cin>>n;
    for (i=0; i<n; i++)
    {
        cout<<"a["<<i<<" ] = ";
        cin>>a[i];
    }
}
void sapxep()
{
    int i,j;
    i=0; j=n-1;
    do
    {
        while ((a[i] % 2)==0)

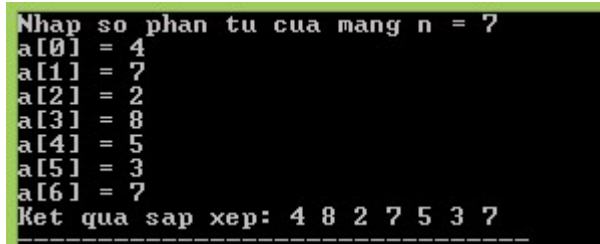
```

```

        i=i+1;
while (a[j] % 2 !=0)
    j = j-1;
if (i<j)
{
    int tg = a[i];
    a[i]=a[j];
    a[j]=tg;
}
}
while (i<j);
cout<<"Ket qua sap xep: ";
for (i=0; i<n; i++)
    cout<<a[i]<<" ";
}
int main()
{
    nhap();
    sapxep();
    return 0;
}

```

Ví dụ về việc thực hiện chương trình như sau:



```

Nhap so phan tu cua mang n = 7
a[0] = 4
a[1] = 7
a[2] = 2
a[3] = 8
a[4] = 5
a[5] = 3
a[6] = 7
Ket qua sap xep: 4 8 2 7 5 3 7

```

Ví dụ 6.9. Cho dãy số a_1, a_2, \dots, a_n là các số thực. Yêu cầu viết chương trình thực hiện:

- Nhập dãy số trên từ bàn phím,
- Tìm dãy con liên tiếp dài nhất không giảm của dãy số trên và in kết quả lên màn hình.

Giải: Để giải bài toán trên, cách đơn giản nhất là xét mọi đoạn $[i, j]$ và kiểm tra xem đó có phải là đoạn không giảm hay không và chọn đoạn có độ dài lớn.

```

#include<iostream>
using namespace std;
int n;
float a[100];

```



```

void nhap()
{
    int i;
    cout<<"Nhap so phan tu cua mang: "; cin>>n;
    for (i=0; i<n; i++)
    {
        cout<<"a["<<i<<" ] = ";
        cin>>a[i];
    }
}
//In doan co do dai khong giam
void inkq()
{
    int i,j,l1,l2,max;
    max =0;
    for (i=0; i<n-1; i++)
        for (j=i+1;j<n;j++)
        {
            int tg=0; int k;
            for (k=i;k<j;k++)
                if(a[k]>a[k+1])
                {
                    tg =1;
                    break;
                }
            if ((tg==0) && (j-i+1>max))
            {
                max = j-i+1;
                l1=i;
                l2=j;
            }
        }
    cout<<"Day con khong giam dai nhat: ";
    for (i=l1; i<=l2; i++)
        cout<<a[i]<<" ";
}
int main()
{
    nhap();
    inkq();
    return 0;
}

```

```
}
```

Ví dụ về kết quả thực hiện chương trình như sau:

```
Nhap so phan tu cua mang: 6
a[0] = 2.5
a[1] = 5.6
a[2] = 7.4
a[3] = 4.5
a[4] = 3.4
a[5] = 5.6
Day con khong giam dai nhat: 2.5 5.6 7.4
```

Ví dụ 6.10. Cho dãy số nguyên a_1, a_2, \dots, a_n ($3 < n < 5000$). Một dãy số a_1, a_2, \dots, a_n được gọi là dãy số dạng sóng nếu $(a_i - a_{i-1}) * (a_i - a_{i+1}) \geq 0, \forall i = 2, \dots, n-1$. Yêu cầu kiểm tra xem dãy số a_1, a_2, \dots, a_n có dạng sóng hay không? Nếu chưa hãy sắp xếp lại để dãy số có dạng sóng. Dữ liệu nhập từ bàn phím và in kết quả lên màn hình.

Giải: Một dãy là không có dạng sóng nếu tồn tại chỉ số i ($1 < i < n$) sao cho $(a_i - a_{i-1}) * (a_i - a_{i+1}) < 0$. Nếu dãy là không có dạng sóng ta sẽ sắp xếp dãy đó theo thứ tự tăng dần, giả sử sau khi sắp được dãy tăng dần b_1, b_2, \dots, b_n , để tạo dãy dạng sóng ta sắp như sau: $b_1, b_n, b_2, b_{n-1}, b_3, b_{n-2}, \dots$. Chương trình như sau:

```
#include<iostream>
using namespace std;
int n;
int a[5000];
void nhap()
{
    int i=0;
    cout<<"n = "; cin>>n;
    for (i=0; i<n; i++)
    {
        cout<<"a["<<i<<"] = ";
        cin>>a[i];
    }
}
void kt_sx()
{
    int i,j,tg;
    tg =0;
    for (i=1; i<n-1; i++)
    if (((a[i] - a[i-1])*(a[i] - a[i+1])) < 0)
    {
```

```

        tg=1;
        break;
    }
    if (tg==0)
        cout<<"Day la day co dang song: ";
    else //sap xep tang dan
    {
        for (i=0; i<n-1; i++)
        {
            int m= i;
            for (j=i+1; j<n; j++)
                if (a[j]<a[m])
                    m =j;
            if (m !=i)
            {
                tg = a[i];
                a[i] = a[m];
                a[m] =tg;
            }
        }
        cout<<"\n";
        for (i=0; i<n/2; i++)
            cout<<a[i]<<" "<<a[n-i-1]<<" ";
        if (n % 2 !=0)
            cout<<a[n/2 +1];
    }
}

int main()
{
    nhap();
    kt_sx();
    return 0;
}

```

Ví dụ 6.11. Cho hai mảng số thực a và b đã được sắp xếp tăng dần tương ứng có m và n phần tử. Yêu cầu xây dựng mảng c có m + n phần tử là hợp của tất cả các phần tử của mảng a và b sao cho các phần tử được sắp xếp tăng dần.

Giải: Chương trình như sau :

```

#include<iostream>
#include<cmath>
using namespace std;

```

```

float a[1000], b[1000], c[2000];
int m,n;
void nhap()
{
    int i;
    cout<<"Nhap so phan tu mang 1: "; cin>>m;
    for (i=0; i<m; i++)
    {
        cout<<"a["<<i<<" ] = ";
        cin>>a[i];
    }
    cout<<"Nhap so phan tu mang 2 "; cin>>n;
    for (i=0; i<n; i++)
    {
        cout<<"b["<<i<<" ] = ";
        cin>>b[i];
    }
}
void hoanhap()
{
    int i,j,k;
    i=0; j=0;k=0;
    while ((i<m) && (j<n))
    {
        if (a[i]<b[j]) //dua a[i] vao mang c
        {
            k=k+1;
            c[k] = a[i];
            i=i+1;
        }
        else
        {
            k=k+1;
            c[k] = b[j]; //dua b[j] vao mang c
            j=j+1;
        }
    }
    if (i<m)
        for (j=i;j<m;j++)
            {k=k+1; c[k]=a[j];}
    if (j<n)

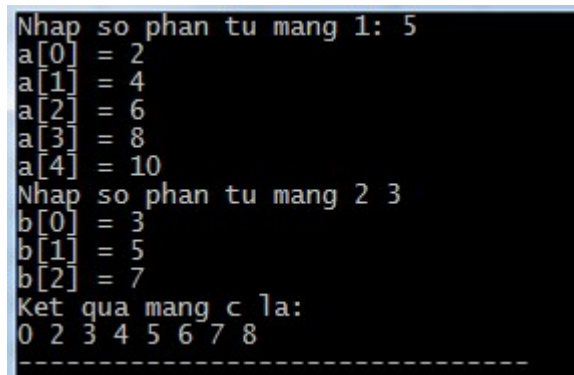
```

```

        for (i=j; i<n; i++)
            {k=k+1; c[k]=b[i];}
    cout<<"Ket qua mang c la: "<<"\n";
    for (i=0; i<m+n; i++)
        cout<<c[i]<<" ";
}
int main()
{
    nhap();
    hoanhap();
    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Nhap so phan tu mang 1: 5
a[0] = 2
a[1] = 4
a[2] = 6
a[3] = 8
a[4] = 10
Nhap so phan tu mang 2 3
b[0] = 3
b[1] = 5
b[2] = 7
Ket qua mang c la:
0 2 3 4 5 6 7 8

```

6.3 Mảng hai chiều

Mảng 2 chiều hay còn gọi là bảng bao gồm các phần tử được bố trí theo hàng và cột. Kích thước mảng hai chiều sẽ đặc trưng bởi số hàng và số cột của nó, mỗi phần tử mảng hai chiều sẽ được xác định bằng hai chỉ số: chỉ số hàng và chỉ số cột.

4	7	6	2	-10
30	2	1	100	1
3	7	-100	2	1

Bảng 6-4. Minh họa mảng hai chiều có 3 hàng, 5 cột

Khai báo mảng hai chiều

<kiểu dữ liệu> <tên mảng>[m][n];

trong đó: m, n tương ứng là số hàng và số cột của mảng, kiểu dữ liệu là kiểu của các phần tử của mảng chẳng hạn như kiểu nguyên, kiểu thực. Để khai báo mảng a có 3 hàng 6 cột

với các phần tử nguyên ta viết `int a[3][4]`; mảng `b` có 10 hàng 10 cột các phần tử là số thực ta viết `double b[10][10]`;

Trong khai báo cũng có thể được khởi tạo bằng dãy các dòng giá trị, các dòng cách nhau bởi dấu phẩy, mỗi dòng được bao bởi cặp ngoặc `{}` và toàn bộ giá trị khởi tạo nằm trong cặp dấu `{}`.

Truy xuất các phần tử mảng hai chiều

Tương tự như mảng một chiều, để truy xuất đến mảng hai chiều ta ghi tên mảng kèm theo các chỉ số của phần tử đó và để trong ngoặc vuông. Chỉ số hàng và chỉ số cột của mảng được đánh số bắt đầu từ 0.

Chẳng hạn sau khi khai báo mảng như trên ta có thể ghi `a[0][0]` để xác định phần tử góc trái trên của mảng, `a[0][1]`,... Để nhập mảng hai chiều từ bàn phím ta phải tiến hành cho từng phần tử một như những biến thông thường; để in các phần tử ra màn hình chúng ta cũng thực hiện in từng phần tử một. Để duyệt các phần tử của mảng thường ta dùng vòng lặp để điều khiển các chỉ số của nó.

Ví dụ 6.12. Cho một mảng gồm m dòng và n cột ghi các giá trị nguyên ($0 < m, n < 100$). Yêu cầu tính và in lên màn hình giá trị trung bình của mảng đó.

Giải: Để tính giá trị trung bình của mảng hai chiều ta sẽ duyệt qua tất cả các phần tử của mảng, tính tổng và tính giá trị trung bình rồi in lên màn hình. Chương trình như sau:

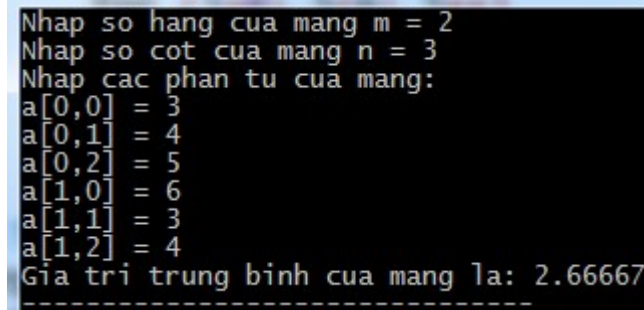
```
#include<iostream>
using namespace std;
int main()
{
    int a[100][100];
    int m, n, i, j, k;
    double tong;
    cout<<"Nhap so hang cua mang m = "; cin>>m;
    cout<<"Nhap so cot cua mang n = "; cin>>n;
    cout<<"Nhap cac phan tu cua mang: "<<endl;
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
        {
            cout<<"a[" << i << ", " << j << "] = ";
            cin >>a[i][j];
        }
}
```

```

// Tính giá trị trung bình các phần tử
tong = 0;
for (i=0; i<m; i++)
    for (j=0; j<n; j++)
        tong = tong + double(a[i][j]);
tong = tong/(double(m*n));
cout<<"Giá trị trung bình của mảng là: "<<tong;
return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Nhập số hàng của mảng m = 2
Nhập số cột của mảng n = 3
Nhập các phần tử của mảng:
a[0,0] = 3
a[0,1] = 4
a[0,2] = 5
a[1,0] = 6
a[1,1] = 3
a[1,2] = 4
Giá trị trung bình của mảng là: 2.66667
-----

```

Ví dụ 6.13. Cho một mảng gồm m dòng và n cột ghi các giá trị nguyên ($0 < m, n < 100$).

Yêu cầu in lên màn hình giá trị lớn nhất và nhỏ nhất của mảng này.

Giải: Quá trình tìm giá trị lớn nhất và nhỏ nhất của mảng được thực hiện bằng cách duyệt toàn bộ các phần tử của mảng. Chương trình như sau:

```

#include<iostream>
using namespace std;
int main()
{
    int a[100][100];
    int m, n, i, j, k, max, min;
    double tong;
    cout<<"Nhập số hàng của mảng m = "; cin>>m;
    cout<<"Nhập số cột của mảng n = "; cin>>n;
    cout<<"Nhập các phần tử của mảng: "<<endl;
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
        {
            cout<<"a[" << i << ", " << j << "] = ";
            cin >>a[i][j];
        }
    // Tính giá trị lớn nhất và nhỏ nhất
}

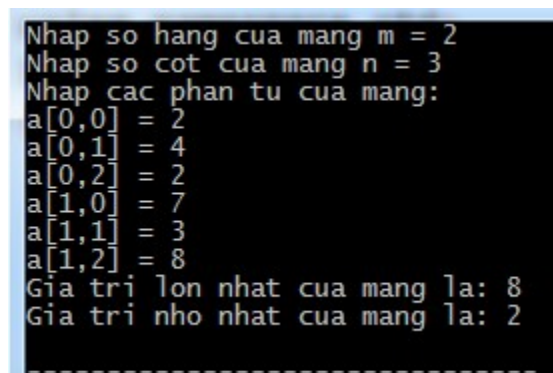
```

```

max=a[0][0];
min = a[0][0];
for (i=0; i<m; i++)
    for (j=0; j<n; j++)
    {
        if (a[i][j]>max)
            max = a[i][j];
        if (a[i][j]< min)
            min = a[i][j];
    }
cout<<"Gia tri lon nhat cua mang la: "<<max<<endl;
cout<<"Gia tri nho nhat cua mang la: "<<min<<endl;
return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Nhap so hang cua mang m = 2
Nhap so cot cua mang n = 3
Nhap cac phan tu cua mang:
a[0,0] = 2
a[0,1] = 4
a[0,2] = 2
a[1,0] = 7
a[1,1] = 3
a[1,2] = 8
Gia tri lon nhat cua mang la: 8
Gia tri nho nhat cua mang la: 2

```

Ví dụ 6.14 Cho ma trận hai chiều có n dòng và m cột trong đó các phần tử là những số nguyên. Yêu cầu:

- Nhập m, n và mảng a vào từ bàn phím
- In lên màn hình giá trị lớn nhất của từng hàng của mảng.

Giải: Để tìm giá trị lớn nhất của từng hàng ta thực hiện giống như mảng một chiều, chương trình như sau:

```

#include<iostream>
#include<cmath>
using namespace std;
int a[100][100];
int m,n,k;
void nhap()
{
    int i, j;

```

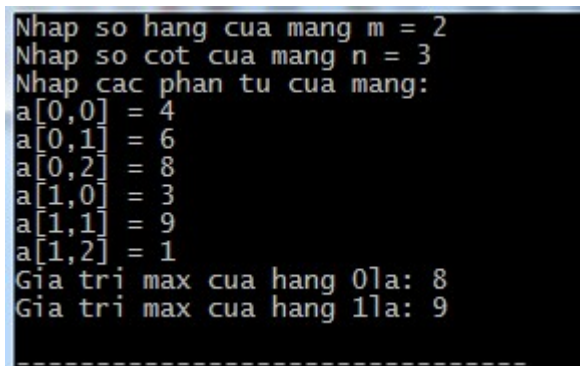


```

cout<<"Nhap so hang cua mang m = "; cin>>m;
cout<<"Nhap so cot cua mang n = "; cin>>n;
cout<<"Nhap cac phan tu cua mang: "<<endl;
for (i=0; i<m; i++)
    for (j=0; j<n; j++)
    {
        cout<<"a[" << i << "," << j << "] = ";
        cin >>a[i][j];}
    }
void inkq()
{
    int i,j, k,l, max;
    for (i=0; i<m;i++)
    {
        max =a[i][0];
        for (j=0; j<n;j++)
            if (a[i][j]> max)
                max = a[i][j];
        cout<<"Gia tri max cua hang "<<i<<"la: "<<max<<endl;
    }
}
int main()
{
    nhap();
    inkq();
    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Nhap so hang cua mang m = 2
Nhap so cot cua mang n = 3
Nhap cac phan tu cua mang:
a[0,0] = 4
a[0,1] = 6
a[0,2] = 8
a[1,0] = 3
a[1,1] = 9
a[1,2] = 1
Gia tri max cua hang 0la: 8
Gia tri max cua hang 1la: 9

```

Ví dụ 6.15. Cho mảng hai chiều kích thước m,n chỉ chứa các số 0 và 1. Yêu cầu :

- Nhập m,n và mảng a từ bàn phím

- In lên màn hình các miền liên thông gồm toàn số 1 của mảng a. Mỗi miền liên thông được định nghĩa là miền lớn nhất toàn số 1 trong đó hai ô bất kì của miền đều liên thông với nhau theo nghĩa nếu muốn đi từ ô này đến ô kia ta có thể di chuyển qua dãy các ô kề cạnh chứa toàn số 1.

Giải : Chúng ta có thể giải bài toán này bằng đệ quy hoặc không đệ quy. Sau đây phương pháp giải bằng đệ quy sẽ được sử dụng. Xuất phát từ một ô chứa số 1, quá trình tìm kiếm miền liên thông sẽ được loang ra 4 ô kề cạnh với nó đến khi nào không loang được nữa thì ta sẽ thu được một miền liên thông. Quá trình sẽ tiếp tục với các ô chứa số 1 khác. Để thuận tiện cho việc tính chỉ số chúng ta sẽ sử dụng các phần tử của mảng bắt đầu từ chỉ số là 1. Chương trình như sau :

```
#include<iostream>
#include<cmath>
using namespace std;
int a[100][100]; int m,n,k;
int kt[100][100];
void nhap()
{
    int i=1;
    do
    {cout<<"Nhap so hang cua mang m = "; cin>>m;
    cout<<"Nhap so cot cua mang n = "; cin>>n;}
    while ((n<100) && (m<100));
    cout<<"Nhap cac phan tu cua mang: ";
    for (i=1; i<=m; i++)
        for (int j=1; j<=n; j++)
        {
            cout<<"a[" << i << "," << j << "] = ";
            cin >>a[i][j];}
    cout<<"\nDu lieu nhap: \n";
    for (i=1; i<=m; i++)
    {
        for (int j=1; j<=n; j++)
            cout<<a[i][j]<<" ";
        cout<<"\n";
    }
    for (i=1; i<=m; i++)
        for (int j=1; j<=n; j++)
```

```

        if (a[i][j]==1)
            kt[i][j] =0;
        else
            kt[i][j]=1;
    }
void thu(int i, int j)
{
    kt[i][j]=1;
    cout<<i<<" "<<j<<"\n";a[i][j]=0;
    if ((i-1>0) && (a[i-1][j]==1))
        thu(i-1,j);
    if ((i+1<=m) && (a[i+1][j]==1))
        thu(i+1,j);
    if ((j+1<=n) && (a[i][j+1]==1))
        thu(i,j+1);
    if ((j-1>0) && (a[i][j-1]==1))
        thu(i,j-1);
}
void inkq()
{ int i,j, dem;
  dem =0;
  for (i=1; i<=m;i++)
  for (j=1;j<=n;j++)
    if ((a[i][j]==1) && (kt[i][j]==0))
    { dem= dem +1;
      cout<<"Mien lien thong thu "<<dem<<":\n";
      thu(i,j);
    }
}
int main()
{
    nhap();
    inkq();
    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:

```

Du lieu nhap:
1 1 1 0 0 0
0 0 0 0 1 1
1 1 1 0 0 0
0 0 1 1 1 0
0 1 1 1 1 1
Mien lien thong thu 1:
1 1
1 2
1 3
Mien lien thong thu 2:
2 5
2 6
Mien lien thong thu 3:
3 1
3 2
3 3
4 3
5 3
5 4
4 4
4 5
5 5
5 6
5 2

```

Ví dụ 6.16. Hãy xếp 8 quân hậu trên bàn cờ quốc tế sao cho không có hai quân hậu nào là không chế nhau.

Giải: Chúng ta giải bài này bằng phương pháp đệ quy quay lui. Chương trình cứ tiến hành đặt lần lượt các quân hậu cho đến khi không đặt được nữa nếu đủ 8 quân thì in kết quả ngược lại chúng ta phải quay lui và chọn phương án khác. Ví dụ về một phương án đặt 8 quân hậu được cho như hình sau (Q: thể hiện vị trí đặt quân hậu) :

Q							
				Q			
							Q
					Q		
		Q					
						Q	
	Q						
			Q				

Để sử dụng đệ quy quay lui trong bài toán này, chúng ta dùng 3 mảng để đánh dấu hàng ngang, hàng dọc và đường chéo của bàn cờ ; trong chương trình tương ứng với mảng a, b và c. Chương trình như sau:

```

#include<iostream>
using namespace std;
int x[10], a[10], b[20], c[20];
int h[9][9];
int dem=0;
void khoitao()
{
    int i,j;
    for (i=1; i<=8;i++)
        a[i]=0;
    for (i=1; i<=20;i++)
    {
        c[i]=0;
        b[i]=0;
    }
    for (i=1;i<=8;i++)
        for (j=1; j<=8; j++)
            h[i][j]=0;
}
void inkq()
{
    int i;
    dem=dem+1;
    cout<<"Phuong an thu "<<dem<<":\n\n";
    for (i=1;i<=8;i++)
        for (int j=1; j<=8; j++)
            h[i][j]=0;
    for (i=1; i<=8;i++)
        h[i][x[i]]=1;
    for (i=1;i<=8;i++)
    {
        for (int j=1; j<=8; j++)
            cout<<h[i][j]<<" ";
        cout<<"\n";
    }
    cout<<"\n";
}
void try1(int i)
{
    int j;
    for (j = 1; j <= 8; j++)

```

```

{
    if (a[j] == 0 && b[i+j] == 0 && c[i-j+7] == 0)
    {
        x[i] = j; a[j] = 1; b[i+j] = 1; c[i-j+7] = 1;
        if (i < 8)
        {
            try1(i+1);
            a[j] = 0; b[i+j] = 0; c[i-j+7] = 0; //quay lui
        }
        else
            inkq();
        a[j] = 0; b[i+j] = 0; c[i-j+7] = 0 ; //quay lui
    }
}
}
int main()
{
    khoitao();
    try1(1);
}

```

Ví dụ về kết quả thực hiện chương trình (phần hiển thị 3 phương án cuối cùng của việc điền 8 quân hậu lên bàn cờ) như sau:

```
Phuong an thu 90:
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 1 0 0 0 0 0
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0

Phuong an thu 91:
0 0 0 0 0 0 0 1
0 0 1 0 0 0 0 0
1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0

Phuong an thu 92:
0 0 0 0 0 0 0 1
0 0 0 1 0 0 0 0
1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
```

6.4 Sử dụng mảng làm tham số trong hàm

Giống như các biến, mảng cũng được dùng làm tham số trong các hàm khi viết chương trình. Có hai cách sử dụng, sử dụng từng phần tử của mảng hoặc sử dụng cả mảng. Chẳng hạn ta có thể viết `min(a[2], a[5])`; cho một lời gọi hàm trong đó `a` là mảng một chiều; hoặc `max(a)`; trong đó `a` là mảng nào đó đã khai báo trước.

Ví dụ 6.17. Cho mảng `a` có `n` phần tử. Yêu cầu in lên màn hình trung bình cộng các số chẵn của mảng.

Giải : Sau đây chúng ta sẽ sử dụng mảng `a` như là tham biến trị cho hàm `tbc()`. Trong hàm trung bình cộng chúng ta khai báo một mảng kiểu nguyên tên là `tg`, giả sử số lượng phần tử là `m`. Khi gọi chúng ta sẽ sử dụng mảng `a` thực sự với số lượng phần tử là `n`.

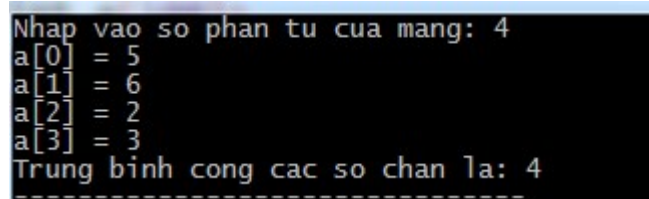
```
#include<iostream>
using namespace std;
int n;
int a[100];
void nhap()
{
```

```

int i;
cout<<"Nhap vao so phan tu = "; cin>>n;
for (i=0; i<n; i++)
{
    cout<<"a["<<i<<"] = ";
    cin>>a[i];
}
}
void tbc(int tg[], int m)
{
    int i, dem;
    double tong;
    dem=0;
    for (i=0;i<m; i++)
        if (tg[i] % 2 ==0)
        {
            dem=dem+1;
            tong = tong + (double) a[i];
        }
    if (dem != 0)
        cout<<"Trung binh cong cac so chan la: "<<tong/dem;
    else
        cout<<"Khong co so chan trong mang.";
}
int main()
{
    nhap();
    tbc(a,n);
    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Nhap vao so phan tu cua mang: 4
a[0] = 5
a[1] = 6
a[2] = 2
a[3] = 3
Trung binh cong cac so chan la: 4

```

Bài tập chương 6

Bài 1. Viết các biểu diễn cho phần tử thứ nhất, thứ ba và thứ bảy của các mảng sau đây:

a) `int grade[20];`

- b) double volts[10];
- c) double amps[16];
- d) double time[100];

Bài 2. Viết chương trình nhập vào các giá trị cho các mảng khai báo trong bài 1, sau đó in giá trị các mảng lên màn hình.

Bài 3. Liệt kê các phần tử của mảng sẽ được in giá trị của các câu lệnh sau:

- a)

```
for (i=0; i<=5; i++)
    cout<< a[i] <<" ";
```
- b)

```
for (k=0; k<=5; k = k + 2)
    cout<< a[k] <<" ";
```
- c)

```
for (j=0; j<=10; j= j +3)
    cout<< a[j] <<" ";
```
- d)

```
for (l=0; l<13; l = l + 2)
    cout<< a[l] <<" ";
```

Bài 4. Cho mảng a có n phần tử là những số nguyên. Yêu cầu nhập mảng a vào từ bàn phím sau đó in lên màn hình giá trị lớn nhất và nhỏ nhất của mảng.

Bài 5. Viết chương trình nhập mảng n phần tử số nguyên, tìm phần tử lớn nhất của mảng.

Bài 6. Viết chương trình nhập n phần tử số nguyên. Nhập phần tử cần tìm kiếm X. Nếu trong n phần tử đã nhập có X thì báo "tim thay", "so lan tim thay" và "cac vi tri tim thay", ngược lại báo "khong tim thay".

Bài 7. Cho dãy số a_1, a_2, \dots, a_n là các số nguyên. Yêu cầu viết chương trình thực hiện:

- a) Nhập dãy số trên từ bàn phím
- b) Tìm dãy con liên tiếp dài nhất tạo thành cấp số cộng của dãy số trên.
- c) Tìm dãy con liên tiếp dài nhất tạo thành cấp số nhân của dãy số trên

Bài 8. Cho dãy số a_1, a_2, \dots, a_n là các số nguyên. Yêu cầu viết chương trình thực hiện:

- a) Nhập dãy số trên từ bàn phím
- b) Sắp xếp dãy số trên thành một dãy số có dạng sóng. Một dãy số có dạng sóng a_1, a_2, \dots, a_n nếu $(a_i - a_{i-1}) * (a_i - a_{i+1}) \geq 0, \forall i = 2, \dots, n-1$.

Bài 9. Để xác định tính ngẫu nhiên của 1 dãy số, người ta phát hiện ra các đoạn lặp trên dãy số đó. Cho dãy n số a_1, a_2, \dots, a_n , đoạn lặp có độ dài k được định nghĩa như sau:

$$a[i] = a[j]$$

$$a[i+1] = a[j+1]$$

$$a[i+k] = a[j+k], (j+k \leq n)$$

Hãy tìm đoạn lặp có độ dài lớn nhất và in lên màn hình đoạn con đó.

Bài 10. Cho dãy số a_1, a_2, \dots, a_n là các số nguyên. Yêu cầu viết chương trình thực hiện:

- a) Nhập dãy số trên từ bàn phím
- b) In lên màn hình các số nguyên tố của dãy.

Bài 11. Cho dãy số a_1, a_2, \dots, a_n là các số nguyên. Yêu cầu viết chương trình thực hiện:

- a) Nhập dãy số trên từ bàn phím
- b) In lên màn hình các số chính phương của dãy trên.

Bài 12. Cho dãy số a_1, a_2, \dots, a_n là các số thực. Yêu cầu viết chương trình thực hiện:

- a) Nhập dãy số trên từ bàn phím
- b) In lên màn hình các bộ ba số có tổng lớn hơn 100.

Bài 13. Cho dãy số a_1, a_2, \dots, a_n là các số nguyên. Yêu cầu viết chương trình thực hiện:

- a) Nhập dãy số trên từ bàn phím

Sắp xếp dãy số trên theo thứ tự giá trị tuyệt đối tăng dần rồi in kết quả lên màn hình

Bài 14. Viết chương trình nhập vào ma trận A có kích thước $m \times n$, hãy tính:

- a) Tính tổng các phần tử âm, dương của ma trận.
- b) Tính tổng các phần tử chẵn, lẻ của ma trận.
- c) Tìm hàng có tổng các phần tử lớn nhất.
- d) Tìm cột có tổng các phần tử lớn nhất.

Bài 15. Viết chương trình nhập vào mảng a kích thước $m \times n$ bao gồm các số nguyên. Yêu cầu:

- a) In lên màn hình phần tử lớn nhất và nhỏ nhất của mảng.
- b) Tính tổng trung bình các phần tử của mảng
- c) Tính và in lên màn hình tổng các cột của mảng
- d) Liệt kê các phần tử là các số chính phương của mảng.

Bài 16. Cho mảng a có 3 dòng và 5 cột trong đó các phần tử là các giá trị điện áp kiểm tra của một bộ khuếch đại. Viết chương trình nhập vào các giá trị điện áp cho mảng, sau đó

in lên màn hình số lượng các giá trị điện áp nhỏ hơn 60; nằm trong khoảng [60, 70); nằm trong khoảng [70, 80); nằm trong khoảng [80, 90), và lớn hơn hoặc bằng 90.

Bài 17. Viết chương trình nhập mảng a một chiều vào từ bàn phím. Sắp xếp giá trị các phần tử của mảng theo thứ tự tăng dần bằng phương pháp *bubble sort*.

Bài 18. Viết chương trình nhập mảng a một chiều vào từ bàn phím. Sắp xếp giá trị các phần tử của mảng theo thứ tự tăng dần bằng phương pháp *quick sort*.

Bài 19. Viết chương trình nhập mảng a một chiều vào từ bàn phím. Sắp xếp giá trị các phần tử của mảng theo thứ tự tăng dần bằng phương pháp *heap sort*.

Bài 20. Viết chương trình với hai hàm có tên là *calvavg()* và *variance()*. Hàm *calvavg* tính và trả về giá trị trung bình của của mảng *testvalue* (mảng nhập từ bàn phím có 20 phần tử). Hàm *variance()* tính giá trị cho mảng *tg* với $tg[i] = testvalue[i] - tb$; trong đó *tb* là giá trị trung bình của mảng *testvalue*. Cả hai hàm trên đều sử dụng *testvalue* làm tham số đầu vào.

Bài 21. Viết chương trình nhập hai mảng a và b có cùng kích thước $m \times n$ gồm các số nguyên. Tính và in lên màn hình mảng c là tổng của hai mảng a và b trong đó: $c[i,j] = a[i,j] + b[i,j]$.

Bài 22. Cho một dãy n số đã sắp xếp, yêu cầu xóa số ở vị trí thứ k sao cho dãy còn lại vẫn giữ nguyên thứ tự.

Bài 23. Giả sử rằng các kí tự sau đây được lưu vào mảng: B, J, K, M, S, Z. Viết hàm *adlet()* nhận mảng trên và một kí tự bất kỳ nào đó làm tham số. Yêu cầu chèn kí tự tham số vào mảng sao cho vẫn giữ nguyên thứ tự như trên (kí tự có mã ASCII bé hơn thì đứng trước kí tự có mã ASCII lớn hơn).

Bài 24. Cho khai báo mảng sau:

```
int volts[500];
```

Viết hai khai dòng tiêu đề khai báo khác nhau cho hàm có tên *sortArray()* sao cho nó nhận mảng *volts* như là tham số được đặt tên là *inArray*.

Bài 25. Cho khai báo mảng sau:

```
double factors[256];
```

Viết hai khai dòng tiêu đề khai báo khác nhau cho hàm có tên findkey() sao cho nó nhận mảng factors như là tham số được đặt tên là select.

Bài 26. Viết chương trình sử dụng ba mảng một chiều volts, current, và resistance với số lượng lưu trữ tối đa là 100 phần tử. Nhập dữ liệu cho hai mảng volts và current sau đó tính giá trị các phần tử cho mảng resistance với $\text{resistance}[i] = \text{volts}[i] * \text{current}[i]$.

In mảng resistance lên màn hình.

Bài 27. Cho mảng hai chiều a có m dòng và n cột với các phần tử là số nguyên, yêu cầu:

- a) Nhập m, n và các phần tử mảng a từ bàn phím
- b) In lên màn hình giá trị lớn nhất của từng hàng
- c) In lên màn hình giá trị nhỏ nhất của từng cột

Bài 28. Cho mảng hai chiều a có m dòng và n cột với các phần tử là số nguyên, yêu cầu:

- a) Nhập m, n và các phần tử mảng a từ bàn phím
- b) In lên màn hình mảng con (2×2) của mảng a sao cho tổng 4 phần tử trong mảng đó là lớn nhất có thể.

Bài 29. Cho mảng vuông b kích thước n gồm các số nguyên, yêu cầu:

- a) Nhập n và mảng b vào từ bàn phím
- b) In lên màn hình tổng các phần tử nằm trên đường chéo chính và đường chéo phụ của mảng.

Bài 30. Trên trục tọa độ cho 2 điểm a, b và n đường thẳng đặc trưng bởi tọa độ đầu và cuối. Hãy lập trình tìm cách phủ kín đoạn $[a, b]$ sao cho số đoạn thẳng phải dùng là ít nhất.

Chương 7. Xâu kí tự

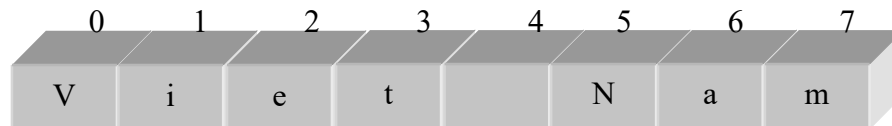
7.1 Khái niệm xâu và cách khai báo

7.1.1 Khái niệm xâu kí tự

Trong thực tế chúng ta gặp các bài toán cần xử lý một dãy các kí tự liên tiếp chẳng hạn như dãy kí tự biểu diễn văn bản, dãy kí tự biểu thị các kí hiệu,... Ngôn ngữ C++ cung cấp kiểu dữ liệu xâu kí tự để biểu diễn và xử lý dữ liệu dạng này. Xâu kí tự được hiểu là một dãy các kí tự của bảng mã ASCII. Ví dụ: xâu “Viet Nam”, xâu chứa các kí tự số “1245872626262”; xâu biểu diễn dãy nhị phân “0000110010000111” v.v. Dấu nháy kép báo hiệu điểm bắt đầu hoặc kết thúc của xâu không được coi là giá trị của xâu.

7.1.2 Khai báo xâu kí tự

Lớp xâu ký tự có một số phương thức để khai báo, tạo và khởi tạo xâu ký tự như trong bảng 7-1. Để sử dụng được các phương thức này cần sử dụng thư viện string. Với mỗi xâu kí tự sẽ bao gồm một số kí tự, mỗi kí tự của xâu sẽ được đặc trưng bằng một chỉ số nguyên trong đó kí tự đầu tiên có chỉ số là 0. Hình 7-1 mô tả xâu "Viet Nam" cùng với các chỉ số của các kí tự tương ứng.



Hình 7-1. Xâu "Viet Nam" cùng với các chỉ số của các kí tự của nó.

Phương thức khai báo	Mô tả	Ví dụ
<code>string Ten_Xau = Gia_tri;</code>	Tạo xâu ký tự và khởi tạo giá trị cho xâu đó. Giá trị khởi tạo có thể là một xâu ký tự, một đối tượng xâu đã được khai báo trước đó hoặc một biểu thức chứa xâu ký tự	<code>string str1 = "Viet Nam";</code> <code>string str2 = str1;</code> <code>string str3 = str1 + str2;</code>
<code>string Ten_Xau(Gia_tri);</code>	Giống phương thức <code>string Ten_Xau = Gia_tri</code>	<code>string str1("Viet");</code> <code>string str2(str1 + "Nam");</code>
<code>string Ten_Xau(str,</code>	Tạo và khởi tạo xâu con của	<code>string str2(str1, 4);</code>

<code>n);</code>	xâu str, từ vị trí đầu tiên của xâu str đi n ký tự	Nếu str1 chứa xâu “Viet Nam”, thì str2 chứa xâu “Viet”
<code>string Ten_Xau(str, n, p);</code>	Tạo và khởi tạo xâu con của xâu str, bắt đầu tại vị trí n của str và chứa p ký tự	<code>string str2(str1, 5, 2)</code> Nếu str2 chứa xâu “Viet Nam”, thì str1 là Na
<code>string Ten_Xau(n, char);</code>	Tạo và khởi tạo xâu gồm n ký tự char	<code>string str1(5, '*')</code> Tạo ra: str1 = “*****”
<code>string Ten_Xau;</code>	Tạo và khởi tạo một xâu rỗng (giống như <code>string Ten_Xau = ""</code>), độ dài xâu này là 0.	<code>string message;</code>

Bảng 7-1 Các phương khai báo xâu kí tự

Ví dụ 7.1. Viết chương trình sử dụng một số phương thức trong bảng 7-1 để tạo xâu và in kết quả ra màn hình.

Giải :

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str1;
    string str2("Viet Nam");
    string str3 = "dat nuoc";
    string str4(str3);
    string str5(str4, 4);
    string str6 = "anh hung";
    string str7(str6, 4, 4);
    string str8=str2+" "+str4+" "+str6;
    string str9(26, '*');
    cout << "str1 = " << str1 << endl;
    cout << "str2 = " << str2 << endl;
    cout << "str3 = " << str3 << endl;
    cout << "str4 = " << str4 << endl;
    cout << "str5 = " << str5 << endl;
    cout << "str6 = " << str6 << endl;
```

```

cout << "str7 = " << str7 << endl;
cout << "str8 = " << str8 << endl;
cout << "str9 = " << str9 << endl;
str1="Que huong toi!";
cout << "str1 = " << str1 << endl;
return 0;
}

```

Kết quả thực hiện chương trình như sau:

```

str1 =
str2 = Viet Nam
str3 = dat nuoc
str4 = dat nuoc
str5 = nuoc
str6 = anh hung
str7 = hung
str8 = Viet Nam dat nuoc anh hung
str9 = *****
str1 = Que huong toi!
-----

```

7.2 Nhập và xuất chuỗi ký tự

Có thể tạo chuỗi bằng các phương thức như trong bảng 7-1, bên cạnh đó, chúng ta còn có thể nhập chuỗi vào từ bàn phím và hiển thị chuỗi ra màn hình như trong bảng 7-2.

Phương thức nhập/xuất chuỗi	Mô tả
cout	Đối tượng cout được dùng để xuất dữ liệu ra màn hình
cin	Đối tượng cin dùng để nhập dữ liệu vào từ bàn phím, dừng việc đọc chuỗi vào khi gặp khoảng trắng hoặc ký tự xuống dòng (\n)
getline(cin, s)	Hàm getline() dùng để nhập dữ liệu vào từ bàn phím. Dữ liệu vào được lưu vào chuỗi s, dừng việc nhận ký tự khi gặp ký tự xuống dòng (\n).

Bảng 7-2 Các phương thức nhập/xuất chuỗi ký tự (yêu cầu sử dụng thư viện string)

Tuy đối tượng cin có thể dùng để nhập chuỗi nhưng hiếm khi chúng ta sử dụng do chuỗi sẽ kết thúc khi gặp ký tự dấu cách. Ví dụ 7.2 mô tả điều này.

Ví dụ 7.2 Viết chương trình sử dụng lệnh cin để nhập dữ liệu cho chuỗi s1 và hiển thị nội dung chuỗi s1 ra màn hình.

Giải:

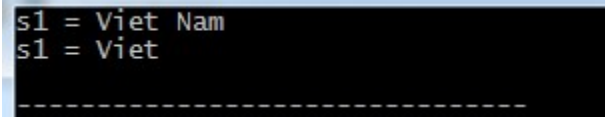
```

#include <iostream>
#include<string>

```

```
using namespace std;
int main()
{
    string s1;
    cout<<"s1 = "; cin>>s1;
    cout<<"s1 = "<<s1<<endl;
    return 0;
}
```

Ví dụ về kết quả thực hiện chương trình như sau:



```
s1 = Việt Nam
s1 = Việt
-----
```

Như vậy, chuỗi s1 được nhập giá trị là "Việt Nam" nhưng do sau "Việt" có dấu cách nên s1 chỉ nhận được giá trị là "Việt".

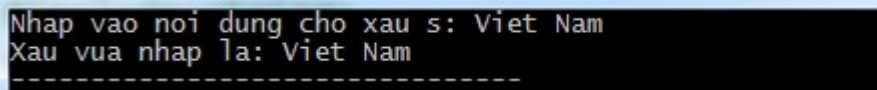
Hàm getline() thuộc lớp string nhận và lưu các ký tự trong bộ đệm bàn phím cho tới khi gặp ký tự xuống dòng (\n hoặc Enter). Để hiểu rõ hơn về phương thức nhập/xuất chuỗi ta xem ví dụ 7.3

Ví dụ 7.3. Nhập chuỗi s từ bàn phím, in lên màn hình chuỗi vừa nhập.

Giải:

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string s;
    cout<<"Nhập vào nội dung cho chuỗi s: ";
    getline(cin, s);
    cout<<"Chuỗi vừa nhập là: "<<s;
    return 0;
}
```

Ví dụ về kết quả thực hiện chương trình như sau:



```
Nhập vào nội dung cho chuỗi s: Việt Nam
Chuỗi vừa nhập là: Việt Nam
-----
```

Chú ý rằng dòng thông báo “Chuỗi vừa nhập là: ” cũng chính là một chuỗi ký tự.

Mặc định hàm `getline()` nhận ký tự kết thúc chuỗi là ký tự xuống dòng, tuy nhiên ta có thể thiết lập ký tự kết thúc chuỗi cho nó bằng cú pháp sau :

```
getline(cin, s, ky_tu_ket_thuc_xau)
```

Ví dụ 7.4. Viết chương trình nhập vào một chuỗi ký tự từ bàn phím. Yêu cầu thiết lập giá trị ‘&’ là giá trị kết thúc chuỗi. Hiển thị chuỗi vừa nhập ra màn hình

Giải :

```
#include <iostream>
#include<string>
using namespace std;
int main()
{
    string s1;
    cout<<"s1 = "; getline(cin, s1, '&');
    cout<<"s1 = "<<s1<<endl;
    return 0;
}
```

Như vậy, khi nhập dữ liệu chúng ta nhập vào chuỗi "Viet Nam&" thì chuỗi s1 chỉ là "Viet Nam", ký tự & ở đây không thuộc chuỗi, nó được coi là ký tự kết thúc chuỗi.

Do hàm `getline()` nhận ký tự từ bộ đệm bàn phím nên nếu trước khi có lệnh gọi hàm `getline(cin, s)` còn có dữ liệu nằm trong bộ đệm bàn phím thì dữ liệu này sẽ được đưa vào chuỗi s. Ví dụ trước khi sử dụng lệnh gọi hàm `getline()` ta sử dụng lệnh `cin` để nhập dữ liệu cho một biến kiểu nguyên. Sau khi nhập dữ liệu cho biến kiểu nguyên, ta nhập Enter (‘\n’) để kết thúc việc nhập biến nguyên, ký tự ‘\n’ này sẽ nằm lưu lại trong bộ đệm bàn phím nên hàm `getline()` sẽ nhận ngay ‘\n’ cho biến s1 và nó hiểu rằng đã kết thúc chuỗi. Do đó, chuỗi được tạo ra là một chuỗi rỗng. Ví dụ 7.5 mô tả điều này.

Ví dụ 7.5a. Viết chương trình nhập vào một biến a kiểu nguyên, tiếp theo nhập vào một chuỗi s. Hiển thị giá trị của a và s ra màn hình.

Giải :

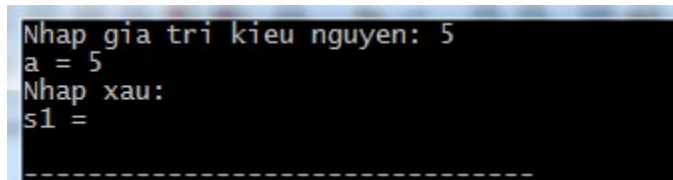
```
#include <iostream>
#include<string>
using namespace std;
int main()
{
    int a;
```

```

    string s1;
    cout<<"Nhập giá trị kiểu nguyên: "; cin>>a;
    cout<<"a = "<<a<<endl;
    cout<<"Nhập xâu: "<<endl;
    getline(cin, s1);
    cout<<"s1 = "<<s1<<endl;
    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Nhập giá trị kiểu nguyên: 5
a = 5
Nhập xâu:
s1 =
-----

```

Để khắc phục lỗi này ta sử dụng thêm lệnh `cin.ignore()` vào trước lệnh `getline(cin, s1);` như trong ví dụ 7.5b :

Ví dụ 7.5b

```

#include <iostream>
#include<string>
using namespace std;
int main()
{
    int a;
    string s1;
    cout<<"Nhập giá trị kiểu nguyên: "; cin>>a;
    cout<<"a = "<<a<<endl;
    cin.ignore();
    cout<<"Nhập xâu: "<<endl;
    getline(cin, s1);
    cout<<"s1 = "<<s1<<endl;
    return 0;
}

```

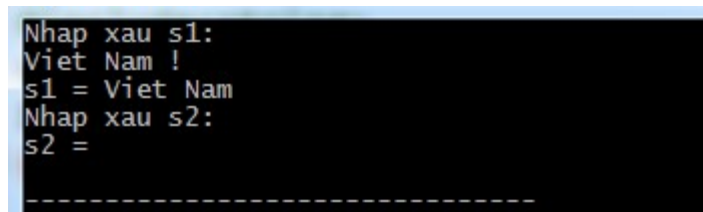
Trong trường hợp ta sử dụng hai lệnh gọi hàm `getline(cin, s1)` để nhập dữ liệu thì lỗi trên sẽ không xảy ra do ký tự ‘\n’ của lệnh `getline(cin, s1)` được coi là ký tự kết thúc xâu s1, nó không còn nằm trong bộ đệm bàn phím. Tuy nhiên, sẽ có vấn đề nếu chúng ta thiết lập ký tự kết thúc xâu cho s1 bằng lệnh `getline(cin, s1, ‘!’)`.

Khi đó s1 sẽ nhận '!' là ký tự kết thúc chuỗi, '\n' sẽ nằm trong bộ đệm bàn phím và được truyền cho s2.

Ví dụ 7.6a

```
#include <iostream>
#include<string>
using namespace std;
int main()
{
    string s1,s2;
    cout<<"Nhập vào s1: "<<endl;
    getline(cin, s1,'!');
    cout<<"s1 = "<<s1<<endl;
    cout<<"Nhập vào s2: "<<endl;
    getline(cin, s2);
    cout<<"s2 = "<<s2<<endl;
    return 0;
}
```

Ví dụ về kết quả thực hiện chương trình như sau:



```
Nhập vào s1:
Việt Nam !
s1 = Việt Nam
Nhập vào s2:
s2 =
-----
```

Để khắc phục lỗi này ta thêm `cin.ignore();` vào trước lệnh `getline()` thứ hai. Ví dụ 7.6a được viết lại như sau:

Ví dụ 7.6b

```
#include <iostream>
#include<string>
using namespace std;
int main()
{
    string s1,s2;
    cout<<"Nhập vào s1: "<<endl;
    getline(cin, s1,'!');
    cout<<"s1 = "<<s1<<endl;
    cout<<"Nhập vào s2: "<<endl;
    cin.ignore();
    getline(cin, s2);
}
```

```

    cout<<"s2 = "<<s2<<endl;
    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:

```

Nhap xau s1:
Viet Nam!
s1 = Viet Nam
Nhap xau s2:
Que huong toi
s2 = Que huong toi
-----

```

7.3 Một số hàm sử dụng trên chuỗi ký tự

Khi làm việc với dữ liệu kiểu chuỗi ký tự, C++ cung cấp một số hàm sau đây để phục vụ cho việc giải quyết các bài toán với chuỗi.

Tên hàm	Ý nghĩa	Ví dụ
<code>length()</code>	Trả về độ dài của chuỗi ký tự	<code>string1.length()</code>
<code>size()</code>	Trả về độ dài chuỗi giống như <code>length()</code>	<code>string1.size()</code>
<code>at(int chỉ_so)</code>	Trả về ký tự tại vị trí thứ chỉ số của chuỗi	<code>string1.at(5)</code>
<code>compare(str)</code>	So sánh chuỗi cần so sánh với chuỗi <code>str</code> , trả lại giá trị âm nếu chuỗi so sánh nhỏ hơn chuỗi <code>str</code> , bằng 0 nếu chúng bằng nhau, và giá trị dương nếu chuỗi so sánh lớn hơn chuỗi <code>str</code>	<code>string1.compare(string2)</code>
<code>empty()</code>	Kiểm tra xem một chuỗi có là chuỗi rỗng	<code>string1.empty()</code>

	hay không	
<code>erase(ind)</code>	Xóa các kí tự từ vị trí <code>ind</code> đến hết xâu	<code>string1.erase(5)</code>
<code>erase(ind, m)</code>	Xóa các kí tự của xâu từ vị trí <code>ind</code> đến kí tự	<code>string1.erase(3, 5)</code>
<code>find(str)</code>	Trả về giá trị đầu tiên của xâu <i>str</i> xuất hiện trong xâu cần tìm	<code>string1.find("phu tho")</code>
<code>find(str, ind)</code>	Trả về giá trị đầu tiên xuất hiện trong xâu cần tìm tính từ vị trí thứ <code>ind</code>	<code>string1.find("ha", 7)</code>
<code>find_first_of(str, ind)</code>	Trả về vị trí đầu tiên xuất hiện của kí tự bất kỳ trong <i>str</i> trong xâu cần tìm, bắt đầu tính từ chỉ số <code>ind</code>	<code>string1.find_first_of("ha", 7)</code>
<code>find_first_not_of(str, ind)</code>	Trả về chỉ số đầu tiên của kí tự không xuất hiện trong <i>str</i> .	<code>string1.find_first_not_of("ha", 7)</code>
<code>insert(ind, str)</code>	Chèn xâu <i>str</i> vào xâu từ vị trí <code>str</code>	<code>string1.insert("ha", 3)</code>
<code>replace(ind, n, str)</code>	Xóa <code>n</code> kí tự từ vị trí <code>ind</code> và chèn xâu <i>str</i> vào xâu bắt đầu từ vị trí <code>ind</code>	<code>string1.replace(3, 7, "ha")</code>
<code>substr(ind, n)</code>	Trả lại một xâu bao	<code>string2 =</code>

	gồm n ký tự lấy ra từ đầu tiên bắt đầu từ chỉ số ind, nếu n lớn hơn số ký tự còn lại thì ký tự kết thúc chuỗi được sử dụng.	string1.substr(0,10)
swap(str)	Hoán đổi nội dung hai chuỗi	string1.swap(string2); hoặc swap(string1, string2);
[ind]	Trả về ký tự có chỉ số ind	string1[5]
=	Phép gán	S1 = "Ha Noi"
+	Phép cộng chuỗi	S1 = S2 + S3
==; !=; <; <=; >; >=	Phép so sánh chuỗi	S1 == S2

Bảng 7-3 Một số hàm xử lý chuỗi ký tự

Ví dụ 7.7 Nhập hai chuỗi s1 và s2 từ bàn phím, chuyển đổi nội dung hai chuỗi s1 và s2 cho nhau và in kết quả lên màn hình.

Giải:

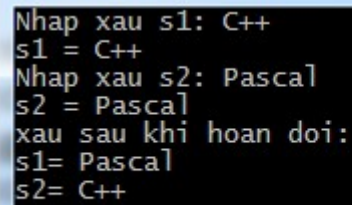
```
#include <iostream>
#include<string>
using namespace std;
int main()
{
    string s1,s2;
    cout<<"Nhap xau s1: ";
    getline(cin, s1);
    cout<<"s1 = "<<s1<<endl;
    cout<<"Nhap xau s2: ";
    getline(cin, s2);
    cout<<"s2 = "<<s2<<endl;
    s1.swap(s2); // hoac dung swap(s1,s2);
    cout<<"xau sau khi hoan doi:"<<endl;
    cout<<"s1= "<<s1<<endl;
    cout<<"s2= "<<s2<<endl;
}
```

```

    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Nhap xau s1: C++
s1 = C++
Nhap xau s2: Pascal
s2 = Pascal
xau sau khi hoan doi:
s1= Pascal
s2= C++
-----

```

Ví dụ 7.8. Ví dụ về một số phép so sánh với chuỗi ký tự.

```

#include <iostream>
#include <string>
using namespace std;
int main()
{
    string s1 = "Lap trinh C++";
    string s2 = "Lap trinh Pascal";
    cout << "s1= " << s1 << endl;
    cout << "Do dai xau 1 la: " << int(s1.length())<< endl;
    cout << "s2= " << s2 << endl;
    cout << "Do dai xau 2 la: " << int(s2.length())<< endl;
    if (s1 < s2)
        cout << s1 << " nho hon " << s2 << endl;
    else if (s1 == s2)
        cout << s1 << " bang " << s2 << endl << endl;
    else
        cout << s1 << " lon hon " << s2 << endl << endl;
    s1 = s1 + s2;
    cout << "Sau khi cong xau, xau 1 la:      \n" << s1 << endl;
    cout << "Do dai xau 1 la: " << int(s1.length()) << endl;
    return 0;
}

```

Ví dụ 7.9: Sử dụng hàm `replace()` và `insert()` để thực hiện thay thế và chèn ký tự trong chuỗi.

Giải:

```

#include <iostream>
#include <string>
using namespace std;

```

```

int main()
{
    string s = "Lap trinh C++";
    cout << "Xau ban dau: " << s<<endl;
    cout<<"Do dai xau: " << int(s.length())<< endl;
    // chen ky tu
    s.insert(10,"bac cao ");
    cout << "Xau sau khi chen: " << s<<endl;
    cout << "Do dai: " << int(s.length())<< endl;
    // thay the ky tu
    s.replace(10, 7, "huong doi tuong");
    cout << "Xau sau khi thay the la: " << s<<endl;
    cout << "Do dai " << int(s.length())<< endl;
    return 0;
}

```

Ví dụ 7.10. Viết chương trình nhập một xâu kí tự s từ bàn phím. Kiểm tra xem trong xâu kí tự s có chứa kí tự số không? In lên các vị trí đó.

Giải:

```

#include<iostream>
#include<string>
using namespace std;
int main()
{
    string s; int i;
    cout<<"Nhap vao noi dung cho xau s: ";
    getline(cin, s);
    for (i=0 ; i<s.length() ;i++)
        if ((s.at(i)>='0') &&(s.at(i)<='9') )
            cout<<"Ki tu so o vi tri: "<<i<<"\n";
    return 0;
}

```

Ví dụ 7.11. Viết chương trình nhập xâu kí tự bất kỳ. In lên màn hình xâu đó có bao nhiêu từ.

Giải: Từ được định nghĩa bao gồm một dãy kí tự liên tiếp không chứa kí tự trắng và giữa các từ được ngăn cách bởi kí tự trắng. Để đếm số từ ta sẽ đi đếm các vị trí xuất hiện từ đó là kí tự thứ i là trắng và kí tự thứ i+1 khác trắng. Chương trình như sau:

```

#include<iostream>

```



```

#include<string>
using namespace std;
int main()
{
    string s; int i, dem;
    cout<<"Nhap vao noi dung cho xau s: ";
    getline(cin, s);
    s= ' '+s ;
    dem =0 ;
    for (i=0 ; i<s.length() ;i++)
        if ((s.at(i) !=' ') &&(s.at(i-1)==' '))
            dem = dem +1 ;
    cout<<"So tu trong xau la: "<<dem;
    return 0;
}

```

Ví dụ 7.12. Cho xâu kí tự s chỉ gồm các kí tự U và V, viết chương trình nén xâu s. Nếu có từ 3 kí tự U liên nhau trở lên thì thay bằng số lượng kí tự U (bằng số) và theo sau bởi U. Tương tự như vậy với kí tự V.

Ví dụ: UUUUUVVVVVVVUUUUUUU -> 5UVVU6V7U

Giải: Ý tưởng cơ bản của thuật toán là duyệt xâu từ kí tự đầu đến kí tự cuối và đếm các kí tự liên tiếp trùng nhau. Chương trình như sau:

```

#include<iostream>
#include<string>
#include <sstream>
using namespace std;
int main()
{
    string s, s1; int i, dem;
    cout<<"Nhap vao noi dung cho xau s: ";
    getline(cin, s);
    dem =0 ;
    s1=' ' ;
    i=0;
    while (i<= s.length()-2)
    {
        dem=1;
        while((i<=s.length()-2) && (s.at(i) == s.at(i+1)))
        {
            i=i+1 ;

```

```

        dem=dem+1 ;
    }
    if (dem<=2)
        s1=s1 +s.at(i) + s.at(i) ;
    else
    {
        ostringstream convert;
        convert<<dem;
        s1=s1 + convert.str() +s.at(i) ;
    }
    i=i+1;
}
cout<<"Xau moi la : "<<s1<<endl;
system("PAUSE");//Lenh dung man hinh
return 0;
}

```

Ví dụ 7.13. Viết chương trình nhập vào chuỗi ký tự s. Hãy thay thế toàn bộ ký tự e của chuỗi s bằng ký tự x rồi in kết quả lên màn hình.

Giải:

```

#include<iostream>
#include<string>
using namespace std;
int main()
{
    string s; int i, dem;
    cout<<"Nhap vao noi dung cho xau s: ";
    getline(cin, s);
    for (i=0 ; i<s.length() ;i++)
        if (s.at(i) =='e')
            s.at(i)='x' ;
    cout<<"Xau thu duoc la: "<<s;
    return 0;
}

```

Ví dụ 7.14. Viết chương trình nhập vào một chuỗi ký tự từ bàn phím, hiện lên màn hình mỗi từ của chuỗi ký tự trên một dòng.

Giải:

```

#include<iostream>
#include<string>
#include <sstream>

```

```

using namespace std;
int main()
{
    string s, s1; int i, dem;
    cout<<"Nhap vao noi dung cho xau s: ";
    getline(cin, s);
    dem =0 ;
    s1="" ;
    s=s+' ' ;
    while (s.length()>1)
    {
        i=0;
        while ((i<s.length()) && (s.at(i)==' '))
            i=i+1;
        if (i<s.length())
        {
            s= s.substr(i, s.length()-i);
            int tg = s.find(" ");
            s1=s.substr(0,tg );
            cout<<"Tu cua xau s la : "<<s1<<"\n";
            s= s.substr(tg, s.length()-tg);
        }
        else
            s=' ';
    }
    return 0;
}

```

Ví dụ 7.15. Nhập xâu ký tự từ bàn phím, đếm xem trong xâu ký tự nào xuất hiện nhiều nhất.

Giải :

```

#include<iostream>
#include<string>
#include <sstream>
using namespace std;
int main()
{
    string s, s1; int i,j, dem, max; char u;
    int a[1000];
    cout<<"Nhap vao noi dung cho xau s: ";
    getline(cin, s);

```

```

dem =0 ;
for (i=0;i<=s.length();i++)
    a[i]=0;
max=0;
for (i=0;i<s.length()-1;i++)
    if (a[i]==0)
    {
        a[i]=1;
        dem=1;
        for (j=i+1;j<s.length();j++)
            if (s.at(j)==s.at(i))
            {
                dem++;
                a[j]=1;
            }
        if (dem>max)
        {
            max=dem;
            u=s.at(i);
        }
    }
    cout<<"Kí tự xuất hiện nhiều nhất là : "<<u<< " với
"<<max<<" lần xuất hiện.""\n";
    return 0;
}

```

Ví dụ 7.16. Cho chuỗi ký tự chỉ bao gồm ba loại ký tự là X, T, D. Yêu cầu chuyển các ký tự T về đầu, ký tự D về cuối của chuỗi, ký tự X ở giữa sao cho số lần đổi chỗ là ít nhất.

Giải: Giả sử số ký tự T, X và D là dt , dx và dd khi đó chuỗi sau khi xếp lại sẽ có dt ký tự X ở đoạn đầu tiên (đoạn của X), dx ký tự T ở đoạn giữa (đoạn của T) và dd ký tự D ở đoạn cuối của chuỗi (đoạn của D). Quá trình đổi chỗ như sau:

- Nếu có ký tự T ở đoạn của X và có ký tự X ở đoạn của T thì sẽ đổi chỗ hai ký tự này cho nhau.
- Nếu có ký tự T ở đoạn của D và có ký tự D ở đoạn của T thì sẽ đổi chỗ hai ký tự này cho nhau.
- Nếu có ký tự T ở đoạn của X và có ký tự X ở đoạn của T thì sẽ đổi chỗ hai ký tự này cho nhau.

- Nếu có kí tự T ở đoạn D, kí tự D ở đoạn của X và kí tự X ở đoạn của T thì sẽ đổi chỗ hoán vị 3 kí tự này cho nhau.

Chương trình như sau:

```
#include<iostream>
#include<string>
#include <sstream>
using namespace std;
int main()
{
    string s, s1; int i,j, dt, dx, dd, dem; char u;
    int a[1000];
    cout<<"Nhap vao noi dung cho xau s: ";
    getline(cin, s);
    dt=0; dx=0; dd=0; dem=0;
    for (i=0;i<s.length();i++)
        if (s.at(i)=='T')
            dt=dt+1;
        else
            if (s.at(i)=='X')
                dx=dx+1;
            else
                if (s.at(i)=='D')
                    dd=dd+1;
    for (i=0; i<dt;i++)
    if (s.at(i)=='X')
    {
        int flag=0;
        for (j=dt; j<dt+dx; j++)
            if (s.at(j) =='T')
            {
                char tg=s.at(i); s.at(i)=s.at(j);
                s.at(j)=tg;
                dem = dem+1;flag=1;
                break;
            }
        if (flag==0)
            for (j=dt+dx; j<dt+dx+dd; j++)
                if (s.at(j) =='T')
                {
                    char tg=s.at(i);    s.at(i)=s.at(j);
```

```

        s.at(j)=tg;
        dem =dem +1;
        break;
    }
}
else
    if (s.at(i)=='D')
    {
        int flag=0;
        for (j=dt+dx; j<dt+dx+dd; j++)
            if (s.at(j) =='T')
            {
                char tg=s.at(i); s.at(i)=s.at(j);
                s.at(j)=tg;
                dem =dem+1; flag=1;
                break;
            }
        if (flag==0)
            for (j=dt; j<dt+dx; j++)
                if (s.at(j) =='T')
                {
                    char tg=s.at(i);
                    s.at(i)=s.at(j);
                    s.at(j)=tg;
                    dem=dem+1; flag=1;
                    break;
                }
    }
for (i=dt; i<dt+dx;i++)
    if (s.at(i)=='D')
    {
        for (j=dt+dx; j<dt+dx+dd; j++)
            if (s.at(j) =='X')
            {
                char tg=s.at(i); s.at(i)=s.at(j);
                s.at(j)=tg; dem=dem+1;
                break;
            }
    }
cout<<"Xau sau khi doi la: "<<s<<" voi so lan doi la:
"<<dem<<"\n";

```

```
    return 0;  
}
```

Bài tập chương 7

Bài 1. Xác định giá trị `s.at(0)`, `s.at(1)`, và `s.at(5)` của các xâu sau đây:

- Hoa học
- Tiếng Anh
- Lập trình C++
- Địa lý

Bài 2. Sử dụng `at()`, viết chương trình đọc xâu kí tự sử dụng `getline()` sau đó hiển thị xâu ngược lại với xâu vừa nhập.

Bài 3. Nhập một xâu kí tự từ bàn phím đếm xem kí tự `e` xuất hiện bao nhiêu lần trong xâu đó.

Bài 4. Viết chương trình sinh ngẫu nhiên một số nguyên từ 0 đến 129. In lên màn hình kí tự tương ứng với mã ASCII của nó đồng thời thông báo là chữ thường, chữ hoa, kí tự số, hoặc kí tự đặc biệt.

Bài 5. Viết chương trình nhập vào một xâu kí tự, in lên màn hình mã của kí tự đó trong hệ Hexa-decimal (hệ 16).

Bài 6. Viết chương trình nhập vào xâu kí tự đến khi gặp dấu chấm biểu thị hết câu. Chuẩn hóa xâu đó theo quy tắc của kí tự viết hoa.

Bài 7. Cho hai xâu kí tự chứa toàn kí tự số biểu thị cho hai số nguyên. Yêu cầu tính tổng hai số nguyên đó rồi in lên màn hình.

Bài 8. Viết hàm tính độ dài xâu (không sử dụng thư viện). Viết chương trình nhập xâu từ bàn phím, sử dụng hàm vừa xây dựng đưa ra độ dài xâu.

Bài 9. Nhập xâu ký tự từ bàn phím, chuẩn hoá xâu đó:

- Loại bỏ khoảng trắng bên trái xâu.
- Loại bỏ khoảng trắng bên phải xâu.
- Loại bỏ các khoảng trắng thừa giữa các từ trong xâu.

Bài 10. Viết chương trình nhập xâu ký tự từ bàn phím, hãy chuyển các ký tự đầu mỗi từ thành ký tự thường.

Bài 11. Viết chương trình sử dụng các hàm tìm kiếm trong chuỗi ký tự.

Bài 12. Viết chương trình sử dụng các hàm chèn và thay thế trong chuỗi ký tự.

Bài 13. Viết chương trình sử dụng các toán tử so sánh trong chuỗi ký tự.

Bài 14. Cho hai chuỗi ký tự s_1 và s_2 , hãy tìm chuỗi con chung lớn nhất của hai chuỗi trên.

Bài 15. Cho chuỗi ký tự s chỉ gồm các ký tự số có độ dài n . Viết chương trình xóa đi k ký tự từ chuỗi s ($k < n$) sao cho chuỗi còn lại biểu thị số lớn nhất có thể.

Bài 16. Cho số nguyên a gồm n con số, yêu cầu xóa đi k con số trong số a để số còn lại có giá trị lớn nhất có thể.

Bài 17. Cho chuỗi ký tự S_1 và S_2 hãy tìm chuỗi con chung lớn nhất giữa hai chuỗi đó.

Bài 18. Cho chuỗi ký tự S và chuỗi ký tự B , hãy kiểm tra xem chuỗi B có xuất hiện trong chuỗi S không? Nếu có tính số lần xuất hiện của nó.

Bài 19. Cho hai chuỗi ký tự chỉ chứa các số biểu diễn hai số nguyên, hãy thực hiện việc cộng hai số nguyên đó và in kết quả lên màn hình.

Bài 20. Cho hai chuỗi ký tự chỉ chứa các số biểu diễn hai số nguyên, hãy thực hiện việc nhân hai số nguyên đó và in kết quả lên màn hình.

Bài 21. Cho chuỗi ký tự S có độ dài tối đa 200 ký tự, hãy mã hóa chuỗi đó theo các bước sau:

- Đọc lần lượt các ký tự của chuỗi và điền vào mảng 15×15 từ trái sang phải, từ trên xuống dưới.
- Đọc lần lượt các ký tự từ mảng ra theo cột từ trên xuống dưới, từ trái sang phải.
- In kết quả của chuỗi mã hóa lên màn hình.

Bài 22. Cho chuỗi ký tự biểu diễn một chuỗi hạt. Yêu cầu tìm vị trí cắt để duỗi thẳng chuỗi hạt ra sao cho tổng các ký tự giống nhau ở hai đầu là lớn nhất.

Ví dụ: Chuỗi hạt cho như sau: AATDDDDXXXTTTTTTTTTTDDDDDDDX

Kết quả: DDDDDDDXAATDDDDXXXTTTTTTTTTT

Bài 23. Cho hai chuỗi ký tự s_1 và s_2 chỉ chứa các ký tự số biểu diễn hai số nguyên bất kỳ, hãy thực hiện việc chia lấy phần nguyên của s_1 cho s_2 và in kết quả lên màn hình.

Chương 8. Lập trình hướng đối tượng với C++

8.1 Giới thiệu

Trong các chương trước chúng ta đã tìm hiểu các chương trình sử dụng các hàm hoặc chỉ có hàm `main()` để giải quyết các bài toán đặt ra. Trong chương giới thiệu về hàm, chúng ta đã biết tác dụng và ưu điểm của việc sử dụng hàm. Việc sử dụng hàm để thiết kế viết các chương trình theo hướng mô đun hóa việc giải quyết bài toán. Trong chương này, chúng ta tìm hiểu một khái niệm hoàn toàn mới đó là khái niệm lập trình hướng đối tượng. Tư tưởng của việc lập trình hướng đối tượng tức là thiết kế các chương trình chứa lớp các đối tượng trong đó mỗi lớp đều có các dữ liệu và phương thức đi kèm với nó. Đây cũng là điểm khác nhau quan trọng giữa ngôn ngữ C++ và ngôn ngữ C. Chúng ta sẽ tìm hiểu vấn đề này trước tiên với việc hiểu cấu trúc của một lớp trong C++. Để khai báo một lớp ta viết như sau:

```
class <tên lớp>
{
    <khai báo các thành phần dữ liệu>;
    <khai báo các hàm dạng nguyên mẫu>
};
<nội dung chi tiết các hàm>
```

Trong khai báo trên các dữ liệu thành phần và các hàm thành phần gọi là các thành viên của lớp. Như vậy mỗi lớp bao gồm dữ liệu và các hàm đi kèm để xử lý các dữ liệu đó.

Ví dụ 8.1. Khai báo và truyền tham số cho các biến trong lớp.

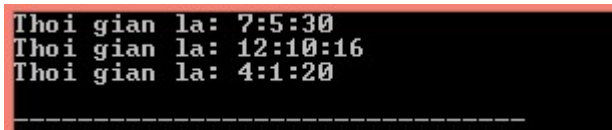
```
#include<iostream>
#include<iomanip>
using namespace std;
class Thoigian
{
    private:
        int gio;
        int phut;
        int giay;
    public:
        Thoigian(int = 7, int = 5, int = 30);
```

```

        void Datthoigian(int, int, int);
        void Hienthoigian();
};
Thoigian::Thoigian(int g, int p, int gi)
{
    gio=g; phut=p; giay=gi;
}
void:: Thoigian::Datthoigian(int g, int p, int gi)
{
    gio=g; phut=p; giay=gi; return;
}
void Thoigian::Hienthoigian()
{
    cout<<"Thoi gian la: ";
    cout<<gio<<": "<<phut<<": "<<giay<<endl;
    return;
}
int main() {
    Thoigian a,b,c(4,1,20);
    b.Datthoigian(12,10,16);
    a.Hienthoigian();
    b.Hienthoigian();
    c.Hienthoigian();
    return 0;
}

```

Kết quả thực hiện chương trình như sau:



```

Thoi gian la: 7:5:30
Thoi gian la: 12:10:16
Thoi gian la: 4:1:20

```

Trong ví dụ trên ta có ba biến `a`, `b`, và `c` được khai báo trong hàm `main()` với `a`, `b` và `c` gọi là các biến đối tượng có kiểu là kiểu lớp `Thoigian`. Như vậy với việc khai báo và xây dựng các lớp sau đó chúng ta có thể dùng như những kiểu dữ liệu mới (kiểu dữ liệu trừu tượng).

Từ khóa **private** và **public** dùng để xác định phạm vi truy xuất của dữ liệu hoặc các hàm thành phần trong lớp. Các thành phần khai báo sau từ khóa `private` chỉ được dùng trong phạm vi của lớp đó, còn với `public` các thành phần khai báo sau nó có thể được sử dụng bởi các hàm hoặc đối tượng bên ngoài lớp.

8.2 Hàm tạo (constructors)

Một hàm gọi là hàm tạo nếu nó trùng tên với chính lớp chứa nó. Một lớp có thể chứa nhiều hàm tạo chỉ cần chúng khác nhau về tham số. Mục đích của hàm tạo là khởi tạo các dữ liệu mỗi khi một đối tượng mới được sinh ra. Như vậy dựa trên số lượng và kiểu dữ liệu của các tham số, một hàm tạo sẽ tự động được gọi khi một đối tượng được tạo lập. Nếu không có hàm tạo nào được viết trong lớp thì trình biên dịch sẽ cung cấp một hàm tạo ngầm định.

Ví dụ 8.2. Cách viết hàm tạo.

```
#include<iostream>
#include<iomanip>
using namespace std;
class Thoigian
{
    private:
        int gio;
        int phut;
        int giay;
    public:
        Thoigian(int = 7, int = 5, int = 30);
        void Datthoigian(int, int, int);
        void Hienthoigian();
};
Thoigian::Thoigian(int g, int p, int gi)
{
    gio=g; phut=p; giay=gi;
}
void::Thoigian::Datthoigian(int g, int p, int gi)
{
    gio=g; phut=p; giay=gi; return;
}
void Thoigian::Hienthoigian()
{
    cout<<"Thoi gian la: ";
    cout<<gio<<":"<<phut<<":"<<giay<<endl;
    return;
}
int main() {
```

```

    Thoigian a,b,c(4,1,20);
    b.Datthoigian(12,10,16);
    a.Hienthoigian();
    b.Hienthoigian();
    c.Hienthoigian();
    return 0;
}

```

Kết quả thực hiện chương trình như sau:

```

Gia tri ve thoi gian: 10, 15, 30
Gia tri ve thoi gian: 10, 15, 30
Gia tri ve thoi gian: 7, 25, 45

```

Các constructor có thể viết dưới dạng hàm inline – loại hàm sẽ được ghép trực tiếp vào trong chương trình, không cần tham chiếu đến chỗ khác, tuy nhiên sẽ phù hợp với các hàm đơn giản. Các constructor có tính chất xếp chồng tức là trong một lớp có thể có một số constructor tuy nhiên chúng được phân biệt bằng tham số và kiểu của tham số đi kèm.

Trái ngược với hàm tạo, C++ cung cấp một hàm gọi là *hàm hủy* (destructor). Các hàm hủy có tên trùng với tên lớp của nó và đặt sau kí hiệu ~. Nếu không có hàm hủy nào được viết thì trình biên dịch C++ cung cấp hàm hủy ngầm định. Lưu ý rằng mỗi lớp chỉ có duy nhất một hàm hủy; chúng không có tham số và cũng không trả về bất cứ giá trị nào. Hàm hủy nhằm mục đích giải phóng các ô nhớ khi chúng không cần thiết nữa, nói chung hàm tạo sẽ phát huy hiệu quả khi trong đối tượng có chứa biến con trỏ.

8.3 Phép gán

Trong các phần trước, chúng ta sử dụng phép gán (=) cho việc gán giá trị vào các biến. Trong chương này chúng ta sẽ tìm hiểu việc gán các đối tượng thuộc lớp với nhau. Chúng ta chú ý rằng lớp là một kiểu dữ liệu trừu tượng nên trên đó chúng ta phải xây dựng các phép thao tác cho nó. Phép gán là một ví dụ trong trường hợp này.

Cấu trúc của phép gán như sau: `void operator = (tên lớp&);` trong đó `void` cho biết hàm này không trả về giá trị, `operator =` biểu thị chúng ta xếp chồng phép gán, `tên lớp&` biểu thị các tham số của operator là một lớp. Với ví dụ ở phần 8.2 chúng ta có thể khai báo: `void operator =(Date&);`

Ví dụ 8.3. Phép gán cho các đối tượng trên lớp Thoigian:

```

#include<iostream>
#include<iomanip>
using namespace std;
class Thoigian
{
    private:
        int gio;
        int phut;
        int giay;
    public:
        Thoigian(int = 7, int =4, int = 2005);
        void operator =(Thoigian&);
        void Inketqua();
};
Thoigian::Thoigian(int x, int y, int z)
{
    gio=x; phut=y; giay=z;
}
void::Thoigian::operator=(Thoigian& Thoigian1)
{
    phut= Thoigian1.phut;
    gio= Thoigian1.gio;
    giay=Thoigian1.giay;
    return;
}
void Thoigian::Inketqua()
{
    cout<<"Ngay, thang, va nam la: ";
    cout<<setfill('0')
        <<setw(2)<<gio<< '/'
        <<setw(2)<<phut<< '/'
        <<setw(4)<<giay;
    cout<<endl;
    return;
}
int main() {
    Thoigian a(4,1,2007), b(12,05,2008);
    cout<<"Gia tri du lieu ban dau: ";
    a.Inketqua();
    a = b;
    cout<<"Sau khi gan la: ";
}

```

```

a.Inketqua();
return 0;
}

```

Câu lệnh `a = b` trong hàm `main()` ở trên chính là câu lệnh gán giá trị các biến tương ứng cho nhau theo phép gán đã xây dựng trong chương trình. Giá trị của `a` bây giờ chính là giá trị của `b`.

Sao chép hàm tạo: Chúng ta thấy rằng phép gán giống như phép khởi tạo những thực chất không có đối tượng mới được tạo ra, đơn giản chỉ là giá trị mới sẽ được gán cho đối tượng đang có. Cấu trúc để sao chép hàm tạo như sau:

```
tên_lớp(const tên_lớp&);
```

Ví dụ như lớp `Thoigian` ở trên ta có: `Thoigian(const Thoigian&)`.

Ví dụ 8.4 Phép gán đối tượng cho lớp số phức:

```

#include<iostream>
#include<iomanip>
using namespace std;
class Sophuc
{
    private:
        int r;
        int i;
    public:
        Sophuc(int = 9, int =4);
        void operator =(Sophuc&);
        void Inketqua();
};
Sophuc::Sophuc(int x, int y)
{
    r = x; i =y;
}
void::Sophuc::operator=(Sophuc& Sophucmoi)
{
    r= Sophucmoi.r;
    i= Sophucmoi.i;

    return;
}
void Sophuc::Inketqua()

```

```

{
    cout<<"so phuc la: ";
    cout<<r<<"+"<<i<<"*i"<<endl;
    return;
}
int main() {
    Sophuc a(7,1), b(6,15);
    cout<<"Gia tri du lieu ban dau ";
    a.Inketqua();
    a = b;
    cout<<"Sau khi gan ta co ";
    a.Inketqua();
    return 0;
}

```

Phạm vi hoạt động của biến:

Như đã đề cập ở chương 5, phạm vi của biến là phạm vi có thể sử dụng của biến đó. Biến địa phương là biến chỉ sử dụng trong hàm nơi nó được khai báo và từ chỗ nó được khai báo đến cuối của hàm. Biến toàn cục là biến được sử dụng trong toàn chương trình bắt đầu từ chỗ nó được khai báo. Tuy nhiên có một số chú ý như sau:

- Nếu có một biến địa phương nào đó được khai báo trùng tên với biến toàn cục thì để sử dụng biến toàn cục tại chỗ đó cần sử dụng toán tử ::.
- Phạm vi của biến toàn cục có thể mở rộng sang các tệp khác bằng cách sử dụng từ khóa `extern`.
- Tên của biến toàn cục có thể được sử dụng lại để khai báo một biến riêng biệt trong tệp đó bằng cách sử dụng từ khóa `static`.

Các biến có kiểu **static** của lớp: Trong một số bài toán chúng ta cần có một biến có giá trị không đổi trong toàn bộ lớp, tất nhiên ta có thể khai báo biến này là biến toàn cục nhưng không phù hợp lắm và hơn nữa làm mất đi tính che dấu dữ liệu trong lập trình hướng đối tượng. Trong trường hợp này chúng ta có thể sử dụng từ khóa `static`, khi khai báo biến loại này vị trí ô nhớ của nó sẽ giữ nguyên đối với mọi đối tượng được tạo mới. Các biến `static` phải được khai báo trong phần mô tả của lớp và có thể khởi tạo giá trị hoặc không. Chúng ta xét khai báo sau đây:

```
class nhanvien
```

```

{ private:
    static double hesothuphi = 0.09;
    ind masonhanvien;
public:
    nhanvien(int);
    void hienthongtin();
}

```

Nếu chúng ta khai báo nhiều đối tượng thì giá trị hesothuphi vẫn sẽ được giữ nguyên cho nó tức là biến hesothuphi sẽ là biến toàn cục của lớp nhanvien.

8.4 Hàm toán tử

Phép gán đơn giản chúng ta đã tìm hiểu trong mục 9.3. Trong phần này chúng ta mở rộng cho các phép toán khác trên lớp. Các hàm dùng cho mục đích định nghĩa các toán tử cho các đối tượng lớp được gọi là hàm toán tử. Một hàm toán tử luôn có dạng `operator<symbol>` trong đó `<symbol>` bao gồm hầu hết các phép toán (một ngôi hoặc hai ngôi) trên các lớp trong đó các toán hạng chính là các đối tượng, chẳng hạn **`operator+`** là hàm cộng, **`operator==`** là hàm so sánh.

Ví dụ 8.5 Xây dựng toán tử so sánh cho hai đối tượng có kiểu `Thoigian` như trong ví dụ 8.1.

```

#include<iostream>
using namespace std;
class Thoigian
{
    private:
        int gio;
        int phut;
        int giay;
    public:
        Thoigian(int = 7, int =4, int =10);
        bool operator==(Thoigian);
};
Thoigian::Thoigian(int x, int y, int z)
{
    gio=x; phut=y; giay = z;
}

```



```

}
bool Thoigian::operator==(Thoigian t)
{
    if ((gio ==t.gio) && (phut == t.phut) && (giay == t.giay))
        return true;
    else
        return false;
}
int main() {
    Thoigian m(4,1,10), n(7, 10, 10);
    cout<<"Vi du ve ham toan tu so sanh == trong C++: "<<endl;
    if (m==n)
        cout<<"m va n bang nhau "<<endl;
    else
        cout<<"m va n khac nhau "<<endl;
    return 0;
}

```

Ví dụ 8.6 Hàm toán tử + cho hai số phức

```

#include<iostream>
using namespace std;
class Sophuc
{
    private:
        int r;
        int i;
    public:
        Sophuc(int = 7, int =4);
        Sophuc operator+(Sophuc);
        insophuc();

};

Sophuc::insophuc()
{
    cout<<r<<"+"<<i<<"*i";
}
Sophuc::Sophuc(int x, int y)
{
    r = x; i = y;
}

```

```

Sophuc Sophuc::operator+(Sophuc t)
{
    Sophuc tg;
    tg.r= r + t.r;
    tg.i= i +t.i;
    return tg;
}

int main() {
    Sophuc m(4,5), n(2, 10),c;
    cout<<"Vi du ve ham toan tu + trong C++: "<<endl;
    c=m+n;
    cout<<"Ket qua phép cong hai so phuc 4+5*i va 2+10*i la: ";
    c.insophuc();
    return 0;
}

```

Ví dụ về kết quả:

```

Vi du ve ham toan tu + trong C++:
Ket qua phép cong hai so phuc 4+5*i va 2+10*i la: 6+15*i

```

8.5 Sự chuyển đổi kiểu dữ liệu trong lớp

Trong chương 2, chúng ta đã biết sự chuyển đổi kiểu dữ liệu trong C++. Kiểu lớp không thể tự chuyển đổi như kiểu cơ sở đã có trong C++, chúng phải được người dùng định nghĩa. Phép chuyển kiểu ngầm định được thực hiện bằng cách sử dụng hàm tạo, phép chuyển đổi kiểu tường minh được thực hiện bằng cách sử dụng toán tử ép kiểu. Trong phần này chúng ta sẽ tìm hiểu việc chuyển đổi các kiểu dữ liệu sau:

- Hàm toán tử chuyển từ kiểu đã xây dựng sẵn sang kiểu lớp,
- Hàm toán tử chuyển đổi kiểu dữ liệu từ kiểu lớp sang kiểu xây dựng sẵn,
- Hàm toán tử chuyển đổi kiểu dữ liệu giữa các lớp.

8.5.1. Hàm toán tử chuyển đổi từ kiểu cơ sở sang kiểu lớp

Để thực hiện việc chuyển đổi kiểu dữ liệu từ cơ sở sang kiểu lớp, chúng ta sử dụng hàm tạo; hàm tạo là một thành viên của lớp. Ví dụ sau đây nhằm mục đích chuyển đổi kiểu dữ liệu từ kiểu long sang kiểu lớp của lớp Thoigian. Chẳng hạn chúng ta muốn sử dụng thời gian dạng giây, sau khi chuyển đổi sang kiểu lớp Thoigian nó sẽ có dạng giờ -

phút - giây trong đó được biểu diễn dưới dạng $3600 * \text{số_giờ} + 60 * \text{số_phút} + \text{số_giây}$.

Chúng ta sẽ có hàm chuyển đổi như sau:

```
Thoigian::Thoigian(long tgl)
{
    gio = int (tgl/3600.0);
    phut =int ((tgl-gio*3600.0)/60.0);
    giay=int(tgl-gio *3600.0-phut*60.0);
}
```

Như vậy hàm trên sẽ nhận một số kiểu long sau đó chuyển sang dạng giờ, phút, giây.

Chương trình hoàn chỉnh như sau:

Ví dụ 8.7

```
#include<iostream>
#include<iomanip>
using namespace std;
class Thoigian
{
    private:
        int gio;
        int phut;
        int giay;
    public:
        Thoigian(int = 5, int =45, int = 20);
        Thoigian(long);
        void Hienthoigian();
};
Thoigian::Thoigian(int g, int p, int gi)
{
    gio =g; phut= p; giay= gi;
}
Thoigian::Thoigian(long tgl)
{
    gio = int (tgl/3600.0);
    phut =int ((tgl-gio*3600.0)/60.0);
    giay=int(tgl-gio *3600.0-phut*60.0);
}
void Thoigian::Hienthoigian()
{
    cout<<setfill('0')
    <<setw(2)<<gio<<" gio "
```

```

        <<setw(2)<<phut<<" phut "
        <<setw(2)<<giay<<" giay"<<endl;
    return;
}
int main() {
    Thoigian a, b(10000L), c(4,1,55);
    cout<<"Gia tri du lieu ban dau: ";
    a.Hienthoigian();
    cout<<"Gia tri thoi gian trong b la: ";
    b.Hienthoigian();
    cout<<"Gia tri thoigian trong c la: ";
    c.Hienthoigian();
    a=Thoigian(65454L);
    cout<<"Thoi gian moi la: ";
    a.Hienthoigian();
    cout<<"\n";
    return 0;
}

```

Kết quả việc thực hiện chương trình như sau:

```

Gia tri du lieu ban dau: 05 gio 45 phut 20 giay
Gia tri thoi gian trong b la: 02 gio 46 phut 40 giay
Gia tri thoigian trong c la: 04 gio 01 phut 55 giay
Thoi gian moi la: 18 gio 10 phut 54 giay

```

8.5.2 Hàm toán tử chuyển đổi từ kiểu lớp sang kiểu cơ sở

Việc chuyển đổi từ kiểu lớp sang kiểu cơ sở được thực hiện bằng việc sử dụng *hàm toán tử chuyển đổi* (conversion operator function). Một hàm toán tử chuyển đổi phải là một hàm thành viên của lớp và phải có tên trùng tên với kiểu cơ sở hoặc kiểu lớp nào đó. Ví dụ muốn xây dựng hàm chuyển đổi từ kiểu lớp sang kiểu long ta sử dụng tên hàm là `operator long()`. Ví dụ sau minh họa việc sử dụng hàm toán tử chuyển đổi để chuyển đổi từ kiểu lớp `Thoigian` sang kiểu long.

Ví dụ 8.8.

```

#include<iostream>
#include<iomanip>
using namespace std;
class Thoigian
{

```

```

private:
    int gio;
    int phut;
    int giay;
public:
    Thoigian(int = 5, int = 15, int = 30);
    operator long();
    void Hienthoigian();
};
Thoigian::Thoigian(int mm, int dd, int yyyy)
{
    gio=mm; phut=dd; giay=yyyy;
}
Thoigian::operator long()
{
    long tg;
    tg = gio*3600 + phut*60 +giay;
    return(tg);
}
void Thoigian::Hienthoigian()
{
    cout<<setfill('0')
        <<setw(2)<<gio<<" gio "
        <<setw(2)<<phut<<" phut "
        <<setw(2)<<giay<<" giay"<<endl;
    return;
}
int main() {
    Thoigian a(4,19,20);
    long b;
    b=a;
    cout<<"Thoi gian cua a la: ";
    a.Hienthoigian();
    cout<<"Thoi gian kieu long la: "<<b;
    return 0;
}

```

Kết quả việc thực hiện chương trình như sau:

```

Thoi gian cua a la: 04 gio 19 phut 20 giay
Thoi gian kieu long la: 15560

```

8.5.3 Hàm toán tử chuyển đổi từ kiểu lớp sang kiểu lớp

Việc chuyển đổi từ kiểu lớp sang kiểu lớp được thực hiện giống như từ kiểu lớp sang kiểu cơ sở - nó được thực hiện bằng việc sử dụng hàm toán tử chuyển đổi. Giả sử có hai lớp `Thoigian` và `Thoigian1`, chúng ta muốn chuyển dạng biểu diễn thời gian là giờ-phút-giây của lớp `Thoigian` sang dạng biểu diễn thời gian bằng giây của lớp `Thoigian1` chúng ta sẽ sử dụng hai hàm toán tử chuyển đổi mỗi hàm đặt ở một lớp và toán tử nằm trong lớp này sẽ có tên trùng với tên của lớp còn lại. Có nghĩa là trong lớp `Thoigian` sẽ có một toán tử có tên `Thoigian1` và ngược lại. Chúng ta sẽ tìm hiểu ví dụ sau đây.

Ví dụ 8.9.

```
#include<iostream>
#include<iomanip>
using namespace std;
class Thoigian1;
class Thoigian
{
private:
    int gio;
    int phut;
    int giay;
public:
    Thoigian(int = 8, int = 5, int = 25);
    operator Thoigian1();
    void Hienthoigian();
};
class Thoigian1
{
private:
    long yyyymmdd;
public:
    Thoigian1(long =0);
    operator Thoigian();
    void Hientg1();
};
Thoigian::Thoigian(int mm, int dd, int yyyy)
{
    gio=mm; phut=dd; giay=yyyy;
```

```

    }
    Thoigian::operator Thoigian1()
    {
        long temp;
        temp=gio*3600+phut*60+giay;
        return(Thoigian1(temp));
    }
void Thoigian::Hienthoigian()
{
    cout<<setfill('0')
        <<setw(2)<<gio<<" gio "
        <<setw(2)<<phut<<" phut "
        <<setw(2)<<giay<<" giay"<<endl;
    return;
}
Thoigian1::Thoigian1(long ymd)
{
    yyyyymmdd=ymd;
}
Thoigian1::operator Thoigian()
{
    int mo, da, yr;
    yr=int(yyyyymmdd/3600.0);
    mo=int(yyyyymmdd-yr*3600.0)/60;
    da=int(yyyyymmdd-yr*3600.0-mo*60.0);
    return(Thoigian(yr,mo,da));
}
void Thoigian1::Hientgl()
{
    cout<<yyyyymmdd<<endl;
    return;
}
int main() {
    cout<<"Chương trình minh hoa chuyen doi kieu du lieu tu lop
sang lop: "<<endl;
    Thoigian a(4,1,20),b;
    Thoigian1 c(6015L), d;
    b=Thoigian(c);
    d=Thoigian1(a);
    cout<<"Thoi gian cua a: "; a.Hienthoigian();
    cout<<"Sau khi chuyen doi kieu la: ";

```

```

d.Hientg1();
cout<<"Thoi gian cua c: ";
c.Hientg1();
cout<<"Sau khi chuyen doi kieu la: ";
b.Hienthoigian();
return 0;
}

```

Kết quả việc thực hiện chương trình như sau:

```

Chương trình minh họa chuyen doi kieu du lieu tu lop sang lop:
Thoi gian cua a: 04 gio 01 phut 20 giay
Sau khi chuyen doi kieu la: 14480
Thoi gian cua c: 6015
Sau khi chuyen doi kieu la: 01 gio 40 phut 15 giay

```

8.6 Thừa kế và sự tương tác giữa các lớp

8.6.1 Thừa kế

Thừa kế là một trong những tính năng quan trọng của C++. Thừa kế là sự thực hiện việc tạo một lớp mới sử dụng một số thành phần đã có ở lớp đã có. Lớp đã có được gọi là lớp cha hay lớp cơ sở. Lớp xây dựng mới được gọi là lớp con hay lớp dẫn xuất. Một lớp dẫn xuất thừa kế tất cả các dữ liệu và phương thức từ lớp cha và bổ sung thêm các dữ liệu cũng như phương thức mới. Như vậy mục đích của thừa kế là sử dụng các đoạn mã đã có làm cho chương trình được phát triển nhanh hơn, tận dụng các khai báo đã có. Để khai báo lớp mới có kế thừa lớp đã có chúng ta viết như sau:

```
class <tên lớp con> : public <tên lớp cơ sở>;
```

Ví dụ 8.10. Ví dụ về tính kế thừa.

```

#include<iostream>
#include<cmath>
const double PI= 2.0 * asin(1.0);
using namespace std;
class Hinhchunhat
{
protected:
    double a,b;
public:

```



```

        Hinhchunhat(double = 1.0, double =2.0);
        double dt();
};
Hinhchunhat::Hinhchunhat(double val, double val2)
{
    a = val; b = val2;
}
double Hinhchunhat::dt()
{
    return (a*b);
}
class Hinhhop : public Hinhchunhat
{
    protected:
        double h;
    public:
        Hinhhop(double h=2.0);
        double dt();
};
Hinhhop::Hinhhop(double p2)
{ h=p2;
}
double Hinhhop::dt()
{
    return (h*Hinhchunhat::dt());
}
int main() {
cout<<"Vi du ve tinh ke thua trong C++"<<endl;
cout<<"Lop hinh hop duoc ke thua tu lop hinh chu nhat:"
"<<endl;
Hinhhop o1(5);
cout<<"The tich hinh hop: "<<o1.dt()<<endl;
return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:

```

Vi du ve tinh ke thua trong C++
Lop hinh hop duoc ke thua tu lop hinh chu nhat:
The tich hinh hop: 10

```

Trong ví dụ trên, lớp Hinhhop kế thừa từ lớp Hinhchunhat việc tính diện tích đáy của hình hộp. Thừa kế là một đặc trưng có nhiều ứng dụng của C++. Khi phát triển các hệ

thống phần mềm lớn các công việc con sẽ được chia thành các lớp có sự tương tác lẫn nhau và kế thừa là một ví dụ cụ thể.

8.6.2 Cách sử dụng các từ khóa *public*, *private* và *protected* trong thừa kế lớp

Các từ khóa *public*, *private*, và *protected* sử dụng trong việc khai báo các biến và các hàm trong một lớp cụ thể nào đó hoặc khai báo thừa kế từ lớp cơ sở. Trong khi viết chương trình chúng ta cần hiểu rõ và sử dụng chúng cho phù hợp nhất là trong việc thừa kế. Sau đây chúng ta sẽ xem xét ba kiểu thừa kế cùng với các quy tắc truy nhập dữ liệu cũng như các hàm thành phần của nó:

- Trường hợp kế thừa kiểu *public*: Khi đó các thành phần *public* của lớp cơ sở sẽ là *public* ở lớp kế thừa; thành phần *protected* ở lớp cơ sở cũng sẽ là thành phần *protected* ở lớp kế thừa, tuy nhiên thành phần *private* sẽ không thể truy cập được trong lớp kế thừa.
- Trường hợp kế thừa lớp kiểu *private*: Các thành phần *protected* và *public* sẽ biến thành thành phần *private* của lớp kế thừa, thành phần *private* của lớp cơ sở sẽ không sử dụng được từ lớp kế thừa.
- Trường hợp kế thừa kiểu *protected*: Các thành phần *protected* và *public* sẽ trở thành thành phần *protected* của lớp kế thừa, thành phần *private* sẽ không sử dụng được.

Chú ý: Một lớp có thể kế thừa từ nhiều lớp khác (đa thừa kế), việc khai báo như sau:

```
class tên_lớp_kế_thừa: <kiểu kế thừa 1> lớp_cơ_sở_1, <kiểu kế thừa 2>  
lớp_cơ_sở_2,...
```

8.7 Tính đa hình

Tính đa hình cho phép hàm trùng tên ở lớp cơ sở và lớp kế thừa, tuy nhiên tùy theo lời gọi và cách khai báo đối tượng, kết quả trả về sẽ nằm ở hàm cơ sở hay hàm kế thừa. Để sử dụng chính xác các hàm xếp chồng trùng tên kiểu này, chúng ta phải sử dụng từ khóa *virtual* đặt trước các hàm xếp chồng. Về nguyên tắc nếu hàm là *virtual* ở lớp cơ sở thì nó cũng là *virtual* ở lớp dẫn xuất ngay sau nó. Khi một hàm được gọi thì

chương trình dịch sẽ xác định hàm xếp chồng nào được thực thi dựa vào đối tượng thực hiện lời gọi đó.

Ví dụ 8.11.

```
#include<iostream>
#include<cmath>
using namespace std;
class Tivi1
{
    protected:
        double gia;

    public:
        Tivi1(double = 2.0);
        double Tong(double);
        double Tong2(double);
};
Tivi1::Tivi1(double x)
{
    gia = x;
}
double Tivi1::Tong(double x)
{
    return (x - 5);
}
double Tivi1::Tong2(double x)
{
    return (Tong(x) +10);
}
class Tivi2 : public Tivi1
{
    public:
        double Tong(double);
};
double Tivi2::Tong(double x)
{
    return (x - 20);
}

int main() {
Tivi1 o1;
```

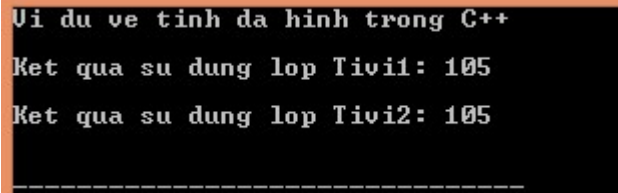
```

Tivi2 o2;

cout<<"Vi du ve tinh da hinh trong C++ "<<endl<<endl;
cout<<"Ket qua su dung lop Tivi1: "<<o1.Tong2(100)<<endl<<endl;
cout<<"Ket qua su dung lop Tivi2: "<<o2.Tong2(100)<<endl<<endl;

return 0;
}

```



```

Vi du ve tinh da hinh trong C++
Ket qua su dung lop Tivi1: 105
Ket qua su dung lop Tivi2: 105

```

Ví dụ 8.12

```

#include<iostream>
#include<cmath>
using namespace std;
class Tivi1
{
    protected:
        double gia;

    public:
        Tivi1(double = 2.0);
        virtual double Tong(double);
        double Tong2(double);
};
Tivi1::Tivi1(double x)
{
    gia = x;
}
double Tivi1::Tong(double x)
{
    return (x - 5);
}
double Tivi1::Tong2(double x)
{
    return (Tong(x) + 10);
}
class Tivi2 : public Tivi1

```

```

{
    public:
        double Tong(double);
};
double Tivi2::Tong(double x)
{
    return (x - 20);
}

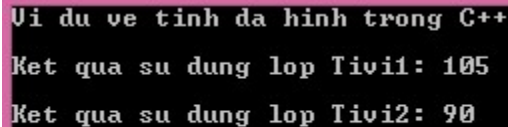
int main() {
    Tivi1 o1;
    Tivi2 o2;

    cout<<"Vi du ve tinh da hinh trong C++ "<<endl<<endl;
    cout<<"Ket qua su dung lop Tivi1: "<<o1.Tong2(100)<<endl<<endl;
    cout<<"Ket qua su dung lop Tivi2: "<<o2.Tong2(100)<<endl<<endl;

    return 0;
}

```

Ví dụ về kết quả thực hiện chương trình như sau:



```

Vi du ve tinh da hinh trong C++
Ket qua su dung lop Tivi1: 105
Ket qua su dung lop Tivi2: 90

```

Trong ví dụ trên, khi gọi hàm Tong2 () cho đối tượng o1, thì kết quả sẽ là 105, tức là hàm Tong () trong lớp Tivi1 sẽ được thực hiện. Ngược lại nếu gọi cho đối tượng o2 thì kết quả sẽ là 90 tức là hàm Tong () ở lớp Tivi2 sẽ được gọi. Đó chính là bản chất của tính đa hình trong C++, cho phép các hàm tên trùng nhau ở các lớp tuy nhiên khi thực hiện sẽ tham chiếu đến đối tượng được gọi.

Bài tập chương 8

Bài 1. Giải thích các khái niệm sau và cho ví dụ minh họa: lớp (class); đối tượng (object), phương thức (method), hàm thành viên (function member), dữ liệu thành viên (data member).

Bài 2. Xác định lỗi trong khai báo lớp sau đây:

```
class employee
```

```

{public:
    int empnum;
    char code;
private:
    class(int =0);
    void showemp(int, char);
}

```

Bài 3. Xác định các mệnh đề sau là đúng hay sai:

- a) Hàm tạo phải có tên trùng với tên lớp của nó.
- b) Một lớp chỉ có duy nhất một hàm tạo
- c) Một lớp có duy nhất một hàm tạo ngầm định
- d) Một hàm tạo ngầm định không có tham số hoặc có nhưng với các giá trị ngầm định
- e) Một hàm tạo phải khai báo cho mỗi lớp
- f) Một hàm tạo sẽ được tự động gọi mỗi khi có một đối tượng mới được tạo lập
- g) Một lớp chỉ có duy nhất một hàm hủy
- h) Một hàm hủy có tên trùng với tên lớp chứa nó và đặt sau dấu ~
- i) Một hàm tạo có thể có các tham số ngầm định
- j) Hàm tạo phải được khai báo cho mỗi lớp
- k) Một hàm tạo sẽ không có ý nghĩa khi lớp chứa dữ liệu kiểu con trỏ.

Bài 4. Viết chương trình sử dụng lớp khai báo hai biến a và b, với các hàm tính diện tích, chu vi, và in kết quả. Khai báo hai đối tượng a1 và a2 để kiểm thử các hàm.

Bài 5. Khai báo lớp `Ngay` có các thuộc tính là ngày, tháng và năm. Viết chương trình xây dựng các hàm toán tử +, so sánh cho các đối tượng thuộc lớp `Ngay`.

Bài 6. Giải thích sự khác nhau giữa phép gán và phép khởi tạo.

Bài 7. Viết khai báo cho lớp `TAMGIAC` trong đó có ba tham số a, b, và c là ba cạnh của tam giác. Các hàm thành phần để tính diện tích, chu vi. Khai báo hai đối tượng với hai bộ tham số khác nhau và hiện các kết quả lên màn hình.

Bài 8. Thừa kế là gì? Cho ví dụ minh họa.

Bài 9. Xây dựng hai lớp `vuong` và `chunhat` để tính diện tích hình vuông và chữ nhật. Trong đó lớp `hinh_vuong` thừa kế lại việc tính diện tích của lớp `chunhat`.

Bài 10. Tính đa hình là gì? Cho ví dụ minh họa.

Chương 9. Kiểu con trỏ và kiểu cấu trúc

9.1 Kiểu con trỏ

9.1.1 Khái niệm kiểu con trỏ

Trong các phần trước, chúng ta đã tìm hiểu việc sử dụng các biến để viết chương trình, mỗi biến được đặc trưng bởi tên và kiểu giá trị của nó. Trong chương này chúng ta sẽ tìm hiểu một loại biến đặc biệt là biến con trỏ, biến con trỏ là biến chứa địa chỉ của một ô nhớ nào đó. Để khai báo một biến con trỏ ta dùng cú pháp sau:

```
<kiểu_dữ_liệu> *tên_biến;
```

Ví dụ về khai báo hai biến con trỏ:

```
int *p;  
double *l;
```

Cũng giống như biến thông thường, biến con trỏ có thể được khởi tạo giá trị, chính là địa chỉ của một biến cụ thể nào đó. Con trỏ đặc biệt hữu ích khi chúng ta làm việc với danh sách liên kết và sử dụng cấp phát bộ nhớ động trong khi viết chương trình.

Để gán giá trị địa chỉ của một biến nào đó cho con trỏ ta ghi: `tên_biến_trỏ = &tên_biến`; trong đó `&` là toán tử lấy ra địa chỉ của một biến thông thường. Để lấy giá trị của biến kiểu con trỏ ta ghi: `*tên_biến_trỏ`; trong đó kí hiệu `*` là toán tử lấy ra nội dung của ô nhớ mà biến con trỏ trỏ tới.

Ví dụ 9.1. Khai báo biến con trỏ và hiện địa chỉ của nó.

Giải:

```
#include <iostream>  
using namespace std;  
int *m;  
int tg = 10;  
int main()  
{  
    m = &tg;  
    cout<<"Địa chỉ của tg là: "<<m;  
    cout<<"Giá trị chứa trong biến con trỏ m là: "<<*m;  
    return 0;  
}
```


Chương trình trên sẽ hiện lên địa chỉ của biến t_g và giá trị lưu tại ô nhớ mà m trỏ vào. Câu lệnh $\&t_g$ cho phép ta lấy địa chỉ của biến t_g để gán cho con trỏ m . Để lấy giá trị lưu tại địa chỉ mà con trỏ m trỏ tới ta viết $*m$, như vậy nếu m là con trỏ thì $*m$ là một biến có địa chỉ lưu tại m . Như vậy việc sử dụng con trỏ cho phép chúng ta thao tác cả với địa chỉ mà ô nhớ được cấp phát cũng như giá trị của ô nhớ đó.

Đến đây chúng ta sẽ đặt câu hỏi về sự khác nhau của tham chiếu và con trỏ là như thế nào ?. Một tham chiếu là một hằng địa chỉ của một biến nào đó đã có trước đó nên giá trị của nó là cố định, trong khi một con trỏ là một biến chứa địa chỉ nên có thể thay đổi được. Hơn nữa, không có tham chiếu NULL và một tham chiếu phải được khởi tạo ngay khi nó được tạo ra. Biến tham chiếu thường chỉ được sử dụng trong việc truyền tham số cho hàm còn biến con trỏ được sử dụng rộng rãi hơn.

9.1.2 Môi liên hệ giữa mảng và con trỏ

Chúng ta đã tìm hiểu cấu trúc về mảng, trong phần này chúng ta sẽ nghiên cứu mối quan hệ giữa mảng và con trỏ, hay nói cách khác là sử dụng con trỏ trong việc thực hiện các thao tác trên mảng. Trong C++, việc khai báo một mảng cũng đồng nghĩa với việc khai báo một con trỏ chỉ vào phần tử đầu tiên của mảng. Xét khai báo: `int a[100];` `int *p;` nếu chúng ta muốn đặt địa chỉ cho con trỏ p vào phần tử đầu tiên của mảng, hai cách viết sau đều hợp lệ: `p = &a[0];` hoặc `p = a;`. Như vậy bản chất của mảng được khai báo chính là một con trỏ trỏ đến phần tử đầu tiên của nó.

Ví dụ 9.2. Nhập một mảng a có n phần tử nguyên. Hiện địa chỉ của từng phần tử lên màn hình.

Giải:

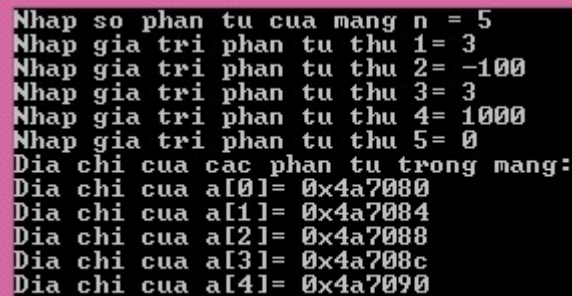
```
#include <iostream>
using namespace std;
int n;
int a[100];
void nhap()
{
    cout<<"Nhập n= "; cin>>n;
    for (int i=0; i<n; i++)
    {
```

```

        cout<<"Nhap gia tri phan tu thu "<<i+1<<"= ";
        cin>>a[i];
    }
}
void indiachi()
{
    cout<<"Dia chi cua cac phan tu la: "<<endl;
    for (int i=0; i<n; i++)
        cout<<"Dia chi a["<<i<<"]="<<&a[i]<<endl;
}
int main(void)
{
    nhap();
    indiachi();
    return 0;
}

```

Ví dụ về việc thực hiện chương trình như sau:



```

Nhap so phan tu cua mang n = 5
Nhap gia tri phan tu thu 1= 3
Nhap gia tri phan tu thu 2= -100
Nhap gia tri phan tu thu 3= 3
Nhap gia tri phan tu thu 4= 1000
Nhap gia tri phan tu thu 5= 0
Dia chi cua cac phan tu trong mang:
Dia chi cua a[0]= 0x4a7080
Dia chi cua a[1]= 0x4a7084
Dia chi cua a[2]= 0x4a7088
Dia chi cua a[3]= 0x4a708c
Dia chi cua a[4]= 0x4a7090

```

Nhìn vào kết quả ta thấy, các phần tử của mảng sẽ được đặt liên tiếp trong bộ nhớ, với các phần tử nguyên trong trường hợp này mỗi phần tử cần 4 byte để lưu trữ.

Ví dụ 9.3. Nhập một mảng a có n phần tử nguyên. Tính trung bình cộng các phần tử của mảng a.

Giải: Chúng ta đã biết cách giải thông thường, sau đây chúng ta sẽ sử dụng con trỏ trong khi viết chương trình.

```

#include <iostream>
using namespace std;
int n;
int a[100];
void nhap()
{
    cout<<"Nhap n= "; cin>>n;
}

```

```

    for (int i=0; i<n; i++)
    {
        cout<<"Nhap gia tri phan tu thu "<<i+1<<"= ";
        cin>>a[i];
    }
}

void tinhhtbc()
{
    int *p;
    int tong=0;
    p=a;
    for (int i=0; i<n;i++)
    {
        tong = tong +*p;
        p = p+1;
    }
    cout<<"Tong = "<<tong;
}

int main(void)
{
    nhap();
    tinhhtbc();
    return 0;
}

```

Trong chương trình trên, trong hàm `tinhhtbc()`, thay vì truy xuất đến từng phần tử của mảng `a` qua chỉ số chúng ta đã sử dụng con trỏ. Đầu tiên con trỏ `p` chỉ vào phần tử đầu tiên, tiếp đó con trỏ sẽ dịch đi một giá trị và đó chính là chỉ vào phần tử kế tiếp, tương tự như vậy sẽ duyệt toàn bộ các phần tử của mảng `a` thông qua con trỏ `p`. Các biến con trỏ cũng giống như các biến thông thường là chứa giá trị, tuy nhiên giá trị ở đây chính là địa chỉ. Đối với biến con trỏ có thể sử dụng các phép toán như `+`, `-`, `==`, `!=`, `<`, `>`,... Tuy nhiên khi sử dụng chúng ta phải hết sức cẩn thận trong quá trình xác định địa chỉ cho các biến con trỏ. Khi khai báo biến con trỏ chúng ta cũng hoàn toàn có thể khởi tạo giá trị đầu cho chúng, tất nhiên đó phải là một địa chỉ.

9.1.3 Truyền tham số là con trỏ cho hàm

Trong chương 5 chúng ta đã tìm hiểu phương pháp sử dụng tham chiếu cho hàm nhằm mục đích trả về nhiều giá trị. Trong phần này chúng ta sẽ tìm hiểu phương pháp sử

dùng con trỏ cho việc truyền tham số trong hàm, hơn nữa chúng ta có thể sử dụng con trỏ để truyền một mảng cho hàm.

Ví dụ 9.4. Viết chương trình đổi chỗ hai giá trị a và b cho nhau.

Giải:

```
#include <iostream>
using namespace std;
void doicho(int *m, int *n)
{
    int tg;
    tg=*n;
    *n=*m;
    *m=tg;
}
int main(void)
{
    int a,b;
    cout<<"a= "; cin>>a;
    cout<<"b= "; cin>>b;
    doicho(&a,&b);
    cout<<"Gia tri cua a va b sau khi doi cho la: "<<endl;
    cout<<"a= "<<a<<endl;
    cout<<"b= "<<b;
    return 0;
}
```

Trong chương trình trên giá trị của a và b sẽ được đổi chỗ cho nhau thông qua việc truy cập trực tiếp vào địa chỉ của nó. Khi truyền tham số cho hàm `doicho()` chúng ta sẽ truyền hai giá trị là địa chỉ của a và b cho hai con trỏ m và n.

Ví dụ 9.5. Cho mảng một chiều a kích thước n. Viết chương trình tìm giá trị nhỏ nhất của mảng a.

Giải: Chúng ta sẽ xây dựng hai hàm là hàm nhập dữ liệu và hàm tìm giá trị nhỏ nhất. Trong hàm tìm giá trị nhỏ nhất (hàm `timmin()`), chúng ta sẽ sử dụng một con trỏ để thay cho một mảng. Khi gọi hàm tìm giá trị nhỏ nhất chúng ta sẽ gán con trỏ vào phần tử đầu tiên của mảng. Trong hàm `timmin()`, câu lệnh `p = p+1` sẽ duyệt qua lần lượt các giá trị của mảng.

```
#include <iostream>
```

```

using namespace std;
int n;
int a[100];
void nhap()
{
    cout<<"Nhap n= "; cin>>n;
    for (int i=0; i<n; i++)
    {
        cout<<"Nhap gia tri phan tu thu "<<i+1<<"= ";
        cin>>a[i];
    }
}
int timmin(int *p, int m)
{
    int i; int min=*p;
    for (i=1; i<m; i++)
    {
        p = p+1;
        if (min>*p)
            min=*p;
    }
    return min;
}
int main()
{
    nhap();
    int *m=a;
    cout<<"Gia tri nho nhat cua mang a la: "<<timmin(m, n);
    return 0;
}

```

9.1.4 Cấp phát bộ nhớ động

Trong khi dịch chương trình, các biến khai báo trong chương trình sẽ được bố trí bộ nhớ cho chúng từ đầu, tuy nhiên biến con trỏ cho phép cấp phát bộ nhớ động tức là thực hiện việc cấp phát cũng như giải phóng bộ nhớ trong quá trình chạy chương trình. Điều này sẽ làm cho không gian nhớ được tiết kiệm hơn giúp cho quá trình thực hiện chương trình được nhanh hơn. Để thực hiện điều này chúng ta sử dụng hai toán tử là `new` và `delete`, trong đó toán tử `new` sẽ dùng để cấp phát số ô nhớ và `delete` sẽ dùng để

giải phóng các ô nhớ của con trỏ khi không cần thiết. Chúng ta sẽ tìm hiểu vấn đề này qua các ví dụ sau.

Ví dụ 9.6. Ví dụ về việc cấp phát bộ nhớ động.

```
#include<iostream>
using namespace std;
int a;
int main()
{
    cout<<"Nhập a = "; cin>>a;
    cout<<"Bạn vừa nhập a = "<<a;
    int *p = new int;
    cout<<"Nhập giá trị cho *p ";
    cin>>*p;
    cout<<"Giá trị của *p là: "<<*p;
    delete p;
    cout<<"Con trỏ p đã được giải phóng ";
    return 0;
}
```

Trong ví dụ trên, con trỏ `p` được sinh ra bằng lệnh `int *p = new int;` sau đó được giải phóng bằng câu lệnh `delete p;` việc cấp phát bộ nhớ động có tính hữu ích cao trong quá trình thực hiện các chương trình lớn cần tồn nhiều bộ nhớ.

Ví dụ 9.7. Cấp phát bộ nhớ động cho mảng.

```
#include<iostream>
using namespace std;
int a;
int main()
{
    int n, i;
    cout<<"Nhập n = "; cin>>n;
    double *p = new double[n];
    for (i=0; i<n; i++)
    {
        cout<<"Nhập giá trị cho phần tử thứ "<<i+1<<" ";
        cin>>p[i];
    }
    double tbc=0;
    for (i=0; i<n; i++)
        tbc = tbc + p[i];
}
```

```

        tbc=tbc/n;
        cout<<"Gia tri trung binh cong cac phan tu vua nhap la:
"<<tbc;;
        delete[] p;
        return 0;
}

```

Trong ví dụ trên, chúng ta đã dùng con trỏ cũng như việc cấp phát bộ nhớ động để thực hiện thao tác trên mảng. Kích thước mảng sau khi xác định bởi mỗi lần chạy chương trình sẽ được cấp phát tại chính thời điểm nó xác định, việc làm này làm tiết kiệm bộ nhớ hơn việc khai báo các biến tĩnh.

9.2 Kiểu cấu trúc

9.2.1 Giới thiệu kiểu cấu trúc

Kiểu cấu trúc nhằm mục đích khai báo các biến dạng bản ghi có nhiều thành phần dữ liệu khác nhau, các thành phần dữ liệu này gọi là các thuộc tính. Trên thực tế có nhiều bài toán mà mỗi phần tử cần xem xét được đặc trưng bởi nhiều thuộc tính khác nhau. Chẳng hạn danh sách học sinh bao gồm các thuộc tính như họ tên, ngày sinh, quê quán, điểm trung bình; danh sách cán bộ bao gồm mã số, họ tên, ngày sinh, đơn vị; danh sách hàng hóa bao gồm các thông tin như tên mặt hàng, đơn vị tính, đơn giá,... Để khai báo cấu trúc ta có thể sử dụng các cách sau:

(i) Khai báo thông qua định nghĩa cấu trúc:

```

struct <tên_cấu_trúc>
{
    Khai báo các thuộc tính thành phần của cấu trúc;
};

```

Sau khi định nghĩa xong cấu trúc chúng ta có thể sử dụng chúng để khai báo các biến giống như kiểu dữ liệu thông thường qua cú pháp:

```

<tên_cấu_trúc> tên_biến1, tên_biến2,...;

```

(ii) Khai báo trực tiếp các biến cấu trúc:

```

struct
{
    Khai báo các thuộc tính thành phần của cấu trúc;
}

```

```
    } tên_biến1, tên_biến2,...;
```

Ví dụ 9.8. Khai báo cấu trúc tên là `mathang` và các thuộc tính của nó sau đó sử dụng chúng để nhập và xuất thông tin về 2 mặt hàng chẳng hạn như mặt hàng Tivi và mặt hàng xi măng.

Giải: Chương trình như sau:

```
#include <iostream>
#include <string>
using namespace std;
struct mathang
{
    string ten;
    string dvt;
    int sl;
    int dg;
};
int main( )
{
    struct mathang mh1;
    struct mathang mh2;
    mh1.ten ="Tivi";
    mh1.dvt="Chiec";
    mh1.sl =10;
    mh1.dg=24;
    cout<<endl;
    cout<<"VI DU VE KHAI BAO VA SU DUNG BIEN KIEU CAU TRUC"<<endl
<<endl;
    cout << "Ten cua mat hang nhat la: " << mh1.ten <<endl;
    cout << "Don vi tinh: " << mh1.dvt <<endl;
    cout << "So luong mat hang thu nhat: " << mh1.sl <<endl;
    cout << "Don gia mat hang thu nhat: " << mh1.dg <<endl;
    cout << "=====" <<endl;
    mh2.ten ="Xi mang";
    mh2.dvt="Tan";
    mh2.sl =100;
    mh2.dg=10;
    cout << "Ten mat hang thu nhat la: " << mh2.ten <<endl;
    cout << "Don vi tinh mat hang thu nhat la: " << mh2.dvt <<endl;
    cout << "So luong mat hang thu nhat la: " << mh2.sl <<endl;
    cout << "Don gia mat hang thu nhat la: " << mh2.dg <<endl;
```



```

return 0;
}

```

Kết quả thực hiện chương trình như sau:

```

VI DU VE KHAI BAO VA SU DUNG BIEN KIEU CAU TRUC
Ten cua mat hang nhat la: Tivi
Don vi tinh: Chiec
So luong mat hang thu nhat: 10
Don gia mat hang thu nhat: 24
=====
Ten mat hang thu nhat la: Xi mang
Don vi tinh mat hang thu nhat la: Tan
So luong mat hang thu nhat la: 100
Don gia mat hang thu nhat la: 10
=====

```

9.2.2 Mảng với các phần tử có kiểu cấu trúc

Chúng ta có thể sử dụng mảng mà trong đó mỗi phần tử có kiểu là kiểu cấu trúc. Chúng ta sẽ tìm hiểu ví dụ sau để nắm được việc sử dụng cũng như truy xuất các phần tử mảng có kiểu cấu trúc.

Ví dụ 9.9. Nhập vào n mặt hàng, mỗi mặt hàng đặc trưng bởi tên, đơn vị tính, số lượng và đơn giá. In danh sách mặt hàng vừa nhập lên màn hình.

Giải:

```

#include <iostream>
#include <string>
//mang kieu cau truc
using namespace std;
struct mathang
{
    string ten;
    string dvt;
    int sl;
    int dg;
};

int main( )
{
    struct mathang a[100];
    int n; // so luong mat hang
    cout<<"Nhap so luong mat hang = "; cin>>n;
    for (int i=0; i<n;i++)
    {cout<<"Nhap thông tin mặt hàng thu "<<i<<endl;
      cout<<"Ten = "; cin>>a[i].ten;
    }
}

```

```

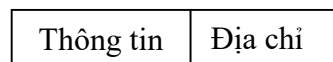
    cout<<"Don vi tinh = "; cin>>a[i].dvt;
    cout<<"So luong "; cin>>a[i].sl;
    cout<<"Don gia= "; cin>>a[i].dg;
}
cout<<"Cac mat hang da nhap la: "<<endl;
for (int i=0; i<n; i++)
{
    cout << "Ten mat hang : " << a[i].ten <<endl;
    cout << "Don vi tinh: " << a[i].dvt <<endl;
    cout << "So luong: " << a[i].sl <<endl;
    cout << "Don gia: " << a[i].dg <<endl;
    cout << "=====" <<endl;
}
return 0;
}

```

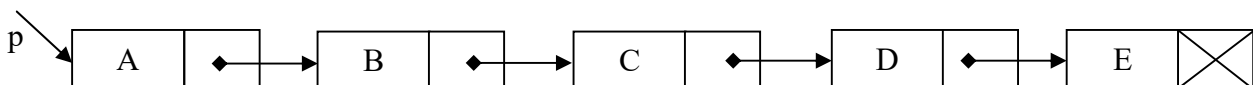
Chúng ta thấy rằng với mảng có các phần tử có kiểu cấu trúc có thể tách ra thành các mảng một chiều riêng biệt, mỗi mảng là một thuộc tính. Tuy nhiên cấu trúc cho chúng ta một cái nhìn gần với thực tế hơn, các phép xử lý cho chúng ta tư duy logic hơn.

9.2.3 Danh sách liên kết trên cấu trúc

Danh sách liên kết là một kiểu bố trí dữ liệu rất gần với thực tế. Mỗi nút của danh sách sẽ chứa thông tin về một bản ghi cụ thể. Điều đặc biệt của danh sách là mỗi nút sẽ chứa một trường là con trỏ trỏ tới địa chỉ của nút tiếp theo. Mỗi nút trong danh sách sẽ gồm có hai phần là phần thông tin và phần địa chỉ. Phần thông tin có thể chứa các trường dữ liệu, trường địa chỉ chính là con trỏ để chứa địa chỉ của nút kế nó. Hình ảnh một nút được minh họa như sau:



Đây là một ứng dụng của kiểu con trỏ. Hình ảnh một danh sách đơn có 5 phần tử như sau:



trong đó con trỏ p trỏ tới nút đầu danh sách, mỗi nút sẽ có trường thông tin chẳng hạn kí hiệu A, B, C,... và một trường con trỏ để chỉ đến địa chỉ của nút tiếp theo. Nếu danh sách rỗng thì p = NULL, như vậy nút cuối cùng sẽ chỉ vào NULL. Khi làm việc với danh sách,

chúng ta sẽ có các thao tác cơ bản như tạo mới danh sách, tìm kiếm phần tử thỏa mãn điều kiện nào đó trong danh sách, chèn phần tử và xóa phần tử khỏi danh sách...

Ví dụ 9.10 Tạo mới một danh sách có n phần tử, mỗi phần tử chứa thông tin về một số nguyên.

Giải: Việc tạo mới một danh sách sẽ tương đương với việc chèn dần phần tử mới vào cuối danh sách. Chúng ta sẽ bố trí một con trỏ luôn trỏ vào đầu danh sách và một con trỏ trỏ vào phần tử cuối cùng của danh sách để mỗi khi có phần tử mới cần thêm vào con trỏ này sẽ chỉ sang địa chỉ của nút mới cần chèn.

```
#include <iostream>
using namespace std;
class dsd{
    // Tao danh sach don chen dan vao cuoi danh sach
    struct nut {
        int a;
        nut *tiep;
    };
public:
    int n; //so luong nut
    dsd(){
        dau = NULL;
    }
    void nhap()
    {
        nut *cuoi;
        cuoi=NULL;
        cout<<"Nhap so luong nut= "; cin>>n;
        for (int i=1; i<=n; i++)
        {
            int gt;
            cout<<"Gia tri cho nut thu "<<i<<" = "; cin>>gt;
            nut *tg = new nut();
            tg->a = gt;
            tg->tiep=NULL;
            if (dau == NULL)
            {
                dau= tg; cuoi=tg;
            }
            else
            {
                cuoi->tiep =tg;
            }
        }
    }
};
```

```

        cuoi=tg;
    }
}

void inds()
{
    nut *tg;
    tg=dau;
    int dem=0;
    while (tg !=NULL)
    {    dem++;
        cout<<"Gia tri nut thu "<<dem<<"= "<<tg->a<<endl;
        tg=tg->tiep;
    }
}

private:
    nut *dau;
};

int main() {
    dsd x;
    x.nhap();
    x.inds();
    return 0;
}

```

Ví dụ trên cho chúng ta cách thức xây dựng một danh sách. Chúng ta lưu ý rằng luôn phải có một con trỏ trỏ vào đầu danh sách và giữa các nút phải được liên kết với nhau; nút trước sẽ chứa thông tin về địa chỉ của nút ngay sau nó.

Ví dụ 9.11 Cho danh sách đơn có n nút trong đó mỗi nút chứa thông tin về một số nguyên. Yêu cầu nhập danh sách vào từ bàn phím và in lên các số chính phương có trong danh sách.

Giải: Trước hết chúng ta sẽ tạo mới danh sách bằng cách nhập liên tiếp các phần tử từ bàn phím.

```

#include <iostream>
#include<cmath>
using namespace std;
class dsd{
    struct nut {
        int a;

```

```

        nut *tiep;
    };
public:
    int n; //so luong nut
    dsd(){
        dau = NULL;
    }
    void nhap()
    {
        nut *cuoi;
        cuoi=NULL;
        cout<<"Nhap so luong nut= "; cin>>n;
        for (int i=1; i<=n; i++)
        {
            int gt;
            cout<<"Gia tri cho nut thu "<<i<<" = "; cin>>gt;
            nut *tg = new nut();
            tg->a = gt;
            tg->tiep=NULL;

            if (dau == NULL)
            {
                dau= tg; cuoi=tg;
            }
            else
            {
                cuoi->tiep =tg;
                cuoi=tg;
            }
        }
    }
    void insochinhphuong()

    {
        nut *tg;
        tg=dau;
        while (tg!=NULL)
        {
            if ((tg->a >0) && ( pow(floor(sqrt(tg->a)),2) == tg->a))
                cout<<tg->a<<" ";
            tg=tg->tiep;
        }
    }
}

```

```
private:
    nut *dau;
};
int main() {
    dsd x;
    x.nhap();
    cout<<"Cac so chinh phuong trong danh sach la: "<<endl;
    x.insochinhphuong();
    return 0;
}
```

Trong ví dụ trên, sau khi tạo lập danh sách chúng ta chỉ việc duyệt lại toàn bộ các nút và kiểm tra xem trường thông tin có phải là số chính phương hay không.

Ví dụ 9.12. Tạo mới một danh sách, thêm vào sau nút chứa giá trị là 10 đầu tiên, nếu không có chèn vào đầu danh sách nút có giá trị bằng 100.

Giải: Sau khi tạo danh sách chúng ta phải thực hiện thao tác tìm kiếm và chèn một phần tử vào danh sách. Khi tìm thấy vị trí cần chèn chúng ta phải thực hiện việc gắn nút cần chèn vào trước rồi mới tách liên kết đến nút đó đến nút mới. Việc chèn một nút a vào sau nút trở bởi b trong danh sách được thực hiện như sau:

Bước 1: Gán trường địa chỉ của nút a bằng địa chỉ của nút b đang chỉ vào.

Bước 2: Gán địa chỉ của nút b đang chỉ vào bằng địa chỉ của nút a.

```
#include <iostream>
using namespace std;
class dsd{
    // Tao danh sach don chen dan vao cuoi danh sach
    struct nut {
        int a;
        nut *tiep;
    };

public:
    int n; //so luong nut
    dsd(){
        dau = NULL;
    }
    void nhap()
    {    nut *cuoi;
        cuoi=NULL;
```

```

cout<<"Nhap so luong nut= "; cin>>n;
for (int i=1; i<=n; i++)
{
    int gt;
    cout<<"Gia tri cho nut thu "<<i<<" = "; cin>>gt;
    nut *tg = new nut();
    tg->a = gt;
    tg->tiep=NULL;
    if (dau == NULL)
    {
        dau= tg; cuoi=tg;
    }
    else
    {
        cuoi->tiep =tg;
        cuoi=tg;
    }
}
}

void chen(int y)
//chen y vao nut co gia tri 10 dau tien neu co,
//khong thi chen vao nut dau tien
{
    nut *tg;
    tg=dau;
    while (tg!=NULL)
    {
        if (tg->a ==10)
            break;
        tg=tg->tiep;
    }
    nut *tg1 = new nut();
    tg1->a = y;
    tg1->tiep=NULL;
    if (tg !=NULL)
    {
        tg1->tiep = tg->tiep;
        tg->tiep = tg1;
    }
    else
    {
        tg1->tiep = dau;
        dau = tg1;
    }
}

```

```

        }
    }
    void inds()
    {
        nut *tg;
        tg=dau;
        int dem=0;
        cout<<tg->a; tg=tg->tiep;
        while (tg !=NULL)
        {
            dem++;
            cout<<" -> "<<tg->a;
            tg=tg->tiep;
        }

    }

private:
    nut *dau;
};
int main() {
    dsd x;
    x.nhap();
    cout<<"Danh sach ban dau: "<<endl;
    x.inds();
    x.chen(100);
    cout<<"\nDanh sach sau khi chen: "<<endl;
    x.inds();
    return 0;
}

```

Ví dụ 9.13. Xóa nút đầu tiên có giá trị bằng k cho trước của một danh sách liên kết đơn trong đó trường thông tin mỗi nút chứa một giá trị nguyên.

Giải: Đầu tiên chúng ta phải tìm xem có hay không nút có trường thông tin bằng k. Nếu có, chúng ta chỉ việc gán trường địa chỉ của nút trở vào nút có giá trị bằng k bằng địa chỉ của nút có giá trị bằng k trở vào. Sau đó chúng ta xóa ô nhớ cấp phát cho nút k đi.

```

#include <iostream>
#include<cmath>
// Xoa mot nut khoi danh sach
using namespace std;
class dsd{

```



```

    struct nut {
        int a;
        nut *tiep;
    };
public:
    int n; //so luong nut
    dsd(){
        dau = NULL;
    }
    void nhap()
    {
        nut *cuoi;
        cuoi=NULL;
        cout<<"Nhap so luong nut= "; cin>>n;
        for (int i=1; i<=n; i++)
        {
            int gt;
            cout<<"Gia tri cho nut thu "<<i<<" = "; cin>>gt;
            nut *tg = new nut();
            tg->a = gt;
            tg->tiep=NULL;
            if (dau == NULL)
            {
                dau = tg; cuoi=tg;
            }
            else
            {
                cuoi->tiep =tg;
                cuoi=tg;
            }
        }
    }
    void xoa(int y)
        //xoa nut dau tien co gia tri bang y khoi danh sach
    {
        nut *tg;
        tg=dau;
        int dem=0;
        while (tg!=NULL)
        {
            dem=dem +1;
            if (tg->a ==y)
                break;
            tg=tg->tiep;
        }
    }

```

```

        }
    if (tg->a ==y)
    {nut *tg1;
    tg1=dau;
    if (dem==1)
        dau=tg->tiep;
    else
    while (dem >2)
    {
        tg1=tg1->tiep;
        dem=dem-1;
    }
    tg1->tiep = (tg1->tiep) ->tiep;
    }
}
void inds()
{
    nut *tg;
    tg=dau;
    int dem=0;
    cout<<tg->a; tg=tg->tiep;
    while (tg !=NULL)
    {    dem++;
        cout<<" -> "<<tg->a;
        tg=tg->tiep;
    }
}
private:
    nut *dau;
};
int main() {
    dsd x;
    x.nhap();
    cout<<"Danh sach ban dau la: "<<endl;
    x.inds();
    int k;
    cout<<"\nBan can xoa nut co gia tri bao nhieu: "; cin>>k;
    x.xoa(k);
    cout<<"\n Danh sach sau khi xoa la: "<<endl;
    x.inds();
    return 0;
}

```

```
}
```

Ví dụ về kết quả thực hiện chương trình như sau:

```
Nhap so luong nut= 9
Gia tri cho nut thu 1 = 5
Gia tri cho nut thu 2 = 3
Gia tri cho nut thu 3 = 100
Gia tri cho nut thu 4 = 3
Gia tri cho nut thu 5 = 1000
Gia tri cho nut thu 6 = 6
Gia tri cho nut thu 7 = 3
Gia tri cho nut thu 8 = 5
Gia tri cho nut thu 9 = 3
Danh sach ban dau la:
5 -> 3 -> 100 -> 3 -> 1000 -> 6 -> 3 -> 5 -> 3
Ban can xoa nut co gia tri bao nhieu: 1000

Danh sach sau khi xoa la:
5 -> 3 -> 100 -> 3 -> 6 -> 3 -> 5 -> 3
```

Ví dụ 9.14 Cho hai danh sách p và q trong đó mỗi nút chứa thông tin về một số nguyên.

Yêu cầu ghép danh sách q vào sau danh sách p rồi in danh sách p lên màn hình.

Giải: Sau khi nhập dữ liệu cho hai danh sách có nút đầu trở bởi p và q. Việc ghép hai danh sách với nhau sẽ được thực hiện bằng việc nút cuối của danh sách này thay vì chỉ đến NULL sẽ chỉ tới địa chỉ của nút đầu của danh sách kia.

```
#include <iostream>
#include<cmath>
using namespace std;
class dsd{
    struct nut {
        int a;
        nut *tiep;
    };
public:
    int m, n;
    dsd(){
        p = NULL;
        q = NULL;
    }

    void nhap()
    {   nut *cuoi;
        cuoi=NULL;
        cout<<"Nhap so luong nut cua danh sach p = "; cin>>n;
        for (int i=1; i<=n; i++)
        {   int gt;
            cout<<"Gia tri cho nut thu "<<i<<" = "; cin>>gt;
            // ... (rest of the code for input and linking) ...
        }
    }
};
```

```

    nut *tg = new nut();
    tg->a = gt;
    tg->tiep=NULL;

    if (p == NULL)
    {
        p = tg; cuoi=tg;
    }
    else
    {
        cuoi->tiep =tg;
        cuoi=tg;
    }
}
cout<<"Nhap so luong nut cua danh sach q = "; cin>>m;
for (int i=1; i<=m; i++)
{
    int gt;
    cout<<"Gia tri cho nut thu "<<i<<" = "; cin>>gt;
    nut *tg = new nut();
    tg->a = gt;
    tg->tiep=NULL;

    if (q == NULL)
    {
        q = tg; cuoi = tg;
    }
    else
    {
        cuoi->tiep =tg;
        cuoi=tg;
    }
}
}
void ghep()
{
    nut *tg;
    tg=p;
    int dem=0;
    while (1)
    {
        if (tg->tiep ==NULL)

```

```

        {
            tg->tiiep = q;
            break;
        }
        tg=tg->tiiep;
    }
}
void inds()
{
    nut *tg;
    tg=p;
    int dem=0;
    cout<<tg->a; tg=tg->tiiep;
    while (tg !=NULL)
    {
        dem++;
        cout<<" -> "<<tg->a;
        tg=tg->tiiep;
    }
}
private:
    nut *p;
    nut *q;
};

int main() {
    dsd x;
    x.nhap();
    x.ghep();
    cout<<"\n Danh sach sau khi ghep la: "<<endl;
    x.inds();
    return 0;
}

```

Trong hàm `ghep()`, chúng ta sẽ tìm đến nút cuối cùng của danh sách có nút đầu trỏ bởi `p` và cho con trỏ chỉ đến địa chỉ của nút đầu tiên (`q`) trong danh sách còn lại là việc ghép hai danh sách đã hoàn thành. Giao diện kết quả thực hiện chương trình.

Ví dụ về kết quả thực hiện chương trình như sau:

```

Nhap so luong nut cua danh sach p = 5
Gia tri cho nut thu 1 = 5
Gia tri cho nut thu 2 = 10
Gia tri cho nut thu 3 = 15
Gia tri cho nut thu 4 = 20
Gia tri cho nut thu 5 = 25
Nhap so luong nut cua danh sach q = 6
Gia tri cho nut thu 1 = 4
Gia tri cho nut thu 2 = 2
Gia tri cho nut thu 3 = 8
Gia tri cho nut thu 4 = 10
Gia tri cho nut thu 5 = 100
Gia tri cho nut thu 6 = 1000

Danh sach sau khi ghep la:
5 -> 10 -> 15 -> 20 -> 25 -> 4 -> 2 -> 8 -> 10 -> 100 -> 1000

```

Bài tập chương 9

Bài 1. Giải thích ý nghĩa của các câu lệnh sau:

- a) `int a = 5;`
- b) `cout<<&a;`
- c) `int *a;`
- d) `int a=4; int &b=a;`

Bài 2. Trong các khai báo sau đây, khai báo nào là khai báo cho biến con trỏ?

- a) `int a;`
- b) `double *p;`
- c) `float m;`
- d) `char *ch;`
- e) `long *t;`

Bài 3. Sử dụng hàm `sizeof()` xác định số byte dành cho các khai báo một biến nguyên, thực, kí tự trên máy tính của bạn đang sử dụng. Gợi ý: `sizeof(*int)` cho biết số byte sử dụng cho khai báo một con trỏ kiểu nguyên.

Bài 4. Cho một mảng có 10 phần tử gồm 2 3 5 7 11 13 15 17 19 23. Viết chương trình sử dụng kiểu con trỏ khai báo và lưu trữ các giá trị trên và in kết quả lên màn hình.

Bài 5. Việc cấp phát bộ nhớ động với con trỏ có tác dụng gì? Cho ví dụ minh họa.

Bài 6. Thay thế các câu lệnh khai báo biến mảng sau bằng câu lệnh cấp phát bộ nhớ động cho mảng tương ứng:

- a) `int a[100];`

- b) char b[20];
- c) float c[50];
- d) double d[100];

Bài 7. Sử dụng con trỏ để khai báo một mảng có 10 phần tử số nguyên. Tính trung bình cộng của các số chẵn của mảng.

Bài 8. Sử dụng con trỏ để khai báo một mảng có 10 phần tử số nguyên. Đếm, tính tổng, hiển thị các phần tử là số nguyên tố của mảng trên.

Bài 9. Sử dụng con trỏ để khai báo một mảng có 10 phần tử số nguyên. Đếm, tính tổng, hiển thị các phần tử là số chính phương của mảng trên.

Bài 10. Khai báo cấu trúc phù hợp cho bảng chứa thông tin về sinh viên của một trường nào đó sau đây:

Mã số	Họ tên	Ngày sinh	Giới tính	Điểm trung bình
K15001	David	12/02/1997	Nu	7.9
K15002	Linda	16/8/1998	Nam	2.5
K15003	Pieere	7/7/1997	Nu	6.3
K15004	Jain	6/9/1998	Nam	3.9
K15005	Vladimir	15/12/1997	Nu	8.7

Bài 11. Sử dụng cấu trúc đã khai báo trong bài 10. Viết chương trình trên C++ nhập danh sách gồm n sinh viên vào từ bàn phím, sau đó sắp xếp các sinh viên theo điểm trung bình từ cao đến thấp rồi in kết quả lên màn hình.

Bài 12. Sử dụng cấu trúc đã khai báo trong bài 10. Viết chương trình trên C++ nhập danh sách gồm n sinh viên vào từ bàn phím, sau đó hiển thị các sinh viên có điểm trung bình từ 7.0 đến 8.5.

Bài 13. Sử dụng cấu trúc đã khai báo trong bài 10. Viết chương trình trên C++ nhập danh sách gồm n sinh viên vào từ bàn phím, sau đó hiển thị các sinh viên nữ có điểm trung bình đạt loại giỏi (điểm giỏi là điểm lớn hơn hoặc bằng 8.5).

Bài 14. Định nghĩa cấu trúc có tên là ngay_thang gồm các trường ngày, tháng, và năm.

Bài 15. Định nghĩa cấu trúc `sinh_vien` trong bài 10, trong đó có sử dụng cấu trúc `ngay_thang` đã định nghĩa bài 14. Yêu cầu nhập danh sách gồm n sinh viên vào từ bàn phím, sau đó hiển thị các sinh viên nam có năm sinh bằng 1998.

Bài 16. Cho mảng a với các phần tử gồm các thông tin họ tên, giới tính, dân tộc, quê quán, hệ số lương, chức vụ. Yêu cầu nhập mảng a vào từ bàn phím và in các phần tử có thông tin về hệ số lương lớn hơn 3.66.

Bài 17. Cho mảng `sinh_vien` gồm các thông tin mã số sinh viên, họ và tên, giới tính, dân tộc, năm sinh. Yêu cầu nhập mảng `sinh_vien` vào từ bàn phím và hiển thị các sinh viên nữ dân tộc nùng của mảng trên.

Bài 18. Cho mảng `mat_hang` gồm các thông tin mã số mặt hàng, tên mặt hàng, giá hàng, ngày nhập hàng. Yêu cầu nhập mảng `mat_hang` vào từ bàn phím, hiển thị các mặt hàng có giá nhỏ hơn 100.000 đồng.

Bài 19. Viết hàm `day()` có ba tham số `ngay`, `thang`, `nam` theo cấu trúc sau:

```
struct ngay_thang
{
    int ngay;
    int thang;
    int nam;
};
```

Hàm `day()` trả về số ngày từ năm 2000 tới ngày tháng, năm là các số được nhập vào từ bàn phím trong hàm `main()` rồi được truyền cho hàm. Yêu cầu hiển thị số ngày tính được. Quy ước rằng một năm có 365 ngày; một tháng có 30 ngày, số năm nhập vào >2000.

Bài 20. Viết chương trình C++ chứa một danh sách móc nối đơn gồm 10 số nguyên, yêu cầu hiển thị các số trong danh sách này.

Bài 21. Tạo danh sách liên kết đơn có n phần tử, thông tin về mỗi phần tử bao gồm họ tên, năm sinh, dân tộc. In lên màn hình thông tin về những nút có năm sinh từ 1990 trở về trước.

Bài 22. Cho danh sách đơn với mỗi nút chứa một số nguyên. Yêu cầu xóa các nút có giá trị bằng một số k cho trước rồi in danh sách lên màn hình.

Bài 23. Cho danh sách móc nối đơn có nút đầu danh sách trở bởi p có n phần tử, trong đó các phần tử được xếp theo thứ tự tăng dần tính từ nút đầu tiên. Yêu cầu chèn một nút có trường giá trị là một số k nhập từ bàn phím vào danh sách sao cho vẫn giữ nguyên thứ tự tăng dần đã có.

Tài liệu tham khảo

- [1]. Gary J. Bronson, *C++ for Engineers and Scientists*, Thomson, 2006.
- [2]. Bjarne Stroustrup, *The C++ programming language*, Addison-Wesley, fourth edition, 2013
- [3]. Balagurusamy, *Object Oriented Programming With C++*, McGraw-Hill, 2014
- [4]. Phạm văn Ất, *Giáo trình C++ và lập trình hướng đối tượng*, NXB Hồng Đức, 2009.
- [5]. Lê Đăng Hưng, Tạ Tuấn Anh, Nguyễn Hữu Đức, Nguyễn Thanh Thúy, *Lập trình hướng đối tượng với C++*, Nhà xuất bản Khoa học và Kỹ thuật, 2005.
- [6]. Dương Tử Cường, *Ngôn ngữ lập trình C++ từ cơ bản đến hướng đối tượng*, Nhà xuất bản Khoa học và Kỹ thuật, 2005
- [7]. Hoàng Trung Sơn, Bùi Thị Xuyên, *Ngôn ngữ lập trình hướng đối tượng với C++*, Nhà xuất bản Khoa học và Kỹ thuật, 2013.

2.5.4 Định dạng giá trị khi in ra màn hình

Đối với câu lệnh `cout<<` khi in ra màn hình, C++ cung cấp một số phương pháp để định dạng phục vụ cho trình bày số liệu khi in ra các biểu thức chẳng hạn như biểu thức giá trị thực, biểu thức giá trị nguyên,...

- `endl`: xuống dòng mới
- `setw(n)`: định dạng n vị trí canh phải để in giá trị ra màn hình
- `setprecision(m)`: chỉ lấy m số thập phân sau dấu phẩy đối với số thực