**Mis on #include <iostream> C++ keeles?**

Kuna C++ põhierinevus C-keelega võrreldes on objektiklasside olemasolu, siis on alternatiivse standardlahendusena realiseeritud loomulikult ka objektiklassidel põhinev andmeedastus. Standardsed andmevood ‘cin’ ja ‘cout’ on loodud klasside ‘istream’ ja ‘ostream’ baasil, andmete liikumise suuna määravad üledefineeritud nihutamis­tehted ‘<<’ ja ‘>>’. Andmete küsimisel ja lihtsal väljastamisel on objektide kasutamine funktsioonidest mugavam, kuid väljundi vormindamine on üksjagu tülikas.

iostream tähistab standardset sisend-väljundvoogu. **#include iostream** deklareerib objektid, mis juhivad standardsetest voogudest lugemist ja nendesse kirjutamist. Teisisõnu, **iostream** teek on **objektorienteeritud teek**, mis pakub voogude abil sisend- ja väljundfunktsioone.

**Voog on baitide jada**. Võite seda pidada abstraktsiooniks, mis esindab seadet. Selle abstraktsiooni kaudu saate seadmes I/O-operatsioone teha. Peate lisama iostreami päisefaili C++ programmi sisendiks ja väljundiks.

**Süntaks**

**#include< iostream>** pakub enimkasutatavaid standardseid sisend- ja väljundvooge**, cin** ja **cout.** Nende kasutamise süntaks on järgmine:

**1. Standardne väljundvoog -- cout**

See on **ostream klassi** eksemplar.

See toodab väljundit standardsele väljundseadmele, st kuvarile.

Peame kasutama voo sisestamise **operaatorit <<,** et sisestada andmed standardsesse väljundvoo väljundisse, mis tuleb ekraanil kuvada

cout << variable\_name;

cout << variable1 << variable2 << ... ;

Sellist mitme voo sisestamise operaatori kasutamist ühe coutiga nimetatakse kaskaadseks. See aitab samale reale printida mitu muutujat kõrvuti.

Mis siis, kui soovite printida eraldi ridadele ja kaskaadiga samaaegselt? Seda saab teha kahel viisil ja mõlemat saab kasutada ka stringliteraalidega:

Kasutades \n, uut reamärki

cout << variable1 << '\n' << variable2

See prindib muutuja1 väärtuse ja prindib uue rea. Lõpuks prindib see muutuja2 väärtuse uuele reale.

Endl, manipulaatori kasutamine

cout << variable1 << endl << variable2;

C++ manipulaator muudab sisend- või väljundvoo käitumist. endl manipulaator on osa std-nimeruumist ja seda saab kasutada ka uue rea sisestamiseks.

cout << "Any string literal";

**2. Standardne sisendvoog -- cin**

See on **istream** klassi eksemplar.

See loeb sisendit standardsest sisendseadmest, st klaviatuurilt.

Klaviatuuriga sisestatud andmete eraldamiseks peame kasutama voo ekstraheerimise **operaatorit >>.**

cin >> variable\_name;

cin >> variable1 >> variable2 >> ... ;

#include <iostream>

using namespace std;

int main() {

 string name = "John Doe";

 *// You can use cout with a variable and also insert a line character,*

 cout << name << '\n';

 *// Or you can also insert the output directly into the screen.*

 cout << "This is a sample text inserted into the output screen...";

 return 0;

}

#include <iostream>

using namespace std;

int main() {

 int a, b, sum;

 cout << "Enter first number: ";

 *// Read input from the standard input device, usually the keyboard.*

 cin >> a;

 cout << "Enter second number: ";

 *// Read another input from the keyboard.*

 cin >> b;

 sum = a + b;

 cout << "The sum of the two numbers is " << sum;

 return 0;

}

Seega #include< iostream> pakub C++ põhilisi I/O-funktsioone. Kõik I/O toimingud kasutavad vooge.

**Voog on iostream klasside keskne kontseptsioon**. Vooobjekt on nutikas fail, mis toimib baitide allikana ja sihtkohana.

Enimkasutatavad iostream klassi objektid on **cin** ja **cout**, mis kasutavad standardseid sisend- ja väljundvooge.

Muud globaalsed vooobjektid, mille annab, sealhulgas iostream, liigitatakse kitsasteks ja laiateks märgivoogudeks.

Kitsas märk kasutab andmetüübina char, laia tähemärk aga wchar\_t.

Kitsas märk sisaldab **cin, cout, cerr ja clog**, samas kui lai märk koosneb wcin, wcout, wcerr ja wclog.

Cout, cerr, clog ja nende laiad vasted on vastavalt ostream ja wostream.

cin ja wcin on vastavalt istream ja wistream.

Cerr on puhverdamata, samas kui ummistus on puhverdatud ja tõhusam. Seetõttu kasutatakse **cerri** kriitiliste vigade kirjutamisel, ummistus aga enamasti logimiseks. Sama kehtib ka wcerri ja wclogi kohta.

#include iostream tuleks eelistada cstd-iole, kuna see on tüübikindel, vähendab vigu, on päritav ja laiendatav.

#include <iostream>

int main() {

 std::cout << "Welcome to Scaler Topics!";

 return 0;

}

#include <iostream>

using namespace std;

int main() {

 int num;

 cout << "Enter a number: ";

 // take integer input

 cin >> num;

 cout << "You entered: " << num;

 return 0;

}

#include <iostream>

using namespace std;

int main() {

 char name[20], address[20];

 cout << "Name: ";

 // use cin with getline()

 cin.getline(name, 20);

 cout << "Address: ";

 cin.getline(address, 20);

 cout << endl << "You entered " << endl;

 cout << "Name = " << name << endl;

 cout << "Address = " << address;

 return 0;

}