BÀI 2 - KIỂU DỮ LIỆU NGUYÊN THỦY, BIẾN VÀ HẰNG TRONG C++.

Trong bài tập 102 ta đã tính được giá trị của 345 + 678, vậy nếu bây giờ muốn tính giá trị của các cặp số khác (15 + 24 = ? ; 37 + 698 = ?;....) thì ta làm thế nào ? Không lẽ với mỗi cặp số ta đều cần viết một chương trình riêng để tính toán ? Nếu thế thì việc lập trình máy tính sẽ không có ý nghĩa.

Ta cần phải viết một chương trình mà nó có khả năng đọc giá trị của hai số nguyên bất kì do người dùng nhập vào, sau đó tính toán và đưa ra kết quả cho người dùng. Ta có thể viết chương trình như sau:

```
1   #include <iostream>
2   using namespace std;
3   int a,b;
4   int main()
5   {
6       cout <<"Nhap gia tri hai so A,B: ";
7       cin >> a >> b;
8       cout << (a + b);
9       return 0;
10  }
```

[3] : Khai báo sử dụng 2 vùng nhớ kiểu số nguyên **int** có dung lượng lưu trữ 4 byte (*phạm vi giá trị từ -2,147,483,648 đến +2,147,483,647*) tên là a,b dùng để chứa giá trị hai số nguyên mà người dùng nhập vào từ bàn phím.

[7]: Lệnh cin >> … sẽ đọc giá trị của hai số mà người dùng nhập từ bàn phím rồi lưu vào hai vùng nhớ a, b. Kí hiệu >> gọi là toán tử nhập, mỗi toán tử nhập nhập dữ liệu vào cho 1 biến.

[8]: in ra giá trị phép cộng 2 giá trị lưu tại hai vùng nhớ a, b (vùng nhớ a,b này trong lập trình được gọi là các biến).

Một số vấn đề cần trả lời:

- *Để lưu trữ được giá trị hai số nguyên do người dùng nhập thì chương trình cần sử dụng bao nhiêu byte bộ nhớ ?*
- *Ngoài kiểu int như trên thì C++ còn có những kiểu dữ liệu nào khác ?*

Thực tế C++ nói riêng và các ngôn ngữ lập trình khác nói chung đều được thiết kế nhiều kiểu dữ liệu khác nhau để xử lí các bài toán. Các kiểu dữ liệu được sử dụng để cho các **biến**(vùng nhớ do người dùng khai báo) biết kiểu dữ liệu mà nó có thể lưu trữ. Bất cứ khi nào một biến được định nghĩa trong C ++, trình biên dịch sẽ cấp phát một số bộ nhớ cho biến đó dựa trên kiểu dữ liệu mà nó được khai báo. Mỗi kiểu dữ liệu yêu cầu một lượng bộ nhớ khác nhau.

Trong C++ người ta thiết kế một số kiểu dữ liệu cơ bản(nguyên thủy) như sau:

Kiểu	Từ khóa	Bộ nhớ	Phạm vi giá trị
Số nguyên	int	4 byte	Số nguyên 4 byte có dấu, phạm vi : ***-2,147,483,648*** Tới: ***2,147,483,647*** *INT_MAX; INT_MIN là hai giá trị max, min của kiểu int*
	unsigned int	4 byte	Số nguyên 4 byte không dấu, phạm vi: ***0*** tới ***4,294,967,295***
	long long	8 byte	Số nguyên 8 byte có dấu, phạm vi : ***-(2^63)*** tới ***(2^63)-1*** LLONG_MAX; LLONG_MIN *là hai giá trị max, min của kiểu long long int*
	unsigned long long	8 byte	Số nguyên 8 byte không dấu, phạm vi : ***0*** tới ***18,446,744,073,709,551,615***
Số thực	float	4 byte	Số thực 4 byte, phạm vi: ***1.17549e-38*** to ***3.40282e+38*** FLT_MAX, FLT_MIN *là hai giá trị max, min của kiểu float*
	double	8 byte	Số thực 8 byte, phạm vi: ***2.22507e-308*** to ***1.79769e+308*** DBL_MAX, DBL_MIN *là hai giá trị max, min của kiểu double*
Kí tự	char	1 byte	Lưu trữ 1 kí tự nằm trong 256 kí tự ASCII
	string	Lưu trữ 1 chuỗi (xâu) kí tự với số lượng không giới hạn mà chỉ phụ thuộc dung lượng bộ nhớ còn lại trong số 64KB cho phép.	
Logic	bool	Lưu trữ 1 trong 2 giá trị logic true/ false hoặc 0 / 1	

Lưu ý

- Để có thể lấy được giá trị các hằng MAX, MIN của kiểu tương ứng thì cần thêm vào tiền xử lí các thư viện sau:

#include<limits.h> // for int,char macros

#include<float.h> // for float,double macros

- Khi khai báo các kiểu dữ liệu có tên quá dài như long long int, unsigned long long int,.. ta có thể định nghĩa nó một cách vắn tắt trước khi sử dụng bằng từ khóa #define như sau:

#define lli long long int;

#define ulli unsigned long long int;

Sau các định nghĩa trên thì sau này trong chương trình thay vì phải khai báo đầy đủ tên kiểu dữ liệu là **long long int** ta chỉ cần viết là **lli**. Ví dụ như sau:

```
1   #include<iostream>
2   #define lli long long int
3   using namespace std;
4   lli a,b;
5   int main()
6   {
7       cin >> a >> b;
8       cout << (a+b);
9       return 0;
10  }
```

II. KHAI BÁO VÀ SỬ DỤNG BIẾN ĐỂ LƯU TRỮ GIÁ TRỊ

1. Cú pháp khai báo biến

Như đã nói ở trên, biến trong lập trình là một vùng nhớ được lập trình viên khai báo và đặt tên để lưu trữ các giá trị dữ liệu trong quá trình tính toán. Để khai báo sử dụng biến trong C++, ta khai báo theo cú pháp như sau:

[kiểu dữ liệu của biến] [danh sách tên các biến cùng kiểu] ;

Như vậy:

- *Một biến là một tên được đặt cho một vị trí bộ nhớ. Nó là đơn vị lưu trữ cơ bản trong một chương trình. Trong C ++, tất cả các biến phải được khai báo trước khi sử dụng.*
- *Giá trị được lưu trữ trong một biến có thể được thay đổi trong quá trình thực thi chương trình.*

✍ *Một biến chỉ là tên được đặt cho một vị trí bộ nhớ, tất cả các thao tác thực hiện trên biến sẽ ảnh hưởng đến vị trí bộ nhớ đó. Tên biến không có dấu cách và các kí tự đặc biệt, không được bắt đầu bằng số.*

Một số ví dụ về khai báo biến:

- int a,b,c; *// khai báo 3 biến a,b,c có cùng kiểu số nguyên 4 byte có dấu*
- float x = 3.5, y, z = 0.1; *// Khai báo 3 biến kiểu số thực 4 byte trong đó x được gán giá trị khởi đầu là 3.5, z có khởi đầu là 0.1, biến y không được gán giá trị khởi đầu nên máy tính sẽ tự gán cho y một giá trị ngẫu nhiên nào đó không xác định.*

✍*Giả sử ta đã khai báo 6 biến a, b, c, x, y, z như trên, theo em thì biến a có thể nhận giá trị thực là 1.25 hay không ?*

Đáp án: biến a đã được khai báo là biến số nguyên 4 byte nên nó không thể tiếp nhận giá trị thực 1.25

✍ *Tại một thời điểm, một biến sẽ lưu trữ được bao nhiêu giá trị ?*

Đáp án: Giá trị của một biến có thể thay đổi tùy ý, nhưng tại một thời điểm một biến chỉ có khả năng lưu một giá trị. Ví dụ tại thời điểm biến a = 5 ta ra lệnh gán cho a = 7 thì giá trị 5 sẽ bị loại bỏ và thay thế bằng giá trị 7.

2. Các kiểu biến căn bản trong một chương trình C++

Có 2 loại biến căn bản

a. Biến cục bộ (biến địa phương): là biến được khai báo trong một khối lệnh (khối lệnh là một /nhóm câu lệnh nằm trong cặp dấu **{ ... }**) hoặc 1 phương thức, 1 hàm.

Các biến này được tạo ra khi khối lệnh được thực hiện hoặc hàm được gọi và sẽ bị hủy sau khi thoát khỏi khối lệnh hoặc khi gặp lệnh trả về từ hàm.

Phạm vi tác dụng biến cục bộ chỉ tồn tại trong khối/ hàm mà biến được khai báo tức là chúng ta chỉ có thể truy cập các biến này trong khối/ hàm đó.

b. Biến toàn cục: là biến được khai báo đầu chương trình (sau các lệnh tiền xử lí #include , khai báo thư viện, #define và trước các hàm.

Biến toàn cục có thể truy cập ở mọi vị trí trong chương trình sau khi được khai báo(từ hàm main, các hàm con,…)

IV. HẰNG TRONG C++

Trong một chương trình, có thể có những giá trị mà lập trình viên sử dụng nhiều lần, nhưng lại không muốn chúng bị thay đổi do sự vô ý nào đó như số Pi, số phút trong một giờ hoặc vận tốc của ánh sáng, tên riêng nào đó, địa chỉ,…. Khi đó lập trình viên có thể định nghĩa các giá trị này là các hằng giá trị.

Trong C++ có hai cách để khai báo hằng là sử dụng từ khóa **const** hoặc sử dụng định nghĩa tiền xử lí #define.

const <data type> <name variable> = <value>;

hoặc <data type> const <name variable> = <value>;

Ví dụ sau đây sẽ định nghĩa hai hằng số pi = 3.14 và gia tốc rơi tự do G = 9.8 với hai cách khác nhau:

```
1  #include <iostream>
2  #define  G 9.8
3
4  using namespace std;
5  float const pi = 3.14;
6
7  int main()
8  {
9      cout <<"Hang so PI =  "<< pi << '\n';
10     cout <<"Hang so gia toc roi tu do g =  " << G;
11     return 0;
12 }
```

III. NHẬP/ XUẤT DỮ LIỆU TRONG C++

1. Nhập dữ liệu vào từ bàn phím và xuất ra màn hình

a. Lệnh nhập dữ liệu từ bàn phím

Để có thể nhập dữ liệu vào cho biến người ta sử dụng lệnh **cin** và toán tử nhập dữ liệu **>>** theo cú pháp:

cin >> biến ; // nhập dữ liệu cho 1 biến duy nhất

cin >> biến 1 >> biến 2 >> ... >> biến thứ n ; // nhập dữ liệu cho nhiều biến

Cách nhập dữ liệu: *gõ trực tiếp từ bàn phím giá trị dữ liệu cần nhập vào biến rồi nhấn Enter để xác nhận đã nhập xong. Trong trường hợp cần nhập giá trị khác nhau cho nhiều biến thì mỗi giá trị sẽ phân biệt nhau bởi một dấu cách. Sau khi đã nhập đủ giá trị và nhấn Enter thì máy tính sẽ đọc các giá trị đã nhập và lần lượt đưa vào lưu trữ trong các biến có trong danh sách nhận dữ liệu.*

Lưu ý: *một toán tử >> chỉ nhập dữ liệu cho 1 biến, không thể nhập đồng thời cho nhiều biến. Vì thế cách viết sau là sai: cin >> a, b, c; mà phải viết: cin >> a >> b >> c ;*

b. Xuất dữ liệu ra màn hình:

Để xuất (*in, hiển thị*) giá trị các biến/ biểu thức lên màn hình C++ sử dụng lệnh **cout** và toán tử xuất **<<** theo cú pháp:

cout << biểu thức ; // xuất giá trị của biểu thức

cout << biểu thức 1 << ... << biểu thức n ; // xuất giá trị của n biểu thức

Trong trường hợp xuất ra giá trị ở kiểu số thực, ta có thể định nghĩa quy cách hiển thị số chữ số thập phân bằng câu lệnh:

cout << fixed << setprecision (k); // với k là số chữ số thập phân cần hiển thị

Sau lệnh này, các giá trị thực xuất ra bởi lệnh cout sẽ hiển thị số chữ số thập phân là k.

```
1  #include <iostream>
2  #include <iomanip>
3  #define  G 9.8
4  using namespace std;
5  float const pi = 3.14;
6
7  int main()
8  {
9      cout << fixed << setprecision (3);
10     cout <<"Hang so PI =  "<< pi << '\n';
11     cout << fixed << setprecision (5);
12     cout <<"Hang so gia toc roi tu do g =  " << G;
13     return 0;
14 }
```

```
Hang so PI =  3.140
Hang so gia toc roi tu do g =  9.80000
```

Chú ý: Để sử dụng được lệnh cout << fixed << setprecision (k); *ta cần khai báo thêm tệp tiêu đề* <iomanip> *vào tiền xử lí như câu lệnh số 2 trong ví dụ trên.*

2. Nhập dữ liệu vào từ file và xuất ra file

Việc nhập dữ liệu trực tiếp từ bàn phím và xuất ra màn hình chỉ thích hợp với các bài toán có dữ liệu nhỏ, đơn giản. Với các bài toán có khối lượng dữ liệu lớn, phức tạp thì không thể nhập liệu trực tiếp từ bàn phím vì sẽ dễ dẫn đến sai sót khi nhập, khi đó sẽ phải tiến hành nhập lại từ đầu. Mặt khác mỗi khi kết thúc chương trình thì toàn bộ dữ liệu đều bị giải phóng không còn lưu trữ trong bộ nhớ nên không thể xem lại được kết quả của bài toán, mỗi lần thực hiện chương trình đều phải tiến hành nhập liệu lại từ đầu. Để có thể khắc phục việc này, các ngôn ngữ lập trình cung cấp thêm một phương thức xử lí dữ liệu khác là nhập và xuất dữ liệu từ các tệp (file).

Trong C++ ta xử línhập/ xuất dữ liệu từ tệp như sau:

freopen("Tên tệp input", "r",stdin);

// r là viết tắt của read, stdin là bộ đệm vào chuẩn

freopen("Tên tệp output", "w",stdout);

// w là viết tắt của write, stdout là bộ đệm ra chuẩn

Ví dụ: Giả sử ta có tệp dữ liệu đầu vào cần xử lí là NUMB.inp, sau khi xử lí xong ta cần ghi kết quả vào tệp NUMB.out thì ta khai báo như sau:

freopen("NUMB.inp", "r",stdin);

freopen("NUMB.out", "w",stdout);

Lưu ý:

- Tệp đầu vào luôn luôn phải được khởi tạo sẵn trước khi thực hiện chương trình, nếu không tồn tại thì khi chạy sẽ bị báo lỗi. Riêng tệp đầu ra thì có thể có hoặc không, nếu có rồi thì nội dung cũ sẽ bị xóa đi và thay bằng nội dung mới, trường hợp tệp đầu ra chưa tồn tại thì hệ thống sẽ tự động tạo ra tệp đầu ra và ghi dữ liệu vào đó.
- Để sử có thể truy cập dữ liệu từ tệp cần khai báo tiền xử lí bổ sung tệp thư viện: #include <fstream>
- Ngoài ra để tăng tốc độ đọc dữ liệu ta cần thêm vào 4 câu lệnh sau trước lệnh mở tệp:

```
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    freopen("Tên tệp dữ liệu đầu vào","r",stdin);
    freopen("Tên tệp chứa kết quả","w",stdout);
```

THỰC HÀNH

Bài tập 200: Viết chương trình nhập vào giá trị hai số nguyên A, B bất kì, in ra giá trị tổng, hiệu của hai số đó. Ràng buộc dữ liệu : |A|,|B| ≤ 10^9.

Bài tập 201: Viết chương trình nhập vào giá trị hai số A, B bất kì, in ra giá trị tổng, hiệu, tích, thương của hai số đó. Ràng buộc dữ liệu : |A|,|B| ≤ 10^37. Độ chính xác lấy đến 3 chữ số thập phân.

Bài tập 202: Viết chương trình thực hiện khởi tạo tệp HELLO.DAT và ghi vào đó 3 từ "Hello", mỗi từ viết trên một dòng.

Bài tập 203: Cho hàm số bậc nhất y = ax + b (với a ≠ 0), hãy tính giá trị của hàm số tại giá trị x cho trước. Dữ liệu đầu vào gồm 3 số nguyên a, b, x phân biệt nhau bởi dấu cách (−10^9 ≤ a, b, x ≤ 10^9) được lưu trữ trong tệp HAMBAC1.inp. Dữ liệu ra là số nguyên duy nhất xác định giá trị của y, ghi vào tệp HAMBAC1.out

Bài tập 204: Viết chương trình đọc vào từ bàn phím độ dài hai cạnh của một hình chữ nhật được cho trong tệp S_HCN.inp. Tính chu vi, diện tích của hình chữ nhật đó. Kết quả ghi ra tệp S_HCN.out, mỗi giá trị trên một dòng. Độ chính xác lấy 2 chữ số thập phân.

Bài tập 205: Tính chu vi, diện tích hình tròn tâm O bán kính R. Biết rằng giá trị của bán kính được lưu trữ trong file CIRCLE.inp. Kết quả ghi ra file CIRCLE.out, mỗi giá trị phân biệt với nhau bởi một dấu cách và được làm tròn đến 4 chữ số sau dấu thập phân.

GỢI Ý PHƯƠNG ÁN

Bài tập 200

```cpp
#include <iostream>
using namespace std;
int a,b;
int main()
{
    cout <<"Nhap gia tri a,b: ";
    cin >> a >> b;
    cout <<"a + b = " << (a+b) << '\n';
    cout <<"a - b = " << (a-b);
    return 0;
}
```

Bài tập 201

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

double a,b;

int main()
{
    cout <<"Nhap gia tri a,b: ";
    cin >> a >> b;
    cout << fixed << setprecision(3);
    cout <<"a + b = " << (a+b) << '\n';
    cout <<"a - b = " << (a-b) << '\n';
    cout <<"a * b = " << (a*b) << '\n';
    cout <<"a : b = " << (a/b) << '\n';
    return 0;
}
```

Bài tập 202:

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

int main()
{
    freopen("HELLO.DAT","w",stdout);
    cout <<"HELLO\nHELLO\nHELLO";
    return 0;
}
```

Bài tập 203

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

int a,b,x;

int main()
{
    freopen("HAMBAC1.inp","r",stdin);
    freopen("HAMBAC1.out","w",stdout);
    cin >> a >> b >> x;
    cout << (a*x + b);
    return 0;
}
```

Bài tập 204:

```cpp
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;

float a,b,cv,s;

int main()
{
    freopen("S_HCN.inp","r",stdin);
    freopen("S_HCN.out","w",stdout);
    cin >> a >> b;
    cv = (a+b)*2;
    s = a*b;
    cout << fixed << setprecision(2);
    cout << cv << '\n' << s;
    return 0;
}
```

Bài tập 205:

```cpp
#include <iostream>
#include <iomanip>
#include <fstream>
#define PI 3.14
using namespace std;

float r,cv,s;

int main()
{
    freopen("CIRCLE.inp","r",stdin);
    freopen("CIRCLE.out","w",stdout);
    cin >> r;
    cv = 2*PI*r;
    s = PI*r*r;
    cout << fixed << setprecision(4);
    cout << cv << '\n' << s;
    return 0;
}
```