SỞ GIÁO DỤC – ĐÀO TẠO ĐẮK NÔNG
TRƯỜNG THPT CHUYÊN NGUYỄN CHÍ THANH

---------ꕥ🕮ꕥ---------

```
int a[10000] = {0}; //Tạo mảng và gán tất cả bằng 0
memset(prime,true,sizeof(prime));

freopen("timso.inp","r",stdin);
freopen("timso.out","w",stdout);
ios_base::sync_with_stdio(0);
cin.tie(0);
cout.tie(0);
```

```
const long long INF = 1e18;
```

SÁCH GIÁO KHOA
TIN HỌC 11 - NGÔN NGỮ LẬP TRÌNH C++

Gia nghĩa, tháng 8 năm 2018

MỤC LỤC

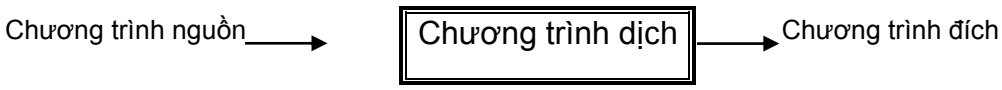
§1. KHÁI NIỆM LẬP TRÌNH VÀ NGÔN NGỮ LẬP TRÌNH

Như đã biết, mọi bài toán có thuật toán đều có thể giải được trên máy tính điện tử. Khi giải bài toán trên máy tính điện tử, sau các bước xác định bài toán và xây dựng hoặc lựa chọn thuật toán khả thi là bước lập trình.

Lập trình là sử dụng cấu trúc dữ liệu và các câu lệnh của ngôn ngữ lập trình cụ thể để mô tả dữ liệu và diễn đạt các thao tác của thuật toán. Chương trình viết bằng ngôn ngữ lập trình bậc cao nói chung không phụ thuộc vào máy, nghĩa là một chương trình có thể thực hiện trên nhiều máy. Chương trình viết bằng ngôn ngữ máy có thể được nạp trực tiếp vào bộ nhớ và thực hiện ngay còn chương trình viết bằng ngôn ngữ lập trình bậc cao phải được chuyển đổi thành chương trình trên ngôn ngữ máy mới có thể thực hiện được.

Chương trình đặc biệt có chức năng chuyển đổi chương trình được viết bằng ngôn ngữ lập trình bậc cao thành chương trình thực hiện được trên máy tính cụ thể được gọi là *chương trình dịch*.

Chương trình dịch nhận đầu vào là chương trình viết bằng ngôn ngữ lập trình bậc cao (chương trình nguồn) thực hiện chuyển đổi sang ngôn ngữ máy (chương trình đích).



Xét ví dụ, bạn chỉ biết tiếng Việt nhưng cần giới thiệu về trường của mình cho đoàn khách đến từ nước Mĩ, chỉ biết tiếng Anh. Có hai cách để bạn thực hiện điều này.

Cách thứ nhất: Bạn nói bằng tiếng Việt và người phiên dịch giúp bạn dịch sang tiếng Anh. Sau mỗi câu hoặc một vài câu giới thiệu trọn một ý, người phiên dịch dịch sang tiếng Anh cho đoàn khách. Sau đó, bạn lại giới thiệu tiếp và người phiên dịch lại dịch tiếp. Việc giới thiệu của bạn và việc dịch của người phiên dịch luân phiên cho đến khi bạn kết thúc nội dung giới thiệu của mình. Cách dịch trực tiếp như vậy được gọi là *thông dịch*.

Cách thứ hai: Bạn soạn nội dung giới thiệu của mình ra giấy, người phiên dịch dịch toàn bộ nội dung đó sang tiếng Anh rồi đọc hoặc trao văn bản đã dịch cho đoàn khách đọc. Như vậy, việc dịch được thực hiện sau khi nội dung giới thiệu đã hoàn tất. Hai công việc đó được thực hiện trong hai khoảng thời gian độc lập, tách biệt nhau. Cách dịch như vậy được gọi là *biên dịch*.

Sau khi kết thúc, với cách thứ nhất không có một văn bản nào để lưu trữ, còn với cách thứ hai có hai bản giới thiệu bằng tiếng Việt và bằng tiếng Anh có thể lưu trữ để dùng lại về sau.

Tương tự như vậy, chương trình dịch có hai loại là *thông dịch* và *biên dịch*.

a) Thông dịch

Thông dịch (interpreter) được thực hiện bằng cách lặp lại dãy các bước sau:

① Kiểm tra tính đúng đắn của câu lệnh tiếp theo trong chương trình nguồn;

② Chuyển đổi câu lệnh đó thành một hay nhiều câu lệnh tương ứng trong ngôn ngữ máy;

③ Thực hiện các câu lệnh vừa chuyển đổi được.

Như vậy, quá trình dịch và thực hiện các câu lệnh là luân phiên. Các chương trình thông dịch lần lượt dịch và thực hiện từng câu lệnh. Loại chương trình dịch này đặc biệt thích hợp cho môi trường đối thoại

giữa người và hệ thống. Tuy nhiên, một câu lệnh nào đó phải thực hiện bao nhiêu lần thì nó phải được dịch bấy nhiêu lần.

Các ngôn ngữ khai thác hệ quản trị cơ sở dữ liệu, ngôn ngữ đối thoại với hệ điều hành,... đều sử dụng trình thông dịch.

b) Biên dịch

Biên dịch (compiler) được thực hiện qua hai bước:

① Duyệt, kiểm tra, phát hiện lỗi, kiểm tra tính đúng đắn của các câu lệnh trong chương trình nguồn;

② Dịch toàn bộ chương trình nguồn thành một chương trình đích có thể thực hiện trên máy và có thể lưu trữ để sử dụng lại khi cần thiết.

Như vậy, trong thông dịch, không có chương trình đích để lưu trữ, trong biên dịch cả chương trình nguồn và chương trình đích có thể lưu trữ lại để sử dụng về sau.

Thông thường, cùng với chương trình dịch còn có một số dịch vụ liên quan như biên soạn, lưu trữ, tìm kiếm, cho biết các kết quả trung gian,... Toàn bộ các dịch vụ trên tạo thành một môi trường làm việc trên một ngôn ngữ lập trình cụ thể. Ví dụ, Turbo Pascal 7.0, Free Pascal 1.2, Visual Pascal 2.1,... trên ngôn ngữ Pascal, Turbo C++, Visual C++,... trên ngôn ngữ C++.

Các môi trường lập trình khác nhau ở những dịch vụ mà nó cung cấp, đặc biệt là các dịch vụ nâng cấp, tăng cường các khả năng mới cho ngôn ngữ lập trình.

§2. CÁC THÀNH PHẦN CỦA NGÔN NGỮ LẬP TRÌNH

1. Các thành phần cơ bản

Mỗi ngôn ngữ lập trình thường có ba thành phần cơ bản là *bảng chữ cái, cú pháp* và *ngữ nghĩa.*

a) Bảng chữ cái là tập các kí tự được dùng để viết chương trình. Không được phép dùng bất kì kí tự nào ngoài các kí tự quy định trong bảng chữ cái.

Trong Pascal, bảng chữ cái bao gồm các kí tự:

- Các chữ cái thường và các chữ cái in hoa của bảng chữ cái tiếng Anh:

  a b c d e f g h i j k l m n o p q r s t u v w x y z
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- 10 chữ số thập phân Ả Rập: 0 1 2 3 4 5 6 7 8 9

- Các kí tự đặc biệt:

+	-	*	/	=		<	>	[	]	.	, (dấu phẩy)
;	#	^	$	@	&	(	)	{	}	:	' (dấu nháy)
dấu cách (mã ASCII 32)									_ (dấu gạch dưới)		

Bảng chữ cái của các ngôn ngữ lập trình nói chung không khác nhau nhiều. Ví dụ, bảng chữ cái của ngôn ngữ lập trình C++ chỉ khác Pascal là có sử dụng thêm các kí tự như dấu nháy kép ("), dấu sổ ngược (\), dấu chấm than (!).

b) Cú pháp là bộ quy tắc để viết chương trình. Dựa vào chúng, người lập trình và chương trình dịch biết được tổ hợp nào của các kí tự trong bảng chữ cái là hợp lệ và tổ hợp nào là không hợp lệ. Nhờ đó, có thể mô tả chính xác thuật toán để máy thực hiện.

c) Ngữ nghĩa xác định ý nghĩa thao tác cần phải thực hiện, ứng với tổ hợp kí tự dựa vào ngữ cảnh của nó.

Ví dụ

Phần lớn các ngôn ngữ lập trình đều sử dụng dấu cộng (+) để chỉ phép cộng. Xét các biểu thức:

```
A + B        (1)
I + J        (2)
```

Giả thiết *A, B* là các đại lượng nhận giá trị thực và *I, J* là các đại lượng nhận giá trị nguyên. Khi đó dấu "+" trong biểu thức (1) được hiểu là cộng hai số thực, dấu "+" trong biểu thức (2) được hiểu là cộng hai số nguyên. Như vậy, ngữ nghĩa dấu "+" trong hai ngữ cảnh khác nhau là khác nhau.

Tóm lại, cú pháp cho biết cách viết một chương trình hợp lệ, còn ngữ nghĩa xác định ý nghĩa của các tổ hợp kí tự trong chương trình.

Các lỗi cú pháp được chương trình dịch phát hiện và thông báo cho người lập trình biết. Chỉ có các chương trình không còn lỗi cú pháp mới có thể được dịch sang ngôn ngữ máy.

Các lỗi ngữ nghĩa khó phát hiện hơn. Phần lớn các lỗi ngữ nghĩa chỉ được phát hiện khi thực hiện chương trình trên dữ liệu cụ thể.

2. Một số khái niệm

a) Tên

Mọi đối tượng trong chương trình đều phải được đặt tên theo quy tắc của ngôn ngữ lập trình và từng chương trình dịch cụ thể.

Trong Turbo Pascal tên là một dãy liên tiếp *không quá 127 kí tự bao gồm chữ số, chữ cái hoặc dấu gạch dưới và bắt đầu bằng chữ cái hoặc dấu gạch dưới*. Trong chương trình dịch Free Pascal, tên có thể có độ dài tới 255 kí tự.

Ví dụ, trong ngôn ngữ Pascal:

- Các tên đúng: `A`
  `R21`
  `P21_c`
  `_45`
- Các tên sai:
  `A BC` (chứa kí tự trắng)
  `6Pq` (bắt đầu bằng chữ số)
  `X#Y` (chứa kí tự "`#`" không hợp lệ)

Ngôn ngữ Pascal không phân biệt chữ hoa, chữ thường trong tên. Một số ngôn ngữ lập trình khác (ví dụ như C++) phân biệt chữ hoa, chữ thường. Ví dụ, *AB* và *Ab* là một tên trong Pascal, nhưng lại là hai tên khác nhau trong C++.

Nhiều ngôn ngữ lập trình, trong đó có Pascal, phân biệt ba loại tên:

- Tên dành riêng;
- Tên chuẩn;
- Tên do người lập trình đặt.

Tên dành riêng

Một số tên được ngôn ngữ lập trình quy định dùng với *ý nghĩa xác định*, người lập trình không được sử dụng với ý nghĩa khác. Những tên này được gọi là *tên dành riêng* (còn được gọi là *từ khoá*).

Ví dụ. Một số tên dành riêng:

Trong Pascal: `program`, `uses`, `const`, `type`, `var`, `begin`, `end`.

Trong C++: `main`, `include`, `if`, `while`, `void`.

Tên chuẩn

Một số tên được ngôn ngữ lập trình dùng với *ý nghĩa nào đó*. Những tên này được gọi là *tên chuẩn*. Tuy nhiên, người lập trình có thể khai báo và dùng chúng với ý nghĩa và mục đích khác.

Ý nghĩa của các tên chuẩn được quy định trong các *thư viện* của ngôn ngữ lập trình.

Ví dụ. Một số tên chuẩn

- Trong Pascal:

```
abs       integer         real
sqr       longint         extended
sqrt      byte            break
```

- Trong C++:

```
cin       cout            getchar
```

Tên do người lập trình đặt

Tên do người lập trình đặt được dùng với ý nghĩa riêng, xác định bằng cách khai báo trước khi sử dụng. Các tên này không được trùng với tên dành riêng.

Ví dụ

Tên do người lập trình đặt:

```
A1
DELTA
CT_Vidu
```

b) Hằng và biến

Hằng

Hằng là các đại lượng có giá trị không thay đổi trong quá trình thực hiện chương trình.

Trong các ngôn ngữ lập trình thường có các hằng số học, hằng lôgic, hằng xâu.

- Hằng số học là các số nguyên hay số thực (dấu phẩy tĩnh hoặc dấu phẩy động), có dấu hoặc không dấu.
- Hằng lôgic là giá trị *đúng* hoặc *sai* tương ứng với *true* hoặc *false*.
- Hằng xâu là chuỗi kí tự trong bảng chữ cái. Khi viết, chuỗi kí tự này được đặt trong cặp dấu nháy (Pascal dùng dấu nháy đơn, còn C++ dùng dấu nháy kép).

Ví dụ

- Hằng số học:

```
2          0           -5          +18
1.5      -22.36      +3.14159      0.5
-2.236E01            1.0E-6
```

- Hằng lôgic:

+ Trong Pascal: `TRUE` `FALSE`

- Hằng xâu:

+ Trong Pascal: `'Informatic'` `'TIN HOC'`

+ Trong C++: `"Informatic"` `"TIN HOC"`

Chú ý: Hằng dấu nháy đơn trong Pascal được viết là ''.

Biến

Biến là đại lượng được đặt tên, dùng để lưu trữ giá trị và giá trị có thể được thay đổi trong quá trình thực hiện chương trình.

Tuỳ theo cách lưu trữ và xử lí, Pascal phân biệt nhiều loại biến. Các biến dùng trong chương trình đều phải khai báo. Việc khai báo biến sẽ được trình bày ở các phần sau.

c) Chú thích

Có thể đặt các đoạn chú thích trong chương trình nguồn. Các chú thích này giúp cho người đọc chương trình nhận biết ngữ nghĩa của chương trình đó dễ hơn. Chú thích không ảnh hưởng đến nội dung chương trình nguồn và được chương trình dịch bỏ qua.

Trong Pascal các đoạn chú thích đặt giữa cặp dấu { và } hoặc (* và *). Một trong những cách tạo chú thích trong C++ là đặt chúng giữa cặp dấu /* và */.

TÓM TẮT

- Cần có chương trình dịch để chuyển chương trình nguồn thành chương trình đích.
- Có hai loại chương trình dịch: thông dịch và biên dịch.
- Các thành phần của ngôn ngữ lập trình: bảng chữ cái, cú pháp và ngữ nghĩa.
- Mọi đối tượng trong chương trình đều phải được đặt tên:
  - Tên dành riêng: Được dùng với ý nghĩa riêng, không được dùng với ý nghĩa khác.
  - Tên chuẩn: Tên dùng với ý nghĩa nhất định, khi cần dùng với ý nghĩa khác thì phải khai báo.
  - Tên do người lập trình đặt: cần khai báo trước khi sử dụng.
- Hằng: Đại lượng có giá trị không thay đổi trong quá trình thực hiện chương trình.
- Biến: Đại lượng được đặt tên. Giá trị của biến có thể thay đổi trong quá trình thực hiện chương trình.

CÂU HỎI VÀ BÀI TẬP

1. Tại sao người ta phải xây dựng các ngôn ngữ lập trình bậc cao?

2. Chương trình dịch là gì? Tại sao cần phải có chương trình dịch?

3. Biên dịch và thông dịch khác nhau như thế nào?

4. Hãy cho biết các điểm khác nhau giữa tên dành riêng và tên chuẩn.

5. Hãy tự viết ra ba tên đúng theo quy tắc của Pascal và có độ dài khác nhau.

6. Hãy cho biết những biểu diễn nào dưới đây không phải là biểu diễn hằng trong Pascal và chỉ rõ lỗi trong từng trường hợp:

a) 150.0 b) -22 c) 6,23 d) '43'
e) A20 f) 1.06E-15 g) 4+6 h) 'C
i) 'TRUE'

§3. CẤU TRÚC CỦA NGÔN NGỮ LẬP TRÌNH C++

Các thành phần cấu tạo nên 1 chương trình C++ bao gồm các câu lệnh, từ khóa, các cấu trúc điều khiển, hằng, biến và các toán tử.

1. Cấu trúc một chương trình C++

- Phần header: bắt đầu bằng từ khóa `#include <tên thư viện>`

```
- Phần khai báo.
- int main()          //Hàm chính
      {
      Các lệnh. Cuối mỗi lệnh là dấu ";"
      }
```

Câu lệnh có hai dạng:

- Lệnh đơn.
- Lệnh ghép: Gồm nhiều câu lệnh đơn được đặt giữa hai dấu { và }. Lệnh ghép còn được gọi là một khối lệnh.

2. Các kiểu dữ liệu chuẩn

Bộ nhớ máy tính có cấu trúc gồm nhiều ô nhớ (bit). Các ô nhớ này được đánh địa chỉ và tổ chức thành các byte liên tiếp. Mỗi byte lưu trữ được $2^8 = 256$ số.

Để biểu diễn những dữ liệu phức tạp hơn, ta ghép các `byte` lại để lưu được các số lớn hơn. Sau đây là các kiểu dữ liệu cơ bản của C++.

a. Kiểu số nguyên

Số nguyên có dấu		Số byte	Số nguyên không dấu	
`char`	`-128..127`	`1 byte`	`unsigned char`	`0..256`
`int`	`-32.768..32.767`	`2 byte`	`unsigned int`	`0..65.535`
`long`	$-2^{31}..2^{31}-1$	`4 byte`	`unsigned long`	$0..2^{32}$
`long long`	$-2^{63}..2^{63}-1$	`8 byte`	`unsigned long long`	$0..2^{64}$

b. Kiểu số thực

`float`	`4 byte`	$3.4.10^{-38}$ ..$3.4.10^{38}$: Số thực với độ chính xác 7 chữ số sau dấu thập phân
`double`	`8 byte`	$1.7.\ 10^{-308}$ ..$1.7.10^{308}$: Số thực với độ chính xác 14 chữ số sau dấu thập phân

c. Kiểu logic: `bool (1 byte)` gồm hai giá trị `true(1)`, `false(0)`

3. Biến: Là đại lượng có giá trị thay đổi được trong quá trình thực thi chương trình.

a. Khai báo: `<tên kiểu dữ liệu><tên biến> ;`

Ví dụ:
```
int a, b; //Khai báo hai biến a,b kiểu số nguyên
float c;  //Khai báo biến c kiểu float
```

b. Lệnh gán: `<tên biến> = <biểu thức>;`

Phép gán sẽ tính giá trị của biểu thức vế phải và gán giá trị đó vào biến ở vế trái. Kết quả của biểu thức và biến phải thuộc cùng một kiểu dữ liệu.

Ví dụ:
```
int a(10),b;          //Khai báo biến a,b và gán a=10
b=a*a+1;              //b = 10² +1 = 101
```

4. Hằng: Là đại lượng có giá trị thay không thay đổi trong toàn bộ chương trình.

a. Khai báo: `<tên kiểu dữ liệu>const<tên hằng> (giá trị);`

Ví dụ:
```
int const maxn(1e9); //Khai báo hằng maxn = 10⁹
float pi = 3.14;     //Khai báo hằng số pi = 3.14
```

5. Chương trình C++ đầu tiên

Bài 1: Viết chương trình in ra màn hình câu chào "Hello world!!!"

```
#include <iostream>          //Khai báo sử dụng thư viện nhập/xuất chuẩn
using namespace std;         //Luôn phải có lệnh này
int main()
{
cout << "Hello world!!!";   //Lệnh cout trong thư viện iostream xuất ra màn
return 0;                    //hình chuỗi "Hello world!!!"
}
```

Để dịch chương trình, ta ấn tổ hợp phím Ctrl-F9. Để dịch và chạy chương trình, ấn F9.

Bài 2: Viết chương trình nhập hai số nguyên `a,b`. Tính tổng `a+b`.

```
#include <iostream>
using namespace std;
int main()
{
   int a,b;
   cout << "Nhap a,b = ";
   cin >> a >> b;           //Mỗi lượt nhập vào phải có lệnh >>
   cout << a+b << endl ;    //Mỗi lượt xuất ra phải có lệnh <<
   return 0;
}
```

Ghi chú: Biến `a,b` trong cách khai báo trên là biến riêng (biến cục bộ) của riêng chương trình chính. Ta có thể khai báo trên dòng `int main()` để `a,b` trở thành biến toàn cục.

Bài 3: Viết chương trình nhập hai số nguyên a,b. Tính thương a/b.

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

```
    cout << "Nhap a,b = "; cin >> a >> b;
    cout << float(a)/b << endl ;
    return 0;
}
```

Ghi chú: Trong đoạn chương trình trên, nếu viết `cout << a/b` thì kết quả nhận được là phần nguyên của phép chia. Để kết quả phép chia là số thực ta dùng toán tử chuyển kiểu với cú pháp `float(a)` hoặc `(float) a` để biến `a` thành số thực.

6. Giới thiệu các toán tử của C++

a. Toán tử số học

Cộng (+), trừ (-), nhân (*).

Phép chia (/): Nếu hai toán hạng đều là số nguyên thì kết quả `a/b` là phần nguyên của phép chia.
Nếu ít nhất một trong hai toán hạnglà số thực thì kết quả là `a/b` .

Phép chia lấy phần dư: % không áp dụng được nếu các toán hạng là số thực.

b. Toán tử so sánh

`> , >= , < , <=,!=(khác)`

So sánh bằng hai toán hạng: `==`

Chú ý: Khi thực hiện phép so sánh giữa hai toán hạng, kết quả thu được thuộc kiểu `bool` (`true` hoặc `false`). Điều này là hợp lý vì các biểu thức so sánh đều là các mệnh đề logic.

Ví dụ: So sánh 10^2 với $6^2 + 8^2$

```
a = 10 ; b = 6 ; c = 8;
bool x = (a*a == b*b+c*c); // Kết quả là true
```

c. Toán tử logic

Các phép toán logic tác động lên các mệnh đề và thu được kết quả là một mệnh đề đúng hoặc sai.

Phép "và": `&&`

Mệnh đề A	Mệnh đề B	Mệnh đề A&&B
Đúng	Đúng	Đúng
Đúng	Sai	Sai
Sai	Đúng	Sai
Sai	Sai	Sai

Phép "hoặc": `||`

Mệnh đề A	Mệnh đề B	Mệnh đề A||B
Đúng	Đúng	Đúng
Đúng	Sai	Đúng
Sai	Đúng	Đúng
Sai	Sai	Sai

Phép phủ định: `!`

Mệnh đề A	Mệnh đề !A
Đúng	Sai

Sai	Đúng

d. Một số hàm toán học thường dùng

Để sử dụng các hàm toán học, ta cần khai báo sử dụng thư viện cmath: `<include cmath>`

Căn bậc hai: `sqrt(x)` Giá trị tuyệt đối `abs(x)` Các hàm lượng giác: `sin,cos,tan`

Logarit Nepe: `log(x)` Logarit thập phân `log10(x)`

Lũy thừa: `pow(a,x)`

Làm tròn lên: `ceil(x).` Ví dụ: `ceil(2.8) = 3` ; `ceil(-3.2) = 3`

Làm tròn xuống: `floor(x)`. Ví dụ: `floor(2.8) = 2` ; `floor(-3.2) = 4`

BÀI TẬP

Bài 1. Nhập ba số nguyên a,b,c. Tính $S = \dfrac{\sqrt{a^3+b^3+c^3}}{|abc|+1}$

Bài 2. Nhập ba số nguyên a,b,c. Tính $d = b^2 - 4ac$

Bài 3. Nhập bán kính một hình tròn. Tính chu vi, diện tích hình tròn đó.

Bài 4. Nhập chiều dài a và chiều rộng b một hình chữ nhật. Tính chu vi và diện tích hình chữ nhật đó.

Bài 5. Nhập cạnh a của một hình vuông. Tính chu vi và diện tích hình vuông đó.

Bài 6. Nhập hai số a và b. Thực hiện hoán vị hai số a và b cho nhau.

Bài 7. Viết chương trình cho phép tính trung bình cộng của bốn số. Nhập bốn số vào bốn biến a, b, c, d.

Bài 8. Viết chương trình cho phép tính trung bình cộng của bốn số với điều kiện chỉ được sử dụng hai biến.

Bài 9. Viết chương trình cho biết chữ số hàng trăm, hàng chục, hàng đơn vị của một số có ba chữ số. Ví dụ khi nhập số 357 thì máy in ra:

- Chữ số hàng trăm: 3.
- Chữ số hàng chục: 5.
- Chữ số hàng đơn vị: 7.

Bài 10. Lập trình nhập từ bàn phím các số thực a, b, c, d và x. Tính và đưa ra màn hình giá trị biểu thức $ax^3 + bx^2 + cx + d$.

Bài 11.Một người đi xe đạp với tốc độ 10km/h và một người đi xe máy với tốc độ 30km/h cùng xuất phát từ một vị trí, cùng một thời điểm và đi cùng một hướng. Lập trình tính khoảng cách giữa hai người sau t giờ (t là số nguyên dương, t ≤ 15). Dữ liệu t nhập từ bàn phím. Kết quả đưa ra màn hình.

Bài 12. Hiền gọi điện thoại trao đổi bài với Minh. Cứ mỗi phút dùng điện thoại phải trả a đồng. Cuộc trao đổi kéo dài t phút. Hãy lập trình tính và đưa ra màn hình số tiền mà mẹ Hiền cuối tháng phải thanh toán cho cuộc trao đổi này. Các số liệu a và t (nguyên dương) được nhập từ bàn phím.

Bài 13. Theo quy định của nhà trường, mỗi trường hợp không đeo thẻ học sinh sẽ bị trừ 3 điểm thi đua của lớp, mỗi trường hợp nói chuyện trong lớp bị trừ 2 điểm và mỗi trường hợp đi muộn trừ 5 điểm. Sổ đầu bài ghi nhận trong tháng lớp có t trường hợp không đeo thẻ, n trường hợp nói chuyện và m trường hợp đi muộn. Hãy nhập các dữ liệu này vào từ bàn phím và đưa ra màn hình số điểm thi đua mà lớp bị trừ trong tháng. Tất cả các dữ liệu đều là số nguyên.

Bài 14. Để bù đắp thiệt hại cho nhân dân trong công tác phòng chống dịch cúm gia cầm H5N1, nhà nước hỗ trợ cho các hộ có gia cầm bị thiêu hủy theo định mức sau: mỗi con gia cầm từ một tháng tuổi trở xuống a đồng, mỗi con gia cầm trên một tháng tuổi và dưới ba tháng tuổi là b đồng, từ ba tháng tuổi trở lên là c đồng một con. Một hộ nông dân có đàn gia cầm n con phải thiêu hủy, trong đàn có m con dưới ba tháng tuổi, trong số đó có k con từ một tháng tuổi trở xuống (0 ≤ k ≤ m ≤ n ≤ 10 000).

§4. CẤU TRÚC RẼ NHÁNH

1. Câu lệnh if:

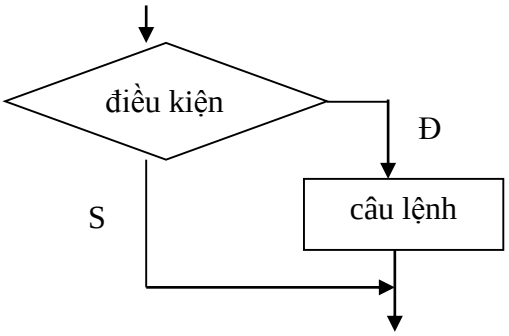
a. Dạng thiếu:

```
if (<điều kiện>)
      <câu lệnh>;
```

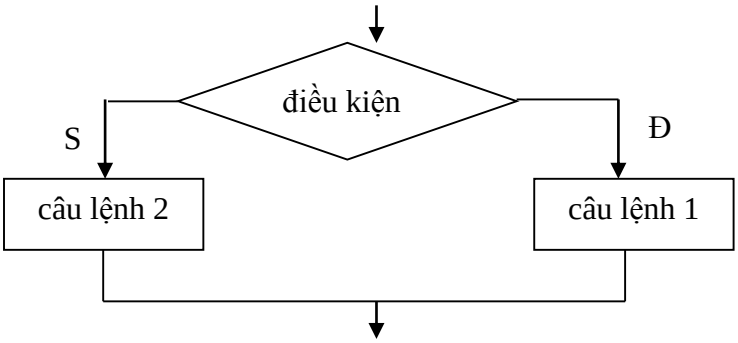
b. Dạng đủ:

```
if (<điều kiện>)
      <câu lệnh 1>;
else
<câu lệnh 2>;
```

Ở dạng thiếu: *điều kiện* sẽ được tính và kiểm tra. Nếu *điều kiện* đúng (có giá trị true) thì *câu lệnh* sẽ được thực hiện, ngược lại thì *câu lệnh* sẽ bị bỏ qua.



Ở dạng đủ: *điều kiện* cũng được tính và kiểm tra. Nếu *điều kiện* đúng (có giá trị true) thì *câu lệnh 1* sẽ được thực hiện, ngược lại thì *câu lệnh 2* sẽ được thực hiện.



Ví dụ 1:

```
if (D<0)
      cout<<"Phuong trinh vo nghiem";
```

Ví dụ 2:

```
if (a % 3 == 0)
      cout<<"a chia het cho 3";
else
      cout<<"a khong chia het cho 3;
```

Ví dụ 3: Để tìm số lớn nhất `max` trong hai số `a` và `b`, có thể thực hiện bằng 2 cách:

- Cách 1:

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

- Cách 2:

```
if (b>a)
```

```
      max=b;
else
      max=a;
```

2. Một số ví dụ:

Ví dụ 1: Tìm nghiệm thực của phương trình bậc hai: $ax^2 + bx + c = 0$, với $a \neq 0$
Input: Các hệ số a, b, c nhập từ bàn phím
Output: Đưa ra màn hình các nghiệm thực hoặc thông báo "Phương trình vô số nghiệm"

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
   float a,b,c;
   cout << "Nhap a,b,c = "; cin >> a >> b>>c;
   float d=b*b-4*a*c;
   if (d<0)
     cout<<"Phuong trinh vo nghiem.";
   else
   {
     float x1=(-b-sqrt(d))/(2*a);
     float x2=-b/a - x1;
     cout<<" x1 = "<<x1<<endl;
     cout<<" x2 = "<<x2<<endl;
   }
   return 0;
}
```

Ví dụ 2: Tìm số ngày của năm N, biết rằng năm nhuận là năm chia hết cho 400 hoặc chia hết cho 4 nhưng không chia hết cho 100. Ví dụ, các năm 2000, 2004 là năm nhuận và có số ngày là 366, các năm 1900, 1945 không phải là năm nhuận và có số ngày là 365.
Input: N nhập từ bàn phím
Output: Đưa ra số ngày của năm N ra màn hình

```
#include <iostream>
using namespace std;
int main()
{
   int N,SN;
   cout << "Nam: "; cin >> N;
   if ((N % 400 == 0)||((N % 4 == 0) && (N % 100 != 0)))
     SN=366;
```

```
    else
      SN=365;
    cout<<"So ngay cua nam ",N," la ",SN);
    return 0;
}
```

3. Toán tử điều kiện:

```
<điều kiện> ? <biểu thức 1>:<biểu thức 2>
```

điều kiện sẽ được tính và kiểm tra. Nếu *điều kiện* đúng thì kết quả là *biểu thức 1*, ngược lại kết quả là *biểu thức 2*.

Ví dụ: Để tìm số lớn nhất max trong hai số a và b:

```
max = a > b ? a : b;
```

BÀI TẬP

Bài 1. Tìm ước chung lớn nhất của hai số nguyên dương M và N

Bài 2. *Bộ số Pi-ta-go*

Biết rằng bộ ba số nguyên a, b, c được gọi là bộ số Pi-ta-go nếu tổng các bình phương của hai số bằng bình phương của số còn lại. Viết chương trình nhập từ bàn phím ba số nguyên dương a, b, c và kiểm tra xem chúng có là bộ số Pi-ta-go không

Ý tưởng: Kiểm tra xem có đẳng thức nào trong ba đẳng thức sau đây xảy ra hay không:

$$a^2=b^2+c^2$$
$$b^2=a^2+c^2$$
$$c^2=a^2+b^2$$

Bài 3. Viết chương trình giải phương trình bậc nhất ax+b=0

Bài 4. Viết chương trình giải phương trình bậc hai $ax^2+bx+c=0$ (a bất kì)

Bài 5. Lập trình nhập từ bàn phím ba số nguyên a, b, c, đưa ra màn hình giá trị lớn nhất trong ba số đó.

Bài 6. Lập trình nhập từ bàn phím bốn số thực a, b, c, d. Đưa ra màn hình giá trị nhỏ nhất và giá trị lớn nhất của các số đó.

Bài 7. Lập trình nhập từ bàn phím hai số nguyên khác nhau m và n, thay số nhỏ hơn bằng hiệu của số lớn và số bé, thay số lớn bằng tổng của hai số ban đầu. Đưa các giá rị mới của m và n ra màn hình.

Bài 8. Cho ba số nguyên m, n, k. Nếu ba số này theo thứ tự nhập vào tạo thành một cấp số cộng thì tăng gấp đôi mỗi số, trong trường hợp ngược lại thì giảm mỗi số một đơn vị. Viết chương trình thực hiện yêu cầu trên.

Bài 9. Viết chương trình xét xem một tam giác có là tam giác đều, tam giác cân hay không khi biết ba cạnh của tam giác.

Bài 10. Ba bạn An, Bình và Cường cùng tham gia một trò chơi như sau: Mỗi bạn nắm trong tay một đồng xu, mỗi đồng xu có hai trạng thái : sấp và ngửa. Theo hiệu lệnh, cả ba bạn cùng đưa đồng xu của mình ra phía trước. Nếu cả ba đồng xu cùng sấp hoặc cùng ngửa thì chưa phát hiện người thua cuộc (hòa nhau). Nếu một bạn có trạng thái đồng xu khác với hai bạn kia(nghĩa là đồng xu của bạn ấy sấp còn hai người kia ngửa và ngược lại đồng xu của bạn ấy ngửa thì hai người kia sấp) thì bạn đó thắng cuộc. Hãy viết chương trình mô phỏng trò chơi trên.

Bài 11. Viết chương trình dịch các ngày trong tuần sang tiếng anh

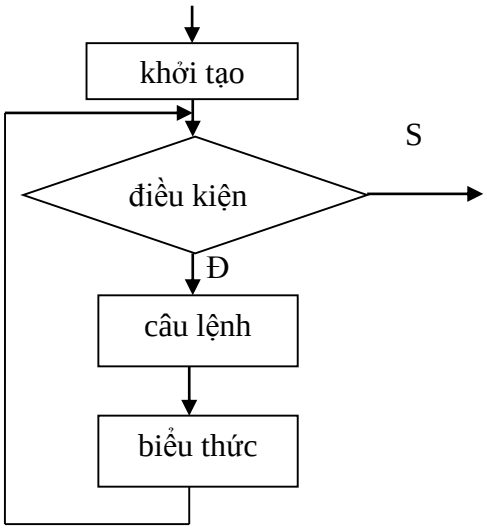
2	3	4	5	6	7	8
Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday

§5. CẤU TRÚC LẶP

1. Câu lệnh for:

Cú pháp: `for (<khởi tạo>;<điều kiện>;<biểu thức>)`
`<câu lệnh>;`

Cơ chế hoạt động của lệnh **`for`**:
- Bước 1: Các lệnh ở phần *khởi tạo* được thực hiện đầu tiên. Khởi tạo có thể có nhiều lệnh và viết ngăn cách nhau bởi dấu phẩy.
- Bước 2: Tính và kiểm tra *điều kiện*. Nếu *điều kiện* sai (có giá trị false) thì thoát khỏi vòng lặp
- Bước 3: Thực hiện *câu lệnh* (có thể là lệnh đơn hoặc lệnh ghép)
- Bước 4: Thực hiện các lệnh ở phần *biểu thức*. *Biểu thức* có thể có nhiều lệnh và viết ngăn cách nhau bởi dấu phẩy. Quay lại bước 2



Một số ví dụ minh họa

Để hiểu rõ hơn về câu lệnh ***for*** ta quan sát cách thực thi của hai đoạn chương trình sau:

Ví dụ 1: Lệnh for đơn giản

```
for (int i=1;i<=10;i++)
      cout<<i*i<<endl;
```

Bước 1: `i` được khởi gán là `1`
Bước 2: So sánh điều kiện `i<=10` → Đúng
Bước 3: Thực hiện lệnh: In ra `1² = 1`. Biến i tăng lên `1` đơn vị là `2`.
Lặp lại bước 2: So sánh điều kiện `i<=10` → Đúng
Bước 3: Thực hiện lệnh: In ra `2² = 4`. Biến `i` tăng lên 1 đơn vị là `3`.
…
Quá trình lặp chỉ kết thúc khi điều kiện không đúng nữa, nghĩa là khi đó `i=11`.
Kết quả ta thu được dãy số `1 4 9 16 … 100` trên màn hình.

Ví dụ 2: Lệnh for với hai biến chạy song song

```
for (int i=1,j=10;i<=5;j>=1;i++,j--)
      cout<<i+j<<endl;
```

Bước 1: `i` được khởi gán là `1, j` khởi gán là `10`

Bước 2: So sánh điều kiện `i<=5 ; j>=1` → Đúng

Bước 3: Thực hiện lệnh: In ra `1+10 = 11`. Biến `i` tăng lên thành `2` ; `j` giảm thành `9`.

Lặp lại bước 2: So sánh điều kiện `i<=5 ; j>=1` → Đúng

Bước 3: Thực hiện lệnh: In ra `2+9 = 11`. Biến `i` tăng lên thành `3` ; `j` giảm thành `8`.

…

Quá trình lặp chỉ kết thúc khi ***một trong hai điều kiện*** bị vi phạm, nghĩa là khi đó `i=6 ; j=5`

Kết quả ta thu được dãy: `11 11 11 11 11` trên màn hình ứng với 5 lượt lặp.

Ví dụ 3: Viết chương trình thực hiện việc nhập từ bàn phím hai số nguyên dương M và N (M<N), tính và đưa ra màn hình tổng các số chia hết cho 3 hoặc 5 trong phạm vi từ M đến N.

```
#include <iostream>
using namespace std;
int main()
{
   int m,n,t;
   cout << "Nhap m,n = "; cin>>m>>n;
   t=0;
   for (int i=m;i<=n;i=i+1)
     if ((i % 3 == 0) || (i % 5 == 0))
        t=t+i;
   cout<<"Ket qua: "<<t;
   return 0;
}
```

Ghi chú: Giới thiệu một số cách viết khác của câu lệnh `for` cho bài toán tính tổng `1+2+3+…+n`

Mã nguồn	Ghi chú

`int sum=0,k=1;`
`for (;sum<n;)`
`{`
`sum=sum+k;`
`k=k+1;`
`}`
	Sử dụng cấu trúc ***for*** nhưng không có phần *khởi tạo* và *biểu thức*.

`int sum=0,k=1;`
`for (;sum<n;k=k+1)`
`sum=sum+k;`
	Sử dụng cấu trúc ***for*** nhưng không có phần *khởi tạo*.

`int sum,k;`
`for (sum=0,k=1;sum<n;k=k+1)`
`sum=sum+k;`
	Sử dụng cấu trúc ***for*** đầy đủ.

`int sum=0,k;`
`for (k=1;sum<n;k=k+1)`
`sum=sum+k;`
	Đây là cách viết thường dùng.

`int sum,k;`	Đây là cách viết cấu trúc ***for*** một cách cô

`for (sum=0,k=1;sum<n;k=k+1,sum=sum+k);`	động. Tuy nhiên cách này ít khi được dùng vì sẽ làm chương trình khó hiểu, khó bảo trì và bắt lỗi.

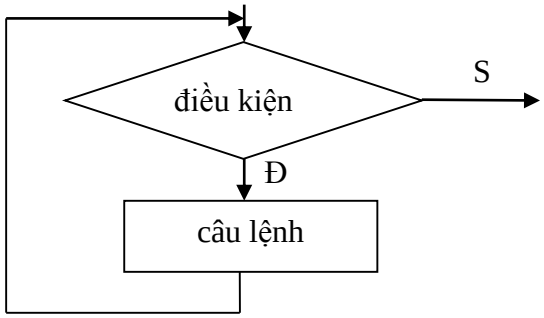
2. Câu lệnh while:

Khác với câu lệnh **For** là lệnh lặp với số lần xác định bởi biến điều khiển, câu lệnh **while** được dùng để lặp đi lặp lại một lệnh với số lần không cần xác định trước.

Cú pháp:
```
while (<điều kiện>)
        <câu lệnh>;
```

Cơ chế hoạt động của câu lệnh while:

Khi *điều kiện* đúng (có giá trị true) thì thực hiện *câu lệnh*, sau đó kiểm tra lại *điều kiện* để quyết định có lặp tiếp hay không. Nói cách khác: *điều kiện* còn đúng (có giá trị **true**) thì tiếp tục thực hiện lệnh sau **while**.



Ví dụ 1: Mật khẩu đơn giản

Lặp lại việc nhập vào từ bàn phím một kí tự cho đến khi kí tự được nhập vào là **'X'**. Nếu kí tự được nhập không phải **'X'** thì in ra thông báo **"password incorrect. Re-enter password"**.

```
#include <iostream>
using namespace std;
int main()
      {
      char ch;
      cout<<"Password =";  cin>>ch;
      while (ch!='X')
            {
            cout<<"Password incorrect. Re-enter password";
            cin>>ch;
            }
   return 0;
}
```

Ví dụ 2: Tổng 1 đến n

Nhập từ bàn phím số nguyên **n**, tính tổng **S=1+2+...+n**

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

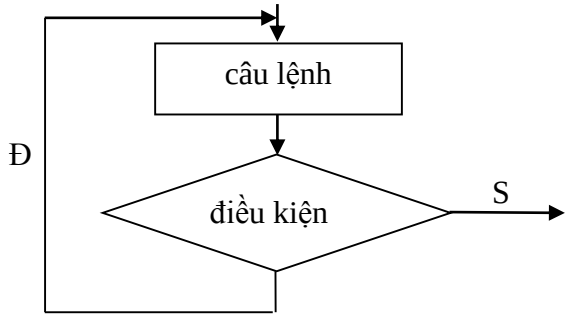
```
{
   int n,s=0,i=1;
   cout << "Nhap n = "; cin>>n;
   while (i<=n)
   {
     s=s+i;
     i=i+1;
   }
   cout<<"Ket qua: "<<s;
   return 0;
}
```

3. Câu lệnh do-while:

```
do
<câu lệnh>;
while (<điều kiện>);
```

Hoạt động của câu lệnh do-while:

- Bước 1: Thực hiện *câu lệnh*.
- Bước 2: Nếu *điều kiện* đúng (có giá trị true) trở lại bước 1.
- Bước 3: Thực hiện lệnh kế tiếp (sau lệnh do while)



Ví dụ 1: Chương trình sau thực hiện việc nhập và tính tổng các số nguyên dương.

```
#include <iostream>
using namespace std;
int main()
{
   int x,s=0
   do
   {
      cout<<"Nhap x: "; cin>>x;
      if (x>0)
         s=s+x;
   }while (x>0);
   cout<<"Ket qua: "<<s;
```

```
    return 0;
}
```

BÀI TẬP

Bài 1: Lập trình tính:

e(n)=$1+\frac{1}{1!}+\frac{1}{2!}+\cdots+\frac{1}{n!}+\cdots$ cho đến khi $\frac{1}{n!}<2.10^{-6}$

Bài 2: Lập trình để giải bài toán cổ sau:

Vừa gà vừa chó
Bó lại cho tròn
Ba mươi sáu con
Một trăm chân chẵn
Hỏi có bao nhiêu con mỗi loại?

Bài 3: Viết chương trình nhập vào N, tính tổng và in ra tất cả các ước số của N

Ví dụ: N=20; - Tập các ước số của 20 là: 1, 2, 4, 5, 10, 20 - Tổng các ước số : 42

Bài 4: Viết chương trình nhập vào số nguyên N, tính tổng:

a. $S_1 = 1 + 2 + 3 + ... + N$

b. $S_2 = 1^2 + 2^2 + 3^2 + ... + N^2$

c. $S_3 = 1 + 3 + 5 + ... + (2N+1)$

d. $S_4 = 2 + 4 + 6 + ... + (2N)$

e. $S_5 = 1 - 2 + 3 – 4 + ... + (-1)^{N+1}N$

Bài 5: Viết chương trình nhập vào số nguyên N và số thực x, tính tổng:

a. $S_1 = x + 2x + 3x + ... + Nx$

b. $S_2 = x + x^2 + x^3 + ... + x^N$

c. $S_3 = x + x^3 + x^5 + ... + x^{2N+1}$

d. $S_4 = x^2 + x^4 + x^6 + ... + x^{2N}$

e. $S_5 = x - x^2 + x^3 – x^4 + ... + (-1)^{N+}x^N$

Bài 6: Viết chương trình nhập vào số nguyên dương N (N<100):

a. In ra tất cả các số nguyên tố nhỏ hơn N

b. Tính tổng các số nguyên tố nhỏ hơn N

Bài 7: Một số có tổng các ước nhỏ hơn nó bằng chính nó được gọi là số hoàn hảo.

Ví dụ: 6 có các ước nhỏ hơn nó là 1, 2, 3. Tổng là 1 + 2 + 3 = 6.

Viết chương trình xét xem một số n được nhập từ bàn phím có phải là số hoàn hảo không.

Bài 9: Viết chương trình tìm các số hoàn chỉnh nhỏ hơn n (Với n được nhập từ bàn phím).

Bài 10: In bảng cửu chương n (Với n nhập từ bàn phím)

Bài 11: Viết chương trình lần lượt in các bảng cửu chương.

Bài 12: Viết chương trình xét xem một số n có phải là số nguyên tố không?

Bài 13: Viết chương trình in ra tất cả các số nguyên tố bé hơn hoặc bằng n?

Bài 14: Viết chương trình nhập vào số nguyên N:

a. Xuất ra màn hình tam giác có chiều cao N có tính chất sau: hàng k gồm k số đầu tiên trong chuỗi Fibonacci: $F_0=1$, $F_1=1$

$F_n = F_{n-1} + F_{n-2}$, n>1

b. Xuất ra tam giác Pascal với chiều cao N.

Bài 15: Viết chương trình nhập vào số nguyên N:

a. In ra các ước số chẵn của N

b. In ra các ước số lẻ của N.

§6. KIỂU MẢNG

1. Kiểu mảng một chiều:

Mảng một chiều là dãy hữu hạn các phần tử cùng kiểu. Mảng được đặt tên và mỗi phần tử của nó có một chỉ số. Để mô tả mảng một chiều cần xác định kiểu của các phần tử và cách đánh số các phần tử của nó.
Để người lập trình có thể xây dựng và sử dụng kiểu mảng một chiều, các ngôn ngữ lập trình có quy tắc, cách thức cho phép xác định:

- Tên kiểu mảng một chiều
- Số lượng phần tử
- Kiểu dữ liệu của phần tử
- Cách khai báo biến mảng
- Cách tham chiếu đến phần tử.

a. Khai báo:

```
<kiểu dữ liệu> <tên mảng>[<kích thước>];
```

Trong đó,
kiểu dữ liệu: mô tả kiểu của mỗi phần tử thuộc mảng, như int, char, double, ...
tên mảng: mô tả tên biến mảng đang định nghĩa. Quy tắc đặt tên cho mảng tương tự như đặt tên biến.
kích thước: chỉ ra số lượng tối đa mà mảng có thể lưu trữ, giá trị này phải là một số nguyên dương.
Ví dụ: Khai báo mảng một chiều có tên là *thang* gồm 12 phần tử là những số nguyên.

```
int thang[12];
```

b. Tham chiếu tới phần tử của mảng một chiều:

Giá trị của chỉ số trong mảng một chiều là số nguyên thuộc đoạn [0, *kích thước* – 1].

Cách tham chiếu tới phần tử của mảng một chiều: `<tên mảng>[<chỉ số>]`

Với biến mảng *tháng* được khai báo ở trên thì *thang[0]* là phần tử đầu tiên, còn *thang[11]* là phần tử cuối cùng.

Lưu ý: Ngôn ngữ C/C++ không kiểm tra giới hạn của mảng trong thời gian biên dịch lẫn thực thi chương trình. Nói cách khác, việc sử dụng chỉ số mảng vượt quá kích thước khai báo (ví dụ thang[12], thang[20], ...) sẽ không khiến cho hệ thống đưa ra bất cứ cảnh báo lỗi nào và đây là một trong những nguyên nhân khiến chương trình hoạt động không ổn định hoặc thậm chí làm dừng chương tình. Để đảm bảo tính an toàn, trong mọi trường hợp, người lập trình phải hết sức chú ý đến chi tiết này khi làm việc với mảng.

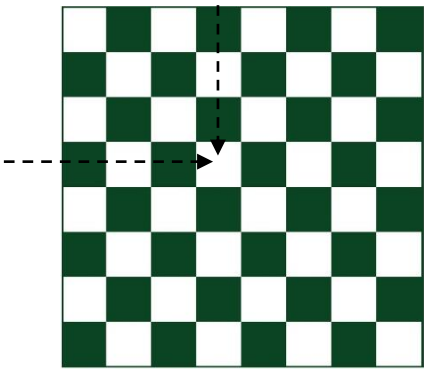
Ví dụ: Tìm phần tử lớn nhất của dãy số nguyên
Input: Số nguyên dương `N(N ≤ 250)` và dãy `N` số nguyên dương `a1, a2, ..., aN`, mỗi số đều không quá `500`.
Output: Chỉ số và giá trị của phần tử lớn nhất trong dãy số đã cho (nếu có nhiều phần tử lớn nhất chỉ cần đưa ra một trong số chúng)

```
#include <iostream>
using namespace std;
int main()
{
   int n;
   int a[250];
```

```
    cout<<"Nhap so luong phan tu cua day so, n= "; cin>>n;
    for(int i=0;i<n;i=i+1)
    {
        cout<<"Phan tu thu "<<i<<" = ";
        cin>>a[i];
    }
    int max=a[0], csmax=0;
    for(int i=1;i<n;i=i+1)
        if (max<a[i])
        {
            max=a[i];
            csmax=i;
        }
    cout<<"Gia tri cua phan tu max: "<<max<<endl;
    cout<<"Chi so cua phan tu max: "<<csmax<<endl;
    return 0;
}
```

2. Kiểu mảng hai chiều:

Một hình ảnh gợi ý trực quan cho mảng hai chiều trong ngôn ngữ lập trình C/C++ là bàn cờ quốc tế có 8 dòng và 8 cột. Để định vị một ô trong bàn cờ này, chúng ta cần đến hai thông tin: mã hiệu dòng và mã hiệu cột.



a. Khai báo:

```
<kiểu dữ liệu> <tên mảng>[<kích thước dòng>][<kích thước cột>];
```

trong đó,

kích thước dòng và *kích thước cột* là hai số nguyên dương chỉ ra số lượng dòng và số lượng cột của mảng hai chiều.

Ví dụ: Khai báo mảng mảng hai chiều có tên là *chessboard* gồm 8 dòng và 8 cột.

```
char chessboard[8][8];
```

b. Tham chiếu tới phần tử của mảng hai chiều:

Giá trị của *chỉ số dòng* là số nguyên thuộc đoạn [0, *kích thước dòng* – 1] và giá trị của chỉ số cột là số nguyên thuộc đoạn [0, *kích thước cột* – 1].

Tham chiếu tới phần tử của mảng hai chiều: **<tên mảng>[<chỉ số dòng>][<chỉ số cột>]**

Ví dụ: Chương trình sau nhập vào từ bàn phím các phần tử của mảng hai chiều b gồm 5 dòng, 7 cột với các phần tử là các số nguyên và một số nguyên k. Sau đó, đưa ra màn hình các phần tử của mảng có giá trị nhỏ hơn k.

```
#include <iostream>
using namespace std;
int main()
{
    int b[5][7];
    cout<<"Nhap cac phan tu cua mang theo dong: "<<endl;
    for(int i=0;i<5;i=i+1)
        for(int j=0;j<7;j=j+1)
          cin>>b[i][j];
    int k;
    cout<<"Nhap vao gia tri k = "; cin>>k;
    int d=0;
    for(int i=0;i<5;i=i+1)
        for (int j=0;j<7;j=j+1)
          if (b[i][j]<k)
          {
             cout<<b[i][j]<<" ";
             d=d+1;
          }
    if(d==0)
      cout<<"Khong co phan tu nao nho hon "<<k;
   return 0;
}
```

BÀI TẬP

Bài 1. Tạo mảng A gồm n (n≤100) số nguyên, mỗi số có giá trị tuyệt đối không vượt quá 300. Tính tổng các phần tử của mảng là bội số của một số nguyên dương k cho trước.

Bài 2. Viết chương trình tìm phần tử có giá trị lớn nhất của mảng và đưa ra màn hình chỉ số và giá trị của phần tử tìm được. Nếu có nhiều phần tử có cùng giá trị lớn nhất thì đưa ra phần tử có chỉ số nhỏ nhất.

Bài 3. Viết chương trình nhập vào mảng N số nguyên (N<100)

a. In các phần tử trong mảng

b. Đếm và in số phần tử chẵn trong mảng

c. Đếm và in số phần tử lẻ trong mảng

d. Đếm và in tất cả các phần tử là số nguyên tố trong mảng

Bài 4. Viết chương trình nhập vào mảng N số nguyên (N<100)

a. Tính tổng và in tất cả các phần tử chẵn trong mảng

b. Tính tổng và in tất cả các phần tử lẻ trong mảng

c. Tính tổng và in tất cả các phần tử âm trong mảng

d. Tính tổng và in tất cả các phần tử lẻ không âm trong mảng

e. Tính tổng và in tất cả các phần tử là số nguyên tố trong mảng.

Bài 5. Viết chương trình nhập vào mảng N số nguyên (N<100), nhập số nguyên x:

a. Kiểm tra x có trong mảng không?

b. Đếm số lần xuất hiện phần tử x trong mảng

c. In ra tất cả các vị trí x xuất hiện trong mảng.

Bài 6. Viết chương trình nhập vào mảng N số nguyên (N<100)

a. Kiểm tra mảng có sắp thứ tự tăng không?

b. Kiểm tra mảng có sắp thứ tự giảm không?

c. Sắp xếp mảng theo thứ tự tăng (nếu mảng chưa được sắp tăng)

d. Sắp xếp mảng theo thứ tự giảm (nếu mảng chưa được sắp thứ tự giảm)

e. Sắp xếp mảng theo thứ tự tăng các phần tử có giá trị chẵn (vị trí các phần tử có giá trị lẻ không thay đổi)

f. Sắp xếp mảng theo thứ tự tăng theo các phần tử có giá trị lẻ (vị trí các phần tử có giá trị chẵn không thay đổi)

g. Sắp xếp mảng theo thứ tự tăng theo các phần tử có giá trị chẵn và giảm dần theo các phần tử có giá trị lẻ (vị trí các phần tử chẵn lẻ không thay đổi)

Bài 7. Viết chương trình nhập vào ma trận M dòng N cột (M<100, N<100)

a. In kết quả ma trận vừa nhập ra màn hình theo dòng cột

b. In các phần tử chẵn trong ma trận

c. In các phần tử lẻ trong ma trận

d. In các phần tử là số nguyên tố trong ma trận

Bài 8. Viết chương trình nhập ma trận M dòng N cột (M<100,N<100)

a. Với mỗi dòng, sắp xếp các phần tử theo thứ tự tăng dần

b. Với mỗi cột, sắp xếp các phần tử theo thứ tự giảm dần

§7. KIỂU DỮ LIỆU CÓ CẤU TRÚC

Xét bài toán quản lý học sinh trong Nhà trường, ta biết rằng mỗi học sinh phải lưu trữ các yếu tố như Họ tên, điểm Toán, Văn, Ngoại ngữ … Để một biến có thể lưu trữ một tập hợp các dữ liệu như vậy ta cần một kiểu dữ liệu mới có tên gọi là kiểu struct (kiểu cấu trúc). Các thành phần của tạo nên struct được gọi là các trường

1. Khai báo kiểu struct:

```
struct [tên cấu trúc]
{
      Định nghĩa trường 1;
      Định nghĩa trường 2;
            ……
};
```

Ví dụ 1:

```
struct hoc_sinh
{
      string hoten;
      float Toan,Van,TB;
};
```

Lúc này ta đã hình thành một kiểu dữ liệu mới có là kiểu hoc_sinh. Ta có thể khai báo các biến thuộc kiểu này.

```
hoc_sinh x, y;          // Hai biến x, y có kiểu hoc_sinh
hoc_sinh ds[10];        // Mảng ds gồm 10 phần tử kiểu hoc_sinh
```

Lưu ý: **struct** phải viết bằng chữ bình thường.

2. Truy cập đến các trường của kiểu struct

Cú pháp : `<tên biến kiểu struct>.<tên trường>` ;

Chẳng hạn muốn gán họ tên cho học sinh x là : « Nguyen Van A » ta thực hiện lệnh sau :

```
x.hoten = « Nguyen Van A » ;
```

3. Một số bài tập cơ bản:

Bài 1: Viết chương trình nhập tên, điểm toán, điểm văn của một học sinh. Xuất ra màn hình điểm tổng môn toán và môn văn của học sinh này?

```
#include<iostream>
using namespace std;
struct hoc_sinh
{                                  } Khai báo cấu trúc struct có tên là hoc_sinh
      string hoten ;
      float toan,van,tb;
};
hoc_sinh x;             // khai báo biến x có kiểu dữ liệu là hoc_sinh
```

```
int main()
{
      cout << "nhap ten hoc sinh";
      cin  >> x.hoten;
      cout << "nhap diem toan, van";
      cin  >> x.toan >> x.van;
      x.tb = (x.toan + x.van)/2;
      cout << " Diem TB  = " << x.Tong <<" " << endl;
      return 0;
}
```

BÀI TẬP:

Bài 1. Viết chương trình nhập tên, lương cơ bản một ngày, số ngày làm trong một tháng của n nhân viên. Tính lương của mỗi nhân viên trong tháng đó. Xuất ra màn hình tên và tiền lương của mỗi nhân viên?

Bài 2. Viết chương trình nhập tên, điểm toán, điểm văn của một lớp gồm n học sinh. Tính điểm trung bình 2 môn này. Xuất ra màn hình tên các học sinh có điểm trung bình lớn hơn 5?

Bài 3. Để chuẩn bị cho Hội khỏe Phù Đổng, nhà trường tổ chức cuộc thi điền kinh chạy 100m, mọi học sinh đều có quyền đăng ký thi. Mỗi người dự thi có một phiếu đăng ký:

- Họ và tên (không quá 35 kí tự)
- Nam/Nữ (1 kí tự : 1 – nam; 0 – nữ)
- Lớp (3 kí tự)
- Thời gian chạy (Số thực)

Sau khi chạy tới đích, thời gian chạy sẽ được ghi vào phiếu đăng kí và các phiếu này được nhập vào máy tính để xử lý. Các lớp có từ 3 học sinh nam (nữ) tham dự trở lên được tính thành tích đồng đội nam (nữ). Thành tích thời gian đồng đội là trung bình thời gian của các học sinh trong lớp.

a. Hãy xây dựng kiểu dữ liệu và khai báo các biến thích hợp.
b. Viết đoạn chương trình tính:

- Tổng số học sinh dự thi, số học sinh nam, số học sinh nữ.
- Danh sách các học sinh nam về nhất, bao gồm *Họ và tên, Lớp*.
- Danh sách các học sinh nữ về nhất, bao gồm *Họ và tên, Lớp*.
- Danh sách các lớp có từ 3 học sinh nam tham dự trở lên (để tính điểm đồng đội).
- Dnah sách các lớp có từ 3 học sinh nữ tham dự trở lên (để tính điểm đồng đội).
- Danh sách lớp đứng thứ nhất đồng đội nam.
- Danh sách lớp đứng thứ nhất đồng đội nữ.

§8. CHƯƠNG TRÌNH CON - HÀM

Một chương trình là một dãy các hàm, trong đó có một hàm chính (hàm main()). Hàm chia các bài toán lớn thành các công việc nhỏ hơn, giúp thực hiện những công việc lặp lại nào đó một cách nhanh chóng mà không phải viết lại đoạn chương trình.

1. Khai báo và định nghĩa hàm

* **Cú pháp:**

```
<Kiểu trả về> <tên hàm> ([Khai báo các tham số hình thức])
      {
      [Khai báo các biến cục bộ]
      [Các câu lệnh]
      [return [biểu thức]];
      }
```

* **Lưu ý:**
  - Hàm có thể có giá trị trả về hoặc không, giá trị trả về phải cùng kiểu với kiểu trả về đã khai báo hàm.
  - Nếu hàm không có giá trị trả về thì đặt từ khóa **void** trước tên hàm.

2. Tham số hình thức, tham số thực, tham chiếu, biến cục bộ, biến toàn cục:

* Tham số hình thức: là các tham số dùng khi khai báo hàm
* Tham số thực: là các tham số được truyền cho hàm khi gọi hàm
* Tham trị: là các tham số được truyền cho hàm khi gọi hàm và có giá trị không thay đổi.
  - Tham biến: là các tham số được cung cấp cho hàm khi gọi hàm và sẽ thay đổi theo tham số biến hình thức tương ứng.
  - Biến cục bộ: là biến chỉ có phạm vi hoạt động trong nội bộ hàm, được khai báo bên trong hàm.
* Biến toàn cục: là biến được khai báo trong chương trình chính. Các biến này có thể được dùng ở mọi nơi trong chương trình.

3. Một số ví dụ:

Ví dụ 1: Cho 2 số a và b. Viết chương trình xuất ra màn hình kết quả tổng của a và b.

```
#include<iostream>
using namespace std;
int a,b;          //khai báo biến toàn cục
void nhap()       //hàm nhập không có giá trị trả về
{
cin >> a >> b;
}
int tong (int x, int y) //hàm tính tổng có giá trị trả về, x và y là tham số
                        hình thức
{
int t; //khai báo biến cục bộ
t=x+y;
```

```
return t;
}
int main()
{
 nhap(); //gọi hàm nhập
 cout <<tong(a,b); // gọi hàm tính tổng, a và b là tham số thực
 return 0;
}
```

Ví dụ 2: Cho chương trình sau:

```
#include<iostream>
using namespace std;
int a,b;
int tinh(int x, int y) //x và y là tham trị
{
x=x-1;
y=y-1;
}
int main()
{
 a=3; b=4;
 tinh(a,b);
}
```

Kết thúc chương trình: a=3; b=4

Ví dụ 3: Cho chương trình sau:

```
#include<iostream>
using namespace std;
int a,b;
int tinh(int x, int & y) // x là tham trị, y là tham biến
{
x=x-1;
y=y-1;
}
int main()
{
 a=3; b=4;
 tinh(a,b);
}
```

Kết thúc chương trình: a=3, b=3

BÀI TẬP:

Bài 1. Viết chương trình nhập 2 số a và b. Xuất ra màn hình hiệu của a và b.

Bài 2. Viết chương trình nhập chiều rộng và chiều dài của hình chữ nhật. Xuất ra màn hình chu vi và diện tích của hình chữ nhật

Bài 3. Viết chương trình nhập vào bán kính của hình tròn. Xuất ra màn hình chu vi và diện tích của hình tròn.

Bài 4. Viết chương trình nhập vào dãy số gồm n số nguyên. Xuất ra màn hình tổng các số nguyên này.

Bài 5. Viết chương trình nhập vào dãy số gồm n số nguyên. Xuất ra màn hình giá trị lớn nhất của dãy số này

Bài 6. Viết chương trình nhập vào dãy số gồm n số nguyên. Xuất ra màn hình dãy số đã sắp xếp theo thứ tự tăng dần.

Bài 7. Hãy mô tả hàm FACT(n) trả về giá trị n! (n giai thừa, n nguyên):

Sử dụng hàm này tính và đưa ra màn hình n! với n = 0, 1, ..., 10.

Bài 8. Hãy mô tả hàm SUMDIGIT(n) trả về giá trị là tổng các chữ số của n, trong đó n là số nguyên không âm. Lập trình lần lượt nhập từ bàn phím các số nguyên dương không vượt quá 10^9, với mỗi số, dùng hàm đã mô tả ở trên, đưa tổng các chữ số của nó ra màn hình. Chương trình kết thúc khi nhập vào là 0 hoặc số âm.

§9. CHUỖI KÍ TỰ

1. Chuỗi kí tự trong C:

Ngôn ngữ C sử dụng mảng các phần tử có kiểu ***char*** để lưu chuỗi ký tự và qui ước ký hiệu kết thúc chuỗi là '\0' (ký tự có mã ASCII là 0). Một mảng ký tự gồm n phần tử lưu được một chuỗi tối đa n - 1 ký tự, đánh chỉ số từ 0 đến tối đa là n - 2, nếu dùng hết n - 1 phần tử của mảng thì phần tử thứ n có chỉ số n – 1 (phần tử cuối mảng) phải chứa ký tự '\0'.

Ví dụ nếu ta khai báo:

```
char str[10]="tab";
```

Thì ta sẽ có `str[0] = 't', str[1] = 'a', str[2] = 'b', str[3] = '\0'`, chuỗi `str` chứa được tối đa `10 – 1 = 9` ký tự. Các chỉ số của mảng `str` có thể lưu ký tự là `0, 1, ..., 8`; ký tự `'\0'` được đặt ngay sau ký tự cuối cùng. Nếu ta gán `str[0] = '\0'` (hay `str[0] = 0`) thì chuỗi str xem như rỗng, tức là không có ký tự nào. Khi đó, những ký tự chứa trong mảng nếu có xem là "*rác*" và không sử dụng.

Trong chuỗi hằng thì ký tự '\' được hiểu đặc biệt, chẳng hạn '\n' là ký tự xuống dòng. Để viết ký tự '\' thì ta viết hai lần ký tự này. Ví dụ, ta phải viết `"d:\\windows\\setup"` cho đường dẫn thư mục. Có thể khai báo biến để lưu giữ một chuỗi hằng dưới dạng:

```
char* name = "Galois";
```

Trong trường hợp này, chương trình dịch sẽ cấp ít nhất 7 byte cho biến name để lưu giữ chuỗi "Galois" (6 byte) và ký tự kết thúc chuỗi (1 byte).

Lưu ý: *Không thể gán giá trị, cũng không thể sử dụng các phép toán như: + (ghép chuỗi), > (lớn hơn), < (bé hơn). Thay vào đó là dùng các hàm thư viện trong* ***<string.h>.***

2. Hàm thao tác trên chuỗi:

Các thao tác trên chuỗi được thực hiện nhờ các hàm chuỗi ký tự của thư viện các hàm khai báo trong <***string.h***>. Những hàm này được mô tả như sau:

```
int strlen(const char  *s);
```

Ví dụ nếu khai báo:

```
char str[20] = "Teaching";
```

thì gọi ***strlen(str)*** sẽ trả về 8.

```
char *strcat(char *dest, const char  *src);
```

Ghép chuỗi src vào chuỗi dest và trả về con trỏ đến chuỗi dest. Chuỗi dest phải có đủ chỗ (kể cả ký tự kết thúc chuỗi) để ghép thêm scr vào.

Với chuỗi str như trên (tối đa 19 ký tự và 1 ký tự kết thúc chuỗi, đã dùng 8 ký tự) có thể ghép thêm tối đa 11 ký tự. Ví dụ khi gọi hàm ***strcat(str, " math.")*** sẽ làm cho chuỗi str có nội dung là: "Teaching math."

```
char *strcpy(char *dest, const char  *src);
```

Chép nội dung chuỗi src chồng lên nội dung chuỗi dest và trả về con trỏ đến chuỗi dest. Chuỗi dest phải có đủ chỗ (kể cả ký tự kết thúc chuỗi) để chép src vào.

Ví dụ với str như trên có thể chép ***strcpy(str, "Hello!")*** sẽ làm cho chuỗi str có nội dung là: "Hello!"

```
int strcmp(const char *s1, const char  *s2);
```

So sánh theo thứ tự alphabet. Trả về giá trị âm nếu s1 nhỏ hơn s2; trả về 0 nếu chuỗi s1 khớp với chuỗi s2; trả về giá trị dương nếu chuỗi s1 lớn hơn s2.

Ví dụ gọi ***strcmp("Lang", "Lanh")*** sẽ trả về giá trị âm vì "Lanh" đứng sau "Lang" trong các danh sách sắp thứ tự (g đứng trước h trong bảng chữ cái).

```
char *strchr(const char *s, int c);
```

Tìm xem ký tự c có nằm trong chuỗi s hay không. Nếu có, hàm trả về con trỏ đến vị trí đầu tiên của chuỗi s mà chứa ký tự c, nếu không hàm trả về NULL.
`char *strstr(const char *s1, const char  *s2);` Xác định xem chuỗi s2 có là chuỗi con của chuỗi s1 hay không. Nếu có, hàm trả về con trỏ đến vị trí đầu tiên của chuỗi s1 mà chứa nội dung chuỗi ký tự s2, nếu không hàm trả về NULL.
`char *strlwr(char *s);` Đổi tất cả ký tự của chuỗi s thành chữ thường và trả về s.
`char *strupr(char  *s);` Đổi tất cả ký tự của chuỗi s thành chữ hoa và trả về s.

3. Kiểu string của ngôn ngữ C++:

Thư viện chuẩn STL của C++ có hỗ trợ kiểu ***string*** cùng với các phép toán và phương thức khá tiện lợi cho người lập trình:

- Phép cộng (+) dùng để ghép hai chuỗi và cũng để ghép một ký tự vào chuỗi.
- Các phép so sánh theo thứ tự từ điển: ==(bằng nhau), != (khác nhau, > (lớn hơn), >= (lớn hơn hay bằng), < (nhỏ hơn), <= (nhỏ hơn hay bằng).
- Phương thức length() và phép lấy chỉ số [] để duyệt từng ký tự của chuỗi: nếu s là biến kiểu string thì s[i] là ký tự thứ i của s với $0 \le i \le s.length()$.
- Phép gán (=) dùng để gán biến kiểu string bằng một hằng chuỗi hay là để sao chép một chuỗi sang chuỗi khác.

Ví dụ 1: Chương trình dưới đây nhập tên của một người và đưa ra lời chào.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string name ;
   cout << "Nhap ten cua ban :";
   cin>>name;
   cout<<"Chao, "<< name <<"\n";
   return 0;
}
```

Ví dụ 2: Chương trình dưới đây nhập hai xâu từ bàn phím và kiểm tra kí tự đầu tiên của xâu thứ nhất có trùng với ký tự cuối cùng của xâu thứ hai không.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string a,b;
   cout << "Nhap xau thu nhat:"; cin>>a ;
   cout << "Nhap xau thu hai:"; cin>>b ;
```

```
    if(a[0]==b[b.length()-1])
        cout<<"Trung nhau" ;
    else
        cout<<"Khac nhau" ;
    return 0 ;
}
```

Ví dụ 3: Chương trình dưới đây nhập một xâu vào từ bàn phím và đưa ra màn hình xâu đó nhưng được viết theo thứ tự ngược lại.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string a;
    cout << "Nhap xau thu nhat:"; cin>>a ;
    for(int i=a.length()-1; i>=0 ;--i)
        cout<<a[i];
    return 0;
}
```

BÀI TẬP:

Bài 1. Nhập vào từ bàn phím một xâu. Kiểm tra xâu đó có phải là xâu đối xứng hay không. Xâu đối xứng có tính chất: đọc nó từ phải sang trái cũng thu được kết quả giống như đọc từ trái sang phải (còn được gọi là xâu palindrome).

Bài 2. Viết chương trình nhập từ bàn phím một xâu kí tự S và thông báo ra màn hình số lần xuất hiện của mỗi chữ cái tiếng Anh trong S (không phân biệt chữ hoa hay chữ thường).

Bài 3. Nhập vào từ bàn phím một xâu. Thay thế tất cả các cụm kí tự 'anh' bằng cụm kí tự 'em'.

Bài 4. Xét xâu S chỉ bào gồm các kí tự *ngoặc mở '('* và *ngoặc đóng ')'*. Xâu S xác định một cách đặt ngoặc đúng, nếu thỏa mãn các điều kiện:

- Số ngoặc mở bằng số ngoặc đóng.
- Nếu duyệt từ trái sang phải, số lượng ngoặc mở luôn luôn lớn hơn hoặc ằng số lượng ngoặc đóng.

Ví dụ: xâu '(((()(())))' xác định một cách đặt ngoặc đúng. Còn xâu '(()()))(())' là một cách đặt ngoặc sai.
Hãy viết đoạn chương trình kiểm tra xem xâu S có xác định một cách đặt ngoặc đúng hay không?

§10. KIỂU TỆP (FILE)

1. Khái niệm: Tệp là tập hợp các dữ liệu được tổ chức theo một cách thức nhất định và được lưu trữ trên bộ nhớ ngoài. Có hai dạng tệp: tệp văn bản và tệp có cấu trúc. Trong chương trình ta chỉ đề cập đến tệp văn bản (TEXT)

2. Tệp văn bản: Là tập hợp các kí tự thuộc bảng mã **ASCII** được tổ chức thành từng dòng. Kí tự kết thúc dòng là **'\n'** . Việc đọc/ ghi dữ liệu trên tệp văn bản thường theo từng dòng.

3. Luồng dữ liệu:

Trong **C++** ta đã biết đến cách nhập/ xuất dữ liệu bằng lệnh **cin** , **cout**. Các lệnh này giao tiếp với chương trình thông qua bàn phím và màn hình.
Để nhập/ xuất dữ liệu thông qua tệp văn bản ta cần sử dụng các luồng vào/ ra được xây dựng trong **header fstream**. Khi đó ta có các luồng sau đây:

- **ifstream**: Luồng vào, để đọc dữ liệu từ tệp.
- **ofstream**: Luồng ra, để ghi dữ liệu lên tệp.

4. Ví dụ về cách đọc, ghi dữ liệu bằng tệp

Ví dụ 1: Giả sử ta tạo sẵn một tệp **input.txt** trên đĩa D có cấu trúc như sau:

Cấu trúc	Ví dụ
Dòng 1 là một chuỗi kí tự	**Khanh**
Dòng 2 là một số nguyên n không quá 100	**6**
Dòng 3 là n số thực a_1 , a_2 , a_3 …, a_n	**3.2 6.1 5.3 7.5 8.8**
Dòng 4 là một kí tự	**T**

Yêu cầu: Hãy tạo một chuỗi S để lưu dòng 1, một số nguyên n để lưu trữ dòng 2, một mảng để lưu dòng 3 và một biến kiểu char để lưu dòng 4.
Ghi lên tệp "**output.txt**" một số là trung bình cộng của **n** số **a1 , a2 , a3 …, an**

Bước 1: Khai báo

```
#include <iostream>
#include <fstream>     //Khai báo header fstream để sử dụng nhập/xuất trên file
using namespace std;
string s;              //Biến s kiểu string để chứa chuỗi ở dòng 1
int n;
float a[101];          //Mảng a chứa được 101 phần tử. Ta khai báo 101 vì
char c;                mảng a bắt đầu từ a[0]
```

Bước 2: Đọc dữ liệu từ file `input.txt`

```
ifstream fi("input.txt");
fi>>s;
fi>>n;
for (int i=1;i<=n;++i) fi>>a[i];
fi>>c;
```

Như vậy thay vì nhập từ luồng chuẩn(từ bàn phím) với lệnh **cin** thì ta chỉ cần đổi hướng sang luồng **fi** (là một luồng kiểu file văn bản). Đến đây ta có thể xuất dữ liệu ra màn hình theo luồng chuẩn với lệnh **cout** hoặc xuất ra file văn bản theo luồng file văn bản.

Bước 3: Xuất dữ liệu

Xuất ra màn hình	Xuất ra file "output.txt

`cout <<s<<'\n';`
`for (int i=1;i<=n;++i)cout<<a[i]<<" ";`
	`//Khai báo luồng ra là file output.txt`
	`fstream fo("output.out");`
	`fo <<s<<'\n';`
	`for (int i=1;i<=n;++i) fo<<a[i]<<" ";`

Kết quả: **Khanh** **3.2 6.1 5.3 7.5 8.8**	**Kết quả:** File **output.txt** **Khanh** **3.2 6.1 5.3 7.5 8.8**

Một cách đơn giản hơn là ta định hướng lại luồng chuẩn thành luồng file bởi lệnh **freopen()** . Khi đó các lệnh **cin, cout** được chuyển hướng trở thành các lệnh nhập/ xuất ra file văn bản chứ không theo luồng chuẩn nữa.

```
int main()
{
    freopen("input.txt","r",stdin);    //Thay luồng nhập chuẩn thành luồng file
    freopen("output.txt","w",stdout); //Thay luồng xuất chuẩn thành luồng file
    cin>>s;
    cin>>n;
    for (int i=1;i<=n;++i) cin>>a[i];
    cin>>c;
//-----------------------------------------
    cout << s << '\n';
    for (int i=1;i<=n;++i) cout<<a[i]<<" ";
    return 0;
}
```

Ví dụ 2: ***Đọc tệp văn bản theo từng dữ liệu rời rạc liên tiếp.***

Giả sử ta tạo sẵn một tệp input.txt trên đĩa D có cấu trúc như sau:

Cấu trúc	Ví dụ
Dòng 1 ghi 2 số nguyên kiểu int Dòng 2 ghi hai số nguyên kiểu long Dòng 3 ghi 3 số thực kiểu double	1 5 3291 -4327 -3.4 2.1 8.98

Yêu cầu: Hãy tạo 2 số nguyên kiểu int để lưu trữ dòng 1, hai biến kiểu long để lưu dòng 2 và 3 biến double để lưu dòng 3. Ghi lên file "output.txt" tổng của 4 số nguyên ở dòng thứ nhất và tổng của hai số thực lên dòng thứ 2.

5. Một số thao tác đặc biệt với kiểu tệp

Lệnh **ofstream fo("tên tệp", ios::app);** Mở tệp ở chế độ ghi nối dữ liệu vào cuối tệp

Lệnh **`ofstream fo("tên tệp", ios::noreplace);`** Mở tệp ở chế độ bảo vệ. Nếu tệp đã có thì tệp cũ không bị xóa. Chế độ mở kiểu này thường đi kèm với việc sử dụng **`(!fo)`**

Ví dụ

```
ofstream fo("D:\\input.txt", ios::noreplace);
if (!fo)
      {
      cout<<"Tep da ton tai, khong the ghi de len duoc";
      return;
      }
```

Lệnh **`ofstream fo("tên tệp", ios::nocreate);`** Kiểm tra tệp đã tồn tại chưa. Nếu chưa tồn tại, biến fo sẽ nhận giá trị false và tệp đã sẵn sàng để ghi dữ liệu được.

Ví dụ

```
ofstream fo("D:\\input.txt", ios::nocreate);
if (!fo)
{
cout<<"Tep chua ton tai"; //Có thể ghi dữ liệu vào tệp D:\\input.txt".
…
}
```

7. Truy cập ngẫu nhiên

Giả sử có biến tệp f. Ta có một số phương thức thường dùng:

Truy cập vào đầu tệp	f.seekg(0)	cin.seekg(0)
Truy cập tới vị trí pos	f.seekg(pos)	cin.seekg(pos)
Kiểm tra kết thúc tệp	eof(f)	cin.eof

BÀI TẬP ÔN TẬP

Bài 1: Viết chương trình in ra màn hình dãy số **`3 6 9 12 15 18 21`**

Bài 2: Nhập $n \geq 1$, tính các tổng :

$$S_1 = 1 + \frac{1}{2^2} + \frac{1}{3^2} + ... + \frac{1}{n^2}$$

$$S_2 = \frac{1+3+5+...+(2n+1)}{2+4+6+...+2n}$$

$$S_3 = 1 + \frac{x}{1!} + \frac{x^2}{2!} + ... + \frac{x^n}{n!}$$, x là số thực, nhập từ bàn phím.

$$S_4 = \underbrace{\sqrt{2+\sqrt{2+...\sqrt{2+\sqrt{2}}}}}_{n \text{ dấu căn bậc hai}}$$

Bài 3: Nhập $n \geq 1$, từ đó nhập vào **`n`** số nguyên.

a) Tính tổng và tính trung bình cộng n số nguyên đã nhập.

b) Tính tổng các số chẵn trong n số đã nhập.

c) Đếm xem có bao nhiêu số lẻ trong n số đã nhập.

d) In số lớn nhất trong n số đã nhập ra màn hình.

Bài 4: Dãy Fibonacci

Dãy Fibonacci được cho bởi công thức $\begin{cases} F_0 = F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \end{cases}$

Viết chương trình tính số hạng thứ n của dãy Fibonacci, $n \geq 1$ là số nguyên dương được nhập từ bàn phím.

Bài 5: Năm âm lịch tính theo Can và Chi. Có 10 can là: Giáp, Ất, Bính, Đinh, Mậu, Kỷ, Canh, Tân, Nhâm, Quý. Có 12 Chi là Tý, Sửu, Dần, Mão, Thìn, Tỵ, Ngọ, Mùi, Thân, Dậu, Tuất, Hợi. Hãy nhập năm dương lịch, tính năm Âm lịch tương ứng biết rằng năm 1999 là năm Kỷ Mão.

Bài 6: Nhập từ bàn phím một số nguyên **N**. Hiện trên màn hình các chữ số của N theo thứ tự ngược lại: Chữ số hàng đơn vị, hàng chục,... Ví dụ, nhập **N = 123** thì kết quả thu được trên màn hình là: **3 2 1**

Bài 7. Nhập *n* số nguyên (*n*≤20). Tính tổng các số chẵn, tổng các số lẻ trong *n* số đã nhập.

Bài 8. Viết chương trình nhập vào một số nguyên *n* và *n* số a_1 ; a_2 ; …, a_n . Xuất ra màn hình dãy (a_n) đã được sắp xếp tăng dần.

Bài 9: Cho một số nguyên N (1≤N≤100) chứa trong file văn bản **nhiphan.inp**. Hãy viết chương trình chuyển số N sang dạng nhị phân. Kết quả chứa trong file văn bản **nhiphan.out**.

Bài 10: Cho một file **Danhsach.txt** chứa dữ liệu về học sinh một lớp có cấu trúc như sau:

- Dòng thứ nhất ghi một số nguyên *n* là số học sinh của lớp
- Trong *n* dòng kế tiếp, mỗi dòng ghi các thông tin của *n* học sinh trong lớp gồm: Họ và tên, điểm trung bình và hạnh kiểm. Giữa các thông tin cách nhau bởi dấu ";"

Hãy viết chương trình xuất ra file văn bản các học sinh có điểm trung bình trên 8.0 và hạnh kiểm loại "Tốt"

Ví dụ:

Dansach.txt	**Danhsach.out**
Nguyen Van A ; 5.6 ; Kha **Tran Quoc B ; 8.7 ; Tot** **Van Thanh C ; 8.9 ; Kha** **Tran My Kieu ; 8.0 ; Tot** **Le Ngoc Hoa ; 7.7 ; Tot**	**Tran Quoc B** **Tran My Kieu**

PHỤ LỤC

§ AI LÀ LẬP TRÌNH VIÊN ĐẦU TIÊN?

Ada Augusta Byron Lovelace

Đó là một phụ nữ, bà Ada Augusta Byron Lovelace, con gái của nhà thơ nổi tiếng thời đó Lord Byron. Ada là một trong những nhân vật ấn tượng nhất trong lịch sử Tin học. Bà sinh ngày 10/12/1815 và là người cùng thời với Charles Babbage, người đầu tiên đưa ra đề án thiết kế chiếc máy tính điều khiển theo chương trình có tên là Analytical Engine (máy giải tích).

Từ nhỏ, bà đã nổi tiếng là một người thông minh, có khả năng đặc biệt về toán học.

Ngay từ khi thiết kế máy giải tích còn ở trên giấy, Ada đã đề xuất với Babbage một kế hoạch chi tiết để máy giải tích tính các số Bernouilli. Ngày nay người ta coi kế hoạch này là *chương trình máy tính đầu tiên* và bà được gọi là *lập trình viên đầu tiên*.

Các ghi chép được công bố của Ada cho tới nay vẫn đặc biệt có ý nghĩa đối với các lập trình viên. Giáo sư J. Von Neumann đã viết rằng các quan sát của Ada "chứng tỏ bà đã hiểu được các nguyên tắc lập trình máy tính trước thời đại của mình hàng thế kỉ".

Như một nhà toán học, Ada đánh giá cao khả năng tự động hoá các công việc tính toán nặng nhọc. Nhưng bà quan tâm hơn đến ***các nguyên tắc của việc lập trình*** các thiết bị đó. Ngay khi máy giải tích còn chưa được xây dựng, Ada đã thí nghiệm viết những dãy lệnh. Bà nhận ra giá trị của một vài thủ thuật đặc biệt trong nghệ thuật mới này và điều thú vị là những thủ thuật này hiện giờ vẫn còn là cơ bản đối với các ngôn ngữ lập trình hiện đại, đó chính là ***chương trình con, vòng lặp và các phép chuyển điều khiển***.

Thay cho việc viết các dãy lệnh lặp đi lặp lại nhiều lần, ta có thể viết chúng dưới dạng các ***chương trình con*** để dùng nhiều lần. Các chương trình con ngày nay là thành phần không thể thiếu được của mọi ngôn ngữ lập trình.

Máy giải tích và các máy tính số thực hiện rất tốt các tính toán nhiều lần một cách nhanh chóng. Thời kì đó, các bìa đục lỗ được sử dụng để đưa dữ liệu và các lệnh vào máy. Bằng việc phát minh ra các lệnh thực hiện việc chuyển thiết bị đọc bìa về một bìa xác định trước nó, sao cho dãy các lệnh có thể được thực hiện một số lần nhất định, Ada đã phát minh ra ***vòng lặp*** – một trong những cấu trúc điều khiển quan trọng trong các ngôn ngữ lập trình.

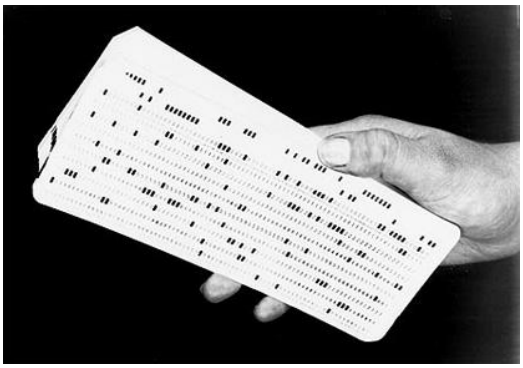
Khả năng lôgic của Ada đã phát huy với ***phép chuyển điều khiển có điều kiện***. Bà nghĩ ra một loại lệnh để thao tác với thiết bị đọc bìa, nhưng thay cho việc quay lại và lặp lại dãy bìa, lệnh này cho phép thiết bị đọc bìa chuyển tới một bìa khác tại bất kì vị trí nào trong dãy, NẾU một điều kiện nào đó được thoả mãn. Việc thêm chữ NẾU đó vào danh sách các lệnh số học thuần tuý trước đây có nghĩa là chương trình có thể làm nhiều hơn là tính toán đơn thuần. Ở dạng thô sơ nhưng về tiềm năng là rất có ý nghĩa, máy giải tích có thể thực hiện các *quyết định*.

Ada mất năm 1852, khi mới qua tuổi 36. Nếu như bà không qua đời sớm như vậy, chắc chắn khoa học lập trình của thế kỉ XIX đã có thể tiến nhanh hơn nhiều.

Để tưởng nhớ công lao của Ada, một ngôn ngữ lập trình do Bộ Quốc phòng Mĩ tạo ra năm 1979 mang tên bà.

§ GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH C++

Lịch sử lập trình đã trải qua một quãng thời gian rất dài trước khi C++ ra đời. Ở thời kì đầu, bàn phím thậm chí còn chưa tồn tại, người ta đã sử dụng những tấm card như hình bên dưới để đưa ra các lệnh cho máy tính. Công việc này đòi hỏi thời gian và sự tỉ mỉ rất cao.



Với sự phát triển của tin học, phần cứng, phần mềm và các ngôn ngữ lập trình đầu tiên cũng ra đời :

1958 : Ngôn ngữ lập trình đầu tiên được phát triển: ALGOL.

1960-1970 : Các nhà khoa học từ Viện Toán của ĐH London và ĐH Cambridge tạo ra một ngôn ngữ mới có tên là CPL, sau đó phát triển thành BCPL, rồi được đổi tên thành ngôn ngữ B

1970 : Trên nền tảng của ngôn ngữ B, hai nhà khoa học người Mỹ là Ken Thompson và Dennis Ritchie phát triển nên một ngôn ngữ lập trình có tên là ngôn ngữ C.

Hình: Dennis Ritchie (trái) và Ken Thompson

1983 : Một thời gian sau, người ta nghĩ đến việc thêm vào C những yếu tố mới để làm nó phát triển hơn. Một lập trình viên người Đan Mạch là Bjarne Stroustrup đã lập nên một ngữ mới trên nền tảng của C là C++. Ngôn ngữ C++ thực chất là C với nhiều yếu tố mới. Các yếu tố mới này gồm những khái niệm tiên tiến như hướng đối tượng, tính đa hình…Bjarne Stroustrup hiện đang giảng dạy về tin học tại đại học Texas A&M ở Mỹ. Ông được xem là một biểu tượng lớn của thế giới tin học.

Rất nhiều ngôn ngữ khác được xây dựng lấy ý tưởng từ C++, chẳng hạn như Java.

C++, mặc dù đã tồn tại trong một thời gian tương đối dài, vẫn đang tiếp tục được cải tiến và hoàn thiện. Một phiên bản mới, có tên gọi C++1x, đang trong gian đoạn xây dựng và phát triển. Nó không phải một ngôn ngữ mới mà là một phiên bản nâng cấp của C++.

Nếu như C++ là một bước cải tiến của C thì tại sao ngày nay vẫn còn rất nhiều người lập trình bằng C ?

Vì không phải ai cũng cần đến những bước cải tiến của C++. Bản thân ngôn ngữ C cũng đủ mạnh để lập nên các hệ điều hành như Linux, Mac OS X và Windows.

§ GIỚI THIỆU IDE CODE:BLOCK

1. Các công cụ cần thiết để lập trình

Để bắt đầu việc lập trình ta cần những công cụ gì?

Thứ nhất đó là ***trình biên dịch*** (compiler) giúp dịch ngôn ngữ C++ sang ngôn ngữ nhị phân. Có rất nhiều chương trình như vậy dành cho C++.

Thứ hai là một ***trình soạn thảo*** để viết mã nguồn của chương trình bằng C++. Về lí thuyết, mọi phần mềm soạn thảo văn bản đơn giản như Notepad, Notepad++ hay phức tạp như MS-Word đều có thể sử dụng được.

Thứ ba là một bộ ***debugger (trình soát lỗi)*** với các công cụ giúp ta tìm lỗi trong chương trình.

Như vậy chúng ta có 2 lựa chọn :

- Một là chúng ta sử dụng riêng từng công cụ. Cách này khá phức tạp, tuy vậy rất nhiều lập trình viên lựa chọn cách làm này, đặc biệt với những lập trình viên làm việc với hệ điều hành Linux.
- Ta không sử dụng cách trên để tiếp cận với ngôn ngữ C++ mà thay vào đó sẽ sử dụng cách thứ hai là dùng một chương trình "3 trong 1". Chương trình này bao gồm cả trình soạn thảo, trình dịch và trình sửa lỗi. Nó có tên gọi là IDE (Intergrated Development Environment) nghĩa là môi trường phát triển tích hợp.

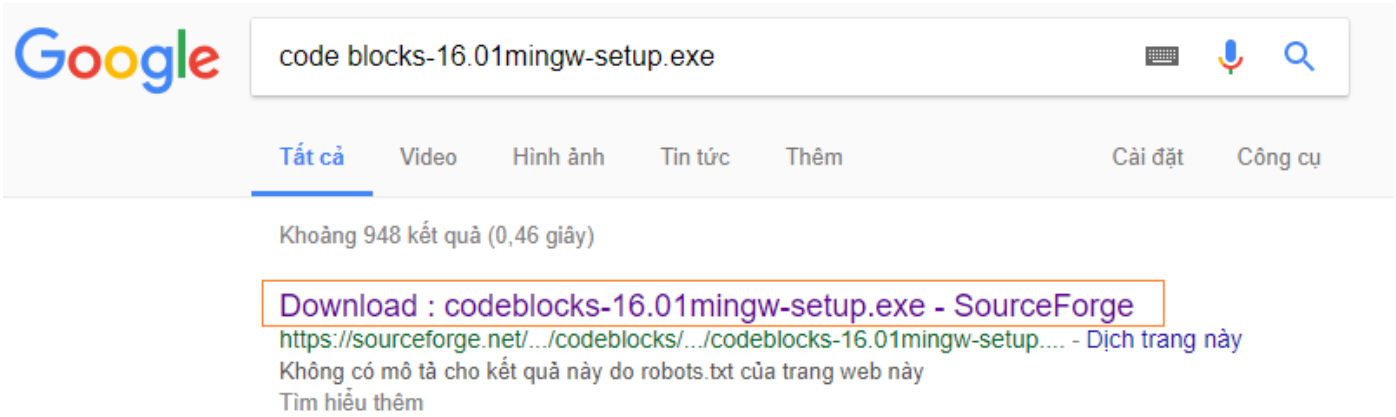
Có rất nhiều IDE dành cho C++ như bộ Visual Studio của Microsoft, DevC, Code:Block (dành cho HĐH Windows) hoặc Eclipse, Netbean, Xcode (dành cho Mac và Linux). Trong tài liệu này, chúng ta sẽ sử dụng IDE Code:Block.

2. Khái niệm Project

Khi xây dựng một chương trình ta cần tạo một project (dự án). Một project được tạo nên bởi nhiều file chẳng hạn file *.cpp, *.layout,*.exe, các file hình ảnh…phục vụ cho chương trình. Code::Block tập hợp tất các các file trong một thư mục và ta có thể truy cập đến chúng thông qua các biểu tượng (icon).

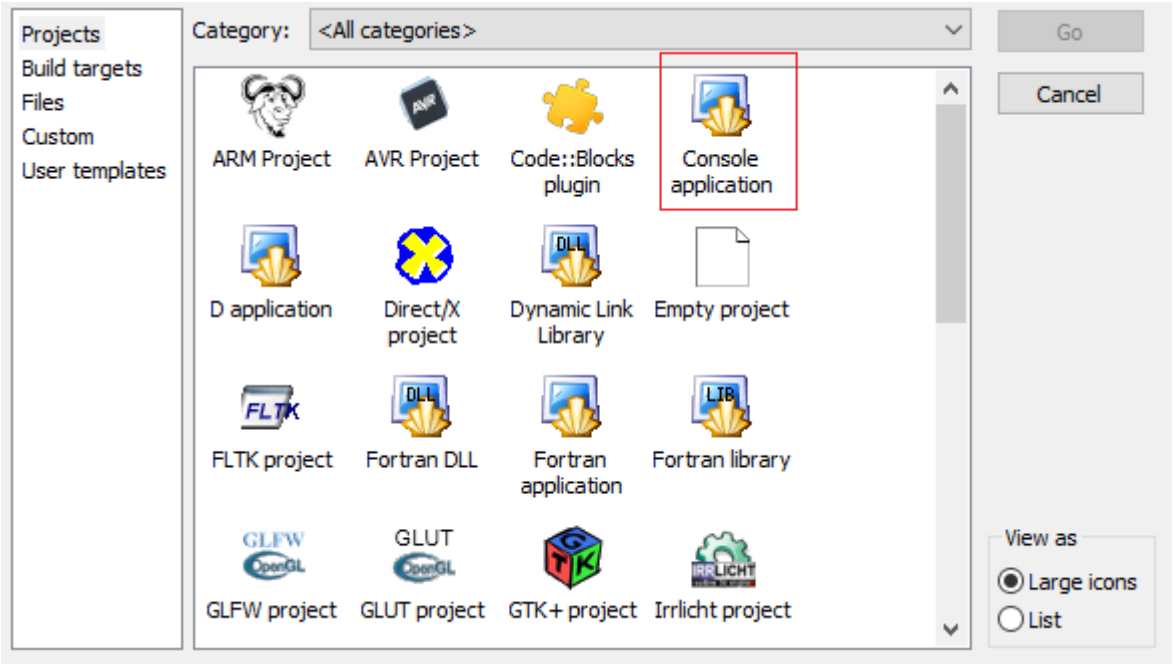
3. Giới thiệu Code::Block

Để tải **Code::Block** ta có thể sử dụng công cụ tìm kiếm Google với từ khóa code blocks mingw.



Nhấn đúp vào file cài đặt **`codeblocks-16.01mingw-setup.exe`** và tiến hành cài đặt như bình thường. Giao diện của Code::Block

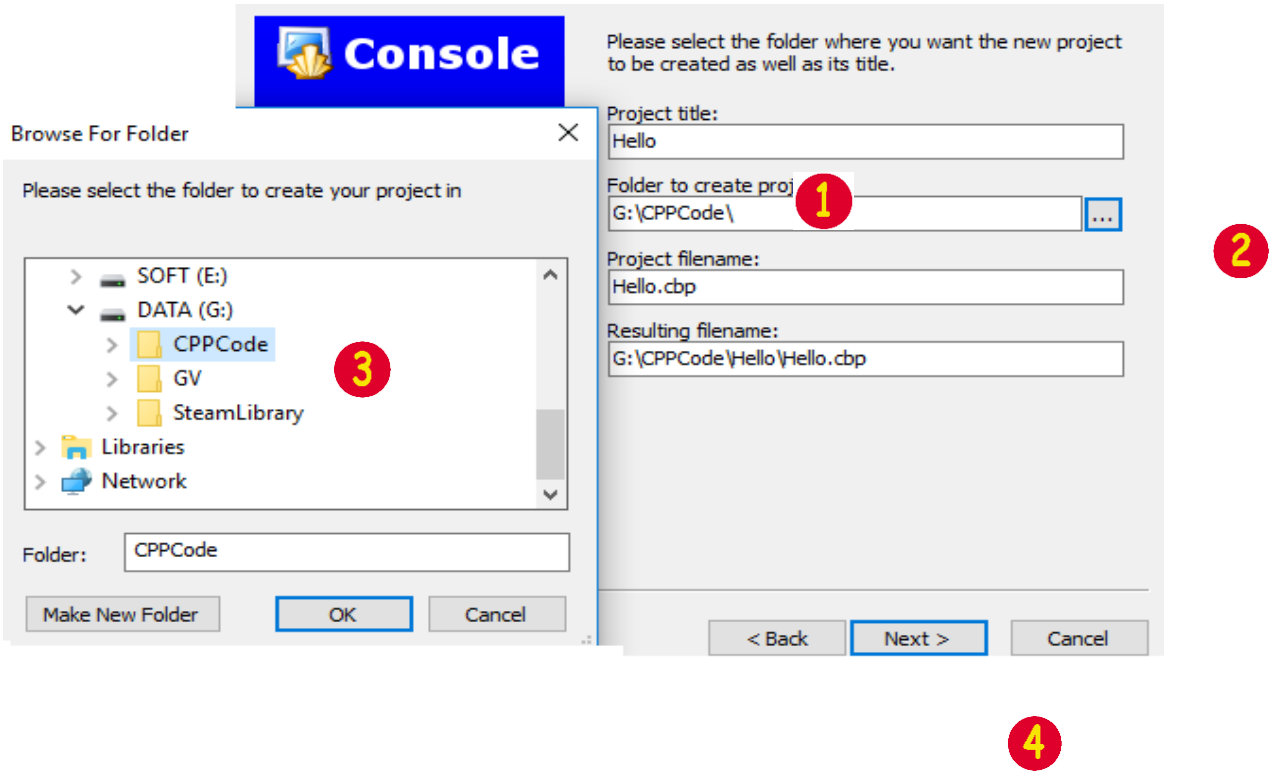
Chọn mục Create a new project



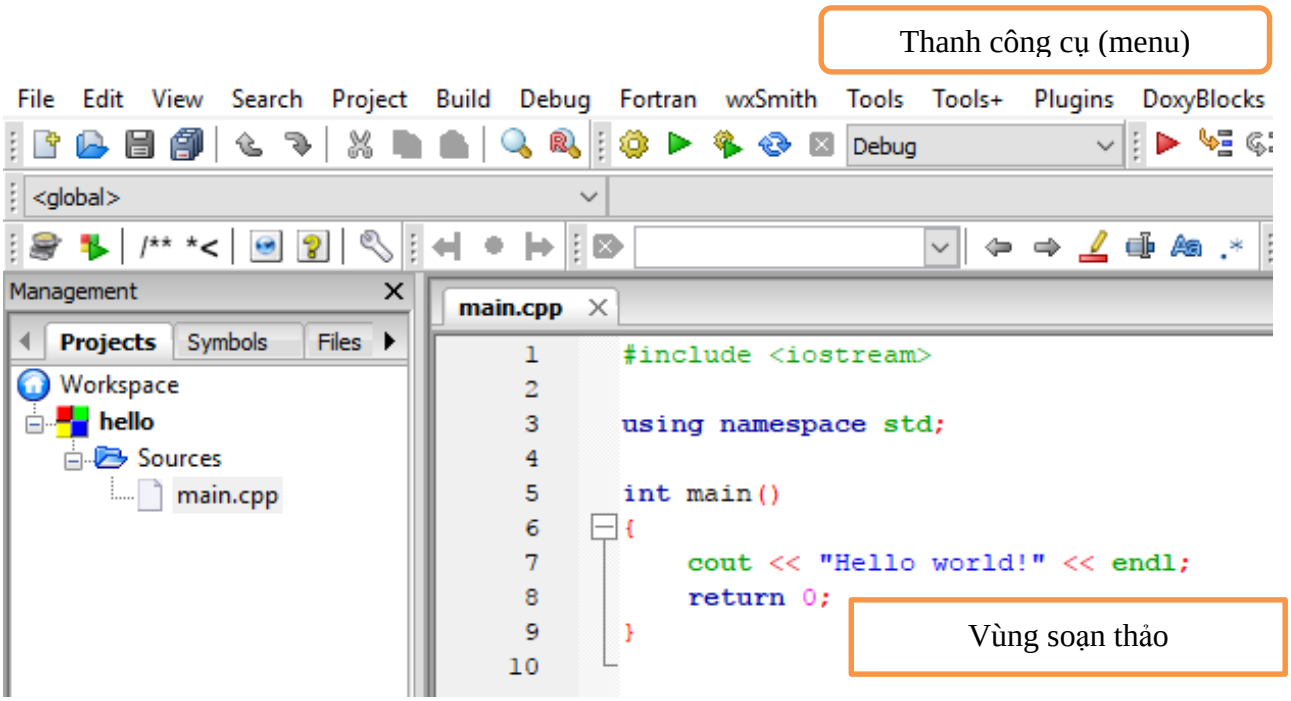
Nhấp đúp vào Console application và chọn C++. Ấn Next

Đặt tên cho Project (1). Ấn chọn nút … (2) và chọn thư mục lưu project vừa tạo. Chúng ta nên tạo một thư mục chẳng hạn CPPCode để lưu tất cả các project sau này như hình, không nên chọn thư mục gốc.

Cuối cùng chọn Next và ấn Finish ở trang kế tiếp. Các mục khác cứ để như mặc định của chương trình.



Ở góc trái, nhấn đúp vào tên project hello, vào mục source ta có giao diện soạn thảo chương trình cho file **`main.cpp`** là file chứa mã thực hiện của chương trình ta đang viết. Các thao tác soạn thảo mã chương trình được thực hiện như mọi chương trình soạn thảo văn bản thông dụng. Ta có thể chọn khối văn bản với chuột, cắt (**`Ctrl-X`**), sao chép (**`Ctrl-C`**), dán (**`Ctrl-V`**)…



Để dịch chương trình ta ấn tổ hợp phím **`Ctrl+F9`** ; để thực thi chương trình ấn **`F9`**
Để thực thi chương trình theo từng bước, ta ấn phím **`F7`** . Để nhảy đến một dòng lệnh ta dùng phím **`F4`**.
Một số chức năng nâng cao khác phục vụ cho việc sửa lỗi nằm ở menu Debug.