

Operācijas (angļu val. *operators*)

#	Category	Operator	Associativity
1.	Highest	() [] -> :: .	3/4®
2.	Unary	! ~ + - ++ -- & * sizeof new delete	¬ 3/4
3.	Member access	. * ->*	3/4®
4.	Multiplicative	* / %	3/4®
5.	Additive	+ -	3/4®
6.	Shift	<< >>	3/4®
7.	Relational	< <= > >=	3/4®
8.	Equality	== !=	3/4®
9.	Bitwise AND	&	3/4®
10.	Bitwise XOR	^	3/4®
11.	Bitwise OR		3/4®
12.	Logical AND	&&	3/4®
13.	Logical OR		3/4®
14.	Conditional	?:	¬ 3/4
15.	Assignment	= *= /= %= += -= &= ^= = <<= >>=	¬ 3/4
16.	Comma	,	3/4®

182

Operācijas ar datiem

```
int x, y;
```

```
x = 4;
```

```
y = x | x >> 1 > 1; // y iegūst vērtību 5
```

```
int a = 20;
```

```
Izteiksmes 5 < a < 10 vērtība ir: 1
```

```
int i, j, m[4];
```

```
i = 2;
```

```
j = ++i; // i iegūst vērtību 3, j iegūst vērtību 3
```

```
i = 2;
```

```
j = i++; // i iegūst vērtību 3, j iegūst vērtību 2
```

```
i = 0;
```

```
m[i++] = 7; // m[0] iegūst vērtību 7, i iegūst vērtību 1
```

183

Operācijas ar datiem

```
char str[] = "*aa*bbb***cccc*****";
```

Noteikt, cik simbolu '*' ir simbolu virknē str.

operācijas * operands ir
"vecā", nepalielinātā s vērtība

```
int stars = 0;
char *s = str;
while (*s) stars += *s++ == '*'; // stars += (*(s++) == '*')

// "saprota māka" realizācija
int stars = 0;
int len = strlen(str);
for (int i = 0; i < len; ++i)
    if (str[i] == '*') ++stars;
```

184

Operācijas ar datiem

n Binārā operācijā, operandu aprēķināšanas secība nav noteikta, izņemot operācijas |, && un , (komats).

```
k = 5;
alpha = ++k + MyFun(k); // MyFun() saņems 5 vai 6?
```

```
int total, sum;
total = 0;
sum = (++total) + (total = 3);
cout << total << sum << endl; // 3 6
```

```
total = 0;
sum = (total = 3) + (++total);
cout << total << sum << endl; // 4 7
```

185

Operācijas ar datiem

- n Loģiskajās operācijās && un || operandu vērtības aprēķina no kreisās puses uz labo.
- n Var būt gadījumi, kad ne visas operandu izteiksmes tiek aprēķinātas!

```
if (a > 0 || ++k > 10)
    //ja a > 0, tad k nepalielinās un nepārbaudīs!
```

```
if (++x <= n && ++y <= m)
    //ja ++x > n, tad y nepalielinās un nepārbaudīs!
```

186

Operācijas ar datiem

- n Operācijas "komats" operandus aprēķina no kreisās puses uz labo. Tās vērtība un tips ir pēdējās aprēķinātās izteiksmes vērtība un tips.
- n Šo operāciju visbiežāk lieto, lai secīgi aprēķinātu vairākas izteiksmes vietās, kur C++ sintakse pieļauj tikai vienu izteiksmi.

```
izteiksme1, izteiksme2, ..., izteiksmen

for (i = 0, j = n; i <= n; ++i, --j)
    a[i] = b[j];
...

int someFun(int, int, int);
...
int p = someFun(i, (j = 1, j + 4), k);
// someFun() saņem 3 parametrus: i, 5, k
```

187

Operācijas new un delete

n Operācijas new vērtība ir attiecīgā tipa rādītājs uz iedalīto atmiņu:

```
new tips
new tips[izteiksme]
```

```
int *pp1, *pp2, *pp3;
pp1 = new int;
pp2 = new int[2048];
int n = 5;
pp3 = new int[1024 * n];
```

```
Triangle *pt = new Triangle;
```

n Operācija delete atbrīvo ar new iedalīto atmiņu:

delete rādītājs

```
delete pp1;
delete pp2;
delete pp3;
delete pt;
```

188

Operācijas new un delete

n Ieteikumi darbā ar rādītājiem:

- § Nelietot neinicializētu rādītāju;
- § Neatbrīvot ar delete atmiņu, kas nav iedalīta ar new;
- § Neizpildīt delete vairākkārtīgi vienam un tam pašam rādītājam.

```
#define NULL 0
```

```
char *buf = NULL;
```

```
...
```

```
buf = new char[size];
```

```
if (!buf) exit(1); //atmiņas iedalīšana nebija veiksmīga!
```

```
...
```

```
if (buf) { delete buf; buf = NULL; }
```

189

Funkcijas

n Funkcijas deklarācija:

```
tips vārds (parametru tipu saraksts);  
tips vārds (parametru deklarāciju saraksts);
```

n Funkcijas definīcija:

```
tips vārds (parametru deklarāciju saraksts)  
{ funkcijas ķermenis }
```

n Funkcijas ar mainīgu parametru skaitu:

```
int varFun(char *c, ...);
```

n Funkciju izsaucot, faktiskie parametri tiek pārveidoti atbilstoši deklarācijā norādītajiem tipiem un vērtības tiek ievietotas stekā.

n Funkcija savā darbā parametru vērtības izmanto no steka, t.i. izmanto faktisko parametru kopijas.

190

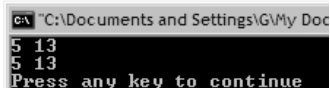
Funkcijas

n Uzdevums: Uzrakstīt funkciju, kas samaina vietām savu divu argumentu vērtības.

```
void main()  
{ void swap(int, int); // deklarācija  
  int x = 5, y = 13;  
  cout << x << " " << y << endl;  
  swap(x, y);  
  cout << x << " " << y << endl;  
}
```

```
void swap(int a, int b)  
{  
  int c = a; a = b; b = c;  
}
```

Kas tiks izvadīts uz ekrāna?



```
C:\Documents and Settings\G\My Doc  
5 13  
5 13  
Press any key to continue
```

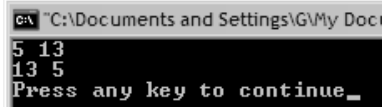
191

Funkcijas

n Uzdevums: Uzrakstīt funkciju, kas samaina vietām savu divu argumentu vērtības.

```
void main()
{
    int x = 5, y = 13;
    void swap(int*, int*); // deklarācija
    cout << x << " " << y << endl;
    swap(&x, &y);
    cout << x << " " << y << endl;
}
```

```
void swap(int *a, int *b)
{
    int c = *a; *a = *b; *b = c;
}
```



```
C:\Documents and Settings\G\My Documents
5 13
13 5
Press any key to continue_
```

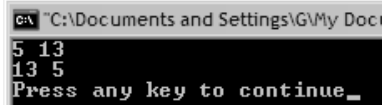
192

Funkcijas

n Uzdevums: Uzrakstīt funkciju, kas samaina vietām savu divu argumentu vērtības.

```
void main()
{
    int x = 5, y = 13;
    void swap(int&, int&); // deklarācija
    cout << x << " " << y << endl;
    swap(x, y);
    cout << x << " " << y << endl;
}
```

```
void swap(int& a, int& b)
{
    int c = a; a = b; b = c;
}
```



```
C:\Documents and Settings\G\My Documents
5 13
13 5
Press any key to continue_
```

193

Rekursīvas funkcijas

```
int pw(int n, int k)
{ if (k == 1) return n;
  else return n * pw(n, k - 1);
}
```

$$n^k = n * n^{k-1}$$

```
void main()
{
    int y;
    int x = 2;

    y = pw(x, 3);
    cout << y << endl;
}
```

194

Rekursīvas funkcijas

Uzlabota versija

```
int pw(int n, int k)
{
    if (k == 0)
        return 1;
    else if (k == 1)
        return n;
    else
        return n * pw(n, k - 1);
}
```

195