Expressions

Donald Davendra

Department of Computing Science, FEI VSB-TU Ostrava

Expression

composition of **operators** and **operands**
C++: every operator except overloading
?:, ::, ., .*, sizeof, typeid, const_cast,
dynamic_cast, reinterpret_cast, static_cast

operator - function - the context, example $\ll$

Square root Pythogrous Therum

$$c = \sqrt{a^2 + b^2}$$

Operators Attributes

```
int a, b, c;
a = b = c = -1;
```

rvalue – **r-value**, *value of expression*
lvalue – **l-value**, *value of address*
ambiguous:

```
cc = cc++ * 2;
a[i] = i++;
```

Arithmetic Expressions

+, −, *, /, %

```
123 * 456 = 56088
295 /   2 =   147
295 %   2 =     1

watch this:
     20000 *       20001 = 400020000
 400020000 *          10 = -294767296
```

Arithmetic Expressions – List 1

```
void main() {
 int o1 = 123, o2 = 456,
     o3 = 295, v1, v2, v3;
 int c1 = 20000, c2 = 20001, vc;
 v1 = o1 * o2;
 v2 = o3 / 2;
 v3 = o3 % 2;

 cout << o1 << " * " << o2 << " = "
      << setw(5) << v1 << endl;
 cout << o3 << " /   2 = " << setw(5)
      << v2 << endl;
 cout << o3 << " %   2 = " << setw(5)
      << v3 << endl;
```

Arithmetic Expressions – List 2

```
  vc = c1 * c2;
  cout << endl << "watch this:" << endl;
  cout << setw(10) << c1 << " * " << setw(10)
       << c2 << " = " << vc << endl;
  cout << setw(10) << vc << " * " << setw(10)
       << 10 << " = " << vc * 10 << endl;
} // void main()
```

Mixed arithmetic expressions

```
  j = i = 5;
  j *= i;
  r = j / 3;
  x = j * 3;

  cout << "i=" << i << " j=" << j << " r="
       << r << " x=" << x << endl;
} // void main()
```

```
i=5 j=25 r=8 x=75
```

Boolean values and operators

C++ bool, true a false
C integers 1, 0
&&, ||, ! – *conjunction*, *disjunction*, *Negate*

A	B	!A	A && B	A || B
false	false	true	false	false
false	true	true	false	true
true	false	false	false	true
true	true	false	true	true

Relational operators

<, >, <=, >=, ==, !=

smaller than, bigger than, smaller than or equal to, bigger than or equal, equal and unequal

Bit Operators

pouze s celoselnmi hodnotami
<<, >>, &, |, ~, ^
shift left, *shift right*, bit *conjucture*, bit *disjuncture*, bit *negation*, bit *non-equivalence*

b1	b2	~b1	b1 & b2	b1 | b2	b1 ^ b2
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

Bit shift, MSB, LSB
– unsigned, integer division (multiplication) two,
– arithmetic shift, sign bit.
Bit Conjuction &, *disjunction* |, a *nonequivalence* ^ (and, or, xor) conduct a binary operation on each pair of corresponding bits integer operands.
Bit negation ~ is unary, bitwise complement.

32-bit environment:

```
  1 << 1 =      2   0x2
  1 << 7 =    128   0x80
 -1 >> 1 =     -1   0xffffffff
512 >> 8 =      2   0x2
 13  & 6 =      4   0x4
 13  | 6 =     15   0xf
 13  ^ 6 =     11   0xb
  2  & 1 =      0     0
  2  | 1 =      3   0x3
  2  ^ 1 =      3   0x3
```

op-bit01.cpp Part 1

```
int left7 = 1 << 7;
cout.setf(ios::showbase);
cout << "  1 << 1 = " << dec << setw(6)
     << left1 << hex << setw(6) << left1
     << endl;
cout << "  1 << 7 = " << dec << setw(6)
     << left7 << hex << setw(7) << left7
     << endl;

int rigth1 = -1 >> 1;
int rigth8 = 512 >> 8;
cout << " -1 >> 1 = " << dec << setw(6)
     << rigth1 << hex << setw(13)
     << rigth1 << endl;
cout << "512 >> 8 = " << dec << setw(6)
```

op-bit01.cpp Part 2

```
        << rigth8 << hex << setw(6)
        << rigth8 << endl;

int k = 13 & 6;
int d = 13 | 6;
int non = 13 ^ 6;
cout << " 13  & 6 = " << dec << setw(6)
     << k << hex << setw(6) << k << endl;
cout << " 13  | 6 = " << dec << setw(6)
     << d << hex << setw(6) << d << endl;
cout << " 13  ^ 6 = " << dec << setw(6)
     << non << hex << setw(6) << non << endl;

k = 2 & 1;
d = 2 | 1;
```

int-size.cpp

```
using namespace std;

void main() {
 unsigned int ui = ~0;
 int i = 1;
 while (ui >>= 1)
   i++;
 cout << "compiler uses " << i <<
 "bit integer representation" << endl;
} // void main()
```

```
compiler uses 32bit integer representation
```

Address operator

& is unary

```
variable i=123 is located from address: 0xbffff964
```

```
#include <iostream>

using namespace std;

void main() {
 int i = 123, *pi;

 pi = &i;
 cout << "variable i=" << i
  << " is located from address: "
  << pi << endl;
}
```

Reference – link

synonym, alias

```
int  i     = 0;
int & io = i;
io         = 2;
```

Comma operator

progressive evaluation, the lowest priority, from left to right.

Cast expression

(typ), const_cast , static_cast , dynamic_cast, reinterpret_cast
(typ) vyraz
r = j / 3;
r = (double) j / 3;

Conditional operator, op-cond.cpp

ternary

```
void main() {
 int i, abs_i;

 cout << endl << "Enter integer: ";
 cin >> i;

 abs_i = (i < 0) ? -i : i;
 cout << "abs(" << i << ") = "
      << abs_i << endl;
}
```

```
Enter integer: 123
abs(123) = 123
```

```
Enter integer: -123
abs(-123) = 123
```

Memory Management, new-del.cpp Part 1

C++ new a delete.
new object allocation – **constructor**

```
void main() {
 int *ptr = new int(3);
 *ptr += 7;
 cout << "*ptr = " << *ptr << endl;
 delete ptr;

 int count;
 cout << "How many variables shoud array have?: ";
 cin >> count;
```

Memory Management, new-del.cpp Part 2

```
  double *array = new double[count];
  int i = 0;
  while (i < count) {
          array[i] = i + (double) i / 100;
           cout << array[i] << endl;
           i++;
  } // while (i < count)
  delete [] array;
} // void main()
```

Memory Management, new-del.cpp output

```
*ptr = 10
How many variables shoud array have?: 5
0
1.01
2.02
3.03
4.04
```

Expression command, commands

term ended; ;

Empty statement

;

Block Block group of commands between { and }

compound statement

local declarations and definitions, local variables,

memory class auto

The area of the identifier

- At the file level starting point Declaration Declaration of force to the end of the file.
- Declaration on the level of argument has a range of functions from the point of declaration of the argument within the function definition to end a block outside the function definition. If not, and the function definition, the scope of the declaration ends with an argument declaration function.
- The declaration block is valid until the end of the block.

Namespace

solves problems with a potential conflict in the identifiers.
using namespace std;

custom:
namespace identifier list declaration
Operator double colon
::

namespace:: identifier

scope-name.cpp

```
namespace CPP_pro_zelenace {
 const char * id = "Saloun. C++ pro zelenace."
  " Neocortex 2003.";
}

void main() {
 const char *id = "body of main function";
 std::cout << id <<  std::endl; // local
 std::cout << ::id <<  std::endl; // global
 std::cout << "Good book: " <<
 CPP_pro_zelenace::id <<  std::endl;
} // void main()
```

scope-name.cpp input