



# IT001 - NHẬP MÔN LẬP TRÌNH

## CHƯƠNG 3: CÁC KIỂU DỮ LIỆU CƠ SỞ, CÁC PHÉP TOÁN VÀ NHẬP XUẤT TRONG C++

Để một chương trình máy tính có thể giải được bài toán thực tế dựa trên các dữ liệu do người dùng cung cấp thì nó phải biết giá trị dữ liệu đó tương ứng với biến nào, miền giá trị của biến số đó và các phép toán có thể thực hiện để tạo ra giá trị kết quả theo yêu cầu. Trong chương này, chúng ta sẽ tìm hiểu xem cấu trúc của một chương trình như thế nào, những kiểu dữ liệu cơ sở và các phép toán, cách khai báo và sử dụng biến mà ngôn ngữ lập trình quy ước,... để lập trình viên có thể sử dụng khi xây dựng chương trình.

Khoa Khoa học máy tính



# NỘI DUNG

- 3.1 Cấu trúc chương trình C++
- 3.2 Bộ từ vựng (**keywords**) trong C++
- 3.3 Các kiểu dữ liệu cơ sở (**Fundamental data types**)
- 3.4 Biến (**Variable**)
- 3.5 Hằng (**Constant**)
- 3.6 Các phép toán (**Operator**)
- 3.7 Biểu thức và độ ưu tiên toán tử
- 3.8 Nhập xuất dữ liệu
- Bài tập



## 3.1 Cấu trúc chương trình C++



## 3.1 Cấu trúc chương trình C++

# Tiền xử lý

Khai báo biến, hàm

```
int main() {
```

Thân hàm chính

```
}
```

Định nghĩa hàm đã khai báo



### 3.1 Cấu trúc chương trình C++

// Chương trình tính diện tích hình tròn

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

→ Tiền xử lý

```
float TinhDienTich(int);
```

→ Nguyên mẫu hàm

```
int main() {
    float r;
    cin >> r;
    cout << TinhDienTich(r) << endl;
    return 0;
}
```

→ Thân hàm chính

```
float TinhDienTich(int r){
    return r*r*PI;
}
```

→ Định nghĩa hàm



### 3.1 Cấu trúc chương trình C++

1.	// Tính diện tích hình tròn	Dòng 1: dòng ghi chú
2.	#include<iostream>	Dòng 2: Thư viện nhập xuất iostream (cin, cout)
3.	using namespace std;	Dòng 3: Khai báo namespace std (cout, cin)
4.	#define PI 3.14	Dòng 4: Định nghĩa macro PI với giá trị 3.14
5.		Dòng 5: Dòng trống sẽ được bỏ qua khi dịch
6.	float TinhDienTich(int);	Dòng 6: Nguyên mẫu hàm TinhDienTich
7.	int main() {	Dòng 7: Bắt đầu hàm main
8.	float r;	Dòng 8: Khai báo biến r có kiểu float
9.	cin >> r;	Dòng 9: Nhập r
10.	cout << TinhDienTich(r);	Dòng 10: Gọi hàm TinhDienTich và in kết quả
11.	return 0;	Dòng 11: Lệnh return kết thúc hàm main
12.	}	Dòng 12: Kết thúc hàm main
13.	float TinhDienTich(int r){	Dòng 13, 14, 15: Định nghĩa hàm TinhDienTich
14.	return r*r*PI;	
15.	}	



## 3.2 Bộ từ vựng trong C++



## 3.2 Bộ từ vựng trong C++

3.2.1 Ký tự (**Character**)

3.2.2 Từ khóa (**Keyword**)

3.2.3 Dấu chấm phẩy

3.2.4 Tên/ Định danh (**Name/Identifier**)

3.2.5 Câu chú thích (**Comment**)



## 3.2.1 Ký tự (Character)

C++ sử dụng nhiều loại ký tự khác nhau để viết chương trình:

- **Chữ cái ký tự Latin**  
`'A', 'B', 'C', ..., 'Z', 'a', 'b', 'c', ..., 'z'`
- **Chữ số thập phân**  
`'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'`
- **Chú thích**
  - Chú thích một dòng: `//`
  - Chú thích nhiều dòng: `/* ... */`
- **Dấu nháy đơn:** Ký tự được đặt trong dấu nháy đơn `' ' ... ' '`
- **Dấu nháy kép:** Chuỗi được đặt trong dấu nháy kép `“ ” ... “ ”`
- **Ký tự điều khiển:** sử dụng ký tự thoát (escape characters) là dấu gạch chéo ngược `\`` đặt ở đầu: `\n`` (dòng mới), `\t`` (tab ngang), `\\"`` (ký tự gạch chéo ngược), `\\"`` (dấu nháy đơn), `\\"`` (dấu nháy kép), `\0`` (null)



## 3.2.1 Ký tự (Character)

- **Toán tử:** `+`, `-`, `\*`, `/`, `%`, `++`, `--`, `=`, `+=`, `-=`, `\*=`, `/=`, `%=`, `==`, `!=`, `>`, `<`, `>=`, `<=`, `&&` (toán tử điều kiện)
- **Ký tự phân tách:** `;` (dấu chấm phẩy), `,` (dấu phẩy)
- **Ký tự ngoặc:** `( )`, `{ }`, `[ ]`
- **Ký tự toán tử bit:** `&`, `|`, `^`, `~`, `<<`, `>>`
- **Ký tự điều khiển:** `#` (cho các chỉ thị tiền xử lý)
- **Ký tự chấm:** `.` (truy cập thành viên)
- **Ký tự mũi tên:** `->` (truy cập thành viên qua con trỏ)
- **Ký tự hai chấm:** `::` (sử dụng trong toán tử ba ngôi và khai báo phạm vi), `::` (toán tử phạm vi)
- **Ký tự dấu gạch dưới:** `\_\_` (thường được sử dụng trong tên biến hoặc hàm)
- ...



## 3.2.2 Từ khóa (Keyword)

- **Không** thể sử dụng từ khóa để đặt tên cho biến, hàm, tên chương trình con.
- Keyword chỉ chứa ký tự chữ cái thường.
- Không có ký hiệu hoặc dấu câu đặc biệt nào được sử dụng trong từ khóa.

**Một số từ khóa thông dụng:**

- **Nhóm 1: Điều khiển luồng**

if, else, switch, case, default, for, while, do, break,  
continue, goto, return, try, catch, throw

- **Nhóm 2: Khai báo và định nghĩa**

int, char, float, double, void, bool, short, long, signed,  
unsigned, enum, class, struct, union, typedef, constexpr,  
constexpr, concept, decltype, inline, static, extern, mutable,  
thread\_local, register



## 3.2.2 Từ khóa (Keyword)

- Nhóm 3: Truy cập và quản lý bộ nhớ

new, delete, sizeof, alignas, alignof, reinterpret\_cast,  
static\_cast, dynamic\_cast, const\_cast

- Nhóm 4: Lập trình hướng đối tượng

class, public, private, protected, virtual, explicit, friend,  
this, operator

- Nhóm 5: Không gian tên và mô-đun

namespace, using, export, module, import

- Nhóm 6: Từ khóa không đồng bộ và đồng thời

co\_await, co\_return, co\_yield, synchronized, atomic\_cancel,  
atomic\_commit, atomic\_noexcept



## 3.2.2 Từ khóa (Keyword)

- Nhóm 7: Logic và toán tử

and, or, not, bitand, bitor, compl, and\_eq, or\_eq, not\_eq

- Nhóm 8: Các từ khóa khác

asm, auto, bool, false, true, nullptr, static\_assert,  
noexcept, requires, reflexpr

- Nhóm 9: Các kiểu dữ liệu và các từ khóa liên quan đến kiểu

int, char, char8\_t, char16\_t, char32\_t, float, double, void,  
bool

...



### 3.2.3 Dấu chấm phẩy

- Dùng để phân cách các câu lệnh.
- Các câu lệnh có thể viết trên 1 dòng, tuy nhiên luôn được phân cách bằng dấu chấm phẩy ;

```
cout << "Xin chao"; return 0;
```



### 3.2.4 Tên/ Định danh (Name/Identifier)

- Một dãy ký tự dùng chỉ tên hằng, biến, hàm.
- Được tạo thành từ các **chữ cái, chữ số, dấu gạch nối \_**
- Quy ước đặt tên:
  - Không trùng với các từ khóa
  - Ký tự đầu tiên là **chữ cái hoặc \_**
  - Tối đa 255 ký tự
  - Không được sử dụng khoảng trắng ở giữa các ký tự
  - Tên có phân biệt chữ hoa chữ thường (case sensitive).



## 3.2.4 Tên/ Định danh (Name/Identifier)

- Một số ví dụ tên biến đặt sai:

1abc

@mail

x&y

X-1

f(x)

default

case

hello world

- Câu hỏi: cách đặt tên nào sau đây đúng?

-  
void1

main

const

Const

\_cin

cin

include

iostream

isdigit



## 3.2.5 Câu chú thích

- Dùng để mô tả hoặc ghi chú trong source code
- Giúp dễ dàng đọc code sau này
- Có 2 cách để viết chú thích trong C++

**Cách 1:** Viết chú thích trong */\*chú thích\*/*

```
/* Day la cau chu thich 1  
   Day la cau chu thich 2*/
```

**Cách 2:** Viết chú thích sau *// chú thích*

```
// Day la cau chu thich 1  
// Day la cau chu thich 2
```

- Nên sử dụng nhất quán 1 cách. Cách 2 được sử dụng phổ biến hơn
- Khi biên dịch source code thì các phần comments sẽ không được biên dịch



### 3.3 Các kiểu dữ liệu cơ sở



## 3.3 Các kiểu dữ liệu cơ bản

3.3.1 Kiểu ký tự (**Character types**)

3.3.2 Kiểu số nguyên (**Numerical Integer types**)

3.3.3 Kiểu số thực (**Floating-point types**)

3.3.4 Kiểu luận lý/logic (**Boolean type**)

3.3.5 Kiểu void (**void type**)

3.3.6 Kiểu nullptr (C++11)

3.3.7 Từ khoá typedef

3.3.8 Một số hàm hữu ích



# Định nghĩa kiểu dữ liệu

- Kiểu dữ liệu (data type) là kiểu hay loại (type) của dữ liệu (data).

Ví dụ:

- Giá trị dữ liệu 5.5 có kiểu là **số thực**
- Giá trị dữ liệu 7 có kiểu là **số nguyên**
- Mỗi kiểu dữ liệu sẽ **chiếm một không gian lưu trữ** nhất định trong bộ nhớ.
- Đặc điểm của kiểu dữ liệu:
  - Bao gồm một tập hợp các giá trị mà biến thuộc kiểu đó có thể nhận
  - Các phép toán có thể thực hiện
- Các kiểu dữ liệu trong ngôn ngữ lập trình
  - Các kiểu cơ sở
  - Các kiểu được cung cấp bởi thư viện có sẵn
  - Các kiểu do người dùng định nghĩa



# Giới thiệu

- C++ có các kiểu cơ sở như sau:

STT	Tên kiểu	Tên kiểu	Ví dụ
1	Kiểu ký tự (Character types)	char, ...	'A', '?', '1', ...
2	Kiểu số nguyên (Numerical Integer types)	int, long, ...	10, 232, ...
3	Kiểu số thực (Floating-point types)	float, double, long double	3.2415, -29.12, ...
4	Kiểu luận lý (Boolean type)	bool	true, false
5	Kiểu void (void type)	void	<i>Không lưu giá trị</i>
6	Null pointer (C++11)	decltype(nullptr)	nullptr

<http://www.cplusplus.com/doc/tutorial/variables/>



### 3.3.1 Kiểu ký tự

- Kiểu char biểu diễn thông qua bảng mã **ASCII**.

Kiểu dữ liệu	Kích thước	Giá trị
char	1 byte	[0, 255]

- Ví dụ:

Ký tự	ASCII
0, 1, ..., 9	48, 49, ..., 57
A, B, ..., Z	65, 66, ..., 90
a, b, ..., z	97, 98, ..., 122
Enter, ESC, Space	13, 27, 32
“, +, -, *, /	34, 43, 45, 42, 47
<, =, >, @	60, 61, 62, 64

<https://en.wikipedia.org/wiki/ASCII>



### 3.3.1 Kiểu ký tự

- ASCII (American Standard Code for Information Interchange - Chuẩn mã trao đổi thông tin Hoa Kỳ), là hệ thống ngôn ngữ gồm bộ ký tự và bộ mã ký tự được hình thành dựa trên chữ bảng chữ cái Latinh.
- Bộ mã ASCII tương tự như một bảng quy ước giúp máy tính có thể hiểu được và hiển thị được những thông tin bằng những ký tự mà bạn nhập vào máy tính.
- **Bảng mã ASCII tiêu chuẩn** được biểu diễn ở dạng nhị phân với 7 ký tự (7 bits) (số thập phân từ 0 đến 127)
- **Bảng mã ASCII mở rộng** được biểu diễn ở dạng nhị phân với 8 ký tự (8 bits) (số thập phân từ 128 đến 255)



### 3.3.1 Kiểu ký tự

- Bảng mã ASCII 256 ký tự chia thành 3 nhóm:
  - Ký tự điều khiển** (ASCII chuẩn): 0 → 31 và 127 (non-printable)
  - Ký tự văn bản** (ASCII chuẩn): 32 → 126
  - Ký tự đồ họa** (extended ASCII): 128 → 255

ASCII control characters			ASCII printable characters								Extended ASCII characters							
00	NULL	(Null character)	32	space	64	@	96	'	128	Ç	160	á	192	ł	224	Ó		
01	SOH	(Start of Header)	33	!	65	A	97	a	129	Ü	161	í	193	ł	225	ß		
02	STX	(Start of Text)	34	"	66	B	98	b	130	é	162	ó	194	ł	226	ô		
03	ETX	(End of Text)	35	#	67	C	99	c	131	â	163	ú	195	ł	227	ò		
04	EOT	(End of Trans.)	36	\$	68	D	100	d	132	ä	164	ñ	196	—	228	ö		
05	ENQ	(Enquiry)	37	%	69	E	101	e	133	à	165	N	197	+	229	ö		
06	ACK	(Acknowledgement)	38	&	70	F	102	f	134	å	166	ø	198	ã	230	µ		
07	BEL	(Bell)	39	'	71	G	103	g	135	ç	167	º	199	Ä	231	þ		
08	BS	(Backspace)	40	(	72	H	104	h	136	ê	168	¿	200	Ł	232	þ		
09	HT	(Horizontal Tab)	41	)	73	I	105	i	137	ë	169	®	201	Ł	233	Ú		
10	LF	(Line feed)	42	*	74	J	106	j	138	è	170	¬	202	Ł	234	Ú		
11	VT	(Vertical Tab)	43	+	75	K	107	k	139	ï	171	½	203	Ł	235	Ú		
12	FF	(Form feed)	44	,	76	L	108	l	140	î	172	¼	204	Ł	236	ý		
13	CR	(Carriage return)	45	-	77	M	109	m	141	í	173	i	205	=	237	Ý		
14	SO	(Shift Out)	46	.	78	N	110	n	142	Ä	174	«	206	+	238	-		
15	SI	(Shift In)	47	/	79	O	111	o	143	À	175	»	207	¤	239	-		
16	DLE	(Data link escape)	48	0	80	P	112	p	144	É	176	„	208	ð	240	≡		
17	DC1	(Device control 1)	49	1	81	Q	113	q	145	æ	177	„	209	đ	241	±		
18	DC2	(Device control 2)	50	2	82	R	114	r	146	Æ	178	„	210	Ê	242	—		
19	DC3	(Device control 3)	51	3	83	S	115	s	147	ô	179	„	211	Ë	243	¾		
20	DC4	(Device control 4)	52	4	84	T	116	t	148	ö	180	„	212	Ê	244	¶		
21	NAK	(Negative acknowl.)	53	5	85	U	117	u	149	ò	181	À	213	í	245	§		
22	SYN	(Synchronous idle)	54	6	86	V	118	v	150	û	182	Â	214	í	246	÷		
23	ETB	(End of trans. block)	55	7	87	W	119	w	151	ù	183	À	215	í	247	·		
24	CAN	(Cancel)	56	8	88	X	120	x	152	ÿ	184	©	216	ï	248	°		
25	EM	(End of medium)	57	9	89	Y	121	y	153	Ö	185	„	217	„	249	..		
26	SUB	(Substitute)	58	:	90	Z	122	z	154	Ü	186	„	218	„	250	-		
27	ESC	(Escape)	59	:	91	[	123	{	155	ø	187	„	219	„	251	1		
28	FS	(File separator)	60	<	92	\	124		156	£	188	„	220	„	252	3		
29	GS	(Group separator)	61	=	93	]	125	}	157	Ø	189	¢	221	„	253	2		
30	RS	(Record separator)	62	>	94	^	126	~	158	×	190	¥	222	„	254	■		
31	US	(Unit separator)	63	?	95	-			159	f	191	„	223	„	255	nbsp		
127	DEL	(Delete)																



### 3.3.1 Kiểu ký tự

- Ví dụ:

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

int main() {
    char c='D';
    cout << c << endl;
    cout << int(c) << endl;
    return 0;
}
```

```
#include <iostream>
int main() {
    char a = 'a';
    char b = 'b';
    std::cout << (a<b);
    return 0;
}
```



### 3.3.2 Kiểu số nguyên

Kiểu dữ liệu (Type)	Kích thước (Size)	Phạm vi (Range of Value)
signed <b>char</b>	1 byte	[-128; 127]
signed <b>short</b> int	2 bytes	[-32,768; 32,767]
signed <b>int</b> ( hoặc signed)	4 bytes	[-2,147,483,648; 2,147,483,647]
signed <b>long</b>	4 bytes	[-2,147,483,648; 2,147,483,647]
signed <b>long long</b> int	8 bytes	[-9,223,372,036,854,775,808; 9,223,372,036,854,775,807]
<b>unsigned char</b>	1 byte	[0; 255]
<b>unsigned short</b> int	2 bytes	[0; 65,535]
<b>unsigned int</b>	4 bytes	[0; 4,294,967,295]
<b>unsigned long</b> int	4 bytes	[0; 4,294,967,295]
<b>unsigned long long</b> int	8 bytes	[0; 18,446,744,073,709,551,615]

- Phạm vi các kiểu số nguyên n bit **có dấu**:  $[-2^{n-1}, 2^{n-1} - 1]$
- Phạm vi các kiểu số nguyên n bit **không dấu**:  $[0, 2^n - 1]$



### 3.3.2 Kiểu số nguyên

- Kích cỡ kiểu **long** trên một số kiến trúc máy tính được thống kê như sau:

OS	arch	size
Windows	IA-32	4 bytes
Windows	Intel 64	4 bytes
Windows	IA-64	4 bytes
Linux	IA-32	4 bytes
Linux	Intel 64	8 bytes
Linux	IA-64	8 bytes
Mac OS X	IA-32	4 bytes
Mac OS X	Intel 64	8 bytes

- Giá trị giới hạn của các kiểu dữ liệu có thể khác nhau tùy thuộc vào kiến trúc máy tính và trình biên dịch.
- Nên sử dụng các hàm và biến trong `climits` để đảm bảo chương trình hoạt động chính xác trên nhiều nền tảng khác nhau.



# Ví dụ

```
#include <iostream>
```

```
int main() {
    // Khai báo biến số nguyên
    short num1 = 65535;
    unsigned short num2 = 65535;
    signed int num3 = 4000000000;
    unsigned int num4 = 4000000000;
```

```
// In ra giá trị của biến
std::cout << "Giá trị signed short: " << num1 << std::endl;
std::cout << "Giá trị unsigned short: " << num2 << std::endl;
std::cout << "Giá trị signed int: " << num3 << std::endl;
std::cout << "Giá trị unsigned int: " << num4 << std::endl;

return 0;
}
```

Kết quả thực thi:

Giá trị signed short: -1

Giá trị unsigned short: 65535

Giá trị signed int: -294967296

Giá trị unsigned int: 4000000000



### 3.3.2 Kiểu số nguyên

- Toán tử **sizeof** dùng để kiểm tra kích thước của kiểu dữ liệu.

```
//Chương trình kiểm tra kích thước kiểu dữ liệu
#include <iostream>
int main() {
    std::cout << sizeof(char) << " byte\n";
    std::cout << sizeof(short) << " bytes\n";
    std::cout << sizeof(int) << " bytes\n";
    std::cout << sizeof(long) << " bytes\n";
    std::cout << sizeof(long long) << " bytes\n";
    return 0;
}
```

Kết quả thực thi:

1 byte  
2 bytes  
4 bytes  
4 bytes  
8 bytes



### 3.3.3 Kiểu số thực

- Các cách biểu diễn số thực:

- Dạng thập phân:

45.0    -256.45           +122.8           .34        15.

- Dạng khoa học:

$$1.257E + 01 = 1.257 * 10^1 = 12.57$$

$$1257.0E - 02 = 1257 * 10^{-2} = 12.57$$

Kiểu dữ liệu	Kích thước	Phạm vi
float	4 bytes	[1.17549E-38, 3.50282E+38] (~6-7 chữ số)
double	8 bytes	[2.22507E-308, 1.79769E+308] (~15-16 chữ số)
long double	16 bytes	[3.4621E-4932, 1.1893E+4932] (~18-19 chữ số)

[https://en.wikipedia.org/wiki/Long\\_double](https://en.wikipedia.org/wiki/Long_double)



# Ví dụ

```
#include <iomanip> // std::setprecision
#include <iostream> // std::cout
#include <cfloat> // LDBL_DIG
using namespace std;
int main() {
    float PI_float;
    double PI_double;
    long double PI_ldouble;
    PI_float = 3.14159265358979323846264338327950288419716939937510;
    PI_double = 3.14159265358979323846264338327950288419716939937510L;
    PI_ldouble = 3.14159265358979323846264338327950288419716939937510L;
    cout << "PI = 3.14159265358979323846264338327950288419716939937510" << endl;
    cout << setprecision(50) << "PI_float = " << PI_float << endl;
    cout << setprecision(50) << "PI_double = " << PI_double << endl;
    cout << setprecision(50) << "PI_ldouble = " << PI_ldouble << endl;
    return 0;
}
```

Kết quả: (Window x64, GNU GCC++ Compiler in IDE Code-Blocks)  
PI = 3.14159265358979323846264338327950288419716939937510  
PI\_float = 3.1415927410125732421875  
PI\_double = 3.141592653589793115997963468544185161590576171875  
PI\_ldouble = 3.1415926535897932385128089594061862044327426701784



### 3.3.4 Kiểu luận lý/logic

- Kiểu dữ liệu bool (Boolean) trong C++ được sử dụng để biểu diễn các giá trị logic, thường được gọi là "đúng" (true) hoặc "sai" (false).

Kiểu dữ liệu	Kích thước	Giá trị
bool	1 byte	<b>false</b> : giá trị 0 <b>true</b> : giá trị khác 0

- Ví dụ:

```
bool isTrue1 = 1;
bool isFalse1 = 0;
bool isTrue2 = true;
bool isFalse2 = false;
```



### 3.3.5 Kiểu void

- Kiểu dữ liệu rỗng không chứa gì cả
- Thường dùng trong các trường hợp sau:
  - Giá trị trả về cho hàm khi không cần giá trị trả về.

```
void swap(int &a, int &b) {  
    int c = b;  
    b = a;  
    a = b;  
}
```

- Một con trỏ chung không trả về bất kì giá trị nào.

```
int a = 10;  
float b = 5;  
void *c;  
c = &a;  
c = &b;
```



### 3.3.6 Kiểu nullptr (C++11)

- Kiểu dữ liệu **nullptr** biểu thị cho con trỏ null, tức là con trỏ không trỏ đến bất kỳ đối tượng nào. Nó có thể gán được cho bất kỳ kiểu con trỏ nào, nhưng không thể gán cho các kiểu dữ liệu không phải con trỏ.
- Ví dụ:

`int* pi = nullptr; → Hợp lệ`

`double* pd = nullptr; → Hợp lệ`

Lưu ý: **NULL** là một macro được định nghĩa trong thư viện C để biểu diễn con trỏ null. Nó được sử dụng phổ biến trong các phiên bản C++ cũ hơn, nhưng hiện nay được khuyến nghị sử dụng **nullptr** thay thế vì lý do an toàn và rõ ràng.



# Giới thiệu: Kiểu string

- Một trong những điểm mạnh của C++ là nó cung cấp những kiểu dữ liệu kết hợp dựa trên những kiểu cơ bản.
- Biến có kiểu string có thể chứa một chuỗi các ký tự (gọi là chuỗi).
- Để dùng kiểu string cần chèn thư viện `<string>`
- Ví dụ:

```
#include <iostream>
#include <string>
int main () {
    std::string mystring;
    mystring = "Truong Dai hoc Cong nghe Thong tin";
    std::cout << mystring << std::endl;
    mystring = "Chao mung cac em den voi lop hoc NMLT.";
    std::cout << mystring << std::endl;
    return 0;
}
```

## Kết quả thực thi:

```
Truong Dai hoc Cong nghe Thong tin
Chao mung cac em den voi lop hoc NMLT.
```



### 3.3.7 Từ khóa typedef

- **typedef** là một từ khóa dùng để định nghĩa một tên gọi mới cho một kiểu dữ liệu đã có.
- Cú pháp: **typedef   kiểu\_có\_sẵn   tên\_mới;**
- Ví dụ:

```
typedef int Integer;  
Integer a = 10; // Tương đương với int a = 10;
```

```
typedef int* IntPointer;  
IntPointer p = nullptr; // Tương đương với int* p = nullptr;
```



### 3.3.8 Một số hàm hữu ích xử lý kiểu dữ liệu

Thư viện	Ví dụ	Kết quả
<limits>	<code>std::numeric_limits&lt;int&gt;::min()</code>	-2147483648
	<code>std::numeric_limits&lt;int&gt;::max()</code>	2147483647
	<code>std::numeric_limits&lt;float&gt;::max()</code>	3.50282e+038
	<code>std::numeric_limits&lt;float&gt;::digits10</code>	6
	...	
<climits>	<code>INT_MIN</code>	-2147483648
	<code>INT_MAX</code>	2147483647
	...	
<cfloat>	<code>FLT_MAX</code>	3.50282e+038
	<code>FLT_DIG</code>	6
	...	



### 3.3.8 Một số hàm hữu ích xử lý kiểu dữ liệu

```
#include <iostream>
```

```
#include <limits>
```

```
int main() {
    // Truy cập giới hạn tối đa của kiểu int
    int max_int = std::numeric_limits<int>::max();
    std::cout << "Giá trị tối đa của int: " << max_int << std::endl;

    // Truy cập giới hạn tối thiểu của kiểu int
    int min_int = std::numeric_limits<int>::min();
    std::cout << "Giá trị tối thiểu của int: " << min_int << std::endl;

    return 0;
}
```



### 3.3.8 Một số hàm hữu ích xử lý kiểu dữ liệu

```
#include <iostream>
#include <limits>

int main() {
    // Truy cập giới hạn tối đa của kiểu int
    int max_int = std::numeric_limits<int>::max();
    std::cout << "Giá trị tối đa của int: " << max_int << std::endl;

    // Truy cập giới hạn tối thiểu của kiểu int
    int min_int = std::numeric_limits<int>::min();
    std::cout << "Giá trị tối thiểu của int: " << min_int << std::endl;

    return 0;
}
```

Kết quả thực thi:

Giá trị tối đa của int: 2147483647  
Giá trị tối thiểu của int: -2147483648



### 3.3.8 Một số hàm hữu ích xử lý kiểu dữ liệu

```
#include <iostream>
#include <limits>

int main() {
    // Khai báo biến long double
    long double value = 3.14159265358979323846L;

    // In ra giá trị của biến với định dạng khoa học
    std::cout << std::scientific << value << std::endl;

    // Truy cập độ chính xác của long double
    std::cout << "Do chinh xac cua kieu long double: "
        << std::numeric_limits<long double>::digits10 << std::endl;

    return 0;
}
```

Kết quả thực thi:

3.141593e+00

Do chinh xac cua kieu long double: 18



### 3.3.8 Một số hàm hữu ích xử lý kiểu dữ liệu

```
#include <iostream>
#include <limits>

int main() {
    // Khai báo biến float
    float value = 3.14159265359;
    // In ra giá trị của biến với định dạng cố định
    std::cout << std::fixed << value << std::endl;

    // Truy cập và in ra số chữ số thập phân tối đa mà float biểu diễn chính xác
    std::cout << "Số chữ số thập phân tối đa của float: "
        << std::numeric_limits<float>::digits10 << std::endl;

    return 0;
}
```

Kết quả thực thi:

3.141593

Số chữ số thập phân tối đa của float: 6



## 3.4 Biến (Variable)



## 3.4 Biến

3.4.1 Khái niệm biến

3.4.2 Khai báo biến

3.4.3 Địa chỉ của biến

3.4.4 Phạm vi biến

3.4.5 Các loại biến thường gặp



### 3.4.1 Khái niệm biến

- Biến là **một ô nhớ hoặc một vùng nhớ dùng để chứa dữ liệu trong quá trình thực hiện chương trình**
- Mỗi biến có một kiểu dữ liệu cụ thể, kích thước của biến phụ thuộc vào kiểu dữ liệu.
- Giá trị của biến có thể được thay đổi.
- Quy cách đặt tên biến: tương tự như quy tắc đặt Tên và định danh
- Ví dụ:

```
int so_nguyen;
```

```
float so_thuc;
```



# Câu hỏi: Cách đặt tên biến nào sau đây đúng?

```
double _;  
int void;  
int main;  
int const;  
int Const;  
int Cin;  
int cin;  
int include;  
int iostream;
```



## 3.4.2 Khai báo biến

- Cú pháp khai báo biến (có thể khai báo mỗi lần một biến hoặc nhiều biến):

- Cách 1: 

```
<Kiểu_dữ_liệu> <Tên_biến_1>;  
<Kiểu_dữ_liệu> <Tên_biến_2>;
```

- Cách 2: 

```
<Kiểu_dữ_liệu> <Tên_biến_1>, <Tên_biến_2>;
```

- Ví dụ:

```
int i, j, k;  
char c, ch;  
float f, salary;  
double d;
```

- Khai báo và khởi tạo: **kiểu\_dữ\_liệu tên\_biến\_1 = giá\_trị;**

```
int d = 3;  
char x = 'x';  
float f = 2.1;
```



## 3.4.2 Khai báo biến

- **Ví dụ:**

Viết chương trình nhập vào 3 số nguyên a, b và c.

Cho biết a, b, c có tạo thành 3 cạnh của tam giác không ?

→ Cần khai báo bao nhiêu biến?

```
int a;  
int b;  
int c;
```

```
int a, b, c;
```



## 3.4.2 Khai báo biến

- Trong C++ có 3 cách **khởi tạo giá trị** cho biến:

```
int NamSinh = 1997; // Copy initialization
```

```
int NamSinh (1997); //direct initialization
```

```
int NamSinh {1997}; // Uniform initialization C++11
```

- Phép gán: Sử dụng toán tử gán

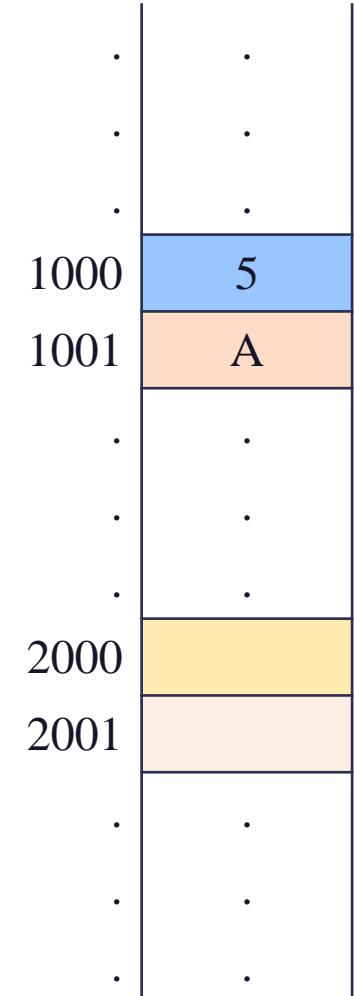
```
int NamSinh;
```

```
NamSinh=1997;
```



### 3.4.3 Địa chỉ của biến

- RAM được tạo nên bởi nhiều ô nhớ.
- Mỗi ô nhớ có kích thước 1 byte.
- Mỗi ô nhớ có địa chỉ duy nhất và được đánh số từ 0 trở đi.
- Mỗi biến khi được khai báo sẽ được cấp 1 vùng nhớ với địa chỉ duy nhất để lưu trữ biến đó.
- Để truy cập vào địa chỉ của một biến ta sử dụng toán tử &.



Main Memory with  
some data



# Quy trình xử lý biến của Trình biên dịch

- Dành riêng một vùng nhớ với địa chỉ duy nhất để lưu biến đó
- Liên kết địa chỉ vùng nhớ đó với tên biến
- Khi gọi tên biến, nó sẽ truy xuất tự động đến vùng nhớ đã liên kết với tên biến

```
#include <iostream>
int main() {
    int a = 5;
    std::cout << "Gia tri cua a: " << a << '\n';
    std::cout << "Dia chi cua a: " << &a << '\n';
    return 0;
}
```

008FF82C  
a 5

Gia tri cua a: 5  
Dia chi cua a: 008FF82C



### 3.4.4 Phạm vi biến

- C bắt buộc khai báo tất cả các biến ở đầu một hàm.
- C++ có thể định nghĩa các biến ở bất kỳ vị trí nào trong hàm.
- Dựa trên phạm vi hoạt động của biến, có thể chia làm 2 loại:
  - Biến toàn cục (global variable)
  - Biến cục bộ (local variable)



# Biến cục bộ (local variable)

- Biến được **định nghĩa** trong một hàm hoặc một khối lệnh
- Biến cục bộ chỉ được sử dụng bên trong hàm hoặc khối lệnh
- Các hàm bên ngoài khác sẽ không truy cập được biến cục bộ
- Ví dụ:

```
#include <iostream>
int main() {
    int a = 2, b = 3;
    int c;
    c = a + b;
    std::cout << c;
    return 0;
}
```



# Biến toàn cục (global variable)

- Biến toàn cục được định nghĩa bên ngoài các hàm, và thường được định nghĩa ở phần đầu tiên xử lý.
- Biến toàn cục sẽ giữ giá trị của biến xuyên suốt chương trình.
- Tất cả các hàm đều có thể truy cập biến toàn cục từ vị trí nó được khai báo.
- Ví dụ:

```
#include <iostream>
int g;
int main() {
    int a = 2, b = 3;
    g = a + b;
    std::cout << g;
    return 0;
}
```

```
#include <iostream>
int g = 20;
int main() {
    int a = 2, b = 3;
    g = a + b;
    std::cout << g;
    return 0;
}
```



## 3.4.5 Các loại biến

- Các loại biến:

- **Biến thường:** (biến này không phải là biến con trỏ) nó được dùng để lưu giá trị dữ liệu (không phải địa chỉ).

Ví dụ: `int a;`

- **Biến con trỏ:** Biến này dùng để lưu trữ địa chỉ của vùng nhớ khác.

Ví dụ: `int *p;`

- **Biến tham chiếu:** Không được cấp phát ô nhớ riêng mà dùng chung ô nhớ với biến được tham chiếu.

Ví dụ:

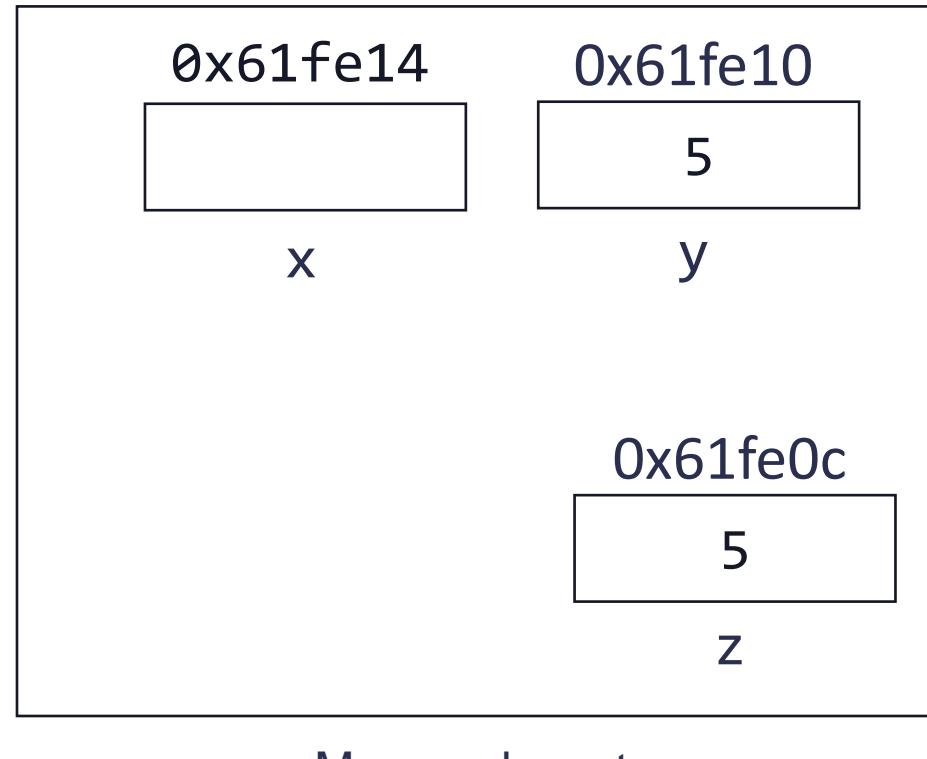
`int a;`

`int &m=a;`



# Ví dụ

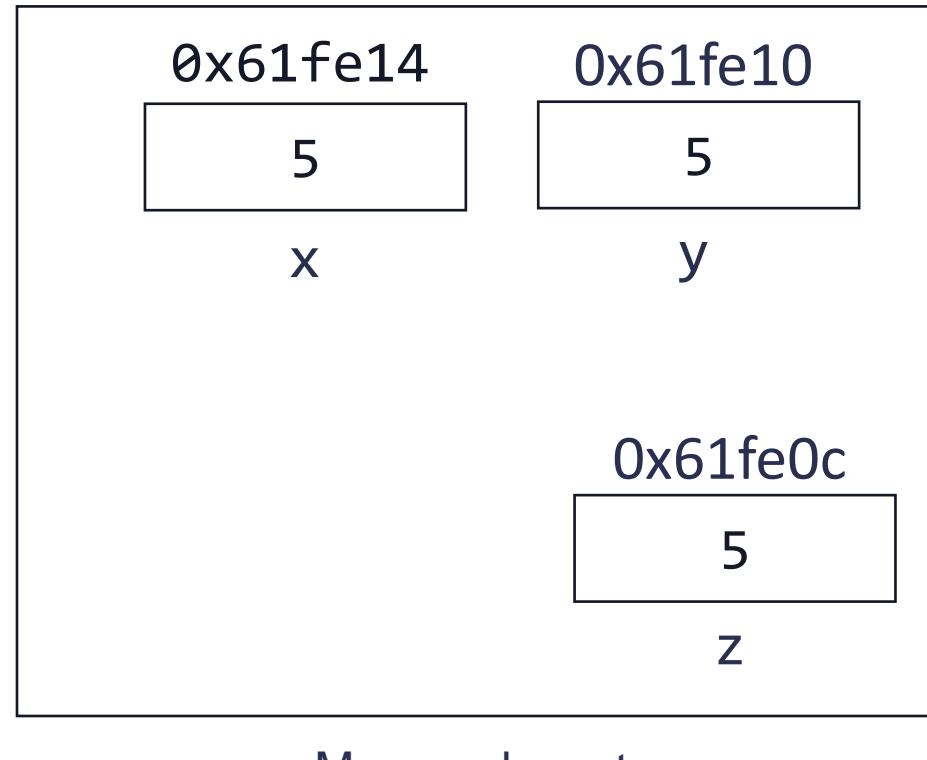
```
#include <iostream>
using namespace std;
int main() {
    int x, y = 5, z = y;
    x = y;
    int* p = &x;
    int &m = x;
    x=6;
    cout << "x: " << x << "\t\t&x: " << &x << endl;
    cout << "y: " << y << "\t\t&y: " << &y << endl;
    cout << "z: " << z << "\t\t&z: " << &z << endl;
    cout << "p: " << p << "\t\t&p: " << &p << endl;
    cout << "m: " << m << "\t\t&m: " << &m << endl;
    return 0;
}
```





# Ví dụ

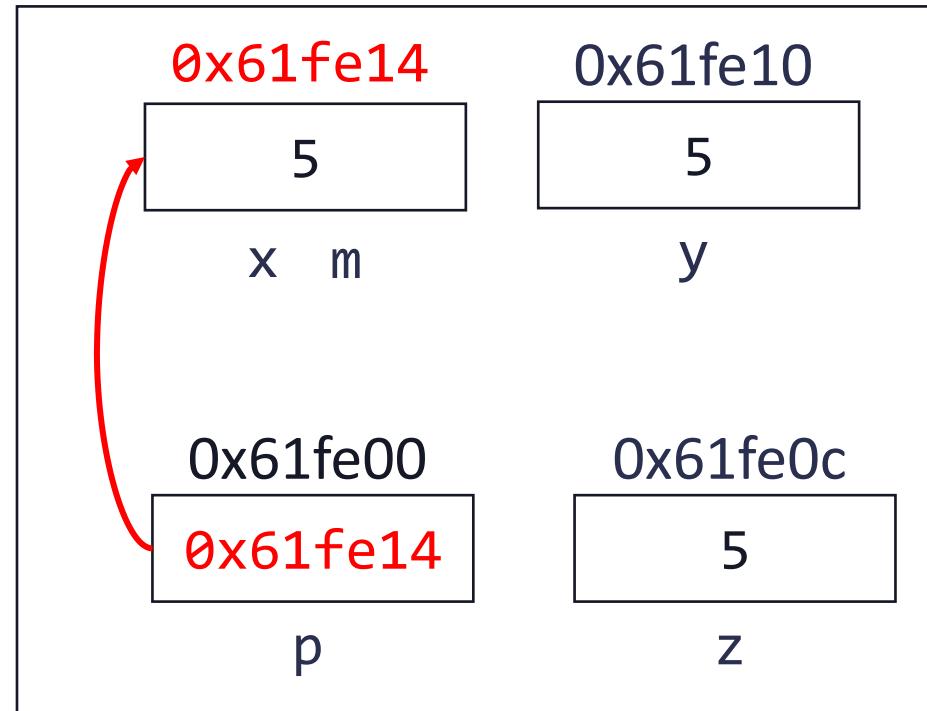
```
#include <iostream>
using namespace std;
int main() {
    int x, y = 5, z = y;
    x = y;
    int* p = &x;
    int &m = x;
    x=6;
    cout << "x: " << x << "\t\t&x: " << &x << endl;
    cout << "y: " << y << "\t\t&y: " << &y << endl;
    cout << "z: " << z << "\t\t&z: " << &z << endl;
    cout << "p: " << p << "\t\t&p: " << &p << endl;
    cout << "m: " << m << "\t\t&m: " << &m << endl;
    return 0;
}
```





# Ví dụ

```
#include <iostream>
using namespace std;
int main() {
    int x, y = 5, z = y;
    x = y;
    int* p = &x;
    int &m = x;
    x=6;
    cout << "x: " << x << "\t\t&x: " << &x << endl;
    cout << "y: " << y << "\t\t&y: " << &y << endl;
    cout << "z: " << z << "\t\t&z: " << &z << endl;
    cout << "p: " << p << "\t\t&p: " << &p << endl;
    cout << "m: " << m << "\t\t&m: " << &m << endl;
    return 0;
}
```

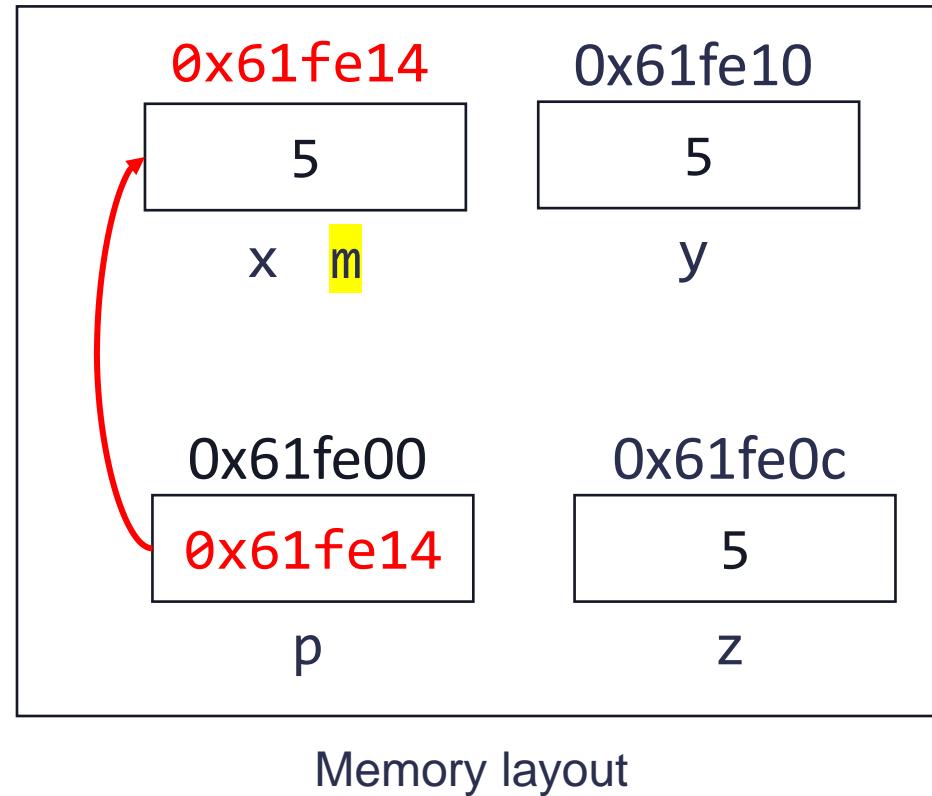


Memory layout



# Ví dụ

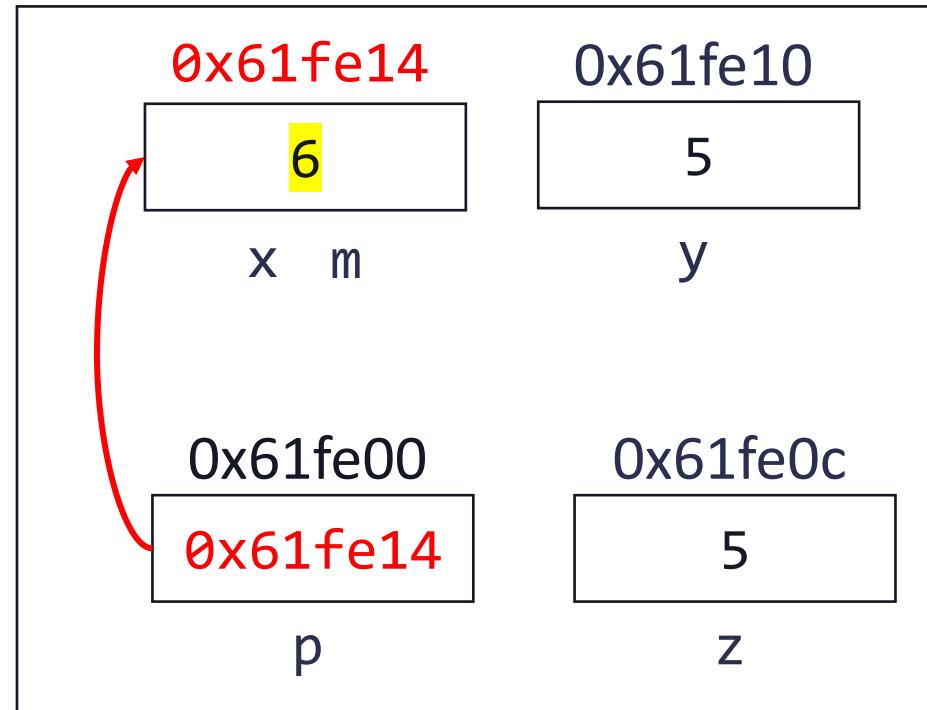
```
#include <iostream>
using namespace std;
int main() {
    int x, y = 5, z = y;
    x = y;
    int* p = &x;
    int& m = x;
    x=6;
    cout << "x: " << x << "\t\t&x: " << &x << endl;
    cout << "y: " << y << "\t\t&y: " << &y << endl;
    cout << "z: " << z << "\t\t&z: " << &z << endl;
    cout << "p: " << p << "\t\t&p: " << &p << endl;
    cout << "m: " << m << "\t\t&m: " << &m << endl;
    return 0;
}
```





# Ví dụ

```
#include <iostream>
using namespace std;
int main() {
    int x, y = 5, z = y;
    x = y;
    int* p = &x;
    int &m = x;
    x=6;
    cout << "x: " << x << "\t\t&x: " << &x << endl;
    cout << "y: " << y << "\t\t&y: " << &y << endl;
    cout << "z: " << z << "\t\t&z: " << &z << endl;
    cout << "p: " << p << "\t\t&p: " << &p << endl;
    cout << "m: " << m << "\t\t&m: " << &m << endl;
    return 0;
}
```

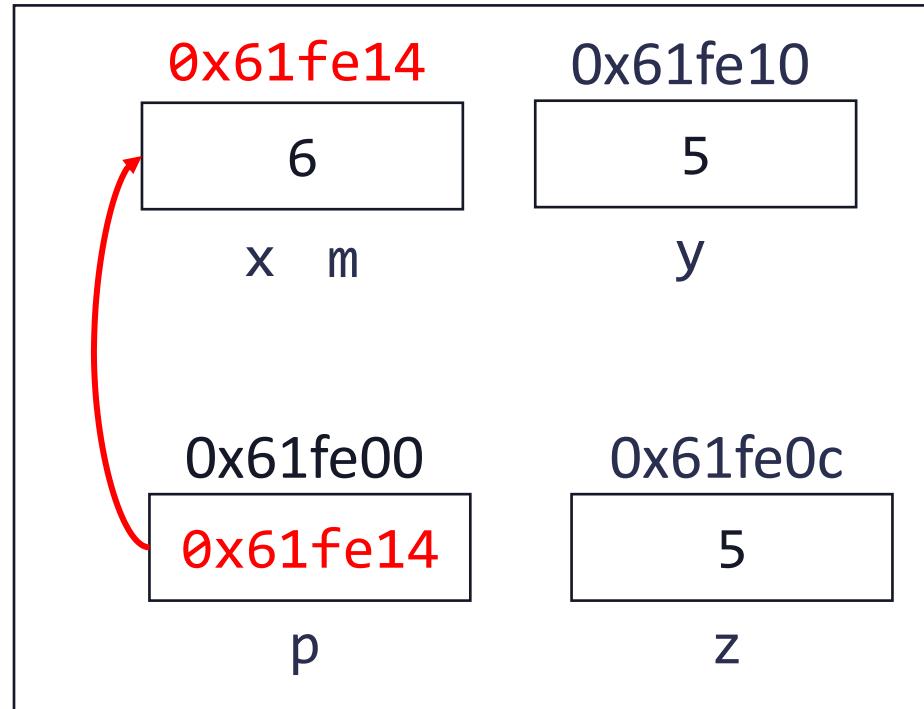


Memory layout



# Ví dụ

```
#include <iostream>
using namespace std;
int main() {
    int x, y = 5, z = y;
    x = y;
    int* p = &x;
    int &m = x;
    x=6;
    cout << "x: " << x << "\t\t&x: " << &x << endl;
    cout << "y: " << y << "\t\t&y: " << &y << endl;
    cout << "z: " << z << "\t\t&z: " << &z << endl;
    cout << "p: " << p << "\t\t&p: " << &p << endl;
    cout << "m: " << m << "\t\t&m: " << &m << endl;
    return 0;
}
```



Memory layout

x: 6	&x: 0x61fe14
y: 5	&y: 0x61fe10
z: 5	&z: 0x61fe0c
p: 0x61fe14	&p: 0x61fe00
m: 6	&m: 0x61fe14



## 3.5 Hằng (constant)



## 3.5 Hằng (constant)

- Hằng số (**constants**) là những giá trị **không thay đổi** trong suốt quá trình thực thi của chương trình.
- **Literal** (giá trị trực tiếp) là loại hằng số rõ ràng nhất. Chúng được sử dụng để thể hiện các giá trị cụ thể trong mã nguồn của chương trình.
- Các hằng số **Literal** có thể được phân loại thành: Boolean, số nguyên, số thực, ký tự, chuỗi, con trỏ và do người dùng định nghĩa.
- Ví dụ:

```
bool b = true;  
int i = 1000;  
double r = 10e2;  
char newline = '\n';  
string str = "Chao ca lop.";  
int *p=nullptr;
```

Các giá trị literal là:

true  
1000  
10e2  
\n  
"Chao ca lop."  
nullptr



## 3.5 Hằng (constant)

- Bảng ví dụ về các loại literal:

Loại hằng	Ví dụ
Hằng số nguyên (Integer literal)	-212, 0213, 0x4b, 18L, 18u, 18l, 18ll, 18ul, 18ull // Integer literal mặc định là int, u or U: unsigned, l or L: long, ll or LL: long long (không phân biệt hoa thường)
Hằng số thực (Floating-point literal)	3.14159, 6.02e23, 1.6e-19, ..., 3.24159L, 6.02e23f // Floating-point literal mặc định là double, l or L: long double, f or F: float (không phân biệt hoa thường)
Hằng luận lý (Boolean literal)	true, false
Hằng ký tự (Char literal)	'y', 'h', ... Escape code: '\0', '\n', '\t', '\x41' (hằng ký tự 'A' mã hexa), ...
Hằng chuỗi (String literal)	"C:\user\username\local", ...
Hằng con trỏ Pointer literal	nullptr
Hằng kiểu dữ liệu được định nghĩa User-defined literal	...



# Hằng số với từ chỉ thị #define

- **Cú pháp:** `#define <tên_hằng> <giá_trị>`
- **Lưu ý:**
  - <giá\_trị>: literal (giá trị cố định), các câu lệnh, các biểu thức, ...
  - #define là một chỉ thị tiền xử lý, nên nó sẽ thay nơi nào có <tên\_hằng> xuất hiện bằng <giá\_trị>. Việc thay thế này được định nghĩa ở bộ tiền xử lý (trước khi biên dịch chương trình)
  - #define có phạm vi toàn cục (có hiệu lực từ điểm định nghĩa) và không tuân theo các quy tắc phạm vi biến của C++



# Hằng số với từ chỉ thị #define

- Ví dụ:

```
#include <iostream>
#define PI 3.14
#define tab '\t'
#define inChuVi std::cout << 2*r*PI;
#define newline std::cout << std::endl
int main() {
    double r = 10;
    std::cout << "Chu vi: " << tab;
    inChuVi
    newline;
    return 0;
}
```

```
int main() {
    double r = 10;
    std::cout << "Chu vi: " << '\t';
    std::cout << 2*r*3.14;
    std::cout << std::endl;
    return 0;
}
```

Kết quả thực thi:

Chu vi: 62.8



# Hằng số với từ khóa const

- **Cú pháp:** `const <kiểu_dữ_liệu> <tên_hằng> = <giá_trị_hằng>;`
- **Lưu ý:**
  - Giá trị của biến hằng số phải được khởi tạo ngay khi khai báo
  - Biến const có kiểu dữ liệu cụ thể
  - Biến hằng số được lưu trữ trong bộ nhớ giống như các biến thông thường.
- **Ví dụ:**

```
#include <iostream>
int main() {
    const float PI = 3.14;
    int r = 2;
    std::cout << r*PI;
    return 0;
}
```



# Hằng số kiểu liệt kê enum

- Enum là một loại hằng số dạng bảng liệt kê.
- Bảng liệt kê là một danh sách các giá trị nguyên.
- **Cú pháp:**

```
enum <ten_enum> {<trang_thai_1> = <gia_tri_1>, <trang_thai_2> = <gia_tri_2>, ... }
```

- Trong đó:
  - <ten\_enum>: tên của enum (được đặt tên như biến/ định danh)
  - <trang\_thai\_1>, ...: tên của các trạng thái (được đặt tên như biến/ định danh)
  - <gia\_tri\_1>, ...: là các số nguyên
- Tên trạng thái trong các bảng liệt kê khác nhau phải khác nhau. Các giá trị không cần phải khác biệt trong cùng một bảng liệt kê.



# Hằng số kiểu liệt kê enum

- **Lưu ý:**

- Nếu các giá trị không được chỉ định cho tên thì mặc định tên đầu tiên có giá trị là 0, tên thứ 2 có giá trị là 1, tiếp tục tăng 1 đơn vị cho các tên tiếp theo.
- Nếu không phải tất cả các giá trị đều được chỉ định, thì các giá trị không xác định sẽ được tiếp tục tiến triển từ giá trị được chỉ định cuối cùng (do trình biên dịch thực hiện).

- **Ví dụ:**

```
enum dayOfWeek { Mon, Tue, Wed, Thur, Fri, Sat, Sun };
```

```
enum escapes { BELL = '\a', BACKSPACE = '\b', TAB = '\t',
               NEWLINE = '\n', VTAB = '\v', RETURN = '\r' };
```

```
enum months { JAN = 1, FEB, MAR, APR, MAY, JUN,
              JUL, AUG, SEP, OCT, NOV, DEC }; // FEB = 2, MAR = 3, ...
```



# Hằng số kiểu liệt kê enum

- Có thể sử dụng enum để thay thế cho tiền chỉ thị `#define` trong một số trường hợp.

```
#define KEYWORD 01
```

```
#define EXTERNAL 02
```

```
#define STATIC 04
```

```
enum { KEYWORD = 01, EXTERNAL = 02, STATIC = 04 };
```



# Hằng số kiểu liệt kê enum

- Ví dụ:

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

```
int main() {
    enum dayOfWeek { Mon, Tue=3, Wed=4, Thur=5, Fri=3, Sat, Sun };
    cout << Mon << endl; //assigned 0 by compiler
    cout << Tue << endl;
    cout << Wed << endl;
    cout << Thur << endl;
    cout << Fri << endl;
    cout << Sat << endl;
    cout << Sun << endl;
    return 0;
}
```

Kết quả thực thi:

```
0
3
4
5
3
4
5
```



# Hằng số kiểu liệt kê enum

- Ví dụ:

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

```
int main() {
    enum Direction {
        UP = 1, //assigned 1 by programmer
        DOWN = 3, //assigned 3 by programmer
        LEFT, //assigned 4 by compiler
        RIGHT //assigned 5 by compiler
    };
    cout << UP << " " << DOWN << " " << LEFT << " " << RIGHT << endl;
    return 0;
}
```



# Hằng số kiểu liệt kê enum

- Ví dụ:

```
#include <iostream>

enum gioi_tinh { nam = 1,nu = 2, khac = 3};

int main() {
    gioi_tinh sv_gt = nam;
    std::cout << sv_gt;
    return 0;
}
```



# Câu hỏi trên lớp

- Khai báo biến sau:

Biến	Kiểu dữ liệu	Ví dụ
Họ tên sinh viên	string	Nguyen Xuan Minh
Giới tính	char	M (Male)/ F (Female)
Điểm Toán	float	8.5
Điểm Tin	float	9
Điểm Trung bình	float	8.5
Xếp loại	string	Giỏi
Kết quả	bool	1 (Đậu)/ 0 (Rớt)



# Câu hỏi trên lớp

- Sửa lại đoạn khai báo biến sau:

```
int _1HoTen;  
float GioiTinh;  
bool Diem_Toan, DiemLy;  
char DiemTrungBinh;  
int _XepLoai;  
string KetQua;
```

=> Viết chương trình  
khai báo các biến dữ  
liệu như bên.



# Bài tập

- Câu 1: Cho biết năm sinh của một người và tính tuổi của người đó.
- Câu 2: Cho 2 số a, b. Tính tổng, hiệu, tích và thương của hai số đó.
- Câu 3: Cho biết tên sản phẩm, số lượng và đơn giá. Tính tiền và thuế giá trị gia tăng phải trả, biết:
  - tiền = số lượng \* đơn giá
  - thuế giá trị gia tăng = 10% tiền
- Câu 4: Cho biết điểm thi và hệ số 3 môn Toán, Lý, Hóa của một sinh viên. Tính điểm trung bình của sinh viên đó.
- Câu 5: Cho biết bán kính của đường tròn. Tính chu vi và diện tích của hình tròn đó.



# Bài tập 1

- Cho biết năm sinh của một người và tính tuổi của người đó?

```
#include <iostream>
int main() {
    int namsinh = 1998;
    int tuoi = 0;
    tuoi = 2016-namsinh;
    return 0;
}
```



# Bài tập 2

- Cho 2 số a, b. Tính tổng, hiệu, tích và thương của hai số đó.

```
#include <iostream>
int main() {
    int a = 10, b = 3, tong = 0;
    int hieu = 0;
    int tich = 0;
    float thuong = 0;
    tong = a + b;
    hieu = a - b;
    tich = a * b;
    thuong = (float)a / b;
    return 0;
}
```



# Bài tập 3

- Cho biết tên sản phẩm, số lượng và đơn giá. Tính tiền và thuế giá trị gia tăng phải trả, biết:
  - tiền = số lượng \* đơn giá
  - thuế giá trị gia tăng = 10% tiền

```
#include <iostream>
int main() {
    int Soluong = 10, Dongia = 500, Tien = 0;
    float Vat = 0;
    Tien = Soluong * Dongia;
    Vat = Tien * 0.1;
    return 0;
}
```



# Bài tập 4

- Cho biết điểm thi và hệ số 3 môn Toán, Lý, Hóa của một sinh viên. Tính điểm trung bình của sinh viên đó.

```
#include <iostream>
int main() {
    float toan = 6.5;          float hstoan = 2.0;
    float ly = 7.0;            float hsly = 1.0;
    float hoa = 7.5;          float hshoa = 1.0;
    float Dtb = 0;
    Dtb = (toan * hstoan + ly * hsly + hoa * hshoa)
          / (hstoan + hsly + hshoa);
    return 0;
}
```



# Bài tập 5

- Cho biết bán kính của đường tròn. Tính chu vi và diện tích của hình tròn đó.

```
#include <iostream>
#define PI 3.14
int main() {
    float r = 6.5;
    float chuvi = 0;
    float dientich = 0;
    chuvi = 2 * PI * r;
    dientich = PI * r * r;
    return 0;
}
```



# Bài tập về nhà

- Câu 1: Cho số xe (gồm 4 chữ số) của bạn. Cho biết số xe của bạn được mấy nút?
- Câu 2: Cho 1 ký tự chữ thường. In ra ký tự chữ hoa tương ứng.
- Câu 3: Cho 3 số nguyên. Cho biết số lớn nhất và nhỏ nhất?
- Câu 4: Cho số thực  $x$ . Tính giá trị các biểu thức sau:
  - a)  $y_1 = 4(x^2 + 10x\sqrt{x} + 3x + 1)$
  - b)  $y_2 = \frac{\sin(\pi x^2) + \sqrt{x^2 + 1}}{e^{2x} + \cos\left(\frac{\pi}{4}x\right)}$
- Câu 5: Viết chương trình cho 2 giờ (giờ, phút, giây) và thực hiện cộng, trừ 2 giờ này.



# Chúc các em học tốt !

