|  |  |
| --- | --- |
| **Arduino Introduction** | |
| **1.Serial Basics** | |
| What is the Overview of Serial?   * It is used for communication purpose between Arduino and   Other devices (i.e. computer etc).   * It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. | |
| **Functions** | |
| **begin()** | **Purpose:** Sets the data rate in bits per second (baud) for serial data transmission. |
|  | **Syntax:** Serial. begin(baudrate)  baudrate defines the speed of the byte communication between Arduino and other device |
| **Example** | void setup() {  Serial. begin(9600); // opens serial port, sets data rate to //9600 bps  }  void loop() {} |
|  | |
| **available()** | **Purpose:** It generates number of bytes (characters) available for reading from serial port. It holds maximum of 64 bytes. It reads from already stored receiver buffer. |
|  | **Syntax:** Serial. available() |
| **Example** | void setup() {  Serial. begin(9600);  }  void loop() {  while (Serial. available())  {  Serial.println(Serial.readString());  }  } |
|  |  |
| **availableForWrite()** | **Purpose:** Get the number of bytes (characters) available for writing in the serial buffer without blocking the write operation. |
|  | **Syntax:** Serial. availableForWrite() |
| **Example** | void setup() {  Serial. begin(9600);  Serial.println(Serial.availableForWrite());  }  void loop() {} |
|  |  |
| **end()** | **Purpose:** Disables serial communication, allowing the RX and TX pins to be used for general input and output.  To re-enable serial communication, call Serial. begin (). |
|  | **Syntax:** Serial. end() |
| **Example** | void setup() {  Serial. begin(9600); // opens serial port, sets data rate to 9600 bps  Serial. end();  }  void loop() {} |
|  |  |
| **find()** | Purpose: Serial.find() reads data from the serial buffer until the target string of given length is found. The function returns true if target string is found, false if it times out. |
|  | Syntax: Serial. find(target)  target : the string to search for (char) |
| Example | void setup() {  Serial. begin(9600);  }  void loop() {  while(Serial. available()>0)  Serial.readString();  {  if (Serial. find("abc"))  {  Serial.println("found");  }  }  } |
|  |  |
| **findUntil()** | Purpose: Serial.findUntil () reads data from the serial buffer until a target string of given length or terminator string is found.  The function returns true if the target string is found, false if it times out. |
|  | Syntax: Serial.findUntil (target, terminal)  target: the string to search for (char)  terminal: the terminal string in the search (char) |
| Example | void setup() {  Serial. begin(9600);  }  void loop() {  while(Serial. available()>0)  Serial.readString();  {  if (Serial.findUntil("abc","ghi"))  {  Serial.println("found");  }  }  } |
|  |  |
| **flush()** | Purpose: Waits for the transmission of outgoing serial data to complete. |
|  | Syntax: Serial. flush () |
| Example | void setup() {  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps  Serial.flush();  }  void loop() {} |
|  |  |
| **read()** | Purpose: Reads incoming serial data.  read () inherits from the Stream utility class. |
|  | Syntax: Serial. read () |
| Example | int a=0;  void setup() {  Serial. begin(9600);  }  void loop() {  if (Serial. available()>0)  {  a=Serial.read();  Serial.println(a,DEC);  }  } |
|  |  |
| **readBytes()** | Purpose: It reads characters from the serial port into a buffer. The function terminates if the determined length has been read, or it times out. Serial.readBytes () returns the number of characters placed in the buffer.  A 0 means no valid data was found. |
|  | Syntax: Serial.readBytes (buffer, length)  buffer: the buffer to store the bytes in (char[] or byte[])  length: the number of bytes to read (int) |
| Example |  |
|  |  |
| **readBytesUntil()** | Purpose: Serial.readBytesUntil () reads characters from the serial buffer into an array. The function terminates if the terminator character is detected, the determined length has been read, or it times out  Serial.readBytesUntil () returns the number of characters read into the buffer. A 0 means no valid data was found. |
|  | Syntax: Serial.readBytesUntil (character, buffer, length)  character: the character to search for (char)  buffer: the buffer to store the bytes in (char[] or byte[])  length: the number of bytes to read (int) |
| Example |  |
|  |  |
| **write()** | Purpose: Writes binary data to the serial port. This data is sent as a byte or series of bytes; to send the characters representing the digits of a number use the print () function instead. |
|  | Syntax: Serial. write(val)  Serial. write(str)  Serial. write (buf, len)  val: a value to send as a single byte  str: a string to send as a series of bytes  buf: an array to send as a series of bytes |
| Example | void setup(){  Serial. begin(9600);  }  void loop(){  //Serial. write(45);  Serial. write("hello");  } |
|  |  |
| **print()** | Purpose: It prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. |
|  | Syntax: Serial. print (Val) (or) Serial. print (Val, format)  Example:  Serial. Print (78) gives "78"  Serial. print ("Hello world.") gives "Hello world."  Serial. print (78, BIN) gives "1001110" |
| Example | void setup() {  Serial. begin(9600);  }  void loop() {  {  Serial. print(" hello ");  delay(800);  }} |
|  |  |
| **println()** | Purpose: Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n'). |
|  | Syntax: Serial. print (Val) (or) Serial. print (Val, format) |
| Example | void setup() {  Serial. begin(9600);  }  void loop() {  {  Serial. println("hello");  delay(800);  }} |
|  |  |
| **peek()** | Purpose Returns the next byte (character) of incoming serial data without removing it from the internal serial buffer. That is, successive calls to peek () will return the same character, as will the next call to read ().  peek () inherits from the Stream utility class. |
|  | Syntax: Serial. peek () |
| Example | void setup(){  Serial.begin(9600);  Serial.println( Serial. peek());  }  void loop(){    } |
|  |  |
| **setTimeout()** | Purpose: Serial.setTimeout () sets the maximum milliseconds to wait for serial data when using serial.readBytesUntil () or serial.readBytes (). It defaults to 1000 milliseconds.Serial.setTimeout () inherits from the Stream utility class. |
|  | Syntax :Serial.setTimeout (time) |
| Example |  |
|  |  |
| **parseFloat()** | Purpose: Serial.parseFloat () returns the first valid floating point number from the Serial buffer. Characters that are not digits (or the minus sign) are skipped. parseFloat () is terminated by the first character that is not a floating point number. |
|  | Syntax: Serial.parseFloat() |
| Example |  |
|  |  |
| **parseInt()** | Purpose: It looks for the next valid integer in the incoming serial stream.parseInt () inherits from the Stream utility class. |
|  | Syntax: stream.parseInt (char skipchar)  skipchar-- use to skip the indicated char in the search. Used for example to skip thousands divider. |
| Example |  |
|  |  |
| **readString()** | Purpose: reads characters from a stream into a string.  The function terminates if it times out |
|  | Syntax: Serial.readString () |
| Example | void setup() {  Serial. begin(9600);  }  void loop() {  while (Serial. available())  {  Serial.println(Serial.readString());  }  } |
|  |  |

|  |  |
| --- | --- |
| **2.Time** | |
| What is the Overview of Time?   * The Time library adds timekeeping functionality to Arduino with or without external timekeeping hardware. | |
| **Functions** | |
| **delay()** | Purpose: Pauses the program for the amount of time (in milliseconds) specified as parameter. |
|  | Syntax: delay (ms)  ms: the number of milliseconds to pause (unsigned long) |
| Example | int ledPin = 13; // LED connected to digital pin 13  void setup()  {  pinMode(ledPin, OUTPUT); // sets the digital pin as output  }  void loop()  {  digitalWrite(ledPin, HIGH); // sets the LED on  delay(10000); // waits for a second  digitalWrite(ledPin, LOW); // sets the LED off  delay(10000); // waits for a second  } |
|  | |
| **delayMicroseconds()** | Purpose: Pauses the program for the amount of time (in microseconds) specified as parameter. |
|  | Syntax: delayMicroseconds (us)  us: the number of microseconds to pause (unsigned int) |
| Example | int op = 13; // digital pin 13  void setup()  {  pinMode(op, OUTPUT); // sets the digital pin as output  }  void loop()  {  digitalWrite(op, HIGH); // sets the pin on  delayMicroseconds(5000); // pauses for 50 microseconds  digitalWrite(op, LOW); // sets the pin off  delayMicroseconds(5000); // pauses for 50 microseconds |
|  | |

|  |  |
| --- | --- |
| **micros()** | Purpose: It returns the number of microseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 70 minutes.  On 16 MHz Arduino boards (e.g. Duemilanove and Nano), this function has a resolution of four microseconds (i.e. the value returned is always a multiple of four). |
|  | Syntax**:** time = micros () |
| Example | unsigned long time;  void setup(){  Serial. begin(9600);  }  void loop(){  Serial. print("Time: ");  time = micros();  Serial.println(time); //prints time since program started  delay(1000); // wait a second so as not to send massive amounts of data  } |
|  | |
| **millis()** | Purpose: Returns the number of milliseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days. |
|  | Syntax: time = millis() |
| Example | unsigned long time;  void setup(){  Serial. begin(9600);}  void loop(){  Serial. print("Time: ");  time = millis();  Serial.println(time); //prints time since program started  delay(1000); // wait a second so as not to send massive amounts of data  } |
|  | |

|  |  |
| --- | --- |
| **3.Math** | |
| What is the Overview of math?   * Math functions are use to perform mathematical operations to make the calculations much easier. | |
| **Functions** | |
| **abs()** | Purpose: Calculate the absolute value of the number. |
|  | Syntax: : abs(x)//where x is a number  x: if x is greater than or equal to 0.  -x: if x is less than 0. |
| Example | void setup() {  Serial. begin(9600);  }  void loop() {  while (Serial. available()>0)  {  int x=Serial. read();  Serial. print(abs(x));  }    } |
|  | |
| **constrain()** | Purpose: Constrains a number to be within a range. |
|  | Syntax: **:** constrain(x, a, b)  x: the number to constrain, all data types  a: the lower end of the range, all data types  b: the upper end of the range, all data types |
| Example |  |
|  | |
| **Map()** | Purpose: Re-maps a number from one range to another. That is, a value of fromLow would get mapped to toLow, a value of fromHigh to toHigh, values in-between to values in-between, etc..  Note that the "lower bounds" of either range may be larger or smaller than the "upper bounds" so the map() function may be used to reverse a range of numbers  Example: y = map(x, 1, 50, 50, 1);  The function also handles negative numbers well  Example: y = map(x, 1, 50, 50, -100); |
|  | Syntax: map (Value, fromLow, fromHigh, toLow, toHigh)  Value: the number to map  fromLow: the lower bound of the value’s current range  fromHigh: the upper bound of the value’s current range  toLow: the lower bound of the value’s target range  toHigh: the upper bound of the value’s target range |
| Example |  |
|  | |
| **max()** | Purpose: Calculates the maximum of two numbers. |
|  | Syntax: max(x, y)  x: the first number, any data type  y: the second number, any data type |
| Example | void setup() {  Serial. begin(9600);  }  int a=30;  int b=20;  void loop() {  {  int c=max(a,b);  Serial.println(c,DEC);    }  } |
|  | |
| **min()** | Purpose: Calculates the minimum of two numbers. |
|  | Syntax: **:** min(x, y)  x: the first number, any data type  y: the second number, any data type |
| Example | void setup() {  Serial. begin(9600);  }  int a=30;  int b=20;  void loop() {  {  int c=min(a,b);  Serial.println(c,DEC);    }  } |
|  | |
| **pow()** | Purpose: Calculates the value of a number raised to a power.  Pow() can be used to raise a number to a fractional power. This is useful for generating exponential mapping of values or curves. |
|  | Syntax: pow(base, exponent)  base: the number (float)  exponent: the power to which the base is raised (float) |
| Example |  |
|  | |
| **sq()** | Purpose: Calculates the square of a number: the number multiplied by itself. |
|  | Syntax: **:** sq(x)  x: the number, any data type |
| Example | void setup() {  Serial.begin(9600);  }  int a=30;  void loop() {  {  int c=sq(a);  Serial.println(c,DEC);  }  } |
|  | |
| **sqrt()** | Purpose Calculates the square root of a number. |
|  | Syntax: sqrt(x)  x: the number, any data type |
| Example | void setup() {  Serial. begin(9600);  }  int a=9;  void loop() {  {  int c=sqrt(a);  Serial.println(c,DEC);  }  } |
|  | |

|  |  |
| --- | --- |
| **3.Data Conversion** | |
| What is the Overview of Data Conversion?   * It is used to convert variables from one data type to another. | |
| **Functions** | |
| **byte()** | Purpose: Converts a value to the byte data type. |
|  | Syntax: byte(x)  x:a value of any type |
| Example |  |
|  | |
| **char()** | Purpose: Converts a value to the char data type. |
|  | Syntax: char(x)  x:a value of any type |
| Example |  |
|  | |
| **float()** | Purpose: Converts a value to the float data type. |
|  | Syntax: float(x)  x:a value of any type |
| Example |  |
|  | |
| **int()** | Purpose: Converts a value to the int data type. |
|  | Syntax: int(x)  x:a value of any type |
| Example |  |
|  | |
| **long()** | Purpose: Converts a value to the longdata type. |
|  | Syntax: long(x)  x:a value of any type |
| Example |  |
|  | |
| **word()** | Purpose: Converts a value to the word data type. |
|  | Syntax: word(x)  word(h,1)  x: a value of any type  h: the high-order (leftmost) byte of the word  l: the low-order (rightmost) byte of the word |
| Example |  |
|  | |
|  | |

|  |  |  |
| --- | --- | --- |
| **4.Character** | | |
| What is the Overview of character?   * A data type that takes up 1 byte of memory that stores a character value. * All Data entered into computers as character, which includes letters, digits, and symbols. * The char data type is a signed data type, encodes numbers from -128 to 127. | | |
| **Functions** |  | |
| **isAlpha()** | **Purpose:**   * If a Char(Character) isAlpha(‘A’). * Returns the variable is true if thisChar contains a letter. | |
|  | *Syntax:* **isAlpha(thisChar)**   * Send a byte of data from the Arduino to a personal computer and the Alphabet results. * thisChar defines the character variable in which the user defines. | |
| *Example* | void setup()  {  Serial.begin(9600);  }  void loop()  {  if(Serial. available())  {  char thisChar=Serial. read();  Serial.print("The Character is:\t");  Serial.write(thisChar)  if (isAlpha(thisChar))  {  Serial.println("\n Is alphabetic");  }  else  {  Serial.println("\n Not a Alphabet");  }  }  } | |
|  | | |
| **isAlphaNumeric()** | **Purpose:**   * If a char is alphanumeric (that is a letter or a numbers). * Returns true if thisChar contains either a number or a letter. | |
|  | *Syntax:***isAlphaNumeric(thisChar)**   * Send a byte of data from the Arduino to a personal computer and the Alpha Numeric results. * thisChar defines the character in which the user defines the Alphanumeric. | |
| *Example* | void setup()  {  Serial.begin(9600);  }  void loop()  {  if(Serial. available())  {  char thisChar=Serial. read();  Serial.print("The Character is:\t");  Serial.write(thisChar);  if (isAlphaNumeric(thisChar))  {  Serial.println("\n Is a Alpha Numeric");  }  else  {  Serial.println("\n Not an Alpha Numeric");  }  }  } | |
|  | | |
| **isAscii()** | **Purpose:**   * If a char is Ascii. * Returns true if thisChar contains an Ascii character. | |
|  | *Syntax:* **isAscii(thisChar)**   * Send a byte of data from the Arduino to a personal computer and the ASCII result. * thisChar defines the character in which the user defines as ASCII Character. | |
| *Example* | void setup()  {  Serial.begin(9600);  }  void loop()  {  if(Serial.available())  {  char thisChar=Serial. read();  Serial.print("The Character is:\t");  Serial.write(thisChar);  Serial.print("\n ASCII Value: ");  Serial.println(thisChar);  if (isAscii(thisChar))  {  Serial.println("\n Is an ASCII");  }  else  {  Serial.println("\n Not an ASCII");  }  }  } | |
| **isControl()** | **Purpose:**   * If a char is a control character. * Returns true if thisChar is a control character. * Controls the devices to the nature of the object. | |
|  | *Syntax:* ***isControl(thisChar)***   * Send a byte of data from the Arduino to a personal computer and the Control Character results. * thisChar defines the character in which the user defines the Control Character. | |
| *Example* | void setup() {  Serial.begin(9600);  }  void loop()  {  if(Serial. available())  {  char thisChar=Serial. read();  Serial.print("The Character is:\t");  Serial.write(thisChar);  if (isControl(thisChar))  {  Serial.println("\n Is a Control Character");  }  else {  Serial.println("\n Not an Control Character");  }  }  } | |
| **Isdigit()** | **Purpose:**   * If a Character is a number. * Returns true thisChar is a number. | |
|  | *Syntax:* **isdigit(thisChar)**   * Send a byte of data from the Arduino to a personal computer and the Digits results. * thisChar defines the character in which the user defines the Number. | |
| *Example* | void setup()  {  Serial.begin(9600);  }  void loop()  {  if(Serial. available())  {  char thisChar=Serial. read();  Serial.print("The Character is:\t");  Serial.write(thisChar);  if (isDigit(thisChar))  {  Serial.println("\n Is a Numeric Digit");  }  else  {  Serial.println("\n Not an Numeric Digit");  }  }  } | |
|  | | |
| **isGraph()** | **Purpose:**   * If a Character is printable with some content (space is printable but has no content). | |
|  | *Syntax:* **isGraph(thisChar)**   * Send a byte of data from the Arduino to a personal computer and graph the result. * thisChar defines the character is printable. | |
| *Example* | void setup()  {  Serial.begin(9600);  }  void loop()  {  if(Serial. available())  {  char thisChar=Serial. read();  Serial.print("The Character is:\t");  Serial.write(thisChar);  if (isGraph(thisChar))  {  Serial.println("\n Is a Graph");  }  else  {  Serial.println("\n Not an Graph");  }  }  } | |
|  | | |
| **isHexadecimalDigit()** | **Purpose:**   * If a Character is an hexadecimal digit(A-F,0-9). * Returns the character contains an hexadecimal value. | |
|  | *Syntax:* **isHexadecimalDigit(thisChar)**   * Send a byte of data from the Arduino to a personal computer and the hexadecimal value results. * thisChar defines the character is an hexadecimal digit. | |
| *Example* | void setup()  {  Serial.begin(9600);  }  void loop() {  if(Serial. available())  {  char thisChar=Serial. read();  Serial.print("The Character is:\t");  Serial.write(thisChar);  if (isHexadecimalDigit(thisChar))  {  Serial.println("\n Is a HexadecimalDigit");  }  else  {  Serial.println("\n Not an HexadecimalDigit");  }  }  } | |
|  | | |
| **isLowerCase()** | **Purpose:**   * If a Character is defined as lower case. * Return true if thisChar contains a lower case. | |
|  | *Syntax:* **isLowerCase(thisChar)**   * Send a byte of data from the Arduino to a personal computer and the Lower Case value results. * thisChar defines the character is a lower case. | |
| *Example* | void setup()  {  Serial.begin(9600);  }  void loop() {  if(Serial. available())  {  char thisChar=Serial. read();  Serial.print("The Character is:\t");  Serial.write(thisChar);  if (isLowerCase(thisChar))  {  Serial.println("\n Is a LowerCase");  }  else  {  Serial.println("\n Not an LowerCase");  }  }  } | |
|  | | |
| **isUpperCase()** | **Purpose:**   * If a Character is defined as upper case. * Return true if thisChar contains a upper case. | |
|  | *Syntax:* **isUpperCase(thisChar)**   * Send a byte of data from the Arduino to a personal computer and the Upper Case value results. * thisChar defines the character is an upper case. | |
| *Example* | void setup()  {  Serial.begin(9600);  }  void loop() {  if(Serial. available())  {  char thisChar=Serial. read();  Serial.print("The Character is:\t");  Serial.write(thisChar);  if (isUpperCase(thisChar))  {  Serial.println("\n Is a UpperCase");  }  else  {  Serial.println("\n Not an UpperCase");  }  }  } | |
|  | | |
| **isPrintable()** | **Purpose:**   * If a character is printable, any character that produce an output, even a blank space. * Returns true if thisChar is printable. * This function can be used in all the above function to print the character. | |
|  | *Syntax:* **isAlpha(thisChar)**   * Send a byte of data from the Arduino to a personal computer and the Printable values results. * thisChar defines the character is printable. | |
| *Example* | void setup()  {  Serial.begin(9600);  }  void loop() {  if(Serial. available())  {  char thisChar=Serial. read();  Serial.print("The Character is:\t");  Serial.write(thisChar);  if (isPrintable(thisChar))  {  Serial.println("\n Is a Printable");  }  else  {  Serial.println("\n Not an Printable");  }  }  } | |
|  | | |
| **isPunct()** | **Purpose:**   * Checks the char or the line is punctuated or terminated. * Returns true if the thisChar is terminated correctly. | |
|  | *Syntax:* **isPunct(thisChar)**   * Send a byte of data from the Arduino to a personal computer and the Punctutation values results. * thisChar defines the character is a punctuation. | |
| *Example* | void setup()  {  Serial.begin(9600);  }  void loop() {  if(Serial. available())  {  char thisChar=Serial. read();  Serial.print("The Character is:\t");  Serial.write(thisChar);  if (isPunct(thisChar))  {  Serial.println("\n Is a Punct");  }  else  {  Serial.println("\n Not an Punct");  }  }  } | |
|  | | |
| **isSpace()** | **Purpose:**   * Checks the character is a spaced character. * Returns if thisChar contains the space character. | |
|  | *Syntax:* **isSpace(thisChar)**   * Send a byte of data from the Arduino to a personal computer and the space values results. * thisChar defines the character is a space character. | |
| *Example* | void setup()  {  Serial.begin(9600);  }  void loop() {  if(Serial. available())  {  char thisChar=Serial. read();  Serial.print("The Character is:\t");  Serial.write(thisChar);  if (isSpace(thisChar))  {  Serial.println("\n Is a Space");  }  else  {  Serial.println("\n Not an Space");  }  }  } | |
|  | | |
| **isWhitespace()** | **Purpose:**   * If a char is a white space, that is formfeed (‘\f’), new line(‘\n’), carriage return (‘\r’), horizontal tab (‘\t’) and vertical tab (‘\v’). Returns true if thisChar contains a white space. | |
|  | *Syntax:* **isWhitespace(thisChar)**   * Send a byte of data from the Arduino to a personal computer and the Whitespace values results. * thisChar defines the character is a white space. | |
| *Example* | void setup()  {  Serial.begin(9600);  }  void loop() {  if(Serial. available())  {  char thisChar=Serial. read();  Serial.print("The Character is:\t");  Serial.write(thisChar);  if (isWhitespace(thisChar))  {  Serial.println("\n Is a Whitespace");  }  else  {  Serial.println("\n Not an Whitespace");  }  }  } | |
|  | | |
| **5.Bits and Bytes** | | |
| What is the Overview of Bits and bytes?   * Computers use Bits to represent information in digital form. * A computer bit is a binary value of (0’s and 1’s). * A Byte is a fixed length sequence of bits. * Modern computers using bytes to increase the data processing efficiency of network. | | |
| **Functions** |  | |
| **bit()** | **Purpose:**   * A bit represents the information digital form as (o’s and 1’s). * A bit is smallest unit of information. | |
|  | *Syntax:* **bit(n)** | |
| *Example* | void setup() {Serial.begin(9600);Serial.setTimeout(50);}void loop(){while(Serial. available()){String content="";byte num=Serial.parseInt();char thisByte=num;content=" The bit value of " + String(num) + " is ";Serial.print(content);Serial.println(bit(num));Serial.print(bitRead(thisByte,0));Serial.print(bitRead(thisByte,1));Serial.print(bitRead(thisByte,2));Serial.print(bitRead(thisByte,3));Serial.print(bitRead(thisByte,4));Serial.print(bitRead(thisByte,5));Serial.print(bitRead(thisByte,6));Serial.println(bitRead(thisByte,7));}} | |
| **bitClear()** | **Purpose:**   * A representation of bit that clears the bit of a numeric variable. | |
|  | *Syntax:* **bitClear(x,n)**   * X – Numeric variable whose bit to clear * N – Which bit to clear, starting from LSB to MSB. | |
| *Example* | void setup()  {  pinMode(13, OUTPUT);  Serial.begin(9600);  }  void loop() {  if (Serial. available())  {  int state=Serial.parseInt();  if (state==1)  {  digitalWrite(13, HIGH);  Serial.println("LED turned ON");  }  if (state==2)  {  digitalWrite(13, LOW);  Serial.println("LED turned OFF");  }  }  } | |
|  | | |
| **bitRead()** | **Purpose:**   * Reads the bit number. | |
|  | *Syntax:*  **bitRead(x,n)**   * X – Number from which to read * N – Which bit to read, starting from LSB to MSB. | |
| *Example* | void setup() {Serial.begin(9600);Serial.setTimeout(50);}void loop(){while(Serial. available()){String content="";byte num=Serial.parseInt();char thisByte=num;content=" The bit value of " + String(num) + " is ";Serial.print(content);Serial.println(bit(num));Serial.print(bitRead(thisByte,0));Serial.print(bitRead(thisByte,1));Serial.print(bitRead(thisByte,2));Serial.print(bitRead(thisByte,3));Serial.print(bitRead(thisByte,4));Serial.print(bitRead(thisByte,5));Serial.print(bitRead(thisByte,6));Serial.println(bitRead(thisByte,7));} } | |
|  | | |
| **bitSet()** | **Purpose:**   * Sets a bit of a numeric variable. | |
|  | *Syntax:* **bitSet(x,n)**   * X – Number from which to Set. * N – Which bit to set, starting from LSB to MSB. | |
| *Example* | void setup()  {  pinMode(13, OUTPUT);  Serial.begin(9600);  }  void loop() {  if (Serial. available())  {  int state=Serial.parseInt();  if (state==1)  {  digitalWrite(13, HIGH);  Serial.println("LED turned ON");  }  if (state==2)  {  digitalWrite(13, LOW);  Serial.println("LED turned OFF");  }  }  } | |
|  | | |
| **bitWrite()** | **Purpose:**   * Writes a bit of a numeric variable | |
|  | *Syntax:* **bitWrite(x,n,b)**   * X – Number from which to write * N – Which bit to write, starting from LSB to MSB. * B – the value to the bit (0 or 1) | |
| *Example* |  | |
|  |  | |
| **highByte()** | **Purpose:**   * Extracts the high-order (MSB) byte of a word or the second lowest byte of larger data type. | |
|  | *Syntax:* **highByte(x)**   * x – a value of any type | |
| *Example* |  | |
| **lowByte()** | **Purpose:**   * Extracts the Low-order (LSB) byte of a variable. | |
|  | *Syntax:* **lowByte(x)**   * x – a value of any type | |
| *Example* |  | |
|  | | |
| **6.Data Types** | | |
| What is the Overview of data types?   * A data type in programming language is an attribute of an data which tells the computer about the kind of data. * This involves setting constraints on datum. | | |
| **Functions** |  | |
| **String()** | **Purpose:**   * Constructs an instance of string class. * Constructing a string from a number contains the ASCII number. | |
|  | *Syntax:* **char String[val]**   * Send a byte of data from the Arduino to a personal computer that allows a string data type like string, char, int, long, unsigned int, unsigned long, float, double. | |
| *Example* | void setup() {  char my\_str[] = "Hello";  Serial.begin(9600);  Serial.println(my\_str);  }  void loop() {  } | |
|  | | |
| **array** | **Purpose:**   * An array is a collection of variables that are accessed with an indexed number. | |
|  | *Syntax:*  **int a[]={}**   * Send a byte of data from the Arduino to a personal computer that allows a variable with an indexed number. | |
| *Example* | void setup() {  Serial.begin(9600);  int my\_array[4]={23,45,35,12};  for(int i=0;i<4;i++)  {  Serial.println(my\_array[i]);  }  }  void loop()  {    } | |
|  | | |
| **boolean** | **Purpose:**   * A Boolean holds one of two values, true or false. | |
|  | *Syntax:* | |
| *Example* |  | |
|  | | |
| **byte** | **Purpose:**   * A byte stores an 8-bit unsigned number, from 0 to 255. | |
|  | *Syntax:* | |
| *Example* |  | |
|  | | |
| **char** | **Purpose:**   * A data type that takes up 1 byte of memory that stores a character | |
|  | *Syntrax:* **char var=val;**   * This character show the logic can be used in either of Alpha Numeric characters. | |
| *Example* |  | |
| **double** | **Purpose:**   * Double precision floating point number. * It is same as the float, with no gain in precision. | |
|  | *Syntax:* **double var=val;**   * The integer value is an decimal value. | |
| *Example* |  | |
|  | | |
| **float** | **Purpose:**   * Data type for floating point numbers that has a decimal point. * Floating point numbers are often used as analog and continuous values, they have greater resolution than integers. | |
|  | *Syntax:* **float var=val;**  var – float variable  val – the value assign to that variable. | |
| *Example* |  | |
|  | | |
| **Int** | **Purpose:**   * The int stores an integer value the variable. | |
|  | *Syntax:* **int var=val;**  var – variable name  val – the value assign to that variable. | |
| *Example* |  | |
|  | | |
| **long** | **Purpose:**   * Long variables are extended size variables for number storage. | |
|  | *Syntax:* **long var=val;**  var – long variable name  val – the value assign to that variable. | |
| *Example* |  | |
|  | | |
| **short** | **Purpose:**   * A short is a 16-bit data type | |
|  | *Syntax:* **short var=val;**  var – short variable name  val – the value assign to that variable. | |
| *Exmple* |  | |
| **string** | **Purpose:**   * Constructs an instance of string class. * Constructing a string from a number contains the ASCII number. | |
|  | *Syntax:* **char String[val]**   * Send a byte of data from the Arduino to a personal computer that allows a string data type like string, char, int, long, unsigned int, unsigned long, float, double. | |
| *Example* |  | |
|  | | |
| **unsigned Char** | **Purpose:**   * An unsigned data type that occupies 1 byte of memory. * Same as byte data type. | |
|  | *Syntax:* **unsigned char mychar=240;** | |
| *Example* |  | |
|  | | |
| **Unsigned int** | **Purpose:**   * Stores the integer value to the variable. | |
|  | *Syntax:* **unsigned int var=val;**   * var – defines the variable name * val – defines the constant value. | |
| *Example* |  | |
| **Unsigned long** | **Purpose:**   * Unsigned long are extended size variables for number storage. | |
|  | *Syntax:* **unsigned long var=val;**  var – defines the variable name  val – defines the constant value. | |
| *Example* | Unsigned long time;  Void setup()  {  Serial.begin(9600);  }  Void loop()  {  Serial.print(“time:”);  Time=mills();  Serial.println(time);  delay(100);  } | |
|  | | |
| **void** | **Purpose:**   * Void keyword is used only for function declaration. * It indicates the function is expected to return no information to the function form. | |
|  | *Syntax:* **void()** | |
| *Example* | Void setup()  {  Statement();  }  Void loop()  { Statement();  } | |
| **Word** | **Purpose:**   * A word stores a 16-bit unsigned number, from 0 to 65535. | |
|  | *Syntax:* **word var=val;**  var – defines the variable name  val – defines the constant value. | |
| *Example* | Word w=100; | |
|  | | |
| **7.Strings** | | |
| What is the overview of Strings?   * Constructs an instance of string class. * Constructing a string from a number contains the ASCII number. | | |
| **Functions** | | |
| **charAt()** | **Purpose:**   * Access a particular value of a string | |
|  | *Syntax:* **string.charAt(n)**   * **String –** a variable of type string * **N** – a variable of type unsigned int | |
| *Example* |  | |
|  | | |
| **compareTO()** | **Purpose:**   * Compare two strings, testing, whether one comes before or after mor they are equal. * The strings are compared by character by character. | |
|  | *Syntax:* **string.compareTo(string2)**   * **String –** a variable of type string * **String2** – another variable of type string | |
| *Example* |  | |
|  | | |
| **concat()** | **Purpose:**   * Appends the parameter of the string. | |
|  | *Syntax:* **string.concat(parameter)**   * Parameter shows the type of data types string, char, byte, int, etc,. | |
| *Example* |  | |
|  | | |
| **c\_str()** | **Purpose:**   * Converts the content of the string as C-style. * When the content is modify the string object, or when it is destroyed, any pointer previously returned by c\_str() becomes invalid. | |
|  | *Syntax:* **string.c\_str()**   * Shows the conversion of the content as C-style. | |
| *Example* |  | |
|  | | |
| **endsWith()** | **Purpose:**   * Tests whether or not a string ends with the character of another string. | |
|  | *Syntax:* **string.endsWith(string2)**   * String – a variable of type string * String 2 – another variable of type string. | |
| *Example* |  | |
|  | | |
| **equals()** | **Pupose:**   * Compare two strings for eqality. * The comparsion case is sensitive. | |
|  | *Syntax:*  **string.equals(string2)**   * **String, string2:** variables of type string. | |
| *Example* |  | |
|  | | |
| **equalsIgnoreCase()** | **Pupose:**   * Compare two strings for equality. * The comparision is not case sensitive. | |
|  | *Syntax:* **string.equalsIgnoreCase(string2)**   * String, string2 – variables of type string. | |
| *Example* |  | |
|  | | |
| **getBytes()** | **Purpose:**   * Copies the string characters to the supplied buffer. | |
|  | *Syntax:* **string.getBytes(buf, len)**   * String – A variable of type String * Buf – The buffer to copy the characters into (byte []) * Len – The size of the buffer (unsigned int) | |
| *Example* |  | |
|  | | |
| **indexOf()** | **Purpose:**   * Locates a character or string within anaother string. | |
|  | *Syntax:* **string.indexOf(val)**  **string.indexOf(val, from)**   * String – a variable type string * Val – the value to search for. * From – the index to start the search from. | |
| *Example* |  | |
|  | | |
| **lastIndexOf()** | **Purpose:**   * Locates a character or string within string. * By default it searches from the end of the string, but also can wotrk backwards from given index. | |
|  | *Syntax:* **string.lastIndexOf(val)**  **string.lastIndexOf(val, from)**   * String – a variable type string * Val – the value to search for. * From – the index to work backwards from. | |
| *Example* |  | |
|  | | |
| **length()** | **Purpose:**   * Return the length of string, in characters. | |
|  | *Syntax:* **string.length()**   * String – a variable of type string. | |
| *Example* |  | |
|  | | |
| **remove()** | **Purpose:**   * Modify in place a string removing chars from the provided index to the end of the string or to the index plus count. | |
|  | *Syntax:* **string.remove(index)**  **string.remove(index, count)**   * Index – A variable of type unsigned int * Count – A variable of type unsigned int | |
| *Example* |  | |
|  | | |
| **replace()** | **Purpose:**   * The string replace() function allows you to replace all instance of a given character with another character. | |
|  | *Syntax:*  **string.replace(substring1, substring2)**   * String – a variable type string * Substring – Another variable of type string. * Substring2 – Another variable of type string. | |
| *Example* |  | |
|  | | |
| **reserve()** | | **Purpose:**   * The String reserve () function allows you to allocate a buffer in memory for manipulating strings. |
|  | | *Syntax:* **string. reserve(size)**   * Size - unsigned int declaring the number of bytes in memory to save for string manipulation |
| *Example* | |  |
|  | | |
| **setCharAt()** | | **Purpose:**   * Sets a character of the String. Has no effect on indices outside the existing length of the String. |
|  | | *Syntax:* **string.setCharAt(index, c)**   * string, string2: a variable of type String |
| *Example* | |  |
|  | | |
| **startsWith()** | | **Purpose:**   * Tests whether or not a String starts with the characters of another String. |
|  | | *Syntax:* **string.startsWith(string2)**   * string, string2: a variable of type String |
| *Example* | |  |
|  | | |
| **substring()** | | **Purpose:**     * Get a substring of a String. * The starting index is inclusive (the corresponding character is included in the substring), but the optional ending index is exclusive (the corresponding character is not included in the substring). * If the ending index is omitted, the substring continues to the end of the String |
|  | | *Syntax:* **string. substring (from) or string. substring(from,to)**   * String – A variable of type String * From – The index to start the substring at * to (optional) – The index to end the substring before |
| *Example* | |  |
|  | | |
| **toCharArray()** | | **Purpose:**   * Copies the string’s characters to the supplied buffer. |
|  | | *Syntax:* **string. toCharArray(buf,len)**   * String – A variable of type String * Buf – The buffer to copy the characters into (char []) * Len – The size of the buffer (unsigned int) |
| *Example* | |  |
| **toInt()** | | **Purpose:**   * Converts a valid String to an integer. * The input string should start with an integer number. * If the string contains non-integer numbers, the function will stop performing the conversion. |
|  | | *Syntax:* **string.toInt()**   * String – A variable of type string. |
| *Example* | |  |
|  | | |
| **toFloat()** | | **Purpose:**   * Converts a valid String to a float. * The input string should start with a digit. If the string contains non-digit characters, the function will stop performing the conversion |
|  | | *Syntax:* **string. toFloat()**   * String – A variable of type string. |
| *Example* | |  |
|  | | |
| **toLowerCase()** | | **Purpose:**   * Get a lower-case version of a String. |
|  | | *Syntax:* **string. toLowerCase()** |
| *Example* | |  |
|  | | |
| **toUpperCase()** | | **Purpose:**   * Get an upper-case version of a String |
|  | | *Syntax:* **string. toUpperCase()** |
| *Example* | |  |
| **trim()** | | **Purpose:**   * Get a version of the String with any leading and trailing whitespace removed |
|  | | *Syntax:* **string. trim()** |
| *Example* | |  |