CÁC KIỂU DỮ LIỆU CƠ SỞ, BIẾN VÀ HẰNG TRONG NGÔN NGỮ LẬP TRÌNH C/C++

Một số lưu ý:
// <=> comment ghi chú, giải thích
; <=> kết thúc câu lệnh, luôn luôn có nếu không sẽ bị lỗi.

I. CÁC KIỂU DỮ LIỆU CƠ SỞ

- Trong ngôn ngữ lập trình C/C++ gồm có 4 kiểu dữ liệu cơ sở do ngôn ngữ lâp trình mặc định sẵn cho người lập trình sử dụng. Tên kiểu dữ liệu phải được *viết thường*, không được phép viết in hoa.

+ *Kiểu kí tự* : char.
+ *Kiểu số nguyên*: int , short, long ..
+ *Kiểu số thực*: float, double…
+ *Kiểu luận lý*: true(đúng), false(sai).

- Bảng mô tả:

Tên	Độ lớn (byte)	Mô tả	Vùng giá trị
char	1	Kí tự hay kiểu số nguyên 8 – bit 256 kí tự trong bảng mã ASCII.	**Có dấu**: -128 to 127 **Không dấu**: 0 to 255
short	2	Kiểu số nguyên 16 – bit	**Có dấu**: -32763to 32762 **Không dấu**: 0 to 65535
long	4	Kiểu số nguyên 32 – bit	**Có dấu**: -2.147.483.648 to 2.147.483.648 **Không dấu**: 0 to 4.294.967.295
int	*	Số nguyên. Độ lớn của nó phụ thuộc vào hệ điều hành, như trong MS-DOS nó là 16-bit, trên Windows 9x/2000/NT là 32 – bit…	*
float	4	Dạng dấu phẩy động với độ chính xác đơn	3.4e-38 -> 3.4e38
double	8	Dạng dấu phẩy động với độ chính xác kép	1.7e-308 -> 1.7e308
bool	1	Gía trị logic. Nó mới được thêm vào chuẩn ANSI-C++. Bởi vậy không phải tất cả các trình biên dịch đều hỗ trợ nó	true hoặc false

- Ví dụ với kiểu float, giá trị dương lớn nhất là 3.4e38 =3.4*10^38 và số dương nhỏ nhất có thể biểu diễn là 3.4e-38 = 3.4*10^-38.
- Tuy nhiên, do số chữ số trong phần định trị là giới hạn nên số chữ số đáng tin cậy (hay ta nói là số chữ số có nghĩa) cũng giới hạn với kiểu float là 7-8 chữ số, double là 15 chữ số, và long double là 18-19 chữ số.

Lưu ý

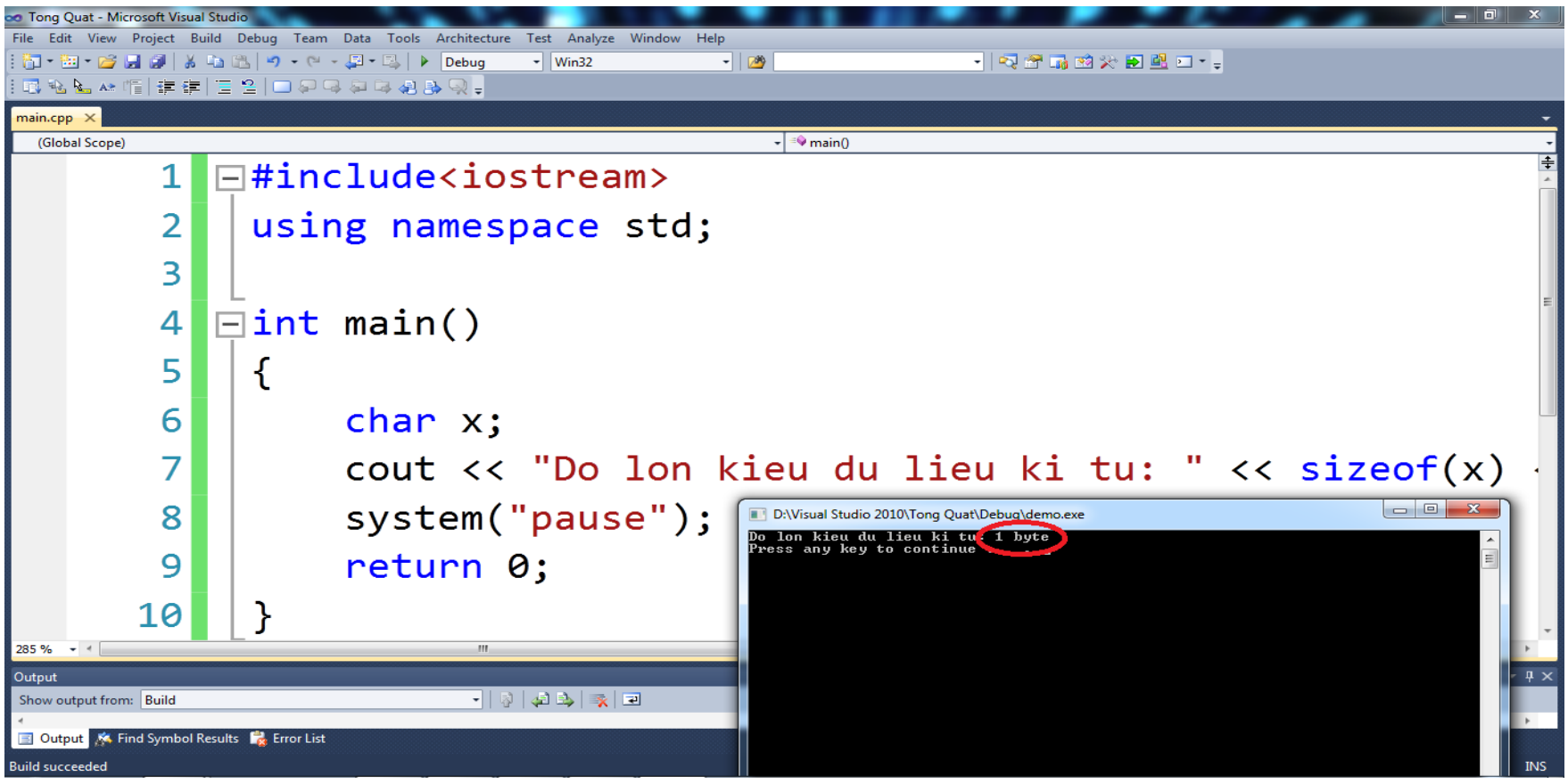
(*) Thông thường thì bạn đã tham khảo qua các loại tài liệu, hay các bài chia sẻ về kích thước (độ lớn) của các kiểu dữ liệu thì vô tình người ta có nhắc đến kiểu *int* – độ lớn của nó là *2 byte*, thì đó có thể là các tài liệu cũ. Tại 1 thời điểm nhất định có thể nó là chuẩn mực được quy ước. Nhưng hiện nay đa số các tài liệu được cập nhật thì kiểu *int* – độ lớn của nó thường là *4 byte*. Hầu hết ở các trình biên dịch hiện đại ngày nay thì các bạn có thể thấy rằng kiểu *int* – được mặc định là *32bit* – tức *4 byte*.

- Có những tranh cãi giữa các lập trình viên đã diễn ra cho vấn đề liệu rằng, kiểu int là *2 byte* hay là *4 byte*. Để đi đến tính thống nhất cho các lập luận của những lập trình viên, các nhà lập trình chuyên nghiệp đã dùng hàm **sizeof()** - để xác định kích thước của 1 kiểu dữ liệu đang sử dụng trên nền tảng và trình biên dịch hiện tại của mỗi lập trình viên. Bởi vì kích thước (độ lớn) của 1 kiểu dữ liệu có thể phụ thuộc vào nền tảng và trình biên dịch compiler mà bạn đang sử dụng. Vì vậy chỉ có thể dùng hàm **sizeof(<kiểu dữ liệu…>)** mới có thể xác định đúng kích thước của 1 kiểu dữ liệu.

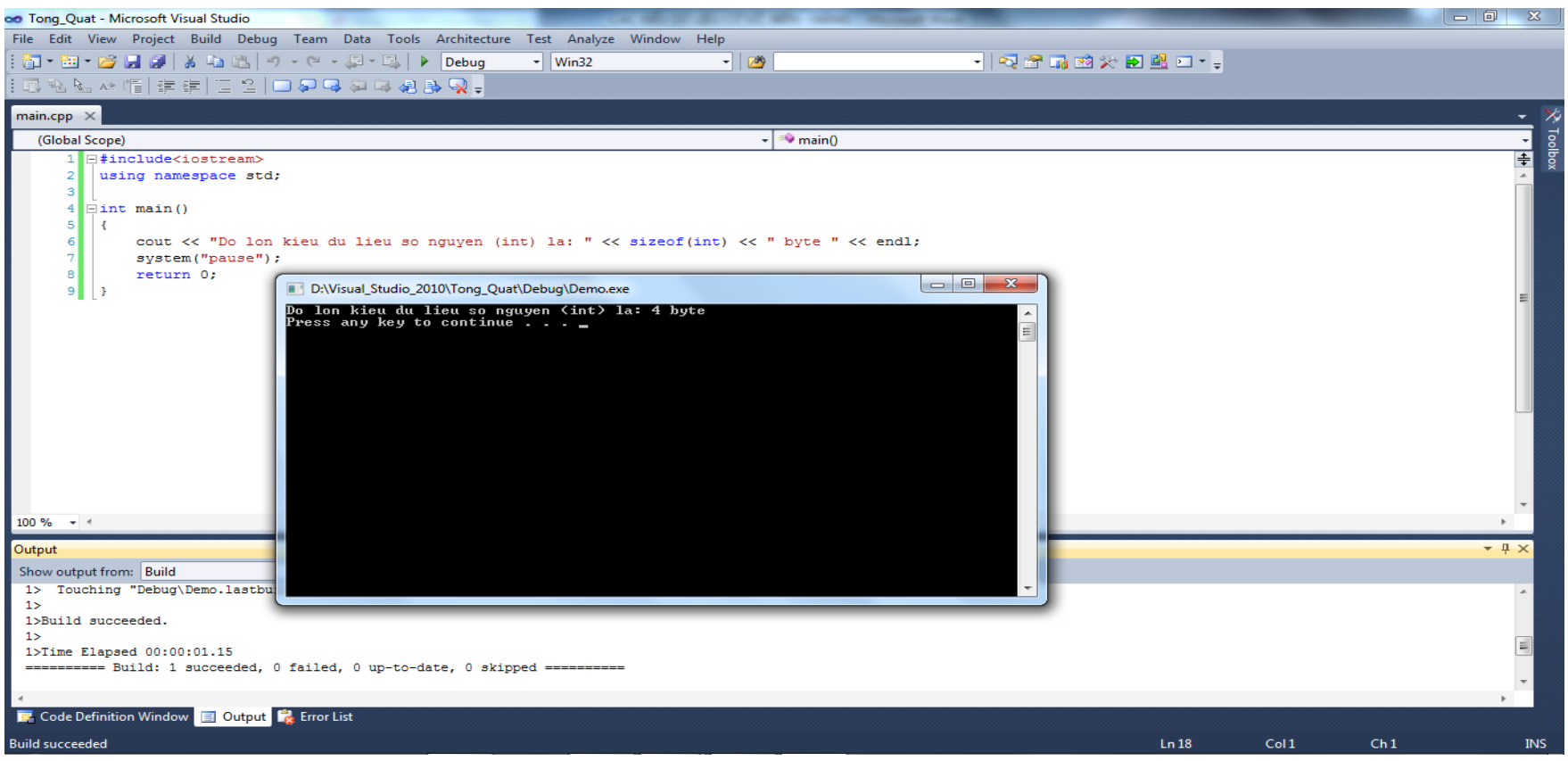
- Nói 1 cách dễ hiểu, **sizeof()** là 1 hàm dùng để xác định kích thước của 1 kiểu dữ liệu trên nền tảng và trình biên dịch mà bạn đang sử dụng. Nó trả về kích thước tính theo *byte* của 1 kiểu dữ liệu cơ sở, con trỏ, mảng, hay 1 kiểu dữ liệu cấu trúc do lập trình viên định nghĩa(khi nào lớn sẽ biết :D).

KIỂM TRA KÍCH THƯỚC KIỂU DỮ LIỆU TRÊN HỆ ĐIỀU HÀNH 64 BIT

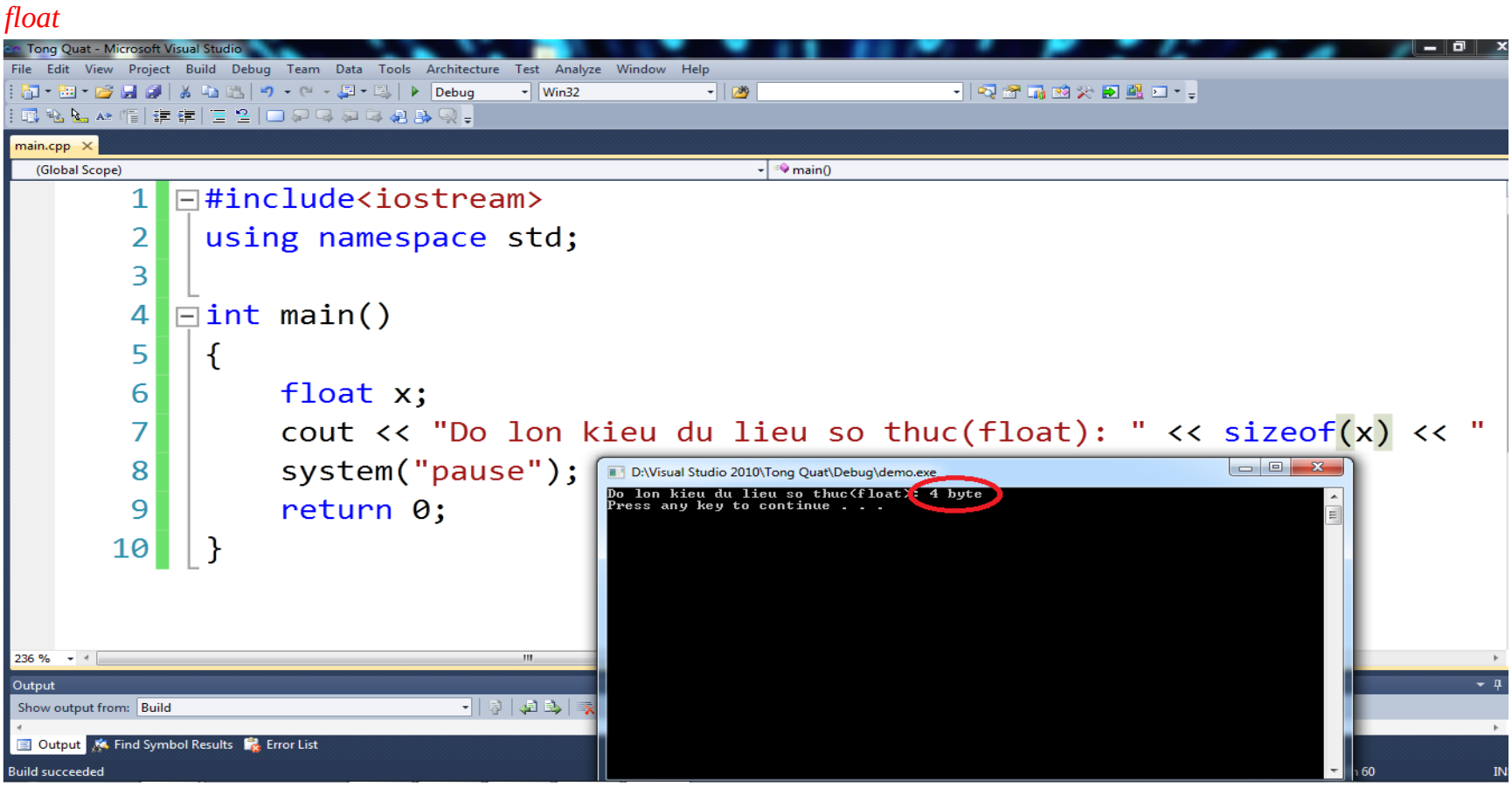
+ *char*



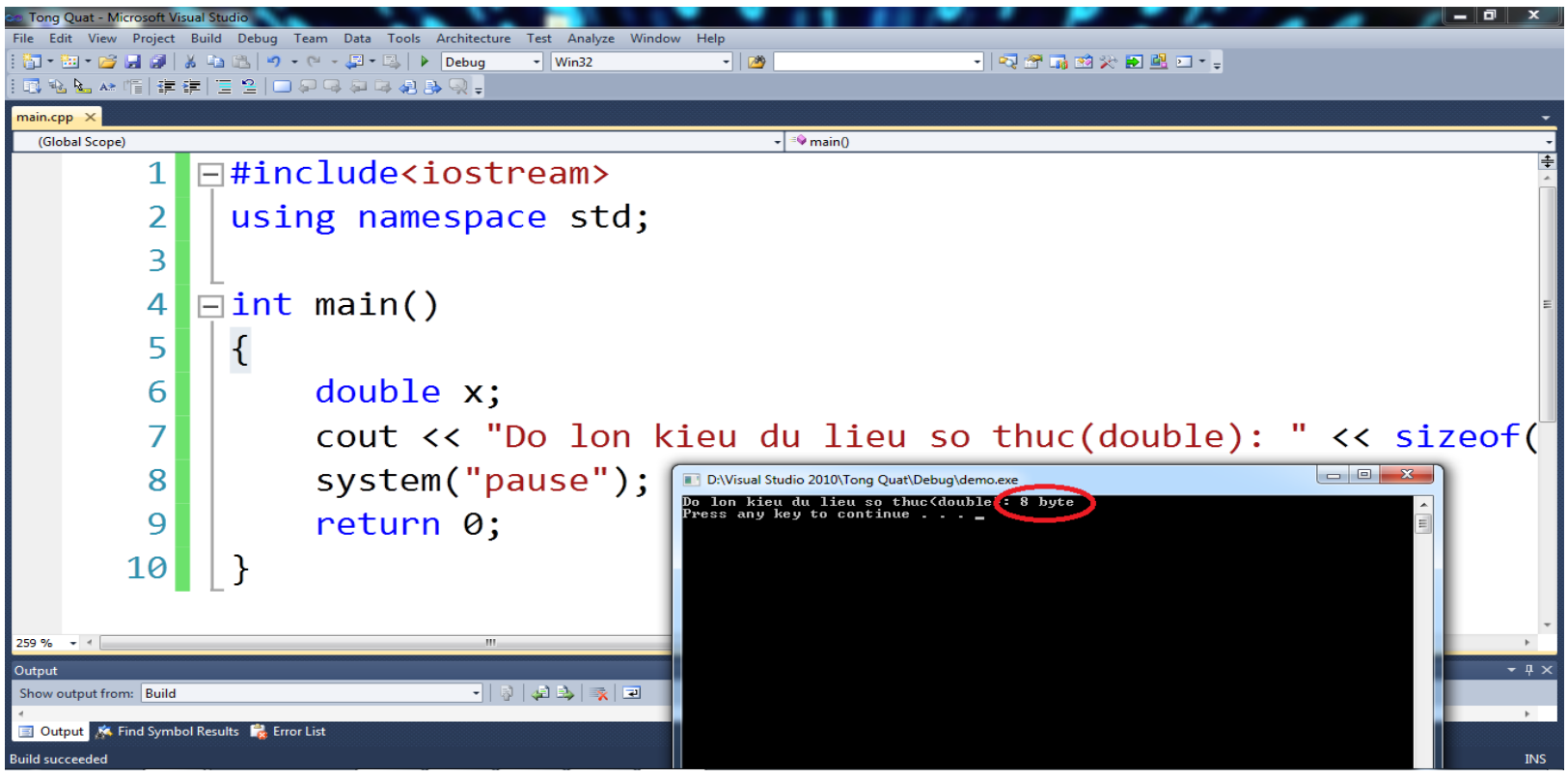
+ *int*



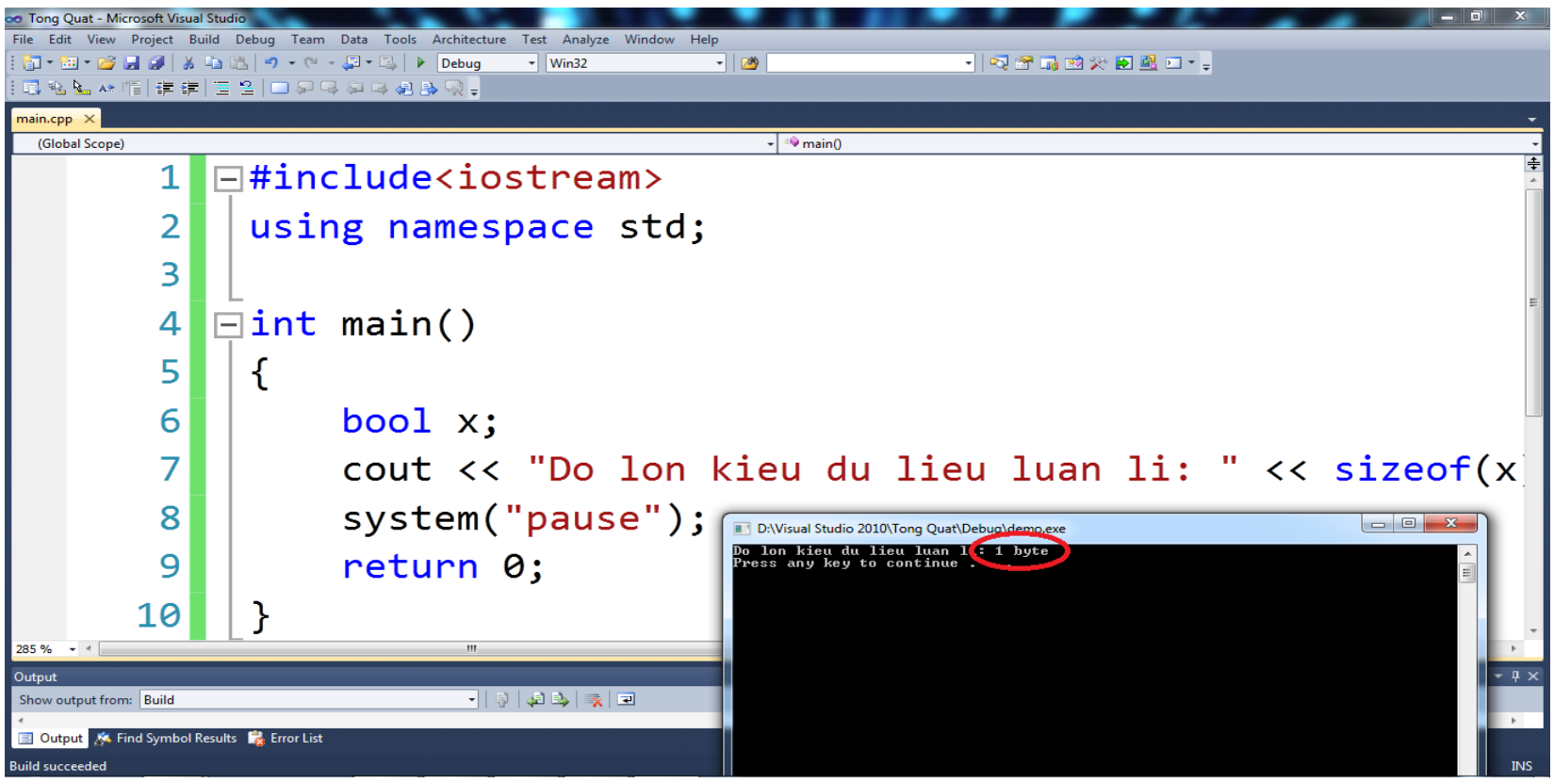
+ *float*



+ *double*



+ *bool*



II. BIẾN, HẰNG

Sau đây là một số từ khoá của C/C++

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

Từ khóa là những từ đã được ngôn ngữ lập trình sử dụng để khai báo kiểu dữ liệu, toán tử và các câu lệnh.

Nói 1 cách dễ hiểu thì từ khóa là những từ hay kí hiệu đã được ngôn ngữ lập trình sử dụng để quy định hoặc quy ước cho nó 1 ý nghĩa xác định, mà người lập trình chỉ có quyền sử dụng lại, chứ không được phép dùng lại từ khóa này với 1 ý nghĩa nào khác.

1. Biến

* ***Khái niệm***

- Biến là đại lượng có giá trị thuộc một kiểu dữ liệu nào đó mà được chấp nhận bởi ngôn ngữ (xem phần các kiểu dữ liệu), giá trị của biến *có thể thay đổi* trong thời gian tồn tại của biến (hay ta nói trong vòng đời của biến).

- Các thành phần của chương trình sẽ được lưu trong bộ nhớ trong và biến cũng không ngoại lệ. Tức là biến cũng được cấp phát một vùng nhớ để lưu giữ giá trị thuộc một kiểu dữ liệu xác định. Vì thế theo một khía cạnh nào đó có thể nói biến là một cái tên đại diện cho ô nhớ trong máy tính, chương trình có thể truy xuất ô nhớ (lấy hoặc ghi giá trị) thông qua tên biến.

- Nói 1 cách dễ hiểu thì biến giống như là khi bạn sinh ra đời – bạn sẽ được đặt cho 1 cái tên(*trường hợp mỗi tên là duy nhất*) – ví dụ là **A**, cái tên **A** đó của bạn có tác dụng là để phân biệt bạn với mọi người xung quanh. Bởi vì trên thế giới này có hàng tỷ người – nếu như bạn không có tên thì sẽ không thể nào phân biệt đâu là bạn – đâu là mọi người xung quanh bạn.

*** Chú ý:**

Một biến nói chung phải có các đặc trưng sau:
- Tên biến.
- Kiểu dữ liệu: kiểu của biến.
- Giá trị hiện tại nó đang lưu giữ - giá trị của biến(nếu có).

- Trong C/C++ cũng như các ngôn ngữ lập trình khác các biến đều phải có tên, các tên biến hay nói chung là tên (gồm tên biến, tên hằng, tên hàm, hoặc từ khoá) là một xâu kí tự và phải tuân theo các quy định của ngôn ngữ lập trình.

QUY TẮC KHAI BÁO BIẾN

+ *Tên chỉ có thể chứa kí tự là chữ cái (‘a’ ,..,’z’; ‘A’,..,’Z’); chữ số(‘0’,..,’9’) và kí tự gạch dưới (_).*
+ *Kí tự đầu tiên của tên phải là **chữ cái** hoặc **kí tự gạch dưới***
+ *Trong tên phân biệt chữ hoa và chữ thường. Tức là hai xâu cùng các kí tự nhưng khác nhau bởi loại chữ hoa hoặc chữ thường là hai tên khác nhau, ví dụ như với 2 xâu kí tự “AB” và “Ab” là hai tên hoàn toàn phân biệt nhau.*
+ *Các từ khoá của ngôn ngữ không được dùng làm tên biến, tên hằng, hay tên hàm. Hay nói khác đi, trong chương trình có thể bạn phải dùng đến tên, tên này do bạn đặt theo ý tưởng của bạn nhưng không được trùng với các từ khoá.*

Ví dụ các tên hợp lệ và không hợp lệ

Tên biến	***Hợp lệ/không hợp lệ***
Temp	Hợp lệ
_hoten	Hợp lệ
A	Hợp lệ
gia-tri	Không hợp lệ vì có kí tự -(dấu trừ)
double	Không hợp lệ vì trùng với từ khóa
9name	Không hợp lệ vì kí tự đầu tiên là số
_1990_tax	Hợp lệ

* ***Cú pháp khai báo biến***

+ < kiểu_dữ_liệu > < tên_biến_1>;

+ < kiểu_dữ_liệu > < tên_biến_1>, <tên_biến_2>, < …>;

Ví dụ:

double x; // khai báo biến x có kiểu dữ liệu là double, giá trị chưa được khởi tạo

int a, b; // khai báo biến a và b có cùng kiểu dữ liệu là int, giá trị chưa được khởi tạo

* ***Cú pháp khởi tạo biến***

+ < kiểu_dữ_liệu > < tên_biến_1 > = < giá_trị_1 >;

+ < kiểu_dữ_liệu > < tên_biến_1> = < giá_ trị_1 >, < tên_ biến_2 > = < giá_trị _2 >, < tên_biến…> = < giá _trị… >;

Ví dụ:

int a = 5, b = 8; // khởi tạo biến a có giá trị ban đầu là 5, b có giá trị ban đầu là 8.

double x = 5.5; // khởi tạo biến x có giá trị ban đầu là 5.5

float y = 6.9, z, i;

trong đó:

+ < **kiểu_dữ_liệu** > là tên một kiểu dữ liệu đã tồn tại, đó có thể là tên kiểu dữ liệu chuẩn hoặc kiểu dữ liệu định nghĩa bởi người lập trình(khi nào lớn sẽ biết :D).
+ < **tên_biến_1** >, < **tên_biến_2**> là các tên biến cần khai báo, các tên này phải tuân theo quy tắc về tên của ngôn ngữ.
+ < **giá_ trị_1** >, < **giá_ trị_2** > là các giá trị khởi đầu cho các biến tương ứng < **tên_biến_1**>, < **tên_biến_2**>. Các thành phần này là tuỳ chọn, nếu có thì giá trị này phải phù hợp với kiểu của biến.

- Trên một dòng lệnh định nghĩa có thể khai báo nhiều biến cùng kiểu, với tên là < **tên_biến_1**>, < **tên_biến_2**>,.. các biến cách nhau bởi dấu phẩy (,) dòng khai báo kết thúc bằng dấu chấm phẩy (;).

- Khi gặp các lệnh định nghĩa biến, chương trình dịch **compiler** sẽ yêu cầu máy tính cấp phát vùng nhớ có kích thước(độ lớn byte) phù hợp với kiểu dữ liệu của biến, nếu có thành phần khởi đầu thì sẽ gán giá trị khởi đầu vào vùng nhớ đó.

- <u>Kết luận</u>: Biến gồm có 2 thành phần là địa chỉ và giá trị.

2. Hằng

* ***Khái niệm***

- Hằng là đại lượng có giá trị thuộc một kiểu dữ liệu nhất định, nhưng *giá trị* của hằng *không thể thay đổi* trong thời gian tồn tại của nó. Nói cách khác là khi ta khởi tạo ra biến hằng thì giá trị của biến đó sẽ không được thay đổi trong suốt chương trình, nếu chúng ta thay đổi thì chương trình sẽ báo lỗi.

- Nói 1 cách dễ hiểu thì hằng trong ngôn ngữ lập trình là 1 đại lượng có giá trị không thể thay đổi trong suốt quá trình chương trình đang thực thi - giống như là: PI = 3.14.., c = 3x10^8(vận tốc ánh sáng)……

* ***Cú pháp khởi tạo hằng***

- Bạn có thể định nghĩa các hằng với tên mà bạn muốn để có thể sử dụng thường xuyên mà không mất tài nguyên cho các biến bằng cách sử dụng chỉ thị **#define**

- Chỉ thị **#define** không phải là một lệnh thực thi, nó là chỉ thị tiền xử lý (preprocessor), đó là lý do trình biên dịch coi cả dòng là một chỉ thị và dòng đó không cần kết thúc bằng dấu chấm phẩy. Nếu bạn thêm dấu chấm phẩy vào cuối dòng, nó sẽ được coi là một phần của giá trị định nghĩa hằng.

#define < tên hằng > <giá trị> // không có dấu ;

- Với tiền tố **const** bạn có thể khai báo các hằng với một kiểu xác định như là bạn làm với một biến.

const < kiểu dữ liệu > < tên hằng >= <giá trị >; // có dấu ;

Ví dụ

```
#define MAX 100  // không có dấu ;

#define PI 3.14   // khống dó dấu ;

const int MAX = 100;

const  int x = 5;  // có dấu ;

const int a = 5; // định nghĩa hằng a kiểu nguyên, có giá trị là 5
const float x = 4; // hằng x kiểu thực, có giá trị là 4.0
const d = 7; // hằng d kiểu int, giá trị là 7
```

---------- ***END*** ----------

- Đây là tài liệu dùng để học và tham khảo.

- Trong quá trình biên soạn bộ tài liệu có:

+ Tham khảo 1 số nguồn tài liệu(google search, ebook, sách chuyên ngành…).

+ Kiến thức – kinh nghiệm cá nhân của tôi.

- Có thể còn 1 số lỗi sai sót – sẽ luôn cập nhật phiên bản mới.

- Mong nhận được sự đóng góp và phản hồi tích cực từ các bạn đọc. Chân thành cảm ơn.