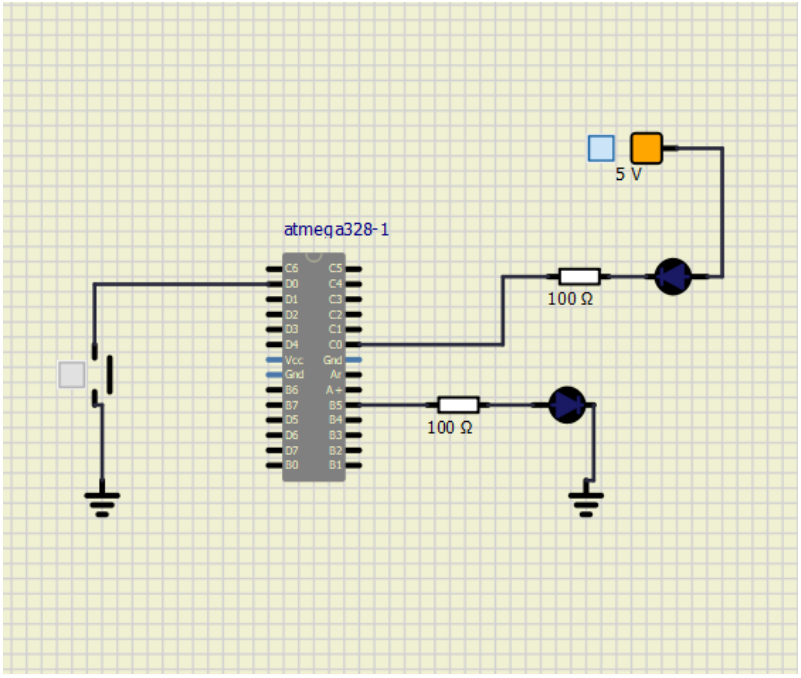
Function:

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

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

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;
}
```

Chart

Data type	Number of bits	Range	Description
uint8_t	8	0-255	Unsigned 8-bit integer
int8_t	8	-128 ...127	Signed 8-bit integer
uint16_t	16	0-65535	Unsigned 16-bit integer
int16_t	16	-32768...32767	Signed 16-bit integer
float	32	-3.4E+38 ..... 3.4E+38	Single precesion floating point
void	Depends on a system	Empty	Empty

Scheme




```
/*****************************************************************
 *
 * 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);
}

/*---------------------------------------------------------------------*/
void GPIO_config_input_nopullup(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_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);
}

/*-------------------------------------------------------------------*/

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

/*-------------------------------------------------------------------*/
/* GPIO_read */
uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num)
{
      if (bit_is_clear(*reg_name, pin_num))
      {

            return 0;
      }
      else
      {
            return 1;

      }

}
```

Main.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
```

```c
#define BTN          PD0
#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_high(&PORTC, LED_RED);


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




    // WRITE YOUR CODE HERE

    // 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;
}
```

Gpio.h

```
#ifndef GPIO_H
#define GPIO_H

/************************************************************************
 *
 * 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.
 *
 ************************************************************************/

/**
 * @file  gpio.h
 * @brief GPIO library for AVR-GCC.
 *
 * @details
 * The library contains functions for controlling AVRs' gpio pin(s).
 *
 * @note
 * Based on AVR Libc Reference Manual. Tested on 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 <avr/io.h>

/* Function prototypes -----------------------------------------------*/
/**
 * @brief Configure one output pin in Data Direction Register.
 * @param reg_name - Address of Data Direction Register, such as &DDRA,
 *                   &DDRB, ...
 * @param pin_num - Pin designation in the interval 0 to 7
 */
void GPIO_config_output(volatile uint8_t *reg_name, uint8_t pin_num);

void GPIO_config_input_nopull(volatile uint8_t *reg_name, uint8_t pin_num);

void GPIO_config_input_pullup(volatile uint8_t *reg_name, uint8_t pin_num);

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

void GPIO_write_high(volatile uint8_t *reg_name, uint8_t pin_num);

void GPIO_toggle(volatile uint8_t *reg_name, uint8_t pin_num);

uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num);

#endif
```

Deklarácia:

- Hovorí compileru ako sa funkcia volá aký ma typ a aké ma parametre.

```
Int8_t calculate(int x, int y );
```

Definicia

- Poskytuje informácie a tele funkcie ako sa funkcia správa.

```
Int8_t calculate(int x, int y);{
Return(x +y);
}
```