```
  {
    if(ctiSP())  //test sepnuteho SP
    {
      nastavLED(1);  //rozsvit LED
    }
    else
    {
      nastavLED(0);  //zhasni LED
    }
  }
}
```

Funkci programu ověříme jednoduše. Držíme-li tlačítko stisknuté, LED svítí. Při uvolnění tlačítka je LED zhasnutá.

Předávání parametrů odkazem

V příkladu **PROG_11** jsme si vystačili s technikou předávání parametrů hodnotou. Připomeňme, že tato technika je použitelná v případě vstupních parametrů (jejich hodnoty lze pouze číst, uvnitř funkce není možná jejich změna).

Existují však případy, kdy potřebujeme zapsat funkci, která má **výstupní parametry**. Funkce tak předá výsledek nejen pomocí návratové hodnoty, ale také v některém z parametrů. V tomto případě je to **referenční parametr**.

Pro deklaraci proměnné typu **odkaz (reference)** používáme symbol &. Proměnné typu odkaz používáme typicky v případě parametrů funkce, které mají fungovat jako výstupní. Říkáme, že parametr je předán odkazem.

Odkazy lze používat pouze v jazyce C++. Jazyk C používá pro realizaci výstupních parametrů komplikovanější způsob předávání adresy proměnné (předávání parametrů přes ukazatel).

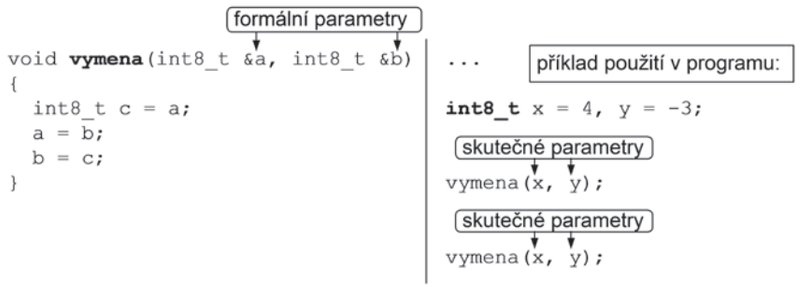
Pro funkce, jež některý ze svých parametrů předávají odkazem, platí:

- formální parametr vystupuje v těle funkce jako zástupné označení (alias) pro skutečný parametr, skutečný a formální parametr jsou tedy propojeny,
- referenční formální parametr se tedy odkazuje na skutečný parametr,
- změna formálního parametru **má vliv** na skutečný parametr,
- jako skutečný parametr muže být použita **pouze** proměnná stejného typu, jaký je typ formálního parametru (literál použít nelze).

Jako příklad funkce, která předává parametry odkazem, lze uvést funkci `vymena` dle obr. 5.3.

Na obr. 5.3 vlevo je uvedena deklarace funkce `vymena`. Parametry `a`, `b` jsou typu odkaz na 8bitové celé číslo se znaménkem. Symbol odkazu (&) obvykle uvádíme u identifikátoru příslušného parametru.

Tato funkce vzájemně vymění hodnoty parametrů `a`, `b`. Původní hodnota parametru `a` bude uložena do parametru `b`. A naopak, původní hodnota parametru `b` bude uložena do parametru `a`.



Obrázek 5.3 Funkce vymena (používá parametry předávané odkazem)

Na obr. 5.3 vpravo vidíme použití této funkce v hlavním programu. Skutečnými parametry jsou 8bitová celá čísla se znaménkem s identifikátory `x`, `y`. Původní hodnoty jsou: `x = 4`, `y = -3`.

Po prvním volání funkce `vymena` dojde k záměně hodnot: x = –3, y = 4. Druhé volání této funkce vede ke druhé záměně, tedy stav proměnných je stejný jako při startu programu: x = 4, y = –3.

PROG_12: univerzální funkce pro manipulaci s bity registrů mikrokontroléru

Nový projekt vytvoříme dle postupu zapsaného u **PROG_02** v kapitole 2. Funkci `printf` však nyní nebudeme používat, proto provedeme pouze kroky 1 až 5.

V příkladu **PROG_12** si předvedeme zápis obecně použitelných funkcí pro manipulaci s bity registrů mikrokontroléru. Připomeňme, že parametr musí být předán odkazem, má-li se chovat jako výstupní (tedy funkce stav příslušného registru změní). Hlavičky funkcí jsou uvedeny níže (při předávání registru do funkce používáme klíčové slovo `volatile`, jeho účel je vysvětlen dále), parametr `reg` odpovídá registru, se kterým pracujeme, parametr `bit` je číslo bitu v rozmezí 0 až 7:

- `void setbit(volatile uint8_t &reg, uint8_t bit)`: nastavení bitu registru,
- `void clearbit(volatile uint8_t &reg, uint8_t bit)`: nulování bitu registru,
- `void negbit(volatile uint8_t &reg, uint8_t bit)`: negace bitu registru,
- `uint8_t getbit(volatile uint8_t reg, uint8_t bit)`: vrátí stav bitu registru.

Realizace těchto funkcí vychází z použití operátorů bitového součtu, součinu, výlučného bitového součtu a posuvu doleva (podobně jako v příkladu **PROG_11**). Obrovskou výhodou těchto funkcí je jejich použitelnost na libovolný registr mikrokontroléru.

Poznámka: Klíčové slovo `volatile` používáme v situacích, kdy chceme překladači zabránit v tom, aby v důsledku optimalizace usoudil, že některá proměnná se nepoužívá, a tudíž pro ni ani nevyhradí místo v paměti.
Obvykle se toto klíčové slovo používá při definici globálních proměnných (jsou použitelné v celém programu), které jsou měněny asynchronně pomocí přerušení.
Při předávání registrů do výše uvedených funkcí je klíčové slovo `volatile` přidáno samotným překladačem. Pro zajištění souladu datových typů formálního a skutečného parametru musíme tedy toto klíčové slovo použít.