Práce s preprocesorem

Základy programování 2

Mgr. Markéta Trnečková, Ph.D.

Palacký University, Olomouc

Zpracování programu

- **Editor** - vytváření kódu
- **Preprocesor** - předzpracování před překladem
- **Compiler** (překladač) - překlad do objektového souboru (jazyk relativních adres) → `.OBJ` + protokol o překladu `.LIS`
- **Linker** (sestavovací program) - přiděluje relativním adresám skutečné adresy → spustitelný soubor `.EXE`
- **Debugger** (ladící program)

Preprocesor

- zpracovává zdrojový kód před překladem
- nekontroluje syntaktickou správnost
- provádí záměnu textů
- odstraňuje z kódu komentáře
- připravuje podmíněný překlad

Preprocesor

- řádky začínají #
- před ani za # by neměla být mezera

- makra bez parametru
- makra s parametry
- podmíněný překlad
- vkládání souborů

Makra bez parametru = konstanty

- Symbolické konstanty
- def. na začátku programu
- náhrada řetězce skutečnou hodnotou = expanze, rozvinutí
- Konvence: název velkými písmeny
- `#define JMENO hodnota`

Example

```
#define PI 3.14
#define AND &&
#define ERROR printf("Chyba programu");
```

Makra bez parametru = konstanty

`#define JMENO hodnota`
Výskyty `JMENO` v kódu nahrazeny výrazem `hodnota`

Example

```
#define PI 3.14

int main()
{
    float r = 3;
    float V = 2*PI*r;  // prevedeno na float V = 2*3.14*r;

    return 0;
}
```

Makra bez parametru = konstanty

Example

```
#define JMENO "Marketa"

int main()
{
    // Vypise Moje jmeno je JMENO
    printf("Moje jmeno je JMENO");

    return 0;
}
```

Jak vypsat jméno?

Makra bez parametru = konstanty

- platnost konstanty od definice do konce souboru
- Změna hodnoty: Nutno konstantu zrušit a definovat znovu
- `#undef JMENO`

Example

```
#define JMENO "Marketa"

#undef JMENO

#define JMENO "TRNECKOVA"
```

Makra bez parametru = konstanty

- Dlouhé konstanty je možné rozdělit na více řádků
- řádek končí \

Example

```
#define DLOUHA_KONSTANTA 1.23456798\
910111213
```

Makra s parametry

- inline funkce
- menší "administrativa" než u funkcí
- není možné použít rekurzi
- Konvence názvy malým písmem
- `#define jmeno(arg1, ..., argN) telo_makra`
- `arg1, ..., argN` argumenty jako u funkci

Makra s parametry

Example

```
#define na2(x) ((x)*(x))

#define na2a(x) x*x

na2(f+g);   // ((f+g)*(f+g))
na2a(f+g); // f+g*f+g
```

Makra s parametry

Vyvarujte se použití argumentů s vedlejším efektem.

Example

```
#define na2(x) ((x)*(x))

int i = 2;
na2(i++);
```

Co bude výsledkem?
Napište funkci fna2, která bude vracet druhou mocninu. Co bude výsledkem pro argument i++?

Podmíněný překlad

- dočasné vynechání části zdrojového kódu při kompilaci

Podmíněný překlad řízený konstantním výrazem

```
#if konstantni_vyraz
    cast 1
#else
    cast 2
#endif
```

```
#if konstantni_vyraz
    cast 1
#elif
    cast 2
#else
    cast 3
#endif
```

Podmíněný překlad řízený konstantním výrazem

Example

```
#define ENG 1

#if ENG
    #define ERROR "error"
#else
    #define ERROR "chyba"
#endif
```

Podmíněný překlad

- `#ifdef JMENO`
- `#ifndef JMENO`

Example

```
#define ENG 1

#ifdef ENG
    #define ERROR "error"
#else
    #define ERROR "chyba"
#endif
```

Podmíněný překlad

- `#if defined(JMENO)`

Example

```
#define ENG 1

#if defined(ENG) && ENG
    #define ERROR "error"
#else
    #define ERROR "chyba"
#endif
```

Vkládání souborů

- `#include <nazev>`
- `#include ''nazev''`

Cvičení

1. Upravte kód ze Slidu 14, tak aby se rozeznávaly 4 různé jazyky pro chybovou hlášku.
2. Napište program, který sečte `N` prvních přirozených čísel a vypíše např. "Soucet prvnich 5-ti cisel je 15."N definujte jako symbolickou konstantu.
3. Napište makro `na_3(x)`, které bude počítat třetí mocninu. Vyzkoušejte ho na výrazech:
   ```
   na_3(3)
   na_3(i)
   na_3(2 + 3)
   na_3(i * j + 2)
   ```
4. Definujte makro `je_velike(c)`, které vrátí 0 není-li znak velké písmeno a 1, pokud je.
5. Definujete makro `cti_int(i)`, které čte z klávesnice celé číslo. Makro musí jít použít i ve výrazu
   `if((j=cti_int(k)) == 0)`
   Můžete použít například funkci getchar.