## Операції зі змінними та константами.

**Типи змінних**

**Глобальна змінна** оголошується на початку скетча поза межами всіх функцій. Глобальна змінна буде видима у будь-якому місці скетча. Приклад глобальної змінної **sound_sensor_pin**:

```
byte sound_sensor_pin = 7; //займає 1 байт пам'яті
void setup() {
  pinMode(sound_sensor_pin,INPUT);
}
```

**Локальна змінна** оголошується у функції, і доступ до неї можна отримати лише в межах цієї функції. Приклад локальної змінної **sound_sensor_pin**:

```
void setup() {
  byte sound_sensor_pin = 6;
  pinMode(sound_sensor_pin,INPUT);
}
```

Локальних змінних може бути кілька з однаковими іменами, але з різними значеннями. Це пов'язано з тим, що локальна змінна вивантажується з пам'яті мікроконтролера при виході із функції. Тобто дві змінні з однаковими іменами та різними значеннями не існують одночасно.

**Константи**

**const**<тип><ім'я> = <значення> – оголосити константу.

Наприклад оголошуємо константу **LedPin** та надаємо їй значення 13:

**const int LedPin** = 13;

```
const int LedPin = 13;
const int pins_in_count=6;      //Входи для підключення перемикачів
void setup() {
}
void loop() {
```

Можна в окремих випадках оголошувати константу та надавати їй значення за допомогою директиви **#define:**

**#define** <ім'я> <значення>

Наприклад, такий рядок на початку скетчу оголосить змінну **sound_sensor_pin** та присвоїть їй значення 7:

**#define sound_sensor_pin 7** //Тут крапка з комою не ставиться!

```
#define sound_sensor_pin 7 //не займає пам'ять
void setup() {
  pinMode(7,INPUT);
}
void loop()  {
  digitalRead(7);
}
```

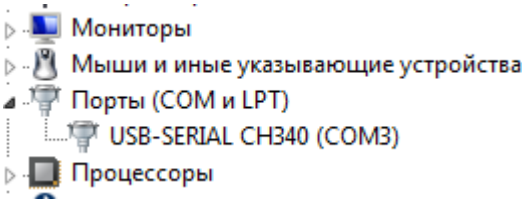
Якщо ви спробуєте змінити значення постійної після її оголошення, ви отримаєте помилку!

**Математичні оператори**

| | | |
|---|---|---|
| **+, -, *, /** | додавання, віднімання, множення та ділення. | |
| **pow(x, y);** | $x^y$, функція pow може навіть зводити в дробовий ступінь: **pow(2, 3.5);** | |
| **sq (x);** | Означає: $x^2$ | |
| **sqrt (x);** | $\sqrt{x}$ | |
| **abs(x);** | модуль числа x: \| x \| | |
| **sin (x), cos (x), tan (x);** | тригонометричні функції синус, косинус, тангенс | |
| **round(x);** | математичне округлення (якщо значення після коми більше або дорівнює 5, округлює у більший бік) | |
| **ceil (x);** | округлює у більший бік | |
| **floor (x);** | округлює в менший бік | |
| **x + = y;** | додасть "y" до "x", або: | x=x+y |
| **x – = y;** | відніме "y" з "x", або: | x=x-y |
| **x * = y;** | множення "x" на "y", або: | x=x*y |
| **x /= y;** | розділить "x" на "y", або: | x=x/y |
| **x + +;** | збільшить "x" на 1, або: | x=x+1 |
| **x – -;** | зменшить "x" на 1, або: | x=x-1 |

## Послідовний порт. Оператори бібліотеки Serial

Підключивши Arduino до комп'ютера, необхідно дізнатися, до якого порту вона приєдналася. Це можна зробити, відкривши «Диспетчер приладів» Windows:




У нашому випадку – це COM3. Тепер необхідно встановити прапорець у меню Arduino: «Інструменти» – «Порт» – вибрати COM3.

Тепер роздивимося, як використовується «Монітор порту». Це меню знаходиться також в «Інструменти». Натискаємо:


**Serial** – об'єкт бібліотеки Serial для роботи з послідовним портом (COM-портом).

**Serial.begin (<швидкість>);** відкрити порт.

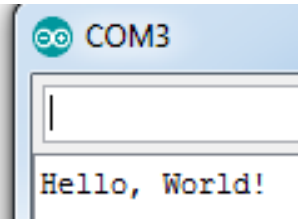
Наприклад, щоб відкрити порт на 9600 БОД (значення по замовчуванню, як на малюнку):

**Serial.begin(9600);** Увага! Швидкість, встановлена в **begin()** повинна дорівнювати швидкості монітору порту (правий нижній кут на самому моніторі – див. малюнок вище).

**Serial.print ();** – виведення в порт. Змінні та цифри пишемо безпосередньо, як є, текстові повідомлення беремо в лапки " ".

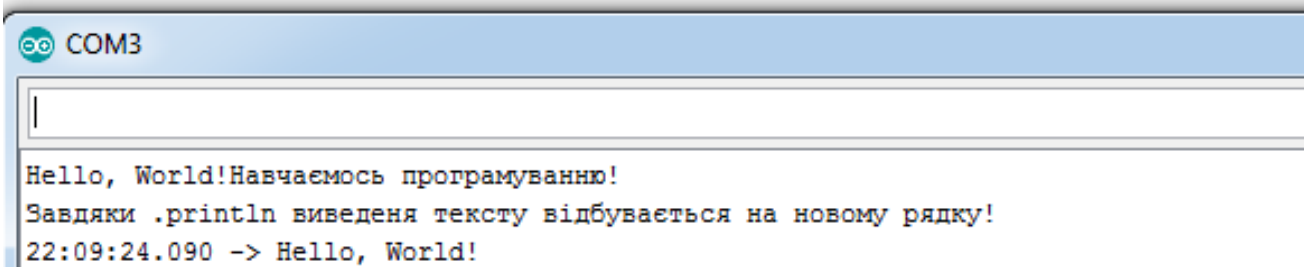
```
void setup() {
  Serial.begin(9600);
  Serial.print("Hello, World!");

}
```




**Serial.println ();** – виведення з переходом на новий рядок.

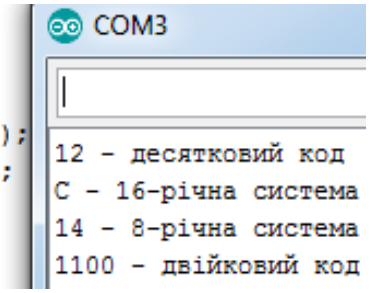
```
void setup() {
  Serial.begin(9600);
  Serial.print("Hello, World!");
  Serial.println("Навчаємось програмуванню!");
  Serial.println("Завдяки .println виведеня тексту відбувається на новому рядку!");
  }
```




**Serial.println (Name, n);** – виведення змінної **Name** (типу **float**), де **n** – кількість десяткових чисел після коми. Знімемо прапорець «Показати відмітки часу» у моніторі порту та запустимо такий код:

```
void setup() {
  Serial.begin(9600);
  Serial.print("Найкраща оцінка ");Serial.print(12);Serial.println(" балів");
  String var = "Це просто текст!";
  Serial.println(var);
  float pi = 3.141592653589793238;
  Serial.println(pi, 3);
  }
```

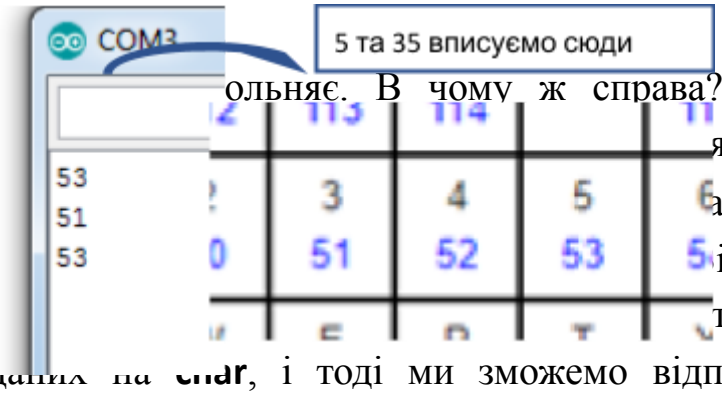
```
void setup() {
  Serial.begin(9600);
  Serial.print(12,DEC);Serial.println(" - десятковий код");
  Serial.print(12,HEX);Serial.println(" - 16-річна система");
  Serial.print(12,OCT);Serial.println(" - 8-річна система");
  Serial.print(12,BIN);Serial.println(" - двійковий код");
  }
```




З виведенням все зрозуміло, а ось з прийняттям даних все складніше. Коли ми відправляємо на Ардуіно дані, вони складаються в буфер, і чекають поки Arduino їх прочитає. Обсяг буфера складає 64 байти. Щоб тисячі разів в секунду не читати порожній буфер, є функція перевірки буфера, **Serial.available().** Вона повертає число байт, які лежать в буфері. Тобто ми можемо використовувати умову "якщо в буфері більше 0 байт", і написати код, який буде виконуватися поки в буфері щось є. Тепер можна спробувати прийняти дані. Створимо змінну типу **integer**, і надамо їй дані з порту за допомогою функції читання **Serial.read().** І відразу відішлемо назад те, що отримали за допомогою вже відомої нам функції **Serial.println().** Тобто Arduino повинна відправити нам те, що ми відправили їй.

Перевіряємо. Відправляю 5 і отримую 53. Відправляю 35 і отримую цілих 2 числа, 51 і 53.

```
void setup() {
  Serial.begin(9600);
  }
void loop() {
  if (Serial.available() >0) {
    int in_data = Serial.read();
    Serial.println(in_data);
  }
}
```

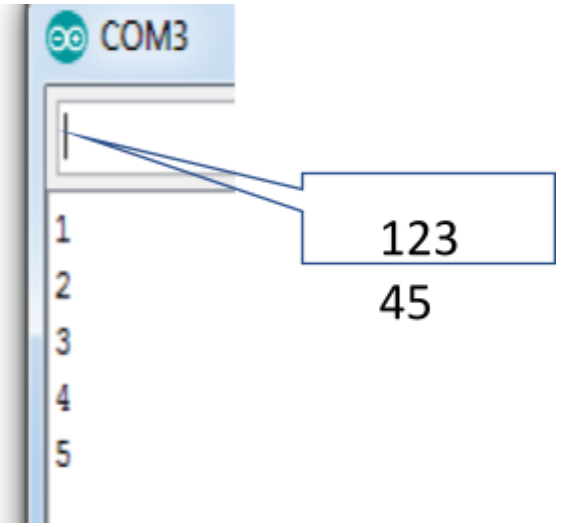



ольняє. В чому ж справа? Функція язку з тим, ндартною ідправимо ти символ, можна змінити тип даних на **char**, і тоді ми зможемо відправляти і приймати символи. Але це все – символи, а не чисельні значення. Тобто

виконувати з ними математичні операції не можна. А якщо ми хочемо приймати саме цифри?

Тоді потрібно відразу переводити дані з порту в чисельне значення. Для цього потрібно відняти символ 0, в одинарних лапках. Це потрібно просто запам'ятати.

```
void setup() {
  Serial.begin(9600);
  }
void loop() {
  if (Serial.available() >0) {
    int in_data = Serial.read()- '0';
    Serial.println(in_data);
  }
}
```



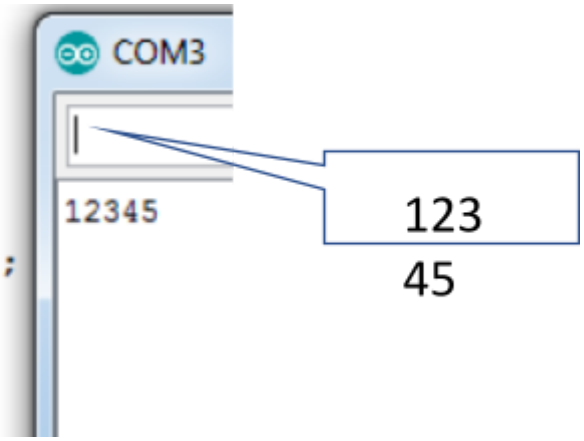

Тепер ми відправляємо і отримуємо саме цифри! Отримані таким чином цифри можна використовувати в математичних та логічних операціях. Але, на жаль, тільки однозначні. Число, що складається більше ніж з однієї цифри, розбивається на шматочки, і виходить, що наша змінна приймає значення тільки останньої цифри відправленого числа. Тобто відправили 12345, і змінна перезаписується 5 разів, і врешті-решт приймає значення 5. Для того, щоб прийняти все число цілком, є функція: **Serial.parseInt();**

Ця функція почекає, поки прийдуть всі відправлені вами цифри і складе з них число, ніяких нулів віднімати більше не потрібно. Тепер наша змінна прийме число повністю, і можна працювати з ним далі. У зв’язку з тим, що ця функція чекає нові цифри, ви можете помітити невелику затримку виконання. Ще є функція:

**Serial.flush(); -** вона очищає буфер порту.

Приклад використання функції **Serial.parseInt():**

```
void setup() {
  Serial.begin(9600);
  }
void loop() {
  if (Serial.available() >0) {
    int in_data = Serial.parseInt();
    Serial.println(in_data);
  }
}
```




**Виділимо головне з описаного вище:**

Дані з комп'ютера попадають в буфер з об'ємом 64 байти і чекають обробки.

**Serial.available();** – перевірити буфер на наявність вхідних даних.

**Serial.read();** – читати вхідні дані в форматі символів, згідно з ASCII.

**Serial.read() - '0';** – читання даних у цілочисельному форматі по одній цифрі.

**Serial.parseInt();** – читання даних у цілочисельному форматі цілком всього числа.

**Serial.flush();** – очистити буфер порту.

Таблиця 2. ASCII кодування клавіатури

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Escape 27 | | F1 112 | F2 113 | F3 114 | | F4 115 | F5 116 | F6 117 | F7 118 | F8 119 | F9 120 | F10 121 | F11 122 | F12 123 | Print Screen | Scroll Lock 145 | Pause 19 | | | |
| ` 192 | 1 49 | 2 50 | 3 51 | 4 52 | 5 53 | 6 54 | 7 55 | 8 56 | 9 57 | 0 48 | - 189 | =+ 187 | Back Space 8 | Insert 45 | Home 36 | Page Up 33 | Num Lock 144 | / доп. 111 | * доп. 106 | + доп. 107 |
| Tab 9 | Q 81 | W 87 | E 69 | R 82 | T 84 | Y 89 | U 85 | I 73 | O 79 | P 80 | [ 219 | ] 221 | | Delete 46 | End 35 | Page Down 34 | 7 доп. 103 | 8 доп. 104 | 9 доп. 105 | |
| Caps Lock 20 | A 65 | S 83 | D 68 | F 70 | G 71 | H 72 | J 74 | K 75 | L 76 | ; 186 | ' 222 | Enter 13 | | | | | 4 доп. 100 | 5 доп. 101 | 6 доп. 102 | Enter доп. 13 |
| Shift 16 | Z 90 | X 88 | C 67 | V 86 | B 66 | N 78 | M 77 | ,< 188 | .> 190 | / 191 | Shift 16 | \| 220 | | | Up 38 | | 1 доп. 97 | 2 доп. 98 | 3 доп. 99 | |
| Ctrl 17 | win | Alt 18 | Space Bar 32 | | | | | | Alt 18 | win | list | Ctrl 17 | | Left 37 | Down 40 | Right 39 | Ins/0 доп. 45/96 | | Del/. доп. 46/110 | |