Bitové operátory a bitová pole

Úvod do programování 2

Tomáš Kühr

Bitové operátory

- Provádějí danou operaci s jednotlivými bity svých operandů
- Operandy bitových operátorů mohou být pouze celočíselné
- Vyhodnocení bitových operátorů je relativně rychlé
- Bitové operátory: & (součin), | (součet), ^ (nonekvivalence, XOR), >> (posun vpravo), << (posun vlevo), ~ (negace)
- Příklady:

```
char c = 'a';
c = c & 0xDF; // prevod na velka pismena
c |= 0x20; // prevod na mala písmena
int x = 5;
x = x << 3; // nasobeni 3. mocninou cisla 2
x >>= 3; // deleni 3. mocninou cisla 2
```

Práce s jednotlivými bity

- Bitové operátory se často používají při práci s více informacemi uloženými v jediné proměnné
- Umožňují tak snížit množství využívané paměti
- A často i zrychlení výpočtu alespoň v některých situacích
- Jen mírně složitěji lze pracovat také se skupinami bitů
- Příklady:

```
#define READ 0x8
#define WRITE 0x10
#define DELETE 0x20
…
unsigned int status;
status |= READ | WRITE | DELETE;
status |= READ | WRITE;
status &= ~(READ | WRITE | DELETE);
```

Bitová pole

- Umožňují mnohem snadnější práci s bity a skupinami bitů
- Definice bitového pole odpovídá strukturovanému typu

```
typedef struct {
    unsigned den : 5;
    unsigned mesic : 4;
    unsigned rok : 7;
} DATUM;
```

- Položky ale mohou být pouze celočíselné (unsigned/signed)
- Číslo za dvojtečkou udává počet bitů pro danou položku
- Rozsah hodnot jednotlivých položek je 0 až $2^{počet\ bitů} - 1$
- Pro uložení znaménka je potřeba 1 bit navíc
- Velikost celého bitového pole bývá omezena na velikost typu int

Práce s bitovým polem

- S bitovým polem lze pracovat podobně jako se strukturou
- Pouze nelze pracovat s adresami jednotlivých položek
- Vytvoření proměnné (s inicializací)

```
DATUM dnes = { 23, 4, 2008 - 1980 };
DATUM zitra = dnes;
```

- Přístup k položkám

```
zitra.den++;
dnes.mesic = 6;
dnes.rok = 2009 - 1980;

if (dnes.rok != d.rok) ...

printf("%u. %u. %u\n", d.den, d.mesic, d.rok + 1980);
```

Další nesouvisející drobnosti

Konstantní proměnné

- Vytváříme pomocí modifikátoru const

```
const static double pi = 3.14159;
const max = 100; // implicitni typ int
```

- Takto vytvořená konstanta má daný typ a rozsah platnosti
- Konstantní ukazatel vs. ukazatel na konstantní hodnotu

```
const char *t1 = "Pointer na const znaky";

char text[] = "Const pointer na znaky";
char *const t2 = text;

const char *const t3 = "Const pointer na const znaky";
```

Generátor pseudonáhodných čísel

- Inicializace generátoru pomocí
```
void srand(unsigned int seed); // v stdlib.h
```
- Jako parametr seed se při inicializaci často používá
```
time_t time(time_t *out); // v time.h
```
  - Funkce vrací počet sekund od 1. ledna 1970 do svého volání
- Generování čísel pomocí
```
int rand(); // v stdlib.h
```
  - Funkce vrací číslo od 0 do konstanty RAND_MAX ze stdlib.h
- Generování čísel v jiném rozsahu (od min do max včetně)
```
cislo = rand() % (max + 1 - min) + min;
```
- Příklad
```
int pole[20];
srand((unsigned int)time(NULL));
for (int i = 0; i<20; i++)
   pole[i] = rand() % 10 + 1; // od 1 do 10
```

Pokročilé možnosti ukončení programu

- Všechny zde zmíněné funkce jsou deklarovány v stdlib.h
- Standardní ukončení programu
  `void exit(int kod);`
  - Korektně uzavře soubory, smaže dočasné soubory, volá funkce registrované pomocí atexit
  - Parametr kod odpovídá návratové hodnotě programu (funkce main)
- Registrace ukončovacích funkcí
  `int atexit(void(*funkce)());`
  - Postupně se spustí při standardním ukončení programu
- Nestandardní ukončení programu
  `void abort();`
  - Rychlé ukončení programu
  - Nezapisuje buffery, nemaže dočasné soubory, nevolá funkce registrované pomocí atexit

Charakteristiky datových typů

- Pro snadnou a bezproblémovou přenositelnost programů
- Existují symbolické konstanty popisující všechny datové typy
- Definovány jsou
  - Ve float.h – pro reálné typy
  - V limits.h – pro celočíselné typy
- Jména konstant jsou intuitivní, definice jsou navíc opatřeny komentáři
- Celočíselné (např. typ int)
  - INT_MAX – maximální hodnota
  - INT_MIN – minimální hodnota,
- Reálné typy (např. typ double)
  - DBL_MIN, DBL_MAX – minimum a maximum (jako u celočíselných)
  - DBL_DIG – počet platných číslic
  - DBL_EPSILON – nejmenší rozlišitelný přírůstek k číslu 1.0
  - A další ...

Práce se znaky

- Makra pro práci se znaky jsou definována v ctype.h
- Názvy maker jsou opět intuitivní
  isalpha, isalnum, isdigit, isxdigit, islower, isupper, isprint, ispunct, isgraph, isspace, iscntrl, tolower, toupper
- Příklad použití:

```
int c, i;
int pismena[26];
FILE *fr = fopen("text.txt", "r");
for (i = 0; i<26; i++) pismena[i] = 0;
while ((c = getc(fr)) != EOF){
    if (isalpha(c)) {
        if (islower(c)) c = toupper(c);
            pismena[c - 'A']++;
    }
}
```