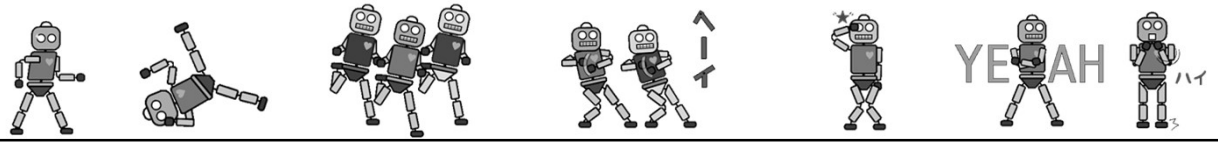
LY TU TRONG COLLEGE

TRƯỜNG CAO ĐẲNG LÝ TỰ TRỌNG TP.HCM
KHOA ĐIỆN – ĐIỆN TỬ

HỌC PHẦN
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG



Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu *(2 tiết)*
2. Chuyển đổi kiểu dữ liệu
3. Hàm *(2 tiết)*
4. Lệnh điều kiện
5. Lệnh vòng lặp
6. Các phép tính toán *(2 tiết)*
7. Xử lý chuỗi ký tự

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.1. void

1.2. int

1.3. unsigned int

1.4. long

1.5. unsigned long

* Hậu tố U và L

1.6. float

1.7. double

1.8. boolean

1.9. char

1.10. byte

1.11. array

1.12. string

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

Numeric types	Bytes	Range	Use
int	2	–32768 to 32767	Represents positive and negative integer values.
unsigned int	2	0 to 65535	Represents only positive values; otherwise, similar to int.
long	4	–2147483648 to 2147483647	Represents a very large range of positive and negative values.
unsigned long	4	4294967295	Represents a very large range of positive values.
float	4	3.4028235E+38 to –3.4028235E+38	Represents numbers with fractions; use to approximate real-world measurements.
double	4	Same as float	In Arduino, double is just another name for float.
boolean	1	false (0) or true (1)	Represents true and false values.
char	1	–128 to 127	Represents a single character. Can also represent a signed value between –128 and 127.
byte	1	0 to 255	Similar to char, but for unsigned values.

Other types	Use
String	Represents arrays of chars (characters) typically used to contain text.
void	Used only in function declarations where no value is returned.

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.1. void

"**void**" là một từ khóa chỉ dùng trong việc khai báo một function
Những function được khai báo với " **void** "
sẽ **<u>không trả về bất kì dữ liệu nào</u>** khi được gọi.

1.2. int

Kiểu **int** là kiểu số nguyên chính được dùng trong chương trình Arduino.
Kiểu int chiếm 2 byte bộ nhớ
Trên mạch Arduino Uno,
kiểu int có giá trị từ -32,768 đến 32,767 (-2^{15} đến $2^{15} - 1$) *(16 bit)*

Cú pháp: **int** ***[tên biến] = [giá trị];***
int ***[tên biến];***

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.3. unsigned int

Kiểu **unsigned int** là kiểu **<u>số nguyên</u>**
nằm trong khoảng từ 0 đến 65535 (0 đến $2^{16} - 1$).
Kiểu unsigned int chiếm 2 byte bộ nhớ

Cú pháp: **unsigned int** ***[tên biến] = [giá trị];***
unsigned int ***[tên biến];***

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.4. long

long là kiểu dữ liệu mở rộng của **int**.
Những biến có kiểu **long**
mang giá trị 32bit từ -2,147,483,648 đến -2,147,483,647.
Kiểu long chiếm 4 byte bộ nhớ

** Khi tính toán với số nguyên (biến kiểu int), phải thêm hậu tố "**L**" phía sau các số nguyên kiểu int để chuyển chúng sang kiểu long.*

Cú pháp: **long** *[tên biến]* **=** *[giá trị]*;

long *[tên biến]*;

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.5. unsigned long

Kiểu **unsigned long** là kiểu **<u>số nguyên</u>**
nằm trong khoảng từ 0 đến 4,294,967,295 (0 đến 2^{32} - 1).
Kiểu unsigned int chiếm 4 byte bộ nhớ

Cú pháp: **unsigned long** *[tên biến]* **=** *[giá trị]*;

unsigned long *[tên biến]*;

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

* Hậu tố U và L

Theo mặc định, một hằng số nguyên được coi là một **số nguyên kiểu int**

Để xác định một hằng số nguyên với một kiểu dữ liệu khác:

+ Dùng 'u' hoặc 'U' để biểu thị kiểu dữ liệu unsigned (không âm).
Ví dụ: 33u

+ Dùng 'l' hoặc 'L' để biểu thị kiểu dữ liệu long.
Ví dụ: 100000L

+ Dùng 'ul' hoặc 'UL' để biểu thị kiểu dữ liệu unsigned long.
Ví dụ: 32767ul

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.6. float

Kiểu **float** là kiểu **số thực**
nằm trong khoảng từ -3.4028235E+38 đến 3.4028235E+38
Kiểu float chiếm 4 byte bộ nhớ

Với kiểu dữ liệu float, có từ 6-7 chữ số có nghĩa nằm ở bên mỗi bên dấu "."
Điều đó có nghĩa rằng có thể đặt một số thực dài đến 15 ký tự
(bao gồm dấu .)

Lưu ý: 2 số thực chia cho nhau thì kết quả sẽ ra một số thực
2 số nguyên chia cho nhau thì kết quả sẽ ra một số nguyên

Cú pháp: **float** *[tên biến]* **=** *[giá trị]*;

float *[tên biến]*;

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.7. double

là tên gọi khác của kiểu **float**

1.8. boolean

Một biến kiểu **boolean** sẽ chỉ nhận một trong hai giá trị: **true** hoặc **false**
Kiểu boolean chiếm 1 byte bộ nhớ

Những cặp giá trị sau là tương đương nhau:

true - false
HIGH - LOW
1 - 0

Cú pháp: **boolean** *[tên biến]* **=** *[giá trị]*;

boolean *[tên biến]*;

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.9. char

char là kiểu dữ liệu biểu diễn cho 1 ký tự
Kiểu char chiếm 1 byte bộ nhớ

Kiểu char chỉ nhận các giá trị trong **bảng mã ASCII**

Kiểu char được lưu dưới dạng 1 byte (có các giá trị từ -127 - 128)

→ *thay vì thiết đặt một biến kiểu char có giá trị là 'A',*
có thể đặt biến với giá trị 65

Cú pháp: **char** *[tên biến]* **=** *[giá trị]*;

char *[tên biến]*;

Ví dụ:
```
char myChar = 'A';
char myChar = 65;
```

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.10. byte

byte là kiểu dữ liệu biểu diễn **số nguyên** nằm trong khoảng từ 0 đến 255

Cú pháp: **byte** *[tên biến]* **=** *[giá trị]*;

byte *[tên biến]*;

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.11. array

Kiểu **array** là kiểu **mảng**

(tập hợp các giá trị có liên quan và được đánh dấu bằng những chỉ số)

Ví dụ:

```
int myInts[6];          // tạo mảng myInts chứa tối đa 6 phần tử (được đánh dấu từ 0-5),
                        // các phần tử này đều có kiểu là int
                        // → khai báo này chiếm 2*6 = 12 byte bộ nhớ
int myPins[] = {2, 4, 8, 3, 6};   // tạo mảng myPins chứa 5 phần tử (lần lượt là 2, 4, 8, 3, 6).
                                  // Mảng này không giới hạn số lượng phần tử vì có khai báo là "[]"
int mySensVals[6] = {2, 4, -8, 3, 2};
                        // tạo mảng mySensVals chứa tối đa 6 phần tử,
                        // trong đó 5 phần tử đầu tiên có giá trị lần lượt là 2, 4, -8, 3, 2
char message[6] = "hello";      // tạo mảng ký tự (dạng chuỗi) có tối đa 6 ký tự
```

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.11. array

Truy cập các phần tử trong mảng

Chú ý: *Phần tử đầu tiên trong mảng luôn được đánh dấu là **số 0***

→ Điều này có nghĩa rằng, việc khai báo một mảng có tối đa 10 phần tử, thì phần tử cuối cùng (thứ 10) được đánh dấu là số 9

Ví dụ:

```
int myArray[10]={9,3,2,4,3,2,7,8,9,11};   // myArray[9]  có giá trị là 11
```

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.11. array

Gán một giá trị cho một phần tử

```
mySensVals[0] = 10;
```

Đọc một giá trị của một phần tử và gán cho một biến nào đó cùng kiểu dữ liệu

```
x = mySensVals[0];
```

→ x = 10

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.12. string

Kiểu **string** là kiểu chuỗi ký tự, sử dụng **<u>mảng ký tự</u>** để biểu diễn chuỗi

Cú pháp: **char** *[tên biến]*[n] **=** *[giá trị]***;**
char *[tên biến]*[n]**;**

Chú ý:
mỗi chuỗi đều cần có 1 ký tự NULL (\0),
nếu không khai báo ký tự NULL (\0) ở cuối chuỗi
thì trình biên dịch sẽ tự động thêm vào

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.12. string

Ví dụ:

```
char Str1[15];      // khai bảo chuỗi có độ dài là 15 ký tự
char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};      //khai báo chuỗi có độ dài tối đa là 8 ký tự
                                                          và đặt nó giá trị ban đầu là arduino (7 ký tự)
char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
char Str4[ ] = "arduino";   // Chương trình dịch sẽ tự động điều chỉnh kích thước cho chuỗi Str4 này
char Str5[8] = "arduino";
char Str6[15] = "arduino";  // Một cách khai báo khác với độ dài tối đa lớn hơn
```

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

1. Các kiểu dữ liệu

1.12. string

Ví dụ: Cũng có thể khai bảo một chuỗi dài như sau:

```
char myString[] = "This is the first line"
                  " this is the second line"
                  " etcetera";
```

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

2. Chuyển đổi kiểu dữ liệu

2.1. Hàm char()

2.2. Hàm byte()

2.3. Hàm int()

2.4. Hàm word()

2.5. Hàm long()

2.6. Hàm float()

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

2. Chuyển đổi kiểu dữ liệu

2.1. Hàm char()

Hàm char() có nhiệm vụ chuyển kiểu dữ liệu của một giá trị về **kiểu char**

Cú pháp: **char(x)**

trong đó, x là một giá trị thuộc bất kỳ kiểu dữ liệu nào

Kết quả trả về là giá trị thuộc **kiểu char**

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

2. Chuyển đổi kiểu dữ liệu

2.2. Hàm byte()

Hàm byte() có nhiệm vụ chuyển kiểu dữ liệu của một giá trị về **kiểu byte**

Cú pháp: **byte(x)**

trong đó, x là một giá trị thuộc bất kỳ kiểu dữ liệu nào

Kết quả trả về là giá trị thuộc **kiểu byte**

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

2. Chuyển đổi kiểu dữ liệu

2.3. Hàm int()

Hàm int() có nhiệm vụ chuyển kiểu dữ liệu của một giá trị về **kiểu int**

Cú pháp: **int(x)**

trong đó, x là một giá trị thuộc bất kỳ kiểu dữ liệu nào

Kết quả trả về là giá trị thuộc **kiểu int**

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

2. Chuyển đổi kiểu dữ liệu

2.4. Hàm word()

Hàm word() có nhiệm vụ chuyển kiểu dữ liệu của một giá trị về **kiểu word**
hoặc ghép 2 giá trị kiểu byte thành 1 giá trị kiểu word

Cú pháp: **word(x)**

word(8bitdau, 8bitcuoi)

trong đó, x là một giá trị thuộc bất kỳ kiểu dữ liệu nào

*8bitdau, 8bitcuoi là một giá trị **kiểu byte***

Kết quả trả về là giá trị thuộc **kiểu word**

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

2. Chuyển đổi kiểu dữ liệu

2.5. Hàm long()

Hàm long() có nhiệm vụ chuyển kiểu dữ liệu của một giá trị về **kiểu long**

Cú pháp: **long(x)**

trong đó, x là một giá trị thuộc bất kỳ kiểu dữ liệu nào

Kết quả trả về là giá trị thuộc **kiểu long**

Chương 2: LẬP TRÌNH C TRÊN ARDUINO IDE

2. Chuyển đổi kiểu dữ liệu

2.6. Hàm float()

Hàm float() có nhiệm vụ chuyển kiểu dữ liệu của một giá trị về **kiểu float**

Cú pháp: **float(x)**

trong đó, x là một giá trị thuộc bất kỳ kiểu dữ liệu nào

Kết quả trả về là giá trị thuộc **kiểu float**

BÀI TẬP CHƯƠNG 2

1. Giải thích hoạt động của đoạn chương trình sau:

```
const int ledPin = 13;

void setup()
{
 pinMode(ledPin, OUTPUT);
}

void loop()
{
 digitalWrite(ledPin, HIGH);
 delay(2000);
 digitalWrite(ledPin, LOW);
 delay(2000);
}
```

BÀI TẬP CHƯƠNG 2

2. Giải thích hoạt động của đoạn chương trình sau:

```
 void setup()
{
 pinMode(2, OUTPUT);
 pinMode(3, OUTPUT);
 pinMode(4, OUTPUT);
 pinMode(5, OUTPUT);
 pinMode(6, OUTPUT)
}
 void loop()
{
 digitalWrite(2, HIGH);
 delay(500);
 digitalWrite(2, LOW);
 digitalWrite(3, HIGH);
 delay(500);
 digitalWrite(3, LOW);
 digitalWrite(4, HIGH);
 delay(500);
 digitalWrite(4, LOW);
 digitalWrite(5, HIGH);
 delay(500);
 digitalWrite(5, LOW);
 digitalWrite(6, HIGH);
 delay(500);
 digitalWrite(6, LOW);
 digitalWrite(5, HIGH);
 delay(500);
 digitalWrite(5, LOW);
 digitalWrite(4, HIGH);
 delay(500);
 digitalWrite(4, LOW);
 digitalWrite(3, HIGH);
 delay(500);
 digitalWrite(3, LOW);
}
```

BÀI TẬP CHƯƠNG 2

3. Giải thích hoạt động của đoạn chương trình sau:

```
#define LED 12
#define BUTTON 7
void setup()
{
 pinMode(LED, OUTPUT);
 pinMode(BUTTON, INPUT);
}
void loop()
{
 if ( digitalRead(BUTTON) == HIGH )
 {
 digitalWrite(LED, HIGH);
 delay(500);
 digitalWrite(LED, LOW);
 }
}
```