HỌC VIỆN KỸ THUẬT QUÂN SỰ
KHOA CÔNG NGHỆ THÔNG TIN

Bài giảng

KỸ THUẬT LẬP TRÌNH

Biên soạn
Hà Đại Dương
Nguyễn Mậu Uyên

Hà Nội - 1/2013

MỤC LỤC

Bài 1 - TỔNG QUAN

Nội dung bài học

I. Giới thiệu
II. Bài toán và việc giải bài toán trên máy tính
III. Công cụ lập trình
IV. Bài tập

I. Giới thiệu

1. Mục đích, Yêu cầu

Mục đích của môn học là cung cấp cho sinh viên những kiến thức cơ bản về kỹ thuật lập trình nói chung và kỹ năng sử dụng công cụ lập trình C/C++ trong việc giải quyết bài toán bằng chương trình phần mềm trên máy tính.
Kết thúc môn học sinh viên được trang bị những kiến thức về:

- Cách thức giải quyết một bài toán trên máy tính;
- Công cụ, kiếm thức về việc thuật toán hóa bài toán;
- Ngôn ngữ lập trình C/C++;
- Một số cấu trúc dữ liệu và giải thuật điển hình.

Yêu cầu đối với sinh viên

- Có hiểu biết cơ bản về cấu trúc máy tính;
- Có hiểu biết cơ bản về hệ điều hành;
- Biết sử dụng các phần mềm hệ thống trong việc quản lý tài nguyên, sao chép/copy dữ liệu;
- Biết sử dụng các phần mền ứng dụng như công cụ soạn thảo văn bản, truy cập internet, web.
- Nghe giảng và làm bài tập.

2. Nội dung môn học

Xem chi tiết ở mục lục.

3. Tài liệu học tập và tham khảo

1. Hà Đại Dương, Nguyễn Mậu Uyên, Tập Bài giảng Lập trình co bản, HVKTQS 2010;
2. Trần Đức Huyên, Phương pháp giải các bài toán trong tin học, Nhà xuất bản giáo dục 1997;
3. Đào Thanh Tĩnh, Hà Đại Dương, Tin học đại cương, Học viện KTQS, 2003;
4. Đỗ Xuân Lôi, Cấu trúc dữ liệu và giải thuật, NXB Giáo dục, 1997;
5. Robert Sedgewick, Algorithns in C++, Addison-Wesley 1992;
6. Niklaus Wirth Bản dịch của Nguyễn Quốc Cường, *Cấu trúc dữ liệu + Giải thuật = Chương trình*, , NXB KHKT, 2001;
7. Giáo trình Tin Học Đại Cương A, Nguyễn Văn Linh, Khoa Công Nghệ Thông Tin, Đại học Cần Thơ, 1991.
8. Giáo trình lý thuyết và bài tập ngôn ngữ C; Nguyễn Đình Tê, Hoàng Đức Hải, Nhà xuất bản Giáo dục, 1999.
9. C - Tham khảo toàn diện, Nguyễn Cẩn, Nhà xuất bản Đồng Nai, 1996.
10. Giúp tự học Lập Trình với ngôn ngữ C, Võ Văn Viện, Nhà xuất bản Đồng Nai, 2002.
11. The C Programming Language, Brain W. Kernighan & Dennis Ritchie, Prentice Hall Publisher, 1988.

II. Bài toán và việc giải bài toán trên máy tính

1. Phương pháp tổng quát để giải một bài toán trên máy tính

Để giải một bài toán trên máy tính cần thực hiện các bước sau:

1. Xác định bài toán;
2. Xác định cấu trúc dữ liệu để mô tả bài toán;
3. Xây dựng thuật toán;
4. Soạn thảo văn bản chương trình, kiểm tra và hoàn thiện chương trình.

2. Xác định bài toán

Khái quát về bài toán

Trong quá trình tồn tại và phát triển, mọi cá nhân luôn phải giải quyết nhiều bài toán đặt ra trong cuộc sống. Có thể nói cuộc sống là một chuỗi các bài toán mà ta phải đối đầu để giải quyết.
Theo nhiều nhà nghiên cứu thì mọi bài toán đều có thể diễn đạt theo một sơ đồ chung như sau:

$$\mathbf{A} \Rightarrow \mathbf{B} \qquad (*)$$

trong đó:

- A là giả thiết, điều kiện ban đầu, thông tin đã cho, đã biết;
- B là kết luận, là mục tiêu cần đạt hoặc cái phải tìm, phải làm ra khi kết thúc bài toán;

$\Rightarrow$ là suy luận, giải pháp cần xác định hoặc chuỗi các thao tác cần thực hiện để có được kết quả B từ cái đã có A.

Xác định bài toán

Theo sơ đồ trên thì việc xác định bài toán có nghĩa là xác định A, B và nếu có thể thì xác định luôn cả các bước thực hiện để "đi" được từ A đến B.

Bài toán trên máy tính

Tương tự như (*), và

- A gọi là đầu vào (INPUT);
- B gọi là đầu ra (OUTPUT);

$\Rightarrow$ là CHƯƠNG TRÌNH MÁY TÍNH cho kết quả B với đầu vào A.

Khó khăn

Việc xác định một bài toán trên máy tính thường gặp khó khăn sau:

- Thông tin về A, B thường không rõ ràng và không đầy đủ;
- Thông báo về điều kiện đặt ra cho cách giải (=>) thường không được nêu ra một cách minh bạch;

Ví dụ 1: Hãy viết chương trình cho phép giải phương trình bậc 2.

A =???, B=???

Ví dụ 2: Giả sử người A có một số tiền X đem gửi tiết kiệm, lãi xuất tháng là L% hỏi rằng sau T tháng thì A có bao nhiêu tiền biết rằng cứ 3 tháng thì tiền lãi được cộng vào gốc.
Ví dụ 3: Bài toán 8 hậu - Hãy tìm cách đặt 8 con hậu trên một bàn cờ vua sao cho không có quân hậu nào có thể ăn quân hậu khác.
Ví dụ 4: Cho dãy số a1, a2, ..., an hãy sắp xếp dãy trên theo thứ tự giảm dần.
Ví dụ 5: Hãy xây dựng hệ thống quản lý hồ sơ và kết quả học tập của sinh viên.

...

A =???, B=???

Nhận xét quan trọng

Việc xác định bài toán là rất rất quan trọng, nó ảnh hưởng tới cách thức và chất lượng của việc giải quyết bài toán;

- Một bài toán cho dù được diễn đạt *chi tiết, rõ ràng* vẫn nên giả định là phần lớn thông tin về A, B là tiềm ẩn trong đầu người giải. Thông tin về A hoặc B thường chỉ là biểu tượng gợi nhớ đến các thông tin tiềm ẩn.
- Bước đầu tiên để xác định một bài toán là phải phát biểu lại bài toán một cách chính xác theo ngôn ngữ của riêng mình vì đó chính là cách tiếp cận bài toán, hiểu bài toán.
- Bước tiếp là tìm hiểu thông tin Input A, Output B và các mối liên hệ giữa chúng;
- Nên xét một vài trường hợp cụ thể để thông qua đó hiểu được cả bài toán , thấy rõ được các thao tác phải làm. Thực tế cho thấy có những bài toán trong tin học chỉ có thể mô tả được thông qua các ví dụ (như:).

3. Cấu trúc dữ liệu và Giải thuật

Cấu trúc dữ liệu

- Trong khoa học máy tính, **cấu trúc dữ liệu** là một cách tổ chức lưu trữ và truy cập dữ liệu trong máy tính sao cho nó có thể được sử dụng một cách hiệu quả (và phụ thuộc cả vào công cụ lập trình).
- Ví dụ (trong C): Mảng (Array), Con trỏ (Pointer), Xâu ký tự (String), File, Stack, Queue
- Thông thường, một cấu trúc dữ liệu được chọn cẩn thận sẽ cho phép thực hiện thuật toán hiệu quả hơn.
- Việc chọn cấu trúc dữ liệu thường bắt đầu từ chọn một cấu trúc dữ liệu trừu tượng. Một cấu trúc dữ liệu được thiết kế tốt cho phép thực hiện nhiều phép toán, sử dụng càng ít tài nguyên, thời gian xử lý và không gian bộ nhớ càng tốt.
- Các cấu trúc dữ liệu được triển khai bằng cách sử dụng các kiểu dữ liệu, các tham chiếu và các phép toán trên đó được cung cấp bởi một ngôn ngữ lập trình.

Thuật toán

- **Thuật toán** , còn gọi là **giải thuật**, là một tập hợp hữu hạn của các chỉ thị hay phương cách được định nghĩa rõ ràng cho việc hoàn tất một số sự việc từ một trạng thái ban đầu cho trước; khi các chỉ thị này được áp dụng triệt để thì sẽ dẫn đến kết quả sau cùng như đã dự đoán.
- Thuật toán là một bộ các qui tắc hay qui trình cụ thể nhằm giải quyết một vấn đề trong một số bước hữu hạn, hoặc nhằm cung cấp một kết quả từ một tập hợp của các dữ kiện đưa vào.

Ví dụ 1: Giả sử có hai bình A và B đựng hai loại chất lỏng khác nhau, A chứa dung dịch Da, B chứa dung dịch Db. Giải thuật để đổi dung dịch Da vào bình B và Db vào A là:

Yêu cầu phải có thêm một bình thứ ba gọi là bình C.
Bước 1: Đổ dung dịch Db vào bình C;
Bước 2: Đổ dung dịch Da vào bình B;
Bước 3: Đổ dung dịch Db vào bình A

Ví dụ 2: Một trong những giải thuật tìm ước chung lớn nhất của hai số a và b là:

Bước 1: Nhập vào hai số a và b.
Bước 2: So sánh 2 số a,b chọn số nhỏ nhất gán cho UCLN.
Bước 3: Nếu hai số a và b chia hết cho UCLN thì
  Thực hiện bước 5.
Bước 4: Giảm UCLN một đơn vị và quay lại bước 3
Bước 5: In UCLN - Kết thúc.

Một thuật toán có các tính chất sau:

- *Tính chính xác*: để đảm bảo kết quả tính toán hay các thao tác mà máy tính thực hiện được là chính xác.
- *Tính rõ ràng*: Thuật toán phải được thể hiện bằng các câu lệnh minh bạch; các câu lệnh được sắp xếp theo thứ tự nhất định.

- *Tính khách quan*: Một thuật toán dù được viết bởi nhiều người trên nhiều máy tính vẫn phải cho kết quả như nhau.
- *Tính phổ dụng*: Thuật toán không chỉ áp dụng cho một bài toán nhất định mà có thể áp dụng cho một lớp các bài toán có đầu vào tương tự nhau.
- *Tính kết thúc*: Thuật toán phải gồm một số hữu hạn các bước tính toán.

Trình tự thực hiện các bước của thuật toán

Giải thuật được thiết kế theo ba cấu trúc suy luận cơ bản sau đây:

1. **Tuần tự (Sequential):** Các công việc được thực hiện một cách tuần tự, công việc này nối tiếp công việc kia.
2. **Cấu trúc lựa chọn (Selection) :** Lựa chọn một công việc để thực hiện căn cứ vào một điều kiện nào đó. Có một số dạng như sau:
   - Cấu trúc 1: Nếu < điều kiện> (đúng) thì thực hiện <công việc>
   - Cấu trúc 2: Nếu < điều kiện> (đúng) thì thực hiện <công việc 1>, ngược lại (điều kiện sai) thì thực hiện <công việc 2>
   - Cấu trúc 3: Trường hợp < i> thực hiện <công việc i>
3. **Cấu trúc lặp (Repeating):** Thực hiện lặp lại một công việc không hoặc nhiều lần căn cứ vào một điều kiện nào đó. Có hai dạng như sau:
   - Lặp xác định: là loại lặp mà khi viết chương trình, người lập trình đã xác định được công việc sẽ lặp bao nhiêu lần.
   - Lặp không xác định: là loại lặp mà khi viết chương trình người lập trình chưa xác định được công việc sẽ lặp bao nhiêu lần. Số lần lặp sẽ được xác định khi chương trình thực thi.

Biểu diễn thuật giải

Ngôn ngữ tự nhiên: Ngôn ngữ tự nhiên là ngôn ngữ của chúng ta đang sử dụng, chúng ta có thể sử dụng ngôn ngữ tự nhiên để mô tả giải thuật giống như các ví dụ ở trên.

Ví dụ: Ta có giải thuật giải phương trình bậc nhất dạng $ax + b = 0$ như sau:

Bước 1: Nhận giá trị của các tham số a, b

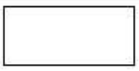
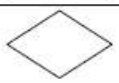
Bước 2: Xét giá trị của a xem có bằng 0 hay không?

Nếu a=0 thì làm bước 3, nếu a khác không thì làm bước 4.

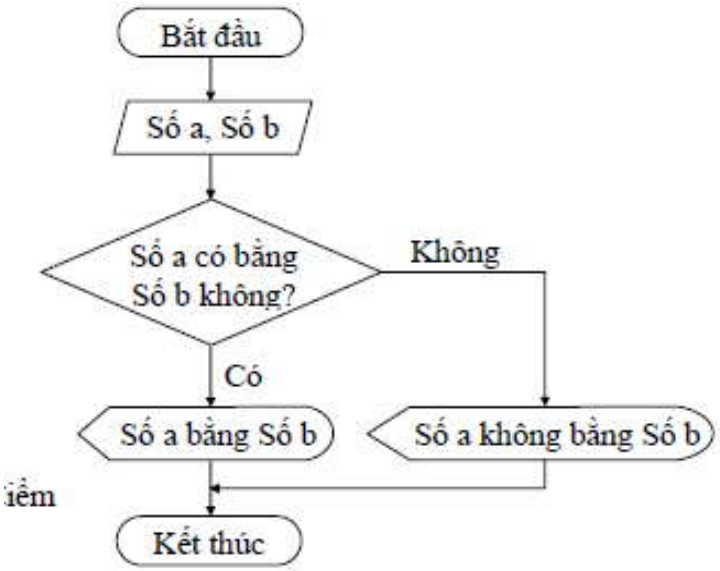
Bước 3: (a bằng 0) Nếu b bằng 0 thì ta kết luận phương trình vô số nghiệm,

nếu b khác 0 thì ta kết luận phương trình vô nghiệm.

Bước 4: (a khác 0) Ta kết luận phương trình có nghiệm x=-b/a

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

Lưu đồ thuật toán: Ngôn ngữ sơ đồ (lưu đồ) là một ngôn ngữ đặc biệt dùng để mô tả giải thuật bằng các sơ đồ hình khối. Mỗi khối qui định một hành động như mô tả ở hình trên. Ví dụ: So sánh 2 số.



Giả mã: (tiếng Anh: *Pseudocode*, xuất phát từ chữ pseudo và code) là một bản mô tả giải thuật ngắn gọn và không chính thức, trong đó sử dụng những quy ước có cấu trúc của một số ngôn ngữ lập trình (thường là Pascal) nhưng thường bỏ đi những chi tiết không cần thiết để giúp hiểu rõ giải thuật hơn.

***Ví dụ*:** Thuật giải phương trình bậc 2

```
Vào: a,b,c
Ra: Kết luận về nghiệm
BEGIN
 Delta: = b*b - 4*a*c;
   If Delta=0 Then
      Phương trình có nghiệm kép x=-b/(2*a);
   else
      begin
           if Delta<0 then
             Phương trình Vô nghiệm
           Else
              Begin
                Phương trình có 2 nghiệm
                x1=(-b+sqrt(Delte))/(2*a)
                x2=(-b+sqrt(Delte))/(2*a)
              end
      end
 END.
```

Một số ví dụ khác về thuật toán và biểu diễn thuật toán

Ví dụ 1: Cần viết chương trình cho máy tính sao cho khi thực hiện chương trình đó, máy tính yêu cầu người sử dụng chương trình nhập vào các số hạng của tổng (n); nhập vào dãy các số hạng a_i của tổng. Sau đó, máy tính sẽ thực hiện việc tính tổng các số a_i này và in kết quả của tổng tính được.

Yêu cầu: Tính tổng n số $S=a_1+a_2+a_3+.....+a_n$.

Chi tiết giải thuật được mô tả bằng ngôn ngữ tự nhiên như sau:

- Bước 1: Nhập số các số hạng n.
- Bước 2: Cho S=0 (lưu trữ số 0 trong S)
- Bước 3: Cho i=1 (lưu trữ số 1 trong i)
- Bước 4: Kiểm tra nếu i<=n thì thực hiện bước 5, ngược lại thực hiện bước 8.
- Bước 5: Nhập a_i
- Bước 6: Cho $S=S+a_i$ (lưu trữ giá trị S + a_i trong S)
- Bước 7: Tăng i lên 1 đơn vị và quay lại bước 4.
- Bước 8: In S và kết thúc chương trình.

Ví dụ 2: Viết chương trình cho phép nhập vào 2 giá trị a, b mang ý nghĩa là các hệ số a, b của phương trình bậc nhất. Dựa vào các giá trị a, b đó cho biết nghiệm của phương trình bậc nhất ax + b = 0. Mô tả giải thuật bằng ngôn ngữ tự nhiên:

- Bước 1: Nhập 2 số a và b
- Bước 2: Nếu a = 0 thì thực hiện bước 3, ngược lại thực hiện bước 4
- Bước 3: Nếu b=0 thì thông báo phương trình vô số nghiệm và kết thúc chương trình, ngược lại thông báo phương trình vô nghiệm và kết thúc chương trình.
- Bước 4: Thông báo nghiệm của phương trình là –b/a và kết thúc.

Ví dụ 3: Viết chương trình cho phép nhập vào 1 số n, sau đó lần lượt nhập vào n giá trị a_1, a_2,…,a_n. Hãy tìm và in ra giá trị lớn nhất trong n số a_1, a_2, …, a_n. Mô tả giải thuật bằng ngôn ngữ tự nhiên:

- Bước 1: Nhập số n
- Bước 2: Nhập số thứ nhất a_1
- Bước 3: Gán $max=a_1$
- Bước 4: Gán i=2
- Bước 5: Nếu i<=n thì thực hiện bước 6, ngược lại thực hiện bước 9
- Bước 6: Nhập a_i
- Bước 7: Nếu $max < a_i$ thì gán $max=a_i$.
- Bước 8: Tăng i lên một đơn vị và quay lại bước 5
- Bước 9: In max - kết thúc

Ví dụ 4: Viết chương trình cho phép nhập vào 1 số n, sau đó lần lượt nhập vào n giá trị a_1, a_2,…,a_n. Sắp theo thứ tự tăng dần một dãy n số a_1, a_2,...a_n nói trên. Có rất nhiều giải thuật để giải quyết bài toán này. Phần trình bày dưới đây là một phương pháp.

Giả sử ta đã nhập vào máy dãy n số a_1, a_2,..., a_n. Việc sắp xếp dãy số này trải qua (n-1) lần:

- Lần 1: So sánh phần tử đầu tiên với tất cả các phần tử đứng sau phần tử đầu tiên. Nếu có phần tử nào nhỏ hơn phần tử đầu tiên thì đổi chỗ phần tử đầu tiên với phần tử nhỏ hơn đó. Sau lần 1, ta được phần tử đầu tiên là phần tử nhỏ nhất.
- Lần 2: So sánh phần tử thứ 2 với tất cả các phần tử đứng sau phần tử thứ 2. Nếu có phần tử nào nhỏ hơn phần tử thứ 2 thì đổi chỗ phần tử thứ 2 với phần tử nhỏ hơn đó. Sau lần 2, ta được phần tử đầu tiên và phần tử thứ 2 là đúng vị trí của nó khi sắp xếp.
- …
- Lần (n-1): So sánh phần tử thứ (n-1) với phần tử đứng sau phần tử (n-1) là phần tử thứ n. Nếu phần tử thứ n nhỏ hơn phần tử thứ (n-1) thì đổi chỗ 2 phần tử này. Sau lần thứ (n-1), ta được danh sách gồm n phần tử được sắp thứ tự.

Mô tả giải thuật bằng ngôn ngữ tự nhiên:

- Bước 1: Gán i=1
- Bước 2: Gán j=i+1
- Bước 3: Nếu i <=n-1 thì thực hiện bước 4, ngược lại thực hiện bước 8

- Bước 4: Nếu j <=n thì thực hiện bước 5, ngược lại thì thực hiện bước 7.
- Bước 5: Nếu $a_i > a_j$ thì hoán đổi a_i và a_j cho nhau (nếu không thì thôi).
- Bước 6: Tăng j lên một đơn vị và quay lại bước 4
- Bước 7: Tăng i lên một đơn vị và quay lại bước 3
- Bước 8: In dãy số $a_1, a_2,..., a_n$ - Kết thúc.

Giải thuật + Cấu trúc dữ liệu = Chương trình

Algorithms + Data Structures = Programs

Niklaus Emil Wirth
In 1975 he wrote the book "Algorithms + Data Structures = Programs", which gained wide recognition and is still useful today.

Biography

Wirth was born in Winterthur, Switzerland, in 1934. In 1959 he earned a degree in Electronics Engineering from the Swiss Federal Institute of Technology Zürich (ETH Zürich). In 1960 he earned an M.Sc. from Université Laval, Canada. Then in 1963 he was awarded a Ph.D.in EECS from the University of California, Berkeley, supervised by the computer designer pioneer Harry Huskey.
From 1963 to 1967 he served as assistant professor of Computer Science at Stanford University and again at the University of Zurich. Then in 1968 he became Professor of Informatics at ETH Zürich, taking two one-year sabbaticals at Xerox PARC in California (1976–1977 and 1984–1985). Wirth retired in 1999.

Programming languages

Wirth was the chief designer of the programming languages Euler, Algol W, Pascal, Modula, Modula-2, Oberon, Oberon-2, and Oberon-07 . He was also a major part of the design and implementation team for the Lilith and Oberon operating systems, and for the Lola digital hardware design and simulation system. He received the *ACM Turing Award* for the development of these languages and in 1994 he was inducted as a Fellow of the ACM. He designed the simple programming language PL/0 to illustrate compiler design. It has formed the basis for many university compiler design classes.

4. Chương trình

Định nghĩa

- Chương trình máy tính: là một tập hợp những câu lệnh được viết bằng một ngôn ngữ lập trình theo một trật tự xác định nhằm tự động thực hiện một số nhiệm vụ hoặc chức năng hoặc giải quyết một bài toán nào đó.
- Là một biểu hiển cụ thể của một giải thuật trên một ngôn ngữ lập trình nào đó cùng với những mô tả về cấu trúc dữ liệu mô tả đầu vào, đầu ra của bài toán cần giải quyết.

Ví dụ 1: Chương trình giải phương trình bậc 2 bằng Pascal
Ví dụ 2: Chương trình giải phương trình bậc 2 bằng C

Tạo ra chương trình máy tính bằng cách nào???

III. Công cụ lập trình

1. Ngôn ngữ lập trình

Khái niệm

- Ngôn ngữ lập trình là một ngôn ngữ dùng để viết chương trình cho máy tính. Ta có thể chia ngôn ngữ lập trình thành các loại sau: ngôn ngữ máy, hợp ngữ và ngôn ngữ cấp cao.
- Ngôn ngữ máy (machine language): Là các chỉ thị dưới dạng nhị phân, can thiệp trực tiếp vào trong các mạch điện tử. Chương trình được viết bằng ngôn ngữ máy thì có thể được thực hiện ngay không cần qua bước trung gian nào. Tuy nhiên chương trình viết bằng ngôn ngữ máy dễ sai sót, cồng kềnh và khó đọc, khó hiểu vì toàn những con số 0 và 1.
- Hợp ngữ (assembly language): Bao gồm tên các câu lệnh và quy tắc viết các câu lệnh đó. Tên các câu lệnh bao gồm hai phần: phần mã lệnh (viết tựa tiếng Anh) chỉ phép toán cần thực hiện và địa chỉ chứa toán hạng của phép toán đó. Ví dụ:

  INPUT a ; Nhập giá trị cho a từ bàn phím
  LOAD a ; Đọc giá trị a vào thanh ghi tổng A
  PRINT a; Hiển thị giá trị của a ra màn hình.
  INPUT b
  ADD b; Cộng giá trị của thanh ghi tổng A với giá trị b

  Trong các lệnh trên thì INPUT, LOAD, PRINT, ADD là các mã lệnh còn a, b là địa chỉ. Để máy thực hiện được một chương trình viết bằng hợp ngữ thì chương trình đó phải được dịch sang ngôn ngữ máy. Công cụ thực hiện việc dịch đó được gọi là Assembler.
- Ngôn ngữ cấp cao (High level language): Ra đời và phát triển nhằm phản ánh cách thức người lập trình nghĩ và làm. Rất gần với ngôn ngữ con người (Anh ngữ) nhưng chính xác như ngôn ngữ toán học. Cùng với sự phát triển của các thế hệ máy tính, ngôn ngữ lập trình cấp cao cũng được phát triển rất đa dạng và phong phú, việc lập trình cho máy tính vì thế mà cũng có nhiều khuynh hướng khác nhau: lập trình cấu trúc, lập trình hướng đối tượng, lập trình logic, lập trình hàm... Một chương trình viết bằng ngôn ngữ cấp cao được gọi là chương trình nguồn (source programs). Để máy tính "hiểu" và thực hiện được các lệnh trong chương trình nguồn thì phải có một chương trình dịch để dịch chương trình nguồn (viết bằng ngôn ngữ cấp cao) thành dạng chương trình có khả năng thực thi.

Chương trình dịch

- Như trên đã trình bày, muốn chuyển từ chương trình nguồn sang chương trình đích phải có chương trình dịch. Thông thường mỗi một ngôn ngữ cấp cao đều có một chương trình dịch riêng nhưng chung quy lại thì có hai cách dịch: thông dịch và biên dịch.
- Thông dịch (interpreter): Là cách dịch từng lệnh một, dịch tới đâu thực hiện tới đó. Chẳng hạn ngôn ngữ LISP sử dụng trình thông dịch.
- Biên dịch (compiler): Dịch toàn bộ chương trình nguồn thành chương trình đích rồi sau đó mới thực hiện. Các ngôn ngữ sử dụng trình biên dịch như Pascal, C...
- Giữa thông dịch và biên dịch có khác nhau ở chỗ: Do thông dịch là vừa dịch vừa thực thi chương trình còn biên dịch là dịch xong toàn bộ chương trình rồi mới thực thi nên chương trình viết bằng ngôn ngữ biên dịch thực hiện nhanh hơn chương trình viết bằng ngôn ngữ thông dịch.
- Một số ngôn ngữ sử dụng kết hợp giữa thông dịch và biên dịch chẳng hạn như Java. Chương trình nguồn của Java được biên dịch tạo thành một chương trình đối tượng (một dạng mã trung gian) và khi thực hiện thì từng lệnh trong chương trình đối tượng được thông dịch thành mã máy.

2. Công cụ lập trình

- Trước những năm 1990 người ta cho rằng ngôn ngữ lập trình quyết định kết quả lập trình. Chẳng hạn, trong một tình huống cụ thể nào đó, chương trình viết bằng C++ thì tốt hơn Pascal, viết bằng Pascal thì tốt hơn Fortran… Khi các công cụ lập trình còn thô sơ và các yêu cầu phần mềm chưa cao thì nhận định này là khá chính xác.

- Sau đó người ta cho rằng công nghệ lập trình mới ảnh hưởng lớn nhất đến sản phẩm cuối cùng, sự thống trị trong thập kỷ 90 của lập trình hướng đối tượng và RAD (viết tắt của Rapid Application Development nghĩa là Công cụ phát triển ứng dụng nhanh, thường gọi là lập trình trực quan hay Visual Programming) đã cho thấy tư duy của người lập trình bị ảnh hưởng bởi nền tảng phát triển phần mềm.
- Không ai phê phán Delphi - phiên bản phát triển từ ngôn ngữ Pascal là kém hơn Java hay Visual C++. Tuy mới có 1/20 thời gian của thế kỷ 21 trôi qua nhưng từ đầu thế kỷ đến nay Công nghệ thông tin đã có những bước phát triển mạnh mẽ trong đó có Công nghệ phần mềm. Nhu cầu sử dụng phần mềm và yêu cầu đối với phần mềm đột nhiên tăng vọt khiến nhiều nhà phát triển phần mềm phải xem lại cách làm việc của mình. Đó chính là cơ sở cho sự phát triển rộng rãi trên toàn thế giới của ngành Công nghiệp phần mềm hiện nay.
- Người ta không chỉ quan tâm đến công nghệ lập trình mà còn quan tâm đến quy trình phát triển phần mềm.
- Các công ty bắt đầu chuẩn hóa và đưa ra quy trình công nghệ phần mềm của mình - ở đó việc lựa chọn bộ công cụ lập trình có vai trò rất quan trọng. Các bộ công cụ lập trình hiện nay có xu hướng thống nhất và tương tác với nhau chặt chẽ. Ý tưởng này đã từng xuất hiện trong thập kỷ trước, tiêu biểu như CORBA của Sun hay Delphi - C++ Builder Project Union của Borland, tuy nhiên khi đó chúng chưa được ưa chuộng.
- Khi Visual Studio.NET của Microsoft ra đời năm 2002, người ta nhận thấy rằng các công cụ lập trình nên đi với nhau thành "bộ". Đến thời điểm hiện nay giải pháp về công cụ lập trình của các nhà phát triển luôn được nhắc đến như "bộ công cụ".
- Hiện nay có rất nhiều ngôn ngữ lập trình (NNLT) và rất nhiều công cụ lập trình (CCLT). Một CCLT có thể gắn liền với một NNLT hoặc không - đây là điều mà một số ít người không có kinh nghiệm không hề biết. Ta có thể phân loại:
    - Theo NNLT: Dòng C có Visual C++, C++ Builder… Dòng Pascal có Borland Pascal, Delphi…
    - Theo phạm vi sử dụng: Dòng lập trình hệ thống có Microsoft Assembly, Borland C… Dòng lập trình trực quan có Visual Basic, Jbuilder… Dòng lập trình mạng có Java, ASP, PHP…
    - Theo phong cách lập trình: Dòng cổ điển có Pascal, Fortran… Dòng hướng đối tượng có C++, SmallTalk, Java…
- Công cụ thường dùng:
    - Microsoft Visual Studio 2005, Microsoft Visual Studio 2008: C++, C##, VisualBasic, ASPX...
    - Java;
    - PHP;
    - Pascal, Turbo C, Dev - C++
- Chức năng của công cụ lập trình:
    1. Biểu diễn chương trình bằng một ngôn ngữ lập trình nào đó
    2. Giao diện tích hợp (IDE - Itergrated Developments Environment) cho phép soạn thảo văn bản chương trình, kiểm lỗi, thử nghiệm;
    3. Biên dịch thành chương trình độc lập trên máy tính.

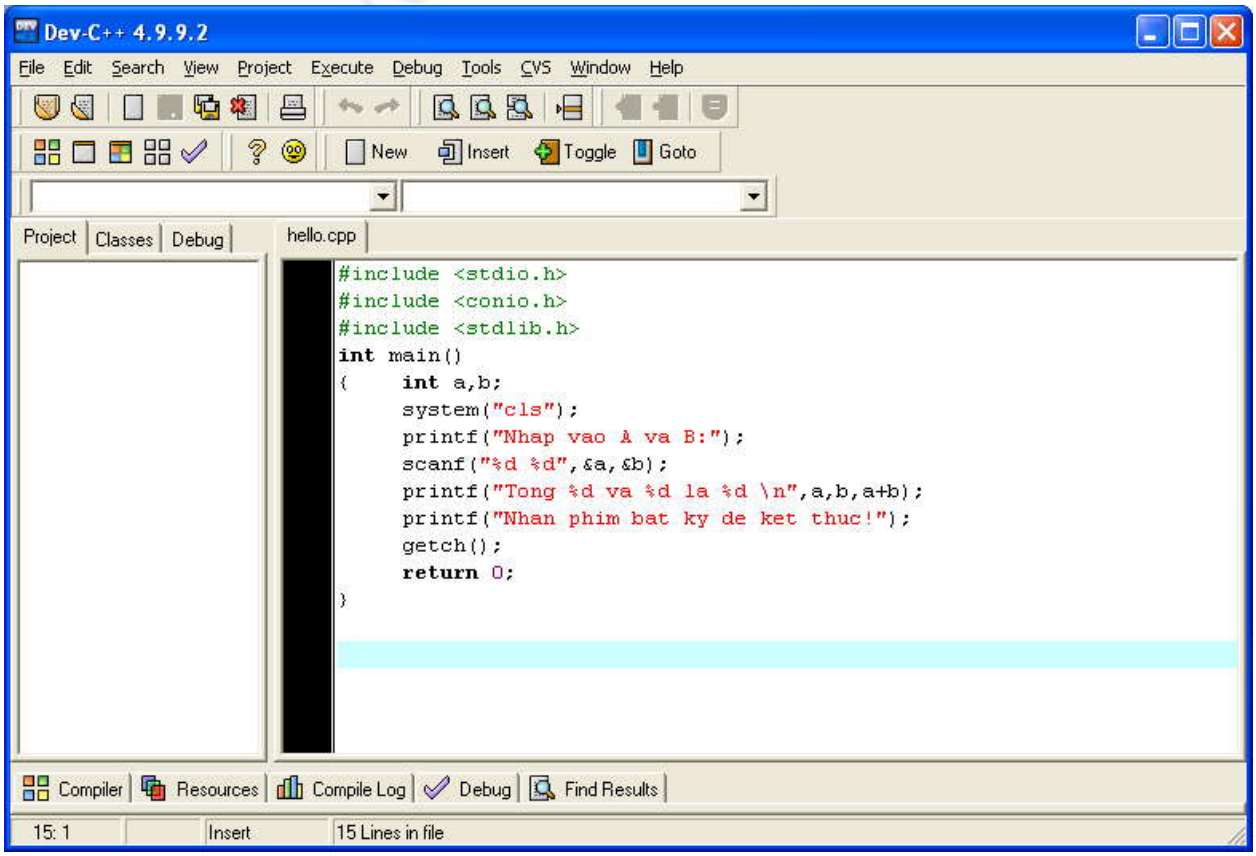
3. Công cụ lập trình Dev-C++

- **Dev-C++** là một công cụ lập trình với giao diện tích hợp cho phép làm việc trên Windows, cho phép lập trình bằng C/C++;

- Dự án phát triển Dev-C++ được lưu trữ trên SourceForge. Dev-C++ nguyên được phát triển bởi một lập trình viên có tên là Colin Laplace và chỉ chạy trên hệ điều hành Microsoft Windows.
- Bloodshed Dev-C++ là một Môi trường Phát triển Tích hợp (IDE) có hỗ trợ đầy đủ tính năng cho ngôn ngữ lập trình C/C++. Nó sử dụng trình MinGW của GCC (Bộ trình dịch GNU) làm trình biên dịch. Dev-C++ cũng có thể được dùng kết hợp với Cygwin hay bất kỳ trình dịch nền tảng GCC nào khác.
- Chương trình cho người dùng có cảm nhận hơi giống với hình thức của chương trình Microsoft Visual Studio vốn được sử dụng rộng rãi hơn. Dev-C++ có một đặc điểm phụ đó là nó sử dung DevPaks, là một phần gồm các gói mở rộng so với môi trường tiêu chuẩn, bao gồm các thư viện, mẫu, và các tiện ích được đưa thêm vào. DevPaks thường có, nhưng không nhất định, tiện ích GUI (giao diện người dùng đồ họa), bao gồm các công cụ phổ biến như GTK+, wxWidgets, và FLTK. Có những DevPaks có chứa các thư viện với thậm chí nhiều hàm chức năng cao hơn.
- Dev-C++ nói chung là một chương trình chỉ chạy trên Windows. Tuy nhiên cũng có một phiên bản cho Linux, nhưng vẫn trong giai đoạn alpha và chưa được cập nhật trong vòng hơn 6 năm qua.

Một số giao diện

Giao diện chính



IV. Tóm tắt nội dung bài học

I. Giới thiệu

1. Mục đích, Yêu cầu
2. Nội dung môn học
3. Tài liệu học tập và tham khảo
4. Hình thức đánh giá kết quả môn học

5. Tổng quan về máy tính

II. Bài toán và việc giải bài toán trên máy tính

1. Xác định bài toán
2. Cấu trúc dữ liệu và Giải thuật
3. Chương trình

III. Công cụ lập trình

1. Ngôn ngữ lập trình
2. Công cụ lập trình
3. Công cụ lập trình Dev-C++

V. Bài tập

1. Biểu diễn thuật toán giải các bài toán sau bằng sơ đồ khối và giả mã:
   a. Giải phương trình bậc 2;
   b. Giải hệ phương trình bậc nhất;
   c. Tính số ngày của một tháng một năm nào đó;
   d. Thuật toán tìm ước số chung lớn nhất;
2. Cài đặt, làm quen với giao diện Dev-C++
3. Tìm hiểu những bài toán thực tế, mô tả thuật toán để giải quyết vấn đề đó.

Bài 2 - NGÔN NGỮ LẬP TRÌNH C/C++

Nội dung bài học

I. Giới thiệu
II. Một số khái niệm cơ bản
III. Cấu trúc chương trình đơn giản trong C
IV. Nhập/Xuất dữ liệu
V. Bài tập

I. Giới thiệu

- C là ngôn ngữ lập trình cấp cao, được sử dụng rất phổ biến để lập trình hệ thống cùng với Assembler và phát triển các ứng dụng.
- Vào những năm cuối thập kỷ 60 đầu thập kỷ 70 của thế kỷ XX, Dennish Ritchie (làm việc tại phòng thí nghiệm Bell) đã phát triển ngôn ngữ lập trình C dựa trên ngôn ngữ BCPL (do Martin Richards đưa ra vào năm 1967) và ngôn ngữ B (do Ken Thompson phát triển từ ngôn ngữ BCPL vào năm 1970 khi viết hệ điều hành UNIX đầu tiên trên máy PDP-7) và được cài đặt lần đầu tiên trên hệ điều hành UNIX của máy DEC PDP-11.
- Năm 1978, Dennish Ritchie và B.W Kernighan đã cho xuất bản quyển "Ngôn ngữ lập trình C" và được phổ biến rộng rãi đến nay.
- Lúc ban đầu, C được thiết kế nhằm lập trình trong môi trường của hệ điều hành Unix nhằm mục đích hỗ trợ cho các công việc lập trình phức tạp. Nhưng về sau, với những nhu cầu phát triển ngày một tăng của công việc lập trình, C đã vượt qua khuôn khổ của phòng thí nghiệm Bell và nhanh chóng hội nhập vào thế giới lập trình để rồi các công ty lập trình sử dụng một cách rộng rãi. Sau đó, các công ty sản xuất phần mềm lần lượt đưa ra các phiên bản hỗ trợ cho việc lập trình bằng ngôn ngữ C và chuẩn ANSI C cũng được khai sinh từ đó.
- Ngôn ngữ lập trình C là một ngôn ngữ lập trình hệ thống rất mạnh và rất "mềm dẻo", có một thư viện gồm rất nhiều các hàm (function) đã được tạo sẵn. Người lập trình có thể tận dụng các hàm này để giải quyết các bài toán mà không cần phải tạo mới. Hơn thế nữa, ngôn ngữ C hỗ trợ rất nhiều phép toán nên phù hợp cho việc giải quyết các bài toán kỹ thuật có nhiều công thức phức tạp. Ngoài ra, C cũng cho phép người lập trình tự định nghĩa thêm các kiểu dữ liệu trừu tượng khác. Tuy nhiên, điều mà người mới vừa học lập trình C thường gặp "rắc rối" là "hơi khó hiểu" do sự "mềm dẻo" của C. Dù vậy, C được phổ biến khá rộng rãi và đã trở thành một công cụ lập trình khá mạnh, được sử dụng như là một ngôn ngữ lập trình chủ yếu trong việc xây dựng những phần mềm hiện nay
- Ngôn ngữ C có những đặc điểm cơ bản sau:
    - Tính cô đọng (compact): C chỉ có 32 từ khóa chuẩn và 40 toán tử chuẩn, nhưng hầu hết đều được biểu diễn bằng những chuỗi ký tự ngắn gọn.
    - Tính cấu trúc (structured): C có một tập hợp những chỉ thị của lập trình như cấu trúc lựa chọn, lặp… Từ đó các chương trình viết bằng C được tổ chức rõ ràng, dễ hiểu.
    - Tính tương thích (compatible): C có bộ tiền xử lý và một thư viện chuẩn vô cùng phong phú nên khi chuyển từ máy tính này sang máy tính khác các chương trình viết bằng C vẫn hoàn toàn tương thích.
    - Tính linh động (flexible): C là một ngôn ngữ rất uyển chuyển và cú pháp, chấp nhận nhiều cách thể hiện, có thể thu gọn kích thước của các mã lệnh làm chương trình chạy nhanh hơn.
    - Biên dịch (compile): C cho phép biên dịch nhiều tập tin chương trình riêng rẽ thành các tập tin đối tượng (object) và liên kết (link) các đối tượng đó lại với nhau thành một chương trình có thể thực thi được (executable) thống nhất.

II. Một số khái niệm cơ bản

1. Bộ ký tự

Bộ chữ viết trong ngôn ngữ C bao gồm những ký tự, ký hiệu sau: (phân biệt chữ in hoa và in thường):

- 26 chữ cái latinh lớn A,B,C...Z ;
- 26 chữ cái latinh nhỏ a,b,c ...z. ;
- 10 chữ số thập phân 0,1,2...9. ;
- Các ký hiệu toán học: +, -, *, /, =, <, >, (,)
- Các ký hiệu đặc biệt: :. , ; " ' _ @ # $! ^ [ ] { } ... ;
- Dấu cách hay khoảng trống.

2. Từ khóa

Từ khóa là các từ dành riêng (reserved words) của C với mục đích đã được xác định trước. Ví dụ:

- Từ khóa **int** dùng để khai báo biến hoặc hàm với kiểu dữ liệu số nguyên;
- Từ khóa **if** dùng để xây dựng câu lệnh cấu trúc chọn;
- Từ khóa **for** dùng để xây dựng câu lệnh cấu trúc lặp.

Danh sách các từ khóa:

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

☞ Các từ khóa phải viết bằng *chữ thường*

Lưu ý: không được dùng từ khóa vào mục đích khác, hoặc đặt tên một đối tượng nào đó (biến, hằng, tên hàm ...) trùng với từ khóa. Các từ khóa của Turbo C 3.0 bao gồm:

3. Kiểu dữ liệu

Các kiểu dữ liệu sơ cấp chuẩn trong C có thể được chia làm 2 dạng : kiểu số nguyên, kiểu số thực.

Kiểu số nguyên

Kiểu số nguyên là kiểu dữ liệu dùng để lưu các giá trị nguyên hay còn gọi là kiểu đếm được. Kiểu số nguyên trong C được chia thành các kiểu dữ liệu con, mỗi kiểu có một miền giá trị khác nhau.

Kiểu số nguyên 1 byte (8 bits): Kiểu số nguyên một byte gồm có 2 kiểu sau:

STT	Kiểu dữ liệu	Miền giá trị (Domain)
1	unsigned char	Từ **0** đến **255** (tương đương 256 ký tự trong bảng mã ASCII)
2	char	Từ **-128** đến **127**

Kiểu unsigned char: lưu các số nguyên dương từ 0 đến 255.

⇒ Để khai báo một biến là kiểu ký tự thì ta khai báo biến kiểu unsigned char. Mỗi số trong miền giá trị của kiểu unsigned char tương ứng với một ký tự trong bảng mã ASCII .

Kiểu char: lưu các số nguyên từ -128 đến 127. Kiểu char sử dụng bit trái nhất để làm bit dấu.

⇒ Nếu gán giá trị > 127 cho biến kiểu char thì giá trị của biến này có thể là số âm (?).

Kiểu số nguyên 2 bytes (16 bits): Kiểu số nguyên 2 bytes gồm có 4 kiểu sau:

STT	Kiểu dữ liệu	Miền giá trị (Domain)
1	enum	Từ -32,768 đến 32,767

2	unsigned int	Từ **0** đến **65,535**
3	Short int	Từ -32,768 đến 32,767
4	int	Từ -32,768 đến 32,767

Kiểu enum, short int, int : Lưu các số nguyên từ -32768 đến 32767. Sử dụng bit bên trái nhất để làm bit dấu.

⇒ Nếu gán giá trị >32767 cho biến có 1 trong 3 kiểu trên thì giá trị của biến này có thể là số âm.

Kiểu unsigned int: Kiểu unsigned int lưu các số nguyên dương từ 0 đến 65535.

Kiểu số nguyên 4 byte (32 bits): Kiểu số nguyên 4 bytes hay còn gọi là số nguyên dài (long) gồm có 2 kiểu sau:

STT	Kiểu dữ liệu	Miền giá trị (Domain)
1	unsigned long	Từ **0** đến **4,294,967,295**
2	long	Từ **-2,147,483,648** đến **2,147,483,647**

Kiểu long : Lưu các số nguyên từ -2147483658 đến 2147483647. Sử dụng bit bên trái nhất để làm bit dấu.

⇒ Nếu gán giá trị >2147483647 cho biến có kiểu long thì giá trị của biến này có thể là số âm.

Kiểu unsigned long: Kiểu unsigned long lưu các số nguyên dương từ 0 đến 4294967295

Kiểu số thực

Kiểu số thực dùng để lưu các số thực hay các số có dấu chấm thập phân gồm có 3 kiểu sau:

STT	Kiểu dữ liệu	Kích thước (Size)	Miền giá trị (Domain)
1	float	4 bytes	Từ **3.4 * 10^{-38}** đến **3.4 * 10^{38}**
2	double	8 bytes	Từ **1.7 * 10^{-308}** đến **1.7 * 10^{308}**
3	long double	10 bytes	Từ **3.4 *10^{-4932}** đến **1.1 *10^{4932}**

Mỗi kiểu số thực ở trên đều có miền giá trị và độ chính xác (số số lẻ) khác nhau. Tùy vào nhu cầu sử dụng mà ta có thể khai báo biến thuộc 1 trong 3 kiểu trên.

Ngoài ra ta còn có kiểu dữ liệu **void**, kiểu này mang ý nghĩa là kiểu rỗng không chứa giá trị gì cả.

4. Tên, Biến, hằng

Tên

Tên hay còn gọi là danh biểu (identifier) được dùng để đặt cho chương trình, hằng, kiểu, biến, chương trình con... Tên có hai loại là tên chuẩn và tên do người lập trình đặt.

- Tên chuẩn là tên do C đặt sẵn như tên kiểu: int, char, float,…; tên hàm: sin, cos...
- Tên do người lập trình tự đặt để dùng trong chương trình của mình.

Sử dụng bộ chữ cái, chữ số và dấu gạch dưới (_) để đặt tên, nhưng phải tuân thủ quy tắc:

- Bắt đầu bằng một chữ cái hoặc dấu gạch dưới;
- Không có khoảng trống ở giữa tên;
- Không được trùng với từ khóa;
- Độ dài tối đa của tên là không giới hạn, tuy nhiên chỉ có 31 ký tự đầu tiên là có ý nghĩa;
- Không cấm việc đặt tên trùng với tên chuẩn nhưng khi đó ý nghĩa của tên chuẩn không còn giá trị nữa.

Ví dụ: tên do người lập trình đặt:

Chieu_dai, Chieu_Rong, Chu_Vi, Dien_Tich

Tên không hợp lệ: Do Dai, 12A2,…

Biến

Biến là một đại lượng được người lập trình định nghĩa và được đặt tên thông qua việc khai báo biến. Biến dùng để chứa dữ liệu trong quá trình thực hiện chương trình và giá trị của biến có thể bị thay đổi trong quá trình này. Mỗi biến phải thuộc về một kiểu dữ liệu xác định và có miền giá trị thuộc kiểu đó.

Cú pháp khai báo biến:

<Kiểu dữ liệu> Danh sách các tên biến cách nhau bởi dấu phẩy;

Ví dụ:

```
int a, b, c; /*Ba biến a, b,c có kiểu int*/
long int chu_vi; /*Biến chu_vi có kiểu long*/
float nua_chu_vi; /*Biến nua_chu_vi có kiểu float*/
double dien_tich; /*Biến dien_tich có kiểu double*/
```

Lưu ý: Để kết thúc 1 lệnh phải có dấu chấm phẩy (;) ở cuối lệnh.

Vị trí khai báo biến trong C

Trong ngôn ngữ lập trình C, ta phải khai báo biến đúng vị trí. Nếu khai báo (đặt các biến) không đúng vị trí sẽ dẫn đến những sai sót ngoài ý muốn mà người lập trình không lường trước (hiệu ứng lề). Chúng ta có 2 cách đặt vị trí của biến như sau:

- Khai báo biến ngoài: Các biến này được đặt bên ngoài tất cả các hàm và nó có tác dụng hay ảnh hưởng đến toàn bộ chương trình (còn gọi là biến toàn cục), ví dụ :

```
int i;          /*Bien ben ngoai */
float pi;       /*Bien ben ngoai*/
int main()
{ … }
```

- Khai báo biến trong: Các biến được đặt ở bên trong hàm, chương trình chính hay một khối lệnh. Các biến này chỉ có tác dụng hay ảnh hưởng đến hàm, chương trình hay khối lệnh chứa nó. Khi khai báo biến, phải đặt các **biến này ở đầu của khối lệnh**, trước các lệnh gán, …

Ví dụ 1:

```
#include <stdio.h>
#include <conio.h>
int bienngoai; /*khai bao bien ngoai*/
int main ()
{
     int j,i; /*khai bao bien ben trong chuong trinh chinh*/
     i=1; j=2;
     bienngoai=3;
     printf("\n Gia tri cua i la %d",i);
               /*%d là số nguyên, sẽ biết sau */
     printf("\n Gia tri cua j la %d",j);
     printf("\n Gia tri cua bienngoai la %d",bienngoai);
     getch();
     return 0;
}
```

Ví dụ 2:

```
#include <stdio.h>
#include<conio.h>
int main ()
{
     int i, j; /*Bien ben trong*/
```

```
        i=4; j=5;
        printf("\n Gia tri cua i la %d",i);
        printf("\n Gia tri cua j la %d",j);
        if(j>i)
        {
              int hieu=j-i; /*Bien ben trong */
              printf("\n Hieu so cua j tru i la %d",hieu);
        }
        else
        {
              int hieu=i-j ; /*Bien ben trong*/
              printf("\n Gia tri cua i tru j la %d",hieu);
        }
        getch();
        return 0;
}
```

Hằng

Là đại lượng không đổi trong suốt quá trình thực thi của chương trình.

Cú pháp:

const <Kiểu dữ liệu> <Tên_hằng> = Giá_trị;

Ví dụ:

const int heso=10;

Hằng số thực

Số thực bao gồm các giá trị kiểu float, double, long double được thể hiện theo 2 cách sau:

- Cách 1: Sử dụng cách viết thông thường mà chúng ta đã sử dụng trong các môn Toán, Lý, …Điều cần lưu ý là sử dụng dấu thập phân là dấu chấm (.);

  Ví dụ: 123.34 -223.333 3.00 -56.0

- Cách 2: Sử dụng cách viết theo số mũ hay số khoa học. Một số thực được tách làm 2 phần, cách nhau bằng ký tự e hay E
  - **Phần giá trị**: là một số nguyên hay số thực được viết theo cách 1.
  - **Phần mũ**: là một số nguyên

  Giá trị của số thực là: Phần giá trị nhân với 10 mũ phần mũ.

  Ví dụ: 1234.56e-3 = 1.23456 (là số $1234.56 * 10^{-3}$)
  -123.45E4 = -1234500 (là $-123.45 * 10^{4}$)

Hằng số nguyên

Số nguyên gồm các kiểu int (2 bytes) , long (4 bytes) được thể hiện theo những cách sau :

- Hằng số nguyên 2 bytes (int) hệ thập phân: Là kiểu số mà chúng ta sử dụng thông thường, hệ thập phân sử dụng các ký số từ 0 đến 9 để biểu diễn một giá trị nguyên.

  Ví dụ: 123, -242

- Hằng số nguyên 2 byte (int) hệ bát phân: Là kiểu số nguyên sử dụng 8 ký số từ 0 đến 7 để biểu diễn một số nguyên.

Cách biểu diễn: 0<các ký số từ 0 đến 7>

Ví dụ : 0345, -020 (số 345, -20 trong hệ bát phân)

- Hằng số nguyên 2 byte (int) hệ thập lục phân: Là kiểu số nguyên sử dụng 10 ký số từ 0 đến 9 và 6 ký tự A, B, C, D, E ,F để biểu diễn một số nguyên.

  Cách biểu diễn: 0x<các ký số từ 0 đến 9 và 6 ký tự từ A đến F>

- Hằng số nguyên 4 byte (long): Số long (số nguyên dài) được biểu diễn như số int trong hệ thập phân và kèm theo ký tự l hoặc L. Một số nguyên nằm ngoài miền giá trị của số int (2 bytes) là số long (4 bytes).

  Ví dụ: 45345L hay 45345l hay 45345

- Các hằng số còn lại: Viết như cách viết thông thường (không có dấu phân cách giữa 3 số)

Hằng ký tự

Hằng ký tự là một ký tự riêng biệt được viết trong cặp dấu nháy đơn (‘). Mỗi một ký tự tương ứng với một giá trị trong bảng mã ASCII. Hằng ký tự cũng được xem như trị số nguyên.

Ví dụ: ‘a’, ‘A’, ‘0’, ‘9’

Chúng ta có thể thực hiện các phép toán số học trên 2 ký tự (thực chất là thực hiện phép toán trên giá trị ASCII của chúng)

Hằng chuỗi ký tự

Hằng chuỗi ký tự là một chuỗi hay một xâu ký tự được đặt trong cặp dấu nháy kép (“). Ví dụ: “Ngon ngu lap trinh C”, “Khoa CNTT-HVKTQS”.

Lưu ý:

- Một chuỗi không có nội dung “” được gọi là chuỗi rỗng;
- Khi lưu trữ trong bộ nhớ, một chuỗi được kết thúc bằng ký tự NULL (‘\0’: mã Ascii là 0);
- Để biểu diễn ký tự đặc biệt bên trong chuỗi ta phải thêm dấu \ phía trước. Ví dụ: “I’m a student” phải viết “I\’m a student”; “Day la ky tu “dac biet”” phải viết “Day la ky tu \”dac biet\””“.

5. Biểu thức

Biểu thức là một sự kết hợp giữa các toán tử (operator) và các toán hạng (operand) theo đúng một trật tự nhất định. Mỗi toán hạng có thể là một hằng, một biến hoặc một biểu thức khác.

Trong trường hợp, biểu thức có nhiều toán tử, ta dùng cặp dấu ngoặc đơn () để chỉ định toán tử nào được thực hiện trước.

Ví dụ: Biểu thức nghiệm của phương trình bậc hai:

(-b + sqrt(Delta))/(2*a)

Trong đó 2 là hằng; a, b, Delta là biến.

Các toán tử số học

Trong ngôn ngữ C, các toán tử +, -, *, / làm việc tương tự như khi chúng làm việc trong các ngôn ngữ khác. Ta có thể áp dụng chúng cho đa số kiểu dữ liệu có sẵn được cho phép