Operátory, operandy a výrazy

Petr Šaloun

katedra informatiky FEI VŠB-TU Ostrava

3. října 2005

Identifikátor

Délka až 1024 znaky

- prvním symbol písmeno nebo podtržítko,
- následuje libovolná kombinace písmen, číslic a podtržítek,
- rozlišují se malá a velká písmena!

Klíčová slova

asm	do	inline	short	typeid
auto	double	int	signed	typename
bool	dynamic_cast	long	sizeof	union
break	else	mutable	static	unsigned
case	enum	namespace	static_cast	using
catch	explicit	new	struct	virtual
char	extern	operator	switch	void
class	false	private	template	volatile
const	float	protected	this	wchar_t
const_cast	for	public	throw	while
continue	friend	register	true	
default	goto	reinterpret_cast	try	

komentář

/* a */, // jednořádkový komentář

```
/* Toto je komentar - je napsan mezi parem znacek,
    a zde je jeho pokracovani na druhem radku. */
...
/* dalsi komentar, tentokrat rozsahlejsi
  if (uk->chyba) {
     ts->pom_info++;
    } // vlozeny radkovy komentar, vnoreny
  else
   stale jsme v komentari
*/
...
```

odsazovač (bílý znak, prázdné místo, *white space*):
mezera, tabulátor, nový řádek, posun řádku, návrat vozíku, nová stránka a vertikální tabulátor.

Čísla v počítači a v C++ I

jen nuly a jedničky, celá čísla přesně, reálná mantisa plus exponent.

datový typ	**bitů**	**význam**
bool	nedef.	logická hodnota
char	8	znak
wchar_t	16	UNICODE znak
short	32	krátké celé číslo
int	32	celé číslo
long	32	dlouhé celé číslo
enum	32	výčtový typ
float	32	racionální číslo
double	64	racionální číslo s dvojitou přesností
long double	80	ještě delší racionální číslo
pointer	32	ukazatel

Čísla v počítači a v C++ II

$$\begin{array}{ccccc} \text{short} & \leq & \text{int} & \leq & \text{long} \\ \text{float} & \leq & \text{double} & \leq & \text{long double} \end{array}$$

dále platí, že char vyžaduje 8 bitů.

`limits.h`, `float.h` norma IEEE 754

Deklarace, definice a konstanty

deklarace určuje typ objektu
definice definuje hodnotu proměnné či posloupnost příkazů funkce.
konstanty pojmenované hodnoty
programátorský styl
magická čísla
konstantní výrazy nesmí

- přiřazení,
- inkrementace a dekrementace,
- funkční volání,
- čárka.

konstant 5-20

```
#include <iostream>
#include <string>

using namespace std;

const bool pravda = true;
const int konstanta = 123;
const int celociselna = -987;
const double CPlanck = 6.6256e-34L;
const char male_a = 'a';
const string retezec = "Konstantni retezec.";
const float meze[2] = {-20, 60};
const char rimska_znk[] = {'I', 'V', 'X', 'L', 'C', 'D
const int  arabska_hodn[] = {1, 5, 10, 50, 100, 500, 1
```

konstant 21-37

```
int main() {
 cout << pravda << endl
    << konstanta << endl
    << celociselna << endl
    << CPlanck  << endl
    << male_a  << endl
    << retezec  << endl;
 for (int i=0;i<sizeof(arabska_hodn)/sizeof(int);i++){
  cout << "i=" << i
    << ", rimsky:" << rimska_znk[i]
    << ", arabsky " << arabska_hodn[i]
    << endl;
 } // konec tela cyklu
 return 0;
} // int main()
```

výstup konstant

sizeof (rimska_znaky)/ sizeof (char).

```
1
123
-987
6.6256e-034
a
Konstantni retezec.
i=0, rimsky:I, arabsky 1
i=1, rimsky:V, arabsky 5
i=2, rimsky:X, arabsky 10
i=3, rimsky:L, arabsky 50
i=4, rimsky:C, arabsky 100
i=5, rimsky:D, arabsky 500
i=6, rimsky:M, arabsky 1000
```

Celočíselné konstanty

- 0 v osmičkové soustavě,
- 0x nebo 0X v šestnáctkové soustavě, (A až F);
- libovolná číslice v desítkové soustavě

desítkový	**šestnáctkový**
123	0x7b
-987	0xfc25
255	0xff
4567	0x11d7
-4567	0xee29

u, U, bez znaménka.
l, L, např. long – 123UL

výpis: ch-opole

```
/*****************************
 * Chyba - konstanty zacinajici
 * nulou jsou osmickove!!!
 *****************************/
int a[] = {128, 231, 127, 162,
           067, 121, 034, 171};
```

Racionální konstanty

mantisa a exponent, double, např. 12.34e5
long double, L, např. 12.34e5L.
Počty platných číslic podle IEEE 754:
7 float , 15 double 19 long double.
ISO C++ norma rozsah exponentů pro všechny racionální datové typy je 10^{-38} až 10^{+38} a že přesnost typu float je nejméně šest platných číslic, přesnost typů double a long double pak nejméně deset platných číslic.
přetečení a **podtečení**

typ	**bitů**	**mant.**	**exp.**	rozsah absolutních hodnot
float	32	24	8	3.4×10^{-38} až $3.4 \times 10^{+38}$
double	64	53	11	1.7×10^{-308} až $1.7 \times 10^{+308}$
long double	80	64	15	3.4×10^{-4932} až $1.1 \times 10^{+4932}$

Alert (Bell)	G	pípnutí	
`\b`	Backspace	H	návrat o jeden znak
`\f`	Formfeed	L	odstránkování
`\n`	Newline	J	na začátek nového řádku
`\r`	Carriage return	M	na začátek aktuálního řádku
`\t`	Horizontal tab	I	na další tabelační pozici
`\v`	Vertical tab	K	stanovený přesun dolů
`\\`	Backslash		zpětné lomítko
`\'`	Single quote		apostrof
`\"`	Double quote		uvozovky
`\?`	Question mark		otazník
`\OOO`			znak zadaný osmičkově
`\xHH`			znak zadaný šestnáctkově

Znakové konstanty II

čeština UNICODE. datový typ wchar_t L'c'.
Konstantní řetězce Řetězec je posloupnost (pole) jednoho či více znaků.
"dve slova" "a" "Ahoj\n"
"Cela tato veta tvori jeden retezec."
UNICODE L"konstantni retezec"

Proměnné

paměťová místa přístupná prostřednictvím identifikátoru.

```
/**********************************
 * promenne a jejich pripadna inicializace
 **********************************/
int a, b, c, pocet = 0;
float x, prumer = 0.0, odchylka = 0.0;
double y;
```

Přímé pojmenování paměťového místa se nazývá proměnná. Nepřímému pojmenování se říká **ukazatel**.
adresový operátor &
dereference adresy *

ukazazatele I

```
// korektni ziskani adresy do ukazatele
int i, *pi;
pi = &i;
*pi = 123;

/**********************
 * Nasleduje chybny kod!
 * adresova fikce:
 **********************/
int *p; p = 1004; *p = 123;
```

Hodnoty	15	0	123	65
Adresy	1002	1003	1004	1005

ukazazatele II

```
/**********************************************
 * pokus o pouziti neinicializovaneho ukazatele
 * pozor, umyslna chyba:
 **********************************************/

int *pi;
*pi = 12345;
```

Operátory jazyka C++, 1

pri.	operátor	popis	asoc.	l-hodn.
1	( )	volání funkce	→	ano
1	[ ]	indexování	→	ano
1	.	přístup k položkám	→	ano
1	−>	přístup k položkám přes ukazatele	→	ano
1	::	rozlišení platnosti	→	ano
2	! ~	logická a bitová negace	←	ne
2	++ −−	inkrementace a dekrementace	←	ne
2	+ −	unární plus a mínus	←	ne
2	(typ)	přetypování	←	ne
2	∗	dereference ukazatele	←	ne
2	&	získání adresy	←	ne
2	sizeof	velikost v bajtech	←	ne
2	new	alokace paměti	←	ne
2	delete	uvolnění paměti	←	ne

Operátory jazyka C++, 2

pri.	operátor	popis	asoc.	l-hodn.
2	const_cast	přetypování	←	ano
2	static_cast	přetypování	←	ano
2	dynamic_cast	přetypování	←	ano
2	reinterpret_cast	přetypování	←	ano
2	typeid	identifikace typu	←	ne
3	.* ->*	deref. třídních ukazatelů	→	ano
4	* / %	násobení, dělení, modulo	→	ne
5	+ -	sčítání, odčítání	→	ne
6	<< >>	bitový posun vlevo, vpravo	→	ne
7	< > <= >=	menší než, větší než, menší nebo rovno, větší nebo rovno	→	ne
8	= !=	rovno, nerovno	→	ne
9	&	bitová konjunkce	→	ne

Operátory jazyka C++, 3

pri.	operátor	popis	asoc.	l-hodn.
10	^	bitová nonekvivalence	→	ne
11	|	bitová disjunkce	→	ne
12	&&	logická konjunkce	→	ne
13	||	logická disjunkce	→	ne
14	?:	podmíněný výraz	←	ano
15	=	přiřazení	←	ano
15	+= -= *= /= %=	složená přiřazení	←	ano
15	&= ^= |=	další složená přiřazení	←	ano
15	>>= <<=	další složená přiřazení	←	ano
16	,	postupné vyhodnocení	→	ano