



# THỰC HÀNH BUỔI 2

**GIẢNG VIÊN: THS.LÂM DU ĐẠT**

Email: [datld@donga.edu.vn](mailto:datld@donga.edu.vn)

Trường Đại học Đông Á : Khoa Điện – Điện tử

# I. CÁC KIỂU DỮ LIỆU CƠ BẢN

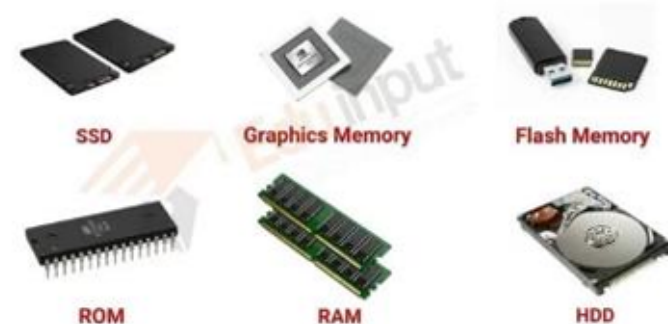
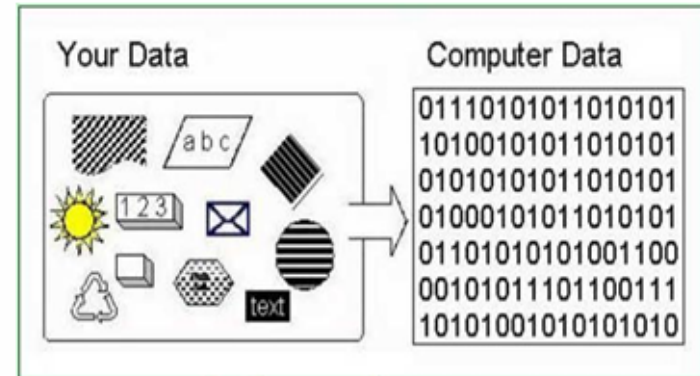
- Kiểu kí tự
- Kiểu số nguyên
- Kiểu số thực
- Kiểu hằng
- Kiểu chuỗi

**Công thức lấy kích thước dữ liệu**

`sizeof(kiểu dữ liệu)`

**Công thức lưu dữ liệu vào bộ nhớ**

`kiểu_dữ_liệu tên_biến = giá_trị`



# 1. KIỂU KÍ TỰ

Kiểu dữ liệu	Kích thước	Phạm vi	Ví dụ khai báo	format khi printf()
<b>char</b>	1 byte	-128 -> 127	<b>char</b> x = -100; <b>char</b> y=100	%c (xuất ra kí tự đồ hoạ) hoặc %d (hệ 10) , %02x (hệ 16)
<b>unsigned char</b>	1 byte	0 -> 255	<b>unsigned char</b> x =200	

# MÃ ASCII

0		20	¶	40	<	60	<	80	P	100	d	120	x	140	â	160	á	180	+	200	£	220	▬	240	≡
1	␣	21	␣	41	>	61	=	81	Q	101	e	121	y	141	í	161	í	181	=	201	␣	221	▬	241	±
2	␣	22	␣	42	*	62	>	82	R	102	f	122	z	142	î	162	ó	182		202	␣	222	▬	242	±
3	␣	23	␣	43	+	63	?	83	S	103	g	123	<	143	ï	163	ô	183		203	␣	223	▬	243	±
4	␣	24	␣	44	,	64	@	84	T	104	h	124	!	144	ÿ	164	æ	184		204	␣	224	▬	244	±
5	␣	25	␣	45	-	65	A	85	U	105	i	125	"	145	ÿ	165	␣	185		205	␣	225	▬	245	±
6	␣	26	␣	46	.	66	B	86	V	106	j	126	#	146	ÿ	166	␣	186		206	␣	226	▬	246	±
7	␣	27	␣	47	/	67	C	87	W	107	k	127	\$	147	ÿ	167	␣	187		207	␣	227	▬	247	±
8	␣	28	␣	48	0	68	D	88	X	108	l	128	%	148	ÿ	168	␣	188		208	␣	228	▬	248	±
9	␣	29	␣	49	1	69	E	89	Y	109	m	129	^	149	ÿ	169	␣	189		209	␣	229	▬	249	±
10	␣	30	␣	50	2	70	F	90	Z	110	n	130	_	150	ÿ	170	␣	190		210	␣	230	▬	250	±
11	␣	31	␣	51	3	71	G	91	[	111	o	131		151	ÿ	171	½	191		211	␣	231	▬	251	±
12	␣	32	␣	52	4	72	H	92	\	112	p	132		152	ÿ	172	¾	192		212	␣	232	▬	252	±
13	␣	33	␣	53	5	73	I	93	]	113	q	133		153	ÿ	173		193		213	␣	233	▬	253	±
14	␣	34	␣	54	6	74	J	94	^	114	r	134		154	ÿ	174		194		214	␣	234	▬	254	±
15	␣	35	␣	55	7	75	K	95	_	115	s	135		155	ÿ	175		195		215	␣	235	▬	255	±
16	␣	36	␣	56	8	76	L	96	a	116	t	136		156	ÿ	176		196		216	␣	236	▬		
17	␣	37	␣	57	9	77	M	97	b	117	u	137		157	ÿ	177		197		217	␣	237	▬		
18	␣	38	␣	58	:	78	N	98	c	118	v	138		158	ÿ	178		198		218	␣	238	▬		
19	␣	39	␣	59	;	79	O	99		119	w	139		159	ÿ	179		199		219	␣	239	▬		

## 2. KIỂU SỐ NGUYÊN

Kiểu dữ liệu	Kích thước	Phạm vi	Ví dụ khai báo	printf()
short	2 byte	$-(2^{16}/2) \rightarrow (2^{16}/2)-1$	short x =10; short y=-50;	%d
unsigned short	2 byte	$0 \rightarrow (2^{16})-1$	unsigned short x = 8	%u
int	4 byte	$-(2^{32}/2) \rightarrow (2^{32}/2)-1$	int x =3000; int y= -5000	%d
unsigned int	4 byte	$0 \rightarrow 2^{32}-1$	unsigned int x =50000	%u
long long	8 byte	$-(2^{64}/2) \rightarrow (2^{64}/2)-1$	long long y=-5000000000	%lld
unsigned long long	8 byte	$0 \rightarrow 2^{64}-1$	unsigned long long y = 90	%llu

## Kiểu số nguyên được chuẩn hóa

Thư viện `<stdint.h>`

Dữ liệu thuộc <code>&lt;stdint.h&gt;</code>	Dữ liệu C chuẩn	Kích cỡ
<code>int8_t</code>	<code>char</code>	1 byte
<code>uint8_t</code>	<code>unsigned char</code>	1 byte
<code>int16_t</code>	<code>short</code>	2 byte
<code>uint16_t</code>	<code>unsigned short</code>	2 byte
<code>int32_t</code>	<code>int</code>	4 byte
<code>uint32_t</code>	<code>unsigned int</code>	4 byte
<code>int64_t</code>	<code>long long</code>	8 byte
<code>uint64_t</code>	<code>unsigned long long</code>	8 byte

### 3. KIỂU SỐ THỰC

Kiểu dữ liệu	Kích thước	Phạm vi	Ví dụ khai báo	printf()
<b>float</b>	4 byte	$3.4 \cdot 10^{-38} \rightarrow 3.4 \cdot 10^{38}$	<b>float</b> x = 10.5;	%f
<b>double</b>	8 byte	$1.7 \cdot 10^{-308} \rightarrow 1.7 \cdot 10^{308}$	<b>double</b> x = 11111.78	%lf

## 4. KIỂU HẲNG

Phân loại	Ví dụ khai báo
Hằng dấu phẩy động	123.15 ; -123.15 ; 12.1E-8 ; 3.2E10
Hằng nguyên	12 ; 12UL ; 12L ;
Hằng nhị phân	0b00001111
Hằng thập lục	0x1a1b
Hằng kí tự	'a' ; '1'
Tên hằng	#define SO_PI 3.14



## 5. KIỂU CHUỖI

Xâu kí tự "HELLO VIETNAM" ; "1"

Bản chất kiểu xâu kí tự là một mảng kiểu kí tự (char) và kết thúc là ký tự null (\0)

- Bộ nhớ lưu chuỗi:



# BÀI TẬP 1

Cho biết kiểu dữ liệu phù hợp với các tình huống sau

- Dữ liệu nhiệt độ
- Dữ liệu độ ẩm
- Dữ liệu trạng thái nút nhấn
- Dữ liệu trạng thái hệ thống
- Dữ liệu góc quay động cơ
- Dữ liệu điểm sinh viên
- Dữ liệu tên cảm biến
- Dữ liệu điện áp
- Số 11111
- Số 1111111111111
- Số 111.1111
- Số 1111111111111.1111

## II. BIẾN

- Biến là tên người lập trình đặt để đại diện cho vùng nhớ
- Dựa vào biến ta biết được thông tin sau
  - Địa chỉ bắt đầu của vùng nhớ
  - kích thước vùng nhớ
  - giá trị lưu trong vùng nhớ
- **Quy tắc đặt tên biến**
  - Chứa các kí tự nằm trong tập kí tự C hỗ trợ
  - không được bắt đầu bằng số
  - không chứa khoảng trắng
  - không được chứa các toán tử
  - không chứa các kí hiệu đặc biệt
  - không trùng với tên từ khoá, tên hằng

# KHỞI TẠO BIẾN

**kiểu\_dữ\_liệu** tên\_biến = **giá\_trị\_khởi\_tạo**

## BÀI TẬP 2:

- Tạo biến `nhiet_do` để lưu giá trị nhiệt độ 35.5 độ
- Tạo biến `do_am` để lưu giá trị độ ẩm 90.5%
- Tạo biến `trang_thai_nut_nhan` để lưu giá trị 0 hoặc 1
- Tạo biến `trang_thai_he_thong` để lưu giá trị "tot", "canh bao" , "loi" ...
- Tạo biến `ten_cam_bien` để lưu giá giá trị "cam bien nhiet phong 1"
- **Xuất tất cả thông tin các biến ra màn hình**

## BÀI TẬP 3:

- Thực hiện tất cả nhiệm vụ trên cho board arduino

# ĐỌC ĐỊA CHỈ CỦA BIẾN: TOÁN TỬ &

- Lấy địa chỉ của biến thông qua toán tử &
- Để lưu địa chỉ ta dùng con trỏ: `kieu_du_lieu * ten_con_tro`
- Thông qua con trỏ ta có thể lấy lại giá trị của ô nhớ

```
int x = 18;
//tạo con trỏ để lưu địa chỉ của biến x
int* p_x = &x; //con trỏ đã lưu địa chỉ của biến x
//in con giá trị con trỏ
printf("gia tri p_x=%p",p_x);
//dùng con trỏ lấy lại giá trị ô nhớ
int y= *p_x;
//in giá trị của y
printf("gia tri x=%d\n",y);
```

# KHỞI TẠO BIẾN HẲNG: TỪ KHÓA `const`

- Khi ta muốn một biến được gán cố định và không cho phép thay đổi trong suốt chương trình ta dùng biến hằng
- Cú pháp: `const` kiểu\_dữ\_liệu **tên\_biến** = giá\_trị\_khởi\_tạo

```
const double pi = 3.1415;  
const pin_lcd_d1=3;
```

- Khi cố tình sửa đổi biến hằng sẽ báo lỗi:

```
pin_lcs_d1=5    //sẽ báo lỗi
```

# PHẠM VI HOẠT ĐỘNG CỦA BIẾN

## BIẾN TOÀN CỤC

- Ta muốn một biến có thể sử dụng mọi nơi trong chương trình
- Khai báo biến nằm ngoài tất cả các hàm
- Chỉ sử dụng biến toàn cục khi cần thiết, nếu lạm dụng sẽ mau hết bộ nhớ

```
#include <stdio.h>
int x =8; //biến toàn cục
int main()
{
    {
        printf("x=%d",x); //biến x sử dụng trong khối lệnh
    }
    //Tới đây biến x vẫn có thể sử dụng tiếp
}
```

## BIẾN CỤC BỘ

- Biến được giải phóng khi thực thi một khối lệnh
- Nên ưu tiên biến cục bộ vì nó tiết kiệm ram

```
#include <stdio.h>
int main()
{
    {
        int x =8; //khai biến cục bộ
        printf("x=%d",x);
    }
    //tới đây biến x đã được giải phóng
}
```



### III. TOÁN TỬ SỐ HỌC

Kí hiệu	Ý nghĩa	Ví dụ
+	Thực hiện phép toán cộng	int x = 10+3 ;
-	Thực hiện phép toán trừ	int x= 10-3;
*	Thực hiện phép toán nhân	int x = 10*3
/	Thực hiện phép chia	int x = 10/3 ; kết quả bằng 3 do x là kiểu nguyên double y = 10/3 ; kết quả bằng 3.33333
%	Thực hiện phép lấy dư	int x = 10%3 ; kết quả bằng 1

# ÉP KIỂU TRONG C

- Ép kiểu tự động:
  - Khi thực hiện tính toán, toán hạng có kiểu dữ liệu thấp sẽ tự chuyển thành kiểu dữ liệu cao
  - Giá trị vế phải chuyển tự động sang kiểu của vế trái

```
int x = 2.5 + 7 ;  
printf("x=%d",x);
```

- Ép kiểu chủ động

```
float x = (double)43/5;  
printf("x=%f",x)
```

# CÁC HÀM TOÁN HỌC KHÁC: THƯ VIỆN <math.h>

Hàm	Chức năng
abs(x)	Giá trị tuyệt đối
sqrt(x)	Căn bậc hai
pow(x, y)	Lũy thừa ( $x^y$ )
sq(x) Bình phương ( $x^2$ )	
sin(x)	Sin (radian)
cos(x)	Cos (radian)
tan(x)	Tang (radian)
min(a, b)	Trả về giá trị nhỏ hơn
max(a, b)	Trả về giá trị lớn hơn
constrain(x, a, b)	Giới hạn giá trị x trong [a, b]
map(x, in_min, in_max, out_min, out_max)	Quy đổi giá trị từ thang đo này sang thang đo khác
random(min, max)	Số ngẫu nhiên trong khoảng [min, max)

- **Tính delta trong phương trình bậc 2:**

- Tạo biến a có giá trị 1
- Tạo biến b có giá trị 10
- Tạo biến c có giá trị 5
- Tạo biến delta tính bởi công thức:  $\Delta = b^2 - 4ac$
- Xuất biến delta ra màn hình
- Tạo biến x1 tính bởi công thức:  $x_1 = \frac{-b + \sqrt{\Delta}}{2a}$
- Xuất biến x1 ra màn hình

## BÀI TẬP 4:

- **Kiến thức cơ bản về input digital arduino:**

- Tín hiệu đưa vào một chân thường có điện áp chạy từ 0 cho đến 5V
- Với tín hiệu **digital** chỉ có 2 mức 0v hoặc 5V
- Khi đọc tín hiệu **digital** vi điều khiển trả về một trong 2 kết quả 0 hoặc 1

- **Kiến thức cơ bản về input analog arduino:**

- Với tín hiệu **analog** thì điện áp nằm trong khoảng 0 cho đến 5 ( 2.3V , 4.8V )
- Với tín hiệu analog thì vi điều khiển sử dụng một bộ ADC để mã hóa điện áp thành các mức số nguyên chạy từ 0-1023
- Sử dụng công thức :  $v_{in} = (muc\_dien\_ap/1023)*5$  sẽ tính được điện áp  $v_{in}$  khi biết các mức số nguyên

- **Hãy viết chương trình thực hiện tác vụ sau**

- Tạo biến `muc_dien_ap` có giá trị 500
- Tạo biến `vin` được tính theo công thức  $v_{in} = (muc\_dien\_ap/1023)*5$
- Xuất kết quả `vin` ra màn hình

## BÀI TẬP 5:

- **Chuyển thang đo:**
- **Hãy viết chương trình thực hiện tác vụ sau**
  - cho biết thang mức điện áp chạy từ 0 đến 1024
  - Cho biết thang góc quay chạy từ 0 đến 360
  - Sử dụng hàm map để qui đổi thang đo mức điện áp sang thang đo góc quay để tính được `goc_quay` khi biết trước `muc_dien_ap = 500`
  - Xuất kết quả `goc_quay` ra màn hình