When you define and use a variable, the generated data within a sketch only lasts as long as the Arduino is powered. If you reset or power off the Arduino, the data stored inside a variable disappears. If you want to keep the data stored for future use you need to use the Arduino EEPROM. The EEPROM can be read, erased, and rewritten electronically. In Arduino, you can read and write from the EEPROM easily using the `EEPROM` library.

**Task.1.** Check the number of available cells in EEPROM.

```
#include<EEPROM.h>
#define BAUDRATE 115200

void setup () {
 Serial.begin(BAUDRATE);
 Serial.print("Arduino Nano EEPROM size: ");
 Serial.println(EEPROM.length());
}

void loop () {
}
```

**Task.2.** Read the content of a single memory cell in EEPROM.

```
#include<EEPROM.h>
#define BAUDRATE 115200

byte t_byte;

void setup () {
 Serial.begin(BAUDRATE);
 t_byte = EEPROM.read(0);
 Serial.print("EEPROM content: ");
 Serial.println(t_byte,HEX);
}

long p_millis = 0;
#define TIM_DELAY 2000
```

```
void loop () { }
```

**Task.3.** Read the content of 64 memory cells in EEPROM.

```
#include<EEPROM.h>
#define BAUDRATE 115200

byte t_byte;

void setup () {
 Serial.begin(BAUDRATE);
 for(int i = 0; i < 64; i++) {
  t_byte = EEPROM.read(i);
  Serial.print("adr: ");
  Serial.print(i);
  Serial.print("\tvalue: ");
  Serial.println(t_byte,HEX);
 }
}

void loop () { }
```

**Task.3.** Write 55h to the EEPROM cell.

**The EEPROM has a finite life. In Arduino, the EEPROM is specified to handle 100 000 write/erase cycles for each position.** However, reads are unlimited. This means you can read from the EEPROM as often as possible without compromising its life expectancy.

```
#include<EEPROM.h>
#define BAUDRATE 115200

byte t_byte;
#define ADR 0

void setup () {
 Serial.begin(BAUDRATE);
```

```
 t_byte = EEPROM.read(ADR);
 Serial.print("address: ");
 Serial.print(ADR);
 Serial.print("\tvalue: ");
 Serial.println(t_byte,HEX);
 if(t_byte == 0xFF) EEPROM.write(ADR,0x55);
}

void loop () { }
```

The EEPROM has limited life expectancy due to limited write/erase cycles, so use the `EEPROM.update()` function instead of the `EEPROM.write()` saves cycles.

```
#include<EEPROM.h>
#define BAUDRATE 115200

byte t_byte;
#define ADR 4

void setup () {
 Serial.begin(BAUDRATE);
 t_byte = EEPROM.read(ADR);
 Serial.print("address: ");
 Serial.print(ADR);
 Serial.print("\tvalue: ");
 Serial.println(t_byte,HEX);
 EEPROM.update(ADR,0x55);
}

void loop () {}
```

**Task.4.** Store the following data structure in EEPROM.

```
struct Car { char make[25]; int motor; int bhp; };
```

```
#include<EEPROM.h>
#define BAUDRATE 115200

struct Car {
  char make[25];
  int motor;
  int bhp;
};

#define ADR 20

void setup () {
 Serial.begin(BAUDRATE);
 Car my_car = {"Kia Sportage", 1580, 180};
 EEPROM.put(ADR,my_car);
 delay(1000);
 memset(my_car.make,'*',sizeof(my_car.make));
 my_car.motor = 0;
 my_car.bhp = 0;
 Serial.print(my_car.make);
 Serial.print("\t");
 Serial.print(my_car.motor);
 Serial.print("\t");
 Serial.println(my_car.bhp);
 EEPROM.get(ADR,my_car);
 Serial.print(my_car.make);
 Serial.print("\t");
 Serial.print(my_car.motor);
 Serial.print("\t");
 Serial.println(my_car.bhp);
}

void loop () { }
```

**For those interested:**

1. A guide to EEPROM:

   docs.arduino.cc/learn/programming/eeprom-guide

2. EEPROM Library:

   docs.arduino.cc/learn/built-in-libraries/eeprom