Streamy (proudy, roury)

Streamy jsou nový způsob vstupů a výstupů zavedený v jazyce C++.

Streamy pro vstup ze standardního zařízení a výstup na standardní zařízení

Jsou začleněny do standardního jmenného prostoru `std`. Při použití streamů musíme buďto uvést, že používáme standardní jmenný prostor:

```
using namespace std;
```

nebo v každém zápisu streamu uvést pomocí operátoru rozlišení :: jeho příslušnost ke standardnímu jmennému prostoru:

```
std::cout << "C++";
```

Streamy

cin	standardní vstupní stream (C: stdin)
cout	standardní výstupní stream (C: stdout)
cerr	standardní stream výstupu chybových zpráv (C: stderr)

Přetížení operátoru (operator overloading) je specifickým druhem polymorfismu. Při přetížení má operátor různou funkci (různé chování) pro různé datové typy operandů.

Pro výstupní streamy se používá přetížení operátoru **<<**:

```
#include <iostream>
using namespace std;

cout << výraz;

cout << 'C' << "++";                    // C++

cout << 0b1111 << endl;                 // 15
```

Od C++14 lze čísla zapisovat v binární soustavě použitím prefixu 0b nebo 0B.

```
int i=2;

cout << i << '+' << 1 << " je " << i+1;  // 2+1 je 3

cout << "3^5 = " << (3^5);              // 3^5 = 6

char c=65;

cout << c << ' ' << (int)c;             // A 65
```

Pro vstupní streamy se používá přetížení operátoru **>>**:

```
unsigned u;
float x;

cin >> u >> x;          // vstup z klávesnice: 5 3.14
```

```
cout << u << " " << x;   // 5 3.14
```

Formátování vstupů a výstupů – manipulátory

endl	vloží do výstupu nový řádek

Platnost manipulátoru začíná jeho uvedením ve streamu. Ukončení platnosti manipulátoru v závislosti na typu manipulátoru může být způsobem:

- uvedením manipulátoru, který jeho platnost ukončuje
- uvedením jiného manipulátoru, který nastavenou vlastnost mění
- jde o manipulátor, jehož platnost je omezena jen na výstup jedné hodnoty

dec	dekadická soustava
hex	hexadecimální soustava
oct	oktalová soustava

```
cout << dec << 10 << endl    // 10
     << hex << 10 << endl    // a
     << oct << 10 << endl;   // 12
cout << 10 << endl;          // 12
```

showbase	zobrazí označení číselné soustavy	**noshowbase**

```
cout << showbase << dec << 10 << endl    // 10
                 << hex << 10 << endl    // 0xa
                 << oct << 10 << endl;   // 012
```

showpos	zobrazí znak + u nezáporných čísel	**noshowpos**

```
cout << showpos   << 0 << endl    // +0
                  << 1 << endl    // +1
     << noshowpos << 2 << endl;   // 2
```

uppercase	velká písmena	**nouppercase**

```
cout << uppercase << showbase << hex << 200 << endl    // 0XC8
     << nouppercase                 << 200 << endl;    // 0xc8
```

setw	nastaví šířku výstupu

```
#include <iomanip>

cout << setw(5) << 10 << endl     //    10
                << 10 << endl;    // 10

cout << setw(5) << 1.2 << endl    //   1.2
```

```
                << 1.2 << endl;   // 1.2

cout << setw(5) << 100'000'000 << endl;   // 100000000
```

Od C++14 lze do zápisu čísla vložit apostrofy jako oddělovače jednotlivých řádů pro lepší přehlednost čísla.

left	zarovnání vlevo
right	zarovnání vpravo
internal	vložení výplňkových znaků dovnitř

```
cout << setw(5) << left     << -10 << endl    // -10
     << setw(5) << right    << -10 << endl    //   -10
     << setw(5) << internal << -10 << endl;   // -  10
```

setfill	nastaví výplňkový znak

```
cout << setfill('*') << setw(10) << 1000 << endl;   // ******1000

cout << setfill('0') << setw(10) << 1234 << endl;   // 0000001234
```

setprecision	nastaví přesnost

```
cout << setprecision(5) << 3.14159 << endl;      // 3.1416

cout << setprecision(5) << 3.14 << endl;         // 3.14

cout << setprecision(5) << 123456.789 << endl;   // 1.2346E+005
```

fixed	nastavení přesnosti se vztahuje na desetinná místa

```
cout << setprecision(4) << fixed << 3.14159 << endl;   // 3.1416

cout << setprecision(4) << fixed << 3.14 << endl;      // 3.1400
```

scientific	zobrazí číslo v semilogaritmickém tvaru

```
cout << scientific << 123.4 << endl;   // 1.234000e+002

cout << setprecision(2)
     << scientific << 123.4 << endl;   // 1.23e+002
```

boolalpha	vypíše hodnoty typu bool klíčovými slovy	**noboolalpha**

```
bool b = true;

cout << b                 << endl   // 1
     << boolalpha << b << endl;   // true
```

skipws	přeskočí na vstupu „bílé“ znaky (whitespaces)	**noskipws**

Implicitně je nastavena volba **skipws**.

```
char a,b,c,d,e,f;

cin >> a >> b >> c >> noskipws >> d >> e >> f;

cout << a << b << c << d << e << f << endl;
```

Vstup:
```
 123
 456
```

Výstup:
```
123
 4
```

ws	na aktuální pozici vstupu odstraní všechny „bílé znaky“

```
char a,b,c,d,e,f;

cin >> noskipws >> a >> b >> c >> ws >> d >> e >> f;

cout << a << b << c << d << e << f << endl;
```

Vstup:
```
1 2 3 4 5 6
```

Výstup:
```
1 23 4
```

Manipulátory používané při zápisu do souboru

ends	vloží do výstupu znak `'\0'`

unitbuf	vyrovnávací paměť je vyprázdněna po každém vložení	**nounitbuf**

flush	vyrovnávací paměť je vyprázdněna po vložení