PHẠM VĂN ẤT (Chủ biên)
NGUYỄN HIẾU CƯỜNG

LẬP TRÌNH
HƯỚNG ĐỐI TƯỢNG
VÀ C++

NHÀ XUẤT BẢN GIAO THÔNG VẬN TẢI

LỜI NÓI ĐẦU

Lập trình hướng đối tượng và C^{++} là một môn học quan trọng đối với sinh viên ngành Công nghệ thông tin và một số ngành học khác. Lập trình hướng đối tượng là phương pháp lập trình chủ đạo hiện nay trong công nghiệp phần mềm và tư tưởng hướng đối tượng được áp dụng trong hầu hết các ngôn ngữ lập trình hiện đại như C^{++}, Visual C^{++}, C#, Java...

Phương pháp lập trình phổ biến nhất trong những năm 70 và 80 của thế kỷ trước là lập trình cấu trúc. Đó là phương pháp tổ chức, phân chia chương trình thành các hàm, thủ tục. Thông qua các ngôn ngữ như Pascal và C, đa số những người làm Tin học đã khá quen biết với phương pháp lập trình này. Tuy nhiên phương pháp lập trình này cũng dần bộc lộ nhiều hạn chế.

Phương pháp lập trình hướng đối tượng đã khắc phục được những hạn chế của lập trình cấu trúc và mở ra một giai đoạn phát triển mới trong công nghiệp phần mềm. Lập trình hướng đối tượng dựa trên việc tổ chức chương trình thành các lớp. Khác với hàm và thủ tục, lớp là một đơn vị bao gồm cả dữ liệu và các phương thức xử lý. Vì vậy lớp có thể mô tả các thực thể một cách chân thực, đầy đủ và chặt chẽ hơn.

Ngôn ngữ C ra đời năm 1973 với mục đích ban đầu là để viết hệ điều hành Unix trên máy tính mini PDP. Sau đó C đã được sử dụng rộng rãi trên nhiều loại máy tính khác nhau và đã trở thành một ngôn ngữ lập trình cấu trúc rất được ưa chuộng. Để đưa C vào thế giới hướng hướng đối tượng, năm 1980 B. Stroustrup đã cho ra đời một ngôn ngữ mới gọi là C^{++}, là một sự phát triển mạnh mẽ của ngôn ngữ C. Ngôn ngữ C^{++} là một ngôn ngữ lai, tức là nó cho phép tổ chức chương trình theo cả các lớp và các hàm. Có thể nói C^{++} đã thúc đẩy ngôn ngữ C vốn đã rất thuyết phục đi vào thế giới lập trình hướng đối tượng và C^{++} đã trở thành ngôn ngữ hướng đối tượng mạnh và được sử dụng rộng rãi nhất từ những năm 1990.

Giáo trình này sẽ trình bầy một cách hệ thống các khái niệm của lập trình hướng đối tượng được cài đặt trong C^{++} như lớp, đối tượng, sự thừa kế, tính tương ứng bội, khuôn hình và các khả năng mới trong xây dựng, sử dụng hàm như: đối tham chiếu, đối mặc định, hàm trùng tên, hàm toán tử. Cuối mỗi chương đều có các bài tập ở những mức độ khác nhau để độc giả tự rèn luyện thêm.

Các vấn đề phức tạp thường đòi hỏi phải phân tích và thiết kế tương đối đầy đủ trước khi có thể viết chương trình. Tuy giáo trình này không tập trung vào phân tích

thiết kế, nhưng trong phụ lục 4 chúng tôi cũng giới thiệu vắn tắt về phương pháp phân tích, thiết kế hướng đối tượng.

Cuốn sách gồm 9 chương và 4 phụ lục.

Chương 1 hướng dẫn cách làm việc với phần mềm TC^{++} 3.0 để thử nghiệm các chương trình, trình bầy sơ lược về các phương pháp lập trình và giới thiệu một số mở rộng đơn giản của C^{++} .

Chương 2 trình bầy các khả năng mới trong việc xây dựng và sử dụng hàm trong C^{++} như biến tham chiếu, đối có kiểu tham chiếu, đối có giá trị mặc định, hàm trực tuyến, hàm trùng tên, hàm toán tử.

Chương 3 nói về một khái niệm trung tâm của lập trình hướng đối tượng.

Chương 4 trình bày chi tiết hơn về định nghĩa chồng các toán tử

Chương 5 trình bầy các vấn đề tạo dựng, sao chép, huỷ bỏ các đối tượng và các vấn đề khác có liên quan.

Chương 6 trình bầy một khái niệm quan trọng tạo nên khả năng mạnh của lập trình hướng đối tượng trong việc phát triển, mở rộng phần mềm, đó là khả năng thừa kế của các lớp.

Chương 7 trình bầy một khái niệm quan trọng khác trong lập trình hướng đối tượng là tính tương ứng bội và phương thức ảo.

Chương 8 nói về việc tổ chức vào/ra trong C^{++}.

Chương 9 trình bầy về khuôn hình (template) trong C^{++}.

Phụ lục 1 trình bầy các phép toán trong C^{++} và thứ tự ưu tiên của chúng.

Phụ lục 2 trình bầy về bảng mã ASCII và mã quét của các ký tự.

Phụ lục 3 là tập hợp một số câu hỏi trắc nghiệm và đáp án để bạn đọc tự kiểm tra lại kiến thức.

Phụ lục 4 trình bầy một cách ngắn gọn phương pháp phân tích, thiết kế và lập trình hướng đối tượng.

Cuối cùng là danh mục một số thuật ngữ chuyên ngành sử dụng trong giáo trình này cùng vị trí tham chiếu để độc giả tiện tra cứu, và một số tài liệu tham khảo chính.

Nội dung chính của giáo trình được PGS. TS. Phạm Văn Ất biên soạn dựa trên nền cuốn "C^{++} & lập trình hướng đối tượng" của tác giả, nhưng có một số bổ sung và sửa chữa. ThS. Nguyễn Hiếu Cường biên soạn chương 4, phụ lục 3, các bài tập cuối mỗi chương và hiệu chỉnh giáo trình.

Khi viết giáo trình này chúng tôi đã hết sức cố gắng để giáo trình được hoàn chỉnh, song chắc không tránh khỏi thiếu sót, vì vậy chúng tôi rất mong nhận được sự góp ý của độc giả.

Các tác giả

Chương 1

CÁC KHÁI NIỆM CƠ BẢN

Chương này trình bầy các vấn đề sau:

- Cách sử dụng phần mềm Turbo C^{++} 3.0
- Tóm lược về các phương pháp lập trình cấu trúc và lập trình hướng đối tượng
- Những mở rộng của C^{++} so với C

§ 1. LÀM VIỆC VỚI TURBO C^{++} 3.0

Các ví dụ trong giáo trình này được viết và thực hiện trên môi trường Turbo C^{++} (TC^{++} phiên bản 3.0). Sau khi cài đặt (giả sử vào thư mục C:\TC) thì trong thư mục TC sẽ gồm có các thư mục con sau:

C:\TC\BGI chứa các tệp đuôi BGI và CHR

C:\TC\BIN chứa các tệp chương trình (đuôi EXE) như TC, TCC, TLIB, TLINK, …

C:\TC\INCLUDE chứa các tệp tiêu đề đuôi H

C:\TC\LIB chứa các tệp đuôi LIB, OBJ

Để vào môi trường của TC^{++} chỉ cần thực hiện tệp chương trình TC.EXE trong thư mục C:\TC\BIN . Sau khi vào môi trường TC^{++} chúng ta thấy vùng soạn thảo chương trình và hệ menu chính của TC^{++} (gần giống như hệ menu quen thuộc của Turbo C). Hệ menu của TC^{++} gồm các menu: File, Edit, Search, Run, Compile, Debug, Project, Options, Window, Help.

Cách soạn thảo, biên dịch và chạy chương trình trong TC^{++} cũng giống như trong TC, ngoại trừ điểm sau: Tệp chương trình trong hệ soạn thảo của TC^{++} có đuôi mặc định là CPP còn trong TC thì tệp chương trình có đuôi là C. Trong TC^{++} có thể thực hiện cả chương trình C và C^{++}.

§ 2. NGÔN NGỮ C VÀ C^{++}

Có thể nói C^{++} là sự mở rộng đáng kể của C. Điều đó có nghĩa là ngoài những khả năng mới của C^{++}, mọi khả năng, mọi khái niệm trong C đều dùng được trong C^{++}.

Vì trong C^{++} sử dụng gần như toàn bộ các khái niệm, định nghĩa, các kiểu dữ liệu, các cấu trúc lệnh, các hàm và các công cụ khác của C, nên sẽ thuận lợi hơn nếu độc

giả đã biết sử dụng tương đối thành thạo ngôn ngữ C. Giáo trình này chủ yếu tập trung vào các khái niệm lập trình hướng đối tượng cùng ngôn ngữ C^{++}, và do đó nó sẽ không trình bày lại các chủ đề cơ bản trong ngôn ngữ C như các kiểu dữ liệu, các cấu trúc điều khiển, …

Vì C^{++} là sự mở rộng của C, nên bản thân một chương trình C đã là chương trình C^{++}. Tuy nhiên Trình biên dịch TC^{++} yêu cầu mọi hàm chuẩn dùng trong chương trình đều phải khai báo nguyên mẫu bằng một câu lệnh #include, trong khi điều này không bắt buộc đối với Trình biên dịch của TC.

Trong C ta có thể dùng một hàm chuẩn mà bỏ qua câu lệnh #include để khai báo nguyên mẫu của hàm được dùng. Điều này không báo lỗi khi biên dịch, nhưng có thể dẫn đến kết quả sai khi chạy chương trình.

Ví dụ khi biên dịch chương trình sau trong môi trường C sẽ không gặp các dòng cảnh báo (warning) và thông báo lỗi (error). Nhưng khi chạy sẽ nhận được kết quả sai.

```
#include <stdio.h>
void main()
  {
    float a,b,c,p,s;
    printf("\nNhap a, b, c ");
    scanf("%f%f%f",&a,&b,&c);
    p=(a+b+c)/2;
    s= sqrt(p*(p-a)*(p-b)*(p-c));
    printf("\nDien tich = %0.2f",s);
    getch();
  }
```

Nếu biên dịch chương trình này trong TC^{++} sẽ nhận được các thông báo lỗi sau:

```
Error:  Funtion ‘sqrt’ should have a prototype
Error:  Funtion ‘getch’ should have a prototype
```

Để biến chương trình trên thành một chương trình C^{++} cần:

+ Đặt tên chương trình với đuôi CPP
+ Thêm hai câu lệnh #include để khai báo nguyên mẫu cho các hàm sqrt và getch:

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

§ 3. LẬP TRÌNH CẤU TRÚC VÀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

3.1. Phương pháp lập trình cấu trúc

Tư tưởng chính của lập trình cấu trúc là tổ chức chương trình thành các chương trình con. Trong PASCAL có hai kiểu chương trình con là thủ tục (procedure) và hàm (fuction). Trong C chỉ có một loại chương trình con là hàm.

Hàm là một đơn vị chương trình độc lập dùng để thực hiện một phần việc nào đó như: Nhập số liệu, in kết quả hay thực hiện một số tính toán. Hàm cần có đối và các biến, mảng cục bộ dùng riêng cho hàm. Việc trao đổi dữ liệu giữa các hàm thực hiện thông qua các đối và các biến toàn bộ.

Các ngôn ngữ như C, PASCAL là các ngôn ngữ cho phép triển khai phương pháp lập trình cấu trúc. Một chương trình cấu trúc gồm các cấu trúc dữ liệu (như biến, mảng, bản ghi, …) và các hàm, thủ tục. Nhiệm vụ chính của việc tổ chức thiết kế chương trình cấu trúc là tổ chức chương trình thành các hàm, thủ tục.

Ví dụ xét yêu cầu sau: Viết chương trình nhập toạ độ (x,y) của một dẫy điểm, sau đó tìm một cặp điểm cách xa nhau nhất.

Trên tư tưởng của lập trình cấu trúc có thể tổ chức chương trình như sau:

+ Sử dụng hai mảng thực toàn bộ x và y để chứa toạ độ dẫy điểm

+ Xây dựng hai hàm:

Hàm nhapsl dùng để nhập toạ độ n điểm, hàm này có một đối là biến nguyên n và được khai báo như sau:

```
void nhapsl(int n);
```

Hàm do_dai dùng để tính độ dài đoạn thẳng đi qua 2 điểm có chỉ số là i và j , nó được khai báo như sau:

```
float  do_dai(int i, int j);
```

Chương trình C cho bài toán trên được viết như sau:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
float x[100], y[100];
float do_dai(int i, int j)
  {
   return sqrt(pow(x[i]-x[j],2)+pow(y[i]-y[j],2));
  }
void nhapsl(int n)
  {
   int i;
   for (i=1;i<=n;++i)
     {
       printf("\nNhap toa do x, y cua diem thu %d : ",i);
       scanf("%f%f", &x[i], &y[i]);
```

```
    }
  }
void main()
  {
    int n,i,j,imax,jmax;
    float d,dmax;
    printf("\nSo diem n = ");
    scanf("%d",&n);
    nhapsl(n);
    dmax=do_dai(1,2);
    imax=1;
    jmax=2;
    for (i=1;i<=n-1;++i)
      for (j=i+1;j<=n;++j)
        {
          d=do_dai(i,j);
          if (d>dmax)
            {
              dmax=d;
              imax=i;
              jmax=j;
            }
        }
    printf("\nDoan thang lon nhat co do dai bang: %0.2f",dmax);
    printf("\n Di qua 2 diem co chi so la %d va %d",imax,jmax);
    getch();
  }
```

3.2. Phương pháp lập trình hướng đối tượng

Khái niệm trung tâm của lập trình hướng đối tượng là lớp (class). Có thể xem lớp là sự kết hợp các thành phần dữ liệu và các hàm. Cũng có thể xem lớp là sự mở rộng của cấu trúc (struct) trong C bằng cách đưa thêm vào các phương thức (methods) hay còn gọi là hàm thành viên (member functions). Một lớp được định nghĩa như sau:

```
class Tên_lớp
  {
    // Khai báo các thành phần dữ liệu
    // Khai báo các phương thức
  };
```

Các phương thức có thể được viết (xây dựng) bên trong hoặc bên ngoài (phía dưới) phần định nghiã lớp. Cách viết một phương thức tương tự như viết một hàm, ngoại trừ quy tắc sau: Khi xây dựng một phương thức bên ngoài định nghĩa lớp thì trong dòng đầu tiên cần dùng tên lớp và hai dấu hai chấm (::) đặt trước tên phương thức để chỉ rõ phương thức đó thuộc lớp nào.

Vì phương thức và các thành phần dữ liệu thuộc cùng một lớp, hơn nữa phương thức được lập lên cốt để xử lý các thành phần dữ liệu, nên trong thân của phương thức có quyền truy nhập đến các thành phần dữ liệu (của cùng lớp).

Sau khi định nghĩa một lớp, có thể dùng tên lớp để khai báo các biến kiểu lớp hay còn gọi là đối tượng. Mỗi đối tượng sẽ có các thành phần dữ liệu và các phương thức. Lời gọi một phương thức cần chứa tên đối tượng để xác định phương thức thực hiện từ đối tượng nào.

Một chương trình hướng đối tượng sẽ bao gồm các lớp có quan hệ với nhau. Việc phân tích, thiết kế chương trình theo phương pháp hướng đối tượng nhằm thiết kế, xây dựng các lớp.

Từ khái niệm lớp nẩy sinh hàng loạt khái niệm khác như: Thành phần dữ liệu, phương thức, phạm vi, sự đóng gói, hàm tạo, hàm huỷ, sự thừa kế, lớp cơ sở, lớp dẫn xuất, tương ứng bội, phương thức ảo, ...

Thiết kế hướng đối tượng là tập trung xác định các lớp để mô tả các thực thể của bài toán. Mỗi lớp đưa vào các thành phần dữ liệu của thực thể và xây dựng luôn các phương thức để xử lý dữ liệu. Như vậy việc thiết kế chương trình xuất phát từ các nội dụng các vấn đề của bài toán.

Các ngôn ngữ thuần tuý hướng đối tượng (như Smalltalk) chỉ hỗ trợ các khái niệm về lớp, không có các khái niệm hàm. C^{++} là ngôn ngữ lai, nó cho phép sử dụng cả các công cụ của lớp và hàm.

Để minh hoạ các khái niệm vừa nêu về lập trình hướng đối tượng ta trở lại xét bài toán tìm độ dài lớn nhất đi qua 2 điểm. Trong bài toán này ta gặp một thực thể là dẫy điểm. Xây dựng lớp dãy điểm (daydiem), trong đó các thành phần dữ liệu của lớp dẫy điểm gồm:

+ Biến nguyên n là số điểm của dẫy

+ Con trỏ x kiểu thực trỏ đến vùng nhớ chứa dẫy hoành độ

+ Con trỏ y kiểu thực trỏ đến vùng nhớ chứa dẫy tung độ

Các phương thức cần đưa vào theo yêu cầu bài toán gồm:

+ Nhập toạ độ một điểm

+ Tính độ dài đoạn thẳng đi qua 2 điểm

Dưới đây là chương trình viết theo thiết kế hướng đối tượng. Để thực hiện chương trình này nhớ đặt tên tệp có đuôi CPP.

Xem chương trình ta thấy thêm một điều mới trong C^{++} là: Các khai báo biến, mảng có thể viết bất kỳ chỗ nào trong chương trình (tất nhiên phải trước khi sử dụng biến, mảng).

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

```
#include <conio.h>
#include <math.h>
#include <alloc.h>
class daydiem
  {
    public:
      int n;
      float *x, *y;
      float do_dai(int i, int j)
        {
          return sqrt(pow(x[i]-x[j],2)+pow(y[i]-y[j],2));
        }
      void nhapsl(void);          // khai báo phương thức
  };

void daydiem::nhapsl(void)  // định nghĩa (xây dựng) phương thức
  {
    int i;
    printf("\nSo diem n = ");
    scanf("%d",&n);
    x=(float*)malloc((n+1)*sizeof(float));
    y=(float*)malloc((n+1)*sizeof(float));
    for (i=1;i<=n;++i)
      {
        printf("\nNhap toa do x, y cua diem thu %d : ",i);
        scanf("%f%f",&x[i],&y[i]);
      }
  }
void main()
  {
    daydiem p;
    int n,i,j;
    int imax,jmax;
    float d, dmax;
    p.nhapsl();
    n=p.n;
    dmax=p.do_dai(1,2); imax=1;jmax=2;
```

```
  for (i=1;i<=n-1;++i)
    for (j=i+1;j<=n;++j)
      {
        d=p.do_dai(i,j);
        if (d>dmax)
          {
            dmax=d;
            imax=i;
            jmax=j;
          }
      }
  printf("\nDoan thang lon nhat co do dai bang: %0.2f",dmax);
  printf("\n Di qua 2 diem co chi so la %d va %d",imax,jmax);
  getch();
}
```

§ 4. MỘT SỐ MỞ RỘNG ĐƠN GIẢN CỦA C++ SO VỚI C

Trong mục này trình bầy một số mở rộng của C++, tuy đơn giản nhưng đem lại khá nhiều tiện lợi.

4.1. Viết các dòng ghi chú

Trong C++ vẫn có thể viết các dòng ghi chú trong các dấu /* và */ như trong C. Cách viết này cho phép viết các ghi chú trên nhiều dòng hoặc trên một dòng. Ngoài ra trong C++ còn cho phép viết ghi chú trên một dòng sau hai dấu gạch chéo rất tiện lợi, ví dụ:

```
int x,y ;   // Khai báo 2 biến thực
```

4.2. Khai báo linh hoạt

Trong C tất cả các câu lệnh khai báo biến, mảng cục bộ phải đặt tại đầu khối. Do vậy nhiều khi vị trí khai báo và vị trí sử dụng của biến khá xa nhau, gây khó khăn trong việc kiểm soát chương trình. C++ đã khắc phục nhược điểm này bằng cách cho phép các lệnh khai báo biến, mảng có thể đặt ở bất kỳ chỗ nào trong chương trình trước khi các biến, mảng đó được sử dụng.

Ví dụ chương trình nhập một dẫy số thực rồi sắp xếp theo thứ tự tăng dần có thể viết trong C++ như sau:

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