Přípravy na kroužek Mechatronika – KA6(A2d)

10. lekce – programování motorů, použití funkcí

Základy práce s funkcemi

Máme-li v našem programu kód, který se stále opakuje, můžeme jej umístit do **funkce**. Ve vlastním programu bude definovaný **pouze 1x** a na místa v programu, kde se kód opakuje, umístíme pouze **odkaz (volání) nadefinované funkce**. Použitím funkcí tedy **ušetříme spoustu času** při psaní kódu, náš **kód bude menší a přehlednější**.

Obecný předpis pro definici funkce:

návratový_typ identifikátor_fce(*parametry_fce a jejich datové typy oddělené čárkou*)

```
{
        příkaz1;
        příkaz2;
        .
        .
        .
        příkazN;
}
```

Návratový typ volíme dle hodnoty vrácené funkcí (int, double apod.). Pokud má funkce vracet hodnotu, přidáme na její konec příkaz **return**.

U funkcí, kde **nedochází** k návratu hodnoty, použijeme typ **void**. To samé platí pro parametry funkce. Funkce bez vracení parametrů by vypadala takto:

void identifikátor_fce(*void*)

```
{
        příkazy;
}
```

Celou funkci pak umístíme v kódu **mimo funkci main**.

Volání funkce

Funkci voláme následovně:

identifikátor_fce(parametry_fce oddělené čárkou);

Funkci můžeme volat **opakovaně** dle potřeby.

Praktické využití funkcí při pohybu vozítka

Příklad:
Založíme nový projekt v Atmel Studiu a v něm vytvoříme následující zdrojový kód:

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
	DDRD = ((1<<PD4)|(1<<PD5)|(1<<PD6)|(1<<PD7));
	while(1)
	{
		//Pohyb vpřed
		PORTD |= ((1<<PORTD5)|(1<<PORTD6));
		_delay_ms(2000);

		//Pohyb vlevo
		for (uint8_t i=0;i<=10;i++) {
			PORTD ^= (1<<PORTD6);
			_delay_ms(100);
		}

		//Pohyb vpravo
		for (uint8_t j=0;j<=10;j++) {
			PORTD ^= (1<<PORTD5);
			_delay_ms(100);
		}

		//Stop
		PORTD = 0x00;
		_delay_ms(2000);

		//Pohyb vzad
		PORTD |= ((1<<PORTD4)|(1<<PORTD7));
		_delay_ms(2000);
		PORTD &= ~((1<<PORTD4)|(1<<PORTD7));
	}
	return 0;
}
```

Kód je bez použití funkcí, pokud bychom chtěli některé z pohybů opakovat, museli bychom celý kód pohybu **psát znovu a znovu**.

Tomu se použitím funkcí můžeme vyhnout.

Kód upravíme následovně:

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>

void vlevo()
{
	for (uint8_t i=0;i<=10;i++) {
		PORTD ^= (1<<PORTD6);
		_delay_ms(100);
	}
}
void vpravo()
{
	for (uint8_t j=0;j<=10;j++) {
		PORTD ^= (1<<PORTD5);
		_delay_ms(100);
	}
}
void vpred()
{
	PORTD |= ((1<<PORTD5)|(1<<PORTD6));
	_delay_ms(2000);
}
void vzad()
{
	PORTD |= ((1<<PORTD4)|(1<<PORTD7));
	_delay_ms(2000);
}
void stop()
{
	PORTD = 0x00;
	_delay_ms(2000);
}


int main(void)
{
	DDRD = ((1<<PD4)|(1<<PD5)|(1<<PD6)|(1<<PD7));
	while(1)
	{
		vpred();
		vlevo();
		vpravo();
		stop();
		vzad();
		stop();
	}
	return 0;
}
```

Ve funkci main pak již voláme pouze příslušné pohybové funkce, můžeme je **opakovat,** kolikrát budeme potřebovat. **Hodnoty** pro cyklus **for** můžeme místo pevného nastavení předávat přes **parametry**.

Vypracoval Radek Zvěřina.