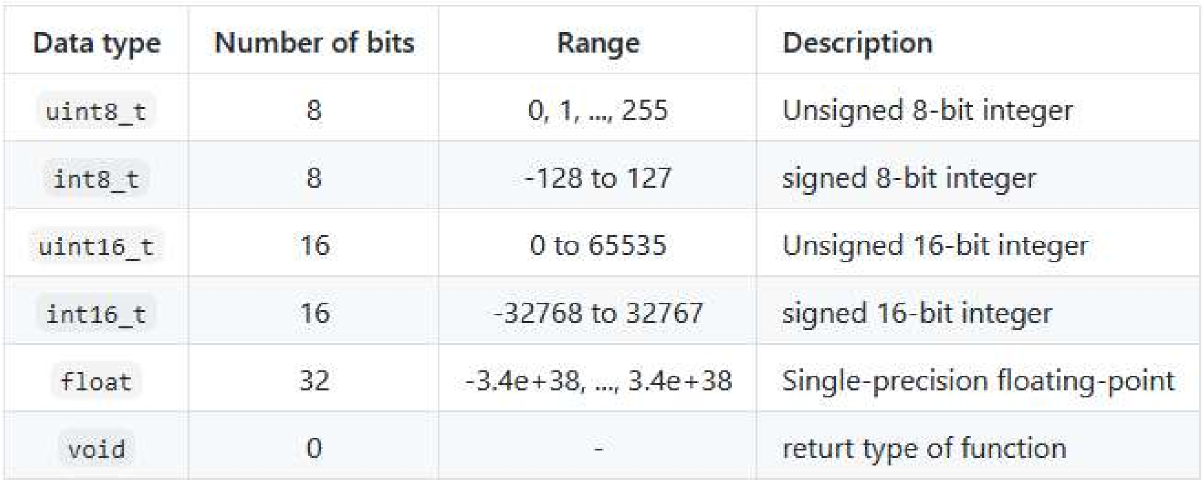
1.



Data type	Number of bits	Range	Description
`uint8_t`	8	0, 1, ..., 255	Unsigned 8-bit integer
`int8_t`	8	-128 to 127	signed 8-bit integer
`uint16_t`	16	0 to 65535	Unsigned 16-bit integer
`int16_t`	16	-32768 to 32767	signed 16-bit integer
`float`	32	-3.4e+38, ..., 3.4e+38	Single-precision floating-point
`void`	0	-	returt type of function

```c
#include <avr/io.h>

// Function declaration (prototype)
uint16_t calculate(uint8_t, uint8_t);

int main(void)
{
	uint8_t a = 156;
	uint8_t b = 14;
	uint16_t c;

	// Function call
	c = calculate(a, b);

	while (1)
	{
	}
	return 0;
}

// Function definition (body)
uint16_t calculate(uint8_t x, uint8_t y)
{
	uint16_t result;    // result = x^2 + 2xy + y^2

	result = x*x;
	result += 2*x*y;
	result += y*y;

	return result;
}
```

2.

gpio.c

```
/***********************************************************************
 *
 * GPIO library for AVR-GCC.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2019-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 **********************************************************************/

/* Includes ----------------------------------------------------------*/
#include "gpio.h"

/* Function definitions ----------------------------------------------*/
void GPIO_config_output(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name | (1<<pin_num);
}

/*--------------------------------------------------------------------*/
/* GPIO_config_input_nopull */
void GPIO_config_input_nopull(volatile uint8_t *reg_name, uint8_t pin_num)
{
       *reg_name = *reg_name & ~(1<<pin_num); //set pin as input
       *reg_name++;
       *reg_name = *reg_name & ~(1<<pin_num); //turn off internal pull-up
}

/*--------------------------------------------------------------------*/
void GPIO_config_input_pullup(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name & ~(1<<pin_num);  // Data Direction Register
    *reg_name++;                    // Change pointer to Data Register
    *reg_name = *reg_name | (1<<pin_num);   // Data Register
}

/*--------------------------------------------------------------------*/
void GPIO_write_low(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name & ~(1<<pin_num);
}

/*--------------------------------------------------------------------*/
/* GPIO_write_high */
void GPIO_write_high(volatile uint8_t *reg_name, uint8_t pin_num)
{
       *reg_name |= (1<<pin_num);
}

/*--------------------------------------------------------------------*/
/* GPIO_toggle */
uint8_t GPIO_toggle(volatile uint8_t *reg_name, uint8_t pin_num)
{
       *reg_name ^= (1<<pin_num);
}

/*--------------------------------------------------------------------*/
/* GPIO_read */
```

```c
uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num)
{
	return(bit_is_set(*reg_name, pin_num));

}
```

main.c

```c
/***********************************************************************
 *
 * Alternately toggle two LEDs when a push button is pressed. Use
 * functions from GPIO library.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2019-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 **********************************************************************/

/* Defines -----------------------------------------------------------*/
#define LED_GREEN   PB5     // AVR pin where green LED is connected
#define LED_RED     PC0     // AVR pin where red LED is connected
#define BTN                 PD0           // AVR pin where button is connected
#define BLINK_DELAY 500
#ifndef F_CPU
#define F_CPU 16000000      // CPU frequency in Hz required for delay
#endif

/* Includes ----------------------------------------------------------*/
#include <util/delay.h>     // Functions for busy-wait delay loops
#include <avr/io.h>         // AVR device-specific IO definitions
#include "gpio.h"           // GPIO library for AVR-GCC

/* Function definitions ----------------------------------------------*/
/**
 * Main function where the program execution begins. Toggle two LEDs
 * when a push button is pressed. Functions from user-defined GPIO
 * library is used instead of low-level logic operations.
 */
int main(void)
{
    /* GREEN LED */
    GPIO_config_output(&DDRB, LED_GREEN);
    GPIO_write_low(&PORTB, LED_GREEN);

    /* second LED */
    GPIO_config_output(&DDRC, LED_RED);
    GPIO_write_low(&PORTB, LED_GREEN);

    /* push button */
    GPIO_config_input_pullup(&DDRD, BTN);

    // Infinite loop
    while (1)
    {
        // Pause several milliseconds
        _delay_ms(BLINK_DELAY);
              if(GPIO_read(&PIND, BTN) == 0)
              {
                     GPIO_toggle(&PORTB, LED_GREEN);
                     GPIO_toggle(&PORTC, LED_RED);
              }

    }

    // Will never reach this
    return 0;
}
```

declaration

It is done in .h file (or in main.c if project is very short). Consist of return type variable, name of function and input name and type of variable(s).

Example

```
void GPIO_write_low(volatile uint8_t *reg_name, uint8_t pin_num);
```

Definition

It is done in .c file. The line is same as in .h file and in brackets - As name suggest is definition of function (the algorithm is written here and return variable is specified – if not void).

Example

```
void GPIO_write_low(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name & ~(1<<pin_num);
}
```