KATEDRA INFORMATIKY
PŘÍRODOVĚDECKÁ FAKULTA
UNIVERZITA PALACKÉHO V OLOMOUCI

Funkce s proměnným počtem parametrů

5. cvičení

Jiří Zacpal

KMI/ZP2 – Základy programování 2

Princip FPPP

- při volání funkce se skutečné parametry kopírují na zásobník (paměť používaná pro uložení lokálních proměných a parametrů funkce)
- pokud tedy známe adresu a typ některého parametru, můžeme přistoupit k následujícímu parametru
- díky makrům ze `stdarg.h` nemusíme znát strukturu zásobníku (je implementačně závislá)
- FPPP musí mít vždy alespoň jeden pevný parametr, od kterého začínáme na zásobníku hledat
- musí být jasné, kolik parametrů v paměti následuje (je znám počet nebo je použita nějaká zarážka)
- musí být známé typy parametrů

Typické příklady FPPP

- počet parametrů a jejich typy jsou dány předaným formátovacím textovým řetězcem (např. funkce `printf`)
- předpokládá se určitý typ parametrů a předá se jejich počet (výpočet sumy, průměru čísel apod.)
- předpokládá se určitý typ parametrů a konec seznamu parametrů je označen zarážkou – domluvenou hodnotou (např. spojení textových řetězců)

Deklarace a volání FPPP

- při deklaraci (definici) FPPP se používá výpustek („...")
- definice funkce:

```
double prumer(int pocet, double prvni,
...){…}
int print(char* format, ...){…}
```

- volání funkce:

```
double prum;

prum = prumer(5, 1.2, 2.3, 4.3, 2.5, 7.1);
print("Průměr je %g. \n", prum);
```

Zpracování parametrů – makra z stdarg.h

- poskytuje programátorům typ `va_list` a makra `va_start()`, `va_arg()` a `va_end()`
- inicializace „ukazatale na zásobník" (proměnné typu `va_list`):
```
va_list param;
va_start(param, prvni);
```
- přístup k hodnotě parametru a posun „ukazatele" na začátek paměti dalšího parametru:
```
cislo = va_arg(param, double);
```
- ukončení práce s „ukazatelem na zásobník":
```
va_end(param);
```
- při práci s FPPP je nutné věnovat zvýšenou pozornost použitým datovým typům
```
prum = prumer(3, 1, (double)3, (double)2);
```

Příklad FPPP

```
double prumer(int pocet, double prvni, ...)

{

   double out;
   va_list param;
   int i;

   out = prvni;
   va_start(param, prvni);
   i = pocet - 1;
   while (i>0)

         {
         out+=va_arg(param, double);
         i--;
         }
   va_end(param);
   out = out/pocet;

   return out;

}
```

Úkol

Napište v jazyku C funkci

`long double prumer(char* format, ...)`,

která vypočítá aritmetický průměr ze zadaných hodnot různých datových typů. Typy předávaných hodnot jsou určeny pomocí parametru `format`, který může tvořit libovolná posloupnost znaků odpovídající typům následujících parametrů - znak "*i*" pro typ int, "*d*" pro typ double a "*l*" pro long double.

Příklad použití:

```
pr = prumer("idld", 1, (double)3,
(long double)2, 3.0); printf("Prumer
je %Lf. \n", pr);
```

Příklad výstupu:

```
Prumer je 2.25.
```

Úkol – řešení

```
long double prumer(char* format, ...)
{
    long double out=0;
    int pocet=strlen(format);
    va_list param;
    va_start(param, format);
    int i = 0;
    while (i<pocet)
    {
        switch(format[i])
        {
            case 'i':out+=va_arg(param, int);break;
            case 'd':out+=va_arg(param, double);break;
            case 'l':out+=va_arg(param, long double);break;
        }
        i++;
    }
    va_end(param);
    out = out/pocet;
    return out;
}
```

Bodovaný úkol

Napište v jazyku C funkci

```
komplexni suma(int pocet, ...),
```

která výpočítá součet předaných komplexních čísel. Počet sčítaných čísel je určen pevným parametrem `pocet`, za nímž pak ve volání funkce následují hodnoty, které má funkce sčítat. Pro práci s komplexními čísly využijte vámi definovaný strukturovaný datový typ `komplexni`.