základy vstupu a výstupu znaků v C++

- neřeší klíčová slova, ale knihovní funkce (součást normy)
- složitý mechanizmus, řešeno pomocí třídy, šablon
- použití (přetížení) operátorů bitových posunů << a >>

```
xstream & operator xx (xstream &, Typ& p) {  }

istream & operator >> (istream &, Typ& p) {  }
ostream & operator << (ostream &, Typ& p) {  }
```

- díky přetížení operátoru není nutné při volání specifikovat/kontrolovat typy (správný operátor hledá překladač)
- pro námi definované typy je nutno tyto operátory napsat

```
struct TPole2D {};

istream& operator>> (istream &is,      TPole2D & p)
{… return is;}
ostream& operator<< (ostream &os,const TPole2D & p)
{… return os;}
```

Pozn.: znak & je symbol pro referenci, nový způsob předávání parametrů

základy vstupu a výstupu znaků v C++

- realizace pomocí streamů – objekt propojující program a V/V zařízení
- pro práci s konzolou slouží předdefinované objekty **`cin, cout, cerr`** uvedené v knihovně **`<iostream>`**
- endl (manipulátor) slouží k vynucení vytištění textu a odřádkování
- objekty jsou v prostoru **`std::`**. Použití **`std::cout, std::endl`**.
- jelikož prvním operandem je "cizí" objekt, jedná se o (friend) funkce

```
int i;
double j;
char k[]=“Ahoj“;
TPole2D p;

cin >> i >> j >> k;
cout << i << ”text” << j << k << p << endl;
```

cout, cin, endl jsou v prostoru **std**, pak správný přístup z našeho programu je **std::cout**. Použijeme-li na začátku modulu **using namespace std**, pak to znamená, že k proměnným z prostoru **std** můžeme přistupovat přímo a tedy psát pouze **cout**
Lépe je být konkrétní a tedy použít **using std::cout; using std::cin; using std::endl;**