Artskart2sosi

Konsollprogram for konvertering av csv-filer til sosi 4.0

Brukermanual

Nedre Eiker kommune
Geodataavdelingen

Bakgrunn

Nedre Eiker kommune har et stort fokus på temakart i sin saksbehandling. I samband med en presentasjon sist år, ble det fremmet en idé om å implementere Artsdatabankens rødlisteregistreringer i kommunens kartsystem. Det finnes to tilnærminger til dette: enten ta i bruk Artsdatabankens wms/wfs-tjeneste, eller laste ned csv-filer og konvertere disse til fullverdige kartfiler på sosi-formatet.

For å øke fleksibiliteten ved visualisering og arealanalyse, har vi valgt siste alternativ. Det er utviklet et lite konsollprogram som forenkler jobben.

Konverteren gjøres tilgjengelig for andre i håp om at den kan være nyttig.

Mjøndalen, 27 april 2009
Rune Bratlie

Vilkår for bruk

Dette er fri programvare. Enhver kan fritt ta i bruk, endre og distribuere programvaren med tilhørende kildekode og dokumentasjon.

Denne friheten gjelder kun programvare og kildekode, ikke de artsdata programvaren håndterer. Ved nedlasting av artsdata må brukeren akseptere Artsdatabankens vilkår vedrørende kildehenvisning og sitering. Overholdelse av disse vilkårene er en forutsetning for å ta konverteren i bruk.

Konverteren stilles til disposisjon som den er. Det gis ingen brukerstøtte eller garantier utover dette dokumentet. Programvaren benyttes på eget ansvar.

Dersom du ønsker å forbedre kildekoden, eller legge på et grafisk brukergrensesnitt, er du velkommen til å melde inn endringene til rune.bratlie@nedre-eiker.kommune.no. Du vil da bli kreditert i dette dokumentet.

Konvertering – trinn for trinn

Konvertering foretas i tre hovedtrinn. Før endelig konvertering finner sted, må det etableres en ini-fil. Ini-filen må tilpasses hos den enkelte bruker.

Geografisk søk

- Gjør et geografisk søk på www.artskart.artsdatabanken.no
- Klikk fanen *Objektinfo* når du er fornøyd med søkeresultatet

Last ned søkeresultatet til en csv-fil

- Klikk *Eksporter data til Excel* nederst på skjermen
- Les og aksepter vilkårene for nedlasting
- Velg alternativet *Last ned foredlede data*
- Velg alternativet *semikolon* som skilletegn
- Klikk *Eksporter data som CSV fil*

Konverter csv-filen til sosi

- Åpne csv-filen i en teksteditor, som f.eks notisblokk (notepad)
- Lagre den på nytt med ansi tegnkoding
- Kjør konverteren

Ini-filen

Første gang programmet kjøres blir det opprettet en ini-fil på programkatalogen. Filen inneholder standard miljøvariabler som bør kontrolleres før sosi-konvertering tar til. De viktigste variablene er beskrevet nedenfor.

...KOORDSYS	Koordinatsystem i henhold til sosi.
...MALEMETODE	Målemetode i henhold til sosi.
...NOYAKTIGHET	Nøyaktighet angitt i hele meter. Objekter stedfestet bedre enn denne verdien blir konvertert.
..KOMM	Kommunenr.
modus	Dersom modus = **konsoll**, vil programmet vise standard info og stoppe opp for å vise kjøreresultat til skjerm. Dersom modus settes til **batch**, vil programmet avslutte automatisk når konverteringen er fullført.
csv	Full sti til csv-filen. Valgfri.
sosi	Full sti til sosi-filen. Valgfri.

De øvrige miljøvariablene bør betraktes som statiske. Tallvariabler, som **Collector** henviser til kolonnenummer i csv-filen. Dersom csv-formatet endres, er det bare å endre kolonnenummeret i ini-filen. Vær oppmerksom på at nummereringen starter på 0.

Programargumenter

Alle ini-variabler kan også benyttes som programargumenter. Dette gjør det mulig å overstyre ini-filen ved konvertering. Programargumenter må benytte følgende syntax:

variabel=verdi
csv=c:\temp\data.csv
..KOMM=0625
..KOMM="0625 0626"

Det skal ikke benyttes mellomrom sammen med likhetstegnet. Argumentnavnet må skrives eksakt som i ini-filen. Argumentenes rekkefølge er likegyldig. Dersom konverteren ikke ligger i samme mappe som csv-filen, må det angis full sti til alle filer.

Følgende programkjøring leser filen arter.csv, henter ut alle objekter stedfestet bedre enn 1000m, og skriver dem til filen artskart_1000.sos. Her ligger konverteren og csv-filen på samme katalog:

artskart2sosi csv=arter.csv sosi=artskart_1000.sos ...NOYAKTIGHET=1000

Et annet aktuelt argument er **..KOMM**. Dersom konverteringen alltid omfatter artsdata i samme kommune, er det bare å legge inn kommunenummeret i ini-fila. Dersom kommunenummeret derimot varierer, er det bedre å legge det til som et programargument.

Hvordan kjøre programmet?

Konverteren kan kjøres på ulike måter.

Statisk konvertering

Ved statisk konvertering er samtlige miljøvariabler, inkludert full sti til csv- og sosi-fil satt i ini-filen. Filene må da hete det samme, og ha samme plassering ved hver konvertering. Konverteringen gjøres ved å dobbeltklikke på programfilen. Eventuelle tidligere sosifiler blir overskrevet.

Batch-konvertering

Batch-konvertering er hensiktsmessig dersom en ønsker å konvertere flere flere filer i en operasjon. Alle konverteringene samles i en bat-fil. Eksemplet nedenfor viser innholdet i en slik bat-fil. Her ligger konverteren og csv-filen på samme katalog:

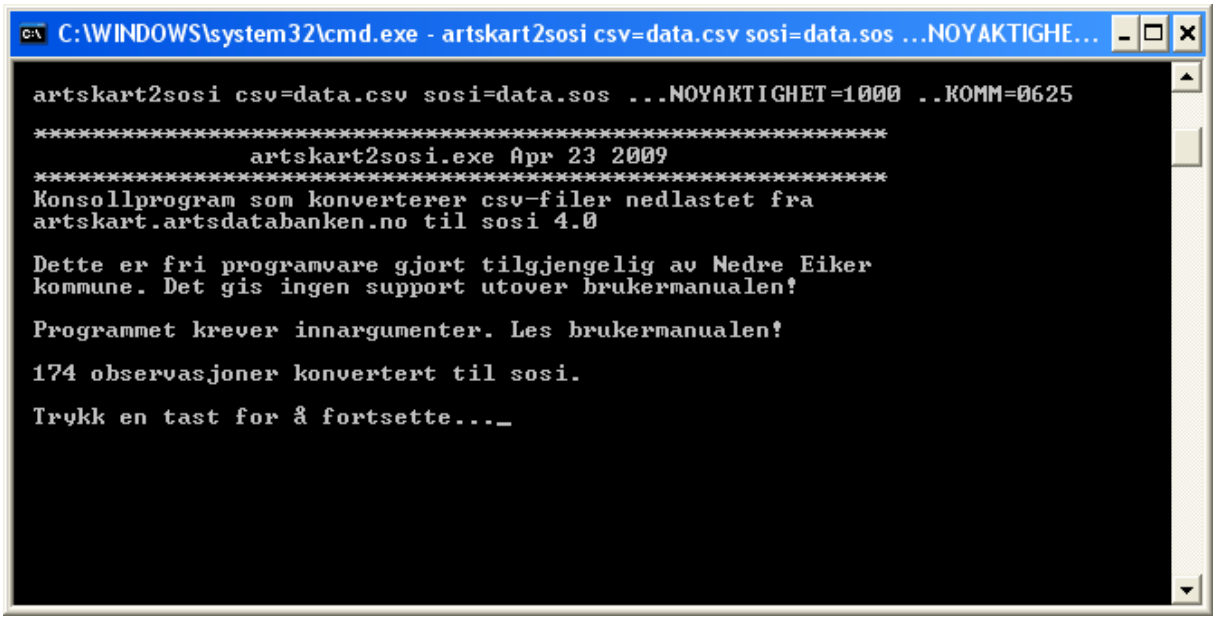
```
@echo off
artskart2sosi csv=arter.csv sosi=artskart_100.sos ...NOYAKTIGHET=100 ..KOMM=0625 modus=batch
artskart2sosi csv=arter.csv sosi=artskart_500.sos ...NOYAKTIGHET=500 ..KOMM=0625 modus=batch
artskart2sosi csv=arter.csv sosi=artskart_1000.sos ...NOYAKTIGHET=1000 ..KOMM=0625 modus=batch
```

Her benyttes **modus=batch** for å overstyre ini-filen, hvor programmodus for tilfellet er satt til **konsoll**. Programmet vil dermed ikke stoppe opp for å vise kjøreresultatet til brukeren.

Konsoll/CGI

Det siste alternativet er å skrive alle argumentene direkte i konsollvinduet som på bildet nedenfor. For å åpne konsollvinduet velger du Kjør… på Start-menyen. Skriv CMD og trykk enter. I eksemplet ligger alle filer på samme katalog.

Dersom konverteren ønskes kjørt som et cgi-script, bør **modus** endres til **batch** i ini-filen. Alle dynamiske variabler må dessuten gis som programargumenter.



Kjente feil og mangler

- Programmet terminerer dersom csv-filen ikke lagres med ansi tegnkoding
- Programmet håndterer ikke løsrevne apostrofer utenom tekststrenger

Kildekode

Kildekoden er skrevet i standard C++. Koden omfatter filene main.cpp, artskart2sosi.h (deklarasjonsfil) og artskart2sosi.cpp (definisjonsfil). Koden er kommentert fortløpende.

```
/*
 * main.cpp
 *
 * Created by Rune Bratlie on 27/04/09.
 * Copyright 2009 Nedre Eiker kommune. All rights reserved.
 *
 */


#include "artskart2sosi.h"

int main(int argc, char **argv) {

	// leser ini-fil og innargumenter
	ini ini;
	ini.sett();
	ini.sett(argc, argv);

	// skriver programinfo til skjerm
	programInfo(ini);

	// leser csv-fil
	csv csv;
	if(csv.sett(ini)) {

		// ini og csv ok, etablerer sosi-fil
		sosi sosi;
		sosi.sett(ini);
		sosi.skrivHode(csv, ini);
		sosi.skrivKropp(csv, ini);
		sosi.skrivSlutt();

	}

	// skriver resultat til skjerm
	programResultat(csv.antall(), ini);

	return 0;
}
```

```
/*
 * artskart2sosi.h
 *
 * Created by Rune Bratlie on 27/04/09.
 * Copyright 2009 Nedre Eiker kommune. All rights reserved.
 *
 */


#ifndef ARTSKART2SOSI_H
#define ARTSKART2SOSI_H

#include <fstream>
#include <vector>
#include <string>
#include <map>
#include <iostream>
#include <algorithm>
#include <ctime>

using std::ifstream;
using std::ofstream;
using std::string;
using std::vector;
using std::ios;
using std::endl;
using std::cout;

#define EXE "artskart2sosi.exe";
#define INI "artskart2sosi.ini";


/*************************************************************************
 * ini: klasse som administrerer programinnstillinger med en tilhørende
 * ini-fil. Dersom filen ikke eksisterer, oppretter klassen en ny
 * standardfil. Ini-filen må tilpasses hos den enkelte bruker.
 *
 * Det kan kun tilordnes en ini-fil til hvert klasseobjekt. Ini-fila kan
 * overstyres med sett(int, char), og eventuelt overskrives med skriv().
 *************************************************************************/

class ini {

        typedef std::map< string, string, std::less< string > > initab;

public:
        ini(const string &fil = INI);
        bool sett();
        bool sett(int argc, char **argv);
        bool skriv();
        string elem(const string &index);

private:
        string fil;
        initab tab;

}; // ini


/*********************************************************************
 * artsfunn: klasse som lagrer egenskaper ved et enkelt artsfunn.
 * STL-kontainere av typen artsfunn kan sorteres etter stigende nord,
 * øst og nordøst.
 *********************************************************************/

class artsfunn {

public:
        artsfunn();
        void catalogNumber(const string &s);
        void collectionCode(const string &s);
        void collectionName(const string &s);
        void collector(const string &s);
        void coordinatePrecision(const string &s);
        void dayCollected(const string &s);
        void institutionCode(const string &s);
        void institutionName(const string &s);
```

```
        void locality(const string &s);
        void monthCollected(const string &s);
        void norskNavn(const string &s);
        void scientificName(const string &s);
        void status(const string &s);
        void url(const string &s);
        void utmNord(const string &s);
        void utmOst(const string &s);
        void utmSone(const string &s);
        void yearCollected(const string &s);

        int    utmNord() const;
        int    utmOst() const;
        int    utmSone() const;
        string collectionCode() const;
        string collectionName() const;
        string collector() const;
        string coordinatePrecision() const;
        string dayCollected() const;
        string institutionCode() const;
        string institutionName() const;
        string locality() const;
        string monthCollected() const;
        string norskNavn() const;
        string scientificName() const;
        string status() const;
        string url() const;
        string yearCollected() const;
        string catalogNumber() const;

        bool artsfunn::operator==(const artsfunn &arg) const;
        bool artsfunn::operator<(const artsfunn &arg) const;

private:
        string CatalogNumber;
        string CollectionCode;
        string CollectionName;
        string DayCollected;
        string MonthCollected;
        string NorskNavn;
        string ScientificName;
        string Status;
        string URL;
        int    UTMnord;
        int    UTMost;
        int    UTMsone;
        string YearCollected;
        string InstitutionCode;
        string InstitutionName;
        string CoordinatePrecision;
        string Locality;
        string Collector;

};  // artsfunn


/*********************************************************************
 * csv: klasse som leser inn csv-filer nedlastet fra artskart (artskart.
 * artsdatabanken.no). Filen må være semikolonseparert og lagret med ANSI
 * tegnsett.
 *
 * Klassen leser kun inn artsfunn registrert med nøyaktighet bedre
 * eller lik ini-egenskapen ...NOYAKTIGHET.
 *
 * Artsfunn leses inn i en STL-kontainer. Antall objekter i
 * konteineren kan hentes ut med antall().
 *
 * Medlemsfunksjonen elem(int) returnerer artsfunn på gitt index
 * dersom index eksisterer og er mindre enn antall().
 *********************************************************************/

class csv {

public:
        csv();
        ~csv();
        bool sett(ini &ini);
```

```
        void minMaxNord(int &min, int &max);
        void minMaxOst(int &min, int &max);
        unsigned int antall() const;
        artsfunn elem(unsigned int index) const;

private:
        string kolonne(string &streng, string &kolonne);
        vector<artsfunn> tab;

}; // csv


/*************************************************************************
 * sosi: klasse som genererer en sosifil fra et csv-objekt med innleste
 * artfunn.
 *
 * Klassemedlemmet ref holder antall objekter skrevet til sosifilen. Filen
 * holdes åpen inntil skrivSlutt() kalles. Dette gjør det mulig å skrive
 * innholdet i flere csv-objekter til sosifilen, forutsatt at innholdet
 * ligger innenfor begrensningene i sosifilens hode.
 *************************************************************************/

class sosi {

public:
        sosi();
        bool sett(ini &ini);
        void skrivHode(csv &csv, ini &ini);
        void skrivKropp(csv &csv, ini &ini);
        void skrivSlutt();
        unsigned int antall() const;

private:
        int ref;
        int presisjon;
        int cos45(int p) const;
        string dato(char *format) const;
        string erstatt(const string &streng, const string &soek, const string &erstatt) const;
        string versaler(const string &streng) const;
        ofstream fil;

        template <class T>
        void skriv( const T &arg, bool mellomrom = false, bool linjeskift = false );

}; // sosi


/*****************************************************************
 * Nyttefunksjoner
 *****************************************************************/

bool stigendeNord(const artsfunn &arg1, const artsfunn &arg2);
bool stigendeOst(const artsfunn &arg1, const artsfunn &arg2);
void programInfo( ini &ini);
void programResultat(unsigned int, ini &ini);
void fylltegn(int antall, char tegn, bool linjeskift);

#endif
```

```
/*
 * artskart2sosi.cpp
 *
 * Created by Rune Bratlie on 27/04/09.
 * Copyright 2009 Nedre Eiker kommune. All rights reserved.
 *
 */


#include "artskart2sosi.h"

/**************************************************************
 * Class ini
 **************************************************************/

ini::ini(const string &filnavn) {
	fil = filnavn;
}

bool ini::sett() {

	// åpner ini-fil
	ifstream inn;
	inn.clear();
	inn.open(fil.c_str(), ios::in);

	// ini-fil lar seg ikke åpne
	if(!inn)
		return skriv();

	// ini-fil lar seg åpne, leser ini-fil
	while(inn.good()) {

		string linje;
		getline(inn, linje);
		string param = linje.substr(0, linje.find(' ', 0));
		string verdi = linje.substr(linje.find(' ', 0) + 1);
		tab.insert(initab::value_type(param, verdi));

	}

	// lukker fila
	inn.close();
	return true;

}

bool ini::sett(int argc, char **argv) {

	// ingen programargumenter gitt
	if(argc == 1)
		return false;

	// programargumenter gitt
	for(int i = 1; i < argc; i++) {

		// gjør om argv til string
		string arg;
		arg = argv[i];

		// finner =
		// !!!ingen feilsjekk her!!!
		int pos = arg.find('=',0);

		// programargumentet er allerede definert i ini-fil - sletter det
		initab::const_iterator j = tab.find(arg.substr(0, pos));

		if(j != tab.end())
			tab.erase(j);

		// setter programargumentet
		tab.insert(initab::value_type(arg.substr(0, pos), arg.substr(pos+1)));

	}

	return true;
```

```
}

bool ini::skriv() {

	// standard csv-innstillinger
	// tallene angir kolonnenummer i csv-fila
	// første filkolonne = 0
	tab.insert(initab::value_type("InstitutionCode", "2"));
	tab.insert(initab::value_type("InstitutionName", "3"));
	tab.insert(initab::value_type("CollectionCode", "4"));
	tab.insert(initab::value_type("CollectionName", "5"));
	tab.insert(initab::value_type("CatalogNumber", "6"));
	tab.insert(initab::value_type("ScientificName", "7"));
	tab.insert(initab::value_type("NorskNavn", "9"));
	tab.insert(initab::value_type("Collector", "26"));
	tab.insert(initab::value_type("YearCollected", "27"));
	tab.insert(initab::value_type("MonthCollected", "28"));
	tab.insert(initab::value_type("DayCollected", "29"));
	tab.insert(initab::value_type("Locality", "36"));
	tab.insert(initab::value_type("CoordinatePrecision", "39"));
	tab.insert(initab::value_type("UTMsone", "57"));
	tab.insert(initab::value_type("UTMost", "58"));
	tab.insert(initab::value_type("UTMnord", "59"));
	tab.insert(initab::value_type("Status", "64"));
	tab.insert(initab::value_type("URL", "66"));

	// oppretter standard sosi-innstillinger
	tab.insert(initab::value_type("..TEGNSETT", "ISO8859-10"));
	tab.insert(initab::value_type("..SOSI-VERSJON", "4.0"));
	tab.insert(initab::value_type("..SOSI-NIVA", "4"));
	tab.insert(initab::value_type("..PROSESS_HISTORIE",
		"Biodiversitetsdata gjort tilgjengelig av: <INSTITUSJON>, (Nedlastet gjennom
Artskart, artskart.artsdatabanken.no, <DATO>)."));
	tab.insert(initab::value_type("..OBJTYPE_punkt", "BmArtOmråde"));
	tab.insert(initab::value_type("..OBJTYPE_kurve", "BmArtGrense"));
	tab.insert(initab::value_type("..OBJTYPE_flate", "BmArtOmråde"));
	tab.insert(initab::value_type("..LINK", "http://artskart.artsdatabanken.no/"));
	tab.insert(initab::value_type("..KOMM", "0625"));
	tab.insert(initab::value_type("...SYNBARHET", "0"));
	tab.insert(initab::value_type("...NOYAKTIGHET", "100"));
	tab.insert(initab::value_type("...MALEMETODE", "99"));
	tab.insert(initab::value_type("...KOORDSYS", "23"));

	// øvrige innstillinger
	string programinfo;
	programinfo = "Generert i ";
	programinfo += EXE;
	programinfo += " (Nedre Eiker kommune, Geodataavdelingen).";

	tab.insert(initab::value_type("Programinfo", programinfo));
	tab.insert(initab::value_type("Bruksinfo",
		"Kvalitet og fullstendighet kan ikke garanteres. Brukere anvender disse data på
egen risiko."));
	tab.insert(initab::value_type("modus","konsoll"));
	tab.insert(initab::value_type("csv", ""));
	tab.insert(initab::value_type("sosi", ""));

	// åpner standard ini-fil
	ofstream ut;
	ut.clear();
	ut.open(fil.c_str(), ios::out);

	// filen lar seg ikke åpne
	if(!ut)
		return false;

	// filen lar seg åpne
	for(initab::const_iterator i = tab.begin(); i != tab.end(); ++i)
		ut << i->first << " " << i->second << endl;

	// lukker filen
	ut.close();

	return true;

}
```

```
string ini::elem(const string &index) {

	// returnerer ini-variabelen på index, dersom eksisterer
	return tab[index];

}


/*********************************************************************
 * Class artsfunn
 *********************************************************************/

artsfunn::artsfunn() {
}

void artsfunn::institutionCode(const string &s) {
	InstitutionCode = s;
}

void artsfunn::institutionName(const string &s){
	InstitutionName = s;
}

void artsfunn::collectionCode(const string &s) {
	CollectionCode = s;
}

void artsfunn::collectionName(const string &s){
	CollectionName = s;
}

void artsfunn::catalogNumber(const string &s) {
	CatalogNumber = s;
}

void artsfunn::yearCollected(const string &s) {
	YearCollected = s;
}

void artsfunn::monthCollected(const string &s) {
	MonthCollected = s;
}

void artsfunn::dayCollected(const string &s) {
	DayCollected = s;
}

void artsfunn::collector(const string &s){
	Collector = s;
}

void artsfunn::norskNavn(const string &s) {
	NorskNavn = s;
}

void artsfunn::scientificName(const string &s) {
	ScientificName = s;
}

void artsfunn::status(const string &s) {
	Status = s;
}

void artsfunn::url(const string &s) {
	URL = s;
}

void artsfunn::utmNord(const string &s) {
	UTMnord = atoi(s.c_str());
}

void artsfunn::utmOst(const string &s) {
	UTMost = atoi(s.c_str());
}

void artsfunn::utmSone(const string &s) {
	UTMsone = atoi(s.c_str());
```

```
}

void artsfunn::coordinatePrecision(const string &s) {
	CoordinatePrecision = s;
}

void artsfunn::locality(const string &s) {
	Locality = s;
}

string artsfunn::catalogNumber() const {
	return CatalogNumber;
}

string artsfunn::collectionCode() const {
	return CollectionCode;
}

string artsfunn::collector() const{
	return Collector;
}

string artsfunn::institutionName() const {
	return InstitutionName;
}

string artsfunn::collectionName() const {
	return CollectionName;
}

string artsfunn::yearCollected() const {
	return YearCollected.length() == 4 ? YearCollected : "1900";
}

string artsfunn::monthCollected() const {

	// tester på antall tegn
	switch(MonthCollected.length()) {

	   case(0):
		return "01";

	   case(1):
		if(MonthCollected == "0")
			return "01";

		else
			return "0" + MonthCollected;

	   default:
		return MonthCollected;

	};

}

string artsfunn::dayCollected() const {

	// tester på antall tegn
	switch(DayCollected.length()) {

	   case(0):
		return "01";

	   case(1):
		if(DayCollected == "0")
			return "01";

		else
			return "0" + DayCollected;

	   default:
		return DayCollected;

	};

}
```

```
string artsfunn::norskNavn() const {
	return NorskNavn;
}

string artsfunn::scientificName() const {
	return ScientificName;
}

string artsfunn::status() const {
	return Status;
}

string artsfunn::url() const {
	return URL;
}

int artsfunn::utmNord() const {
	return UTMnord;
}

int artsfunn::utmOst() const {
	return UTMost;
}

int artsfunn::utmSone() const {
	return UTMsone;
}

string artsfunn::institutionCode() const {
	return InstitutionCode;
}

string artsfunn::coordinatePrecision() const {
	return CoordinatePrecision.empty() ? "*" : CoordinatePrecision;
}

string artsfunn::locality() const {
	return Locality;
}

bool artsfunn::operator==(const artsfunn &arg) const {

	// sjekker om de viktigste klassemedlemmene er identiske
	return (CatalogNumber  == arg.CatalogNumber &&
			ScientificName == arg.ScientificName &&
			Status         == arg.Status &&
			UTMnord        == arg.UTMnord &&
			UTMost         == arg.UTMost &&
			UTMsone        == arg.UTMsone);

}

bool artsfunn::operator<(const artsfunn &arg) const {

	// sortert stigende nord
	if(UTMnord < arg.UTMnord)
		return true;

	// sortert stigende nord + øst
	else if (UTMnord == arg.UTMnord && UTMost < arg.UTMost)
		return true;

	// usortert
	else return false;

}


/*****************************************************************
 * Class csv
 *****************************************************************/

csv::csv() {
}

csv::~csv() {
```

```
	tab.clear();
}

bool csv::sett(ini &i) {

	// sjekker gyldig filnavn
	if(i.elem("csv").empty())
		return false;

	// åpner csvfil
	ifstream inn;
	inn.clear();
	inn.open(i.elem("csv").c_str(), ios::in);

	// fil lar seg ikke åpne
	if(!inn)
		return false;

	// finner hjørneradius
	int pres = atoi(i.elem("...NOYAKTIGHET").c_str());

	// leser inn første linje
	string linje;
	getline(inn, linje);

	// løper gjennom hele filen og leser inn kolonner definert i ini-filen
	while(!inn.eof()) {
		getline(inn, linje);
		artsfunn reg;
		reg.institutionCode(kolonne(linje, i.elem("InstitutionCode")));
		reg.catalogNumber(kolonne(linje, i.elem("CatalogNumber")));
		reg.collectionCode(kolonne(linje, i.elem("CollectionCode")));
		reg.dayCollected(kolonne(linje, i.elem("DayCollected")));
		reg.monthCollected(kolonne(linje, i.elem("MonthCollected")));
		reg.norskNavn(kolonne(linje, i.elem("NorskNavn")));
		reg.scientificName(kolonne(linje, i.elem("ScientificName")));
		reg.status(kolonne(linje, i.elem("Status")));
		reg.url(kolonne(linje, i.elem("URL")));
		reg.utmNord(kolonne(linje, i.elem("UTMnord")));
		reg.utmOst(kolonne(linje, i.elem("UTMost")));
		reg.utmSone(kolonne(linje, i.elem("UTMsone")));
		reg.yearCollected(kolonne(linje, i.elem("YearCollected")));
		reg.coordinatePrecision(kolonne(linje, i.elem("CoordinatePrecision")));
		reg.locality(kolonne(linje, i.elem("Locality")));
		reg.collector(kolonne(linje, i.elem("Collector")));
		reg.institutionName(kolonne(linje, i.elem("InstitutionName")));
		reg.collectionName(kolonne(linje, i.elem("CollectionName")));

		// populerer tabellen dersom registreringen har ønsket presisjon
		if(reg.scientificName().length() &&
			atoi(reg.coordinatePrecision().c_str()) <= pres)
			tab.push_back(reg);
	}

	return true;

}

string csv::kolonne(string &streng, string &kolonne) {

	// tom streng, returnerer tomt strengobjekt
	if(streng.empty())
		return "";

	// indexvariabler
	int kol  = atoi(kolonne.c_str());
	int pos1 = 0;

	// finner kolonneindex
	for(int i = 0; i < kol; i++) {

		pos1++;

		while(streng[pos1] != ';')
			pos1++;

		if(streng[pos1+1] == '\"') {
```

```
                        pos1 += 2;

                        while(streng[pos1] != '\"')
                                pos1++;

                }

        }

        // finner kolonnelengde
        int pos2 = ++pos1;
        if(streng[pos1] == '\"')
                while(streng[pos2] != '\"')
                        pos2++;

        else
                while(streng[pos2] != ';')
                        pos2++;

        // returnerer kolonneinnholdet
        return streng.substr(pos1, pos2 - pos1);

}

unsigned int csv::antall() const {

        // returnerer antall innleste elementer
        return tab.size();

}

artsfunn csv::elem(unsigned int index) const {

        // lovlig index, returnerer tabellelementet
        if(index < antall())
                return tab[index];

        // ulovlig index, returnerer tomt element
        artsfunn ret;
        return ret;

}

void csv::minMaxNord(int &min, int &max) {

        // sorterer tab etter stigende nordkoordinat
        sort(tab.begin(), tab.end(), stigendeNord);

        min = tab[0].utmNord();
        max = tab[tab.size() - 1].utmNord();

}

void csv::minMaxOst(int &min, int &max) {

        // sorterer tab etter stigende østkoordinat
        sort(tab.begin(), tab.end(), stigendeOst);

        min = tab[0].utmOst();
        max = tab[tab.size() - 1].utmOst();

}


/*****************************************************************
 * Class sosi
 *****************************************************************/

sosi::sosi() {
        ref = 0;
}

bool sosi::sett(ini &i) {

        // sjekker gyldig filnavn
        if(i.elem("sosi").empty())
                return false;
```

```
	// åpner fil
	fil.clear();
	fil.open(i.elem("sosi").c_str(), ios::out);

	// fila lar seg ikke åpne
	if(!fil)
		return false;

	// setter filas presisjon, benyttes kun i filhodet
	presisjon = atoi(i.elem("...NOYAKTIGHET").c_str());

	// fila er åpen
	return true;

}

void sosi::skrivHode(csv &csv, ini &ini) {

	// artsfunn eksisterer
	if(csv.antall()) {

		// finner antall lovlige indexer i artstabellen
		unsigned int n = csv.antall() - 1;

		// beregner min-max nord
		int minNord, maxNord;
		csv.minMaxNord(minNord, maxNord);

		// beregner min-max ost
		int minOst, maxOst;
		csv.minMaxOst(minOst, maxOst);

		// beregner filas utstrekning
		minNord -= presisjon;
		maxNord += presisjon;
		minOst  -= presisjon;
		maxOst  += presisjon;

		// skriver filhode
		skriv(".HODE 0:",0,1);

		// skriver tegnsett
		skriv("..TEGNSETT",1);
		skriv(ini.elem("..TEGNSETT"),0,1);

		// skriver transformasjonsparametre
		skriv("..TRANSPAR",0,1);
		skriv("...KOORDSYS",1,0);
		skriv(ini.elem("...KOORDSYS"),0,1);
		skriv("...ORIGO-NØ 0 0",0,1);
		skriv("...ENHET 0.010",0,1);

		// skriver baseområde
		skriv("..OMRÅDE",0,1);
		skriv("...MIN-NØ",1);
		skriv(minNord,1);
		skriv(minOst,0,1);
		skriv("...MAX-NØ",1);
		skriv(maxNord,1);
		skriv(maxOst,0,1);

		// skriver versjon
		skriv("..SOSI-VERSJON",1);
		skriv(ini.elem("..SOSI-VERSJON"),0,1);

		// skriver nivå
		skriv("..SOSI-NIVÅ",1);
		skriv(ini.elem("..SOSI-NIVA"),0,1);

		// skriver overordnet kvalitet
		skriv("..OVERORD_KVALITET",0,1);
		skriv("...PROSESS_HISTORIE",1);
		skriv(versaler(ini.elem("Programinfo")));
		skriv("...INFORMASJON",1);
		skriv(versaler(ini.elem("Bruksinfo")));
		skriv("...DATAUTTAKSDATO",1);
```

```
                skriv(dato("%Y%m%d"),0,1);

        }

}

void sosi::skrivKropp(csv &csv, ini &ini) {

        // csv inneholder artsfunn
        if(csv.antall()) {

                // skriver kartobjektene fortløpende
                for(unsigned int i = 0; i < csv.antall(); i++) {

                        // reduserer funnets presisjon fra hjørneradius til sideradius
                        int pres = cos45(atoi(csv.elem(i).coordinatePrecision().c_str()));

                        // presisjon angitt: oppretter boks med hjørneradius = presisjonen
                        if(pres > 0) {

                                // skriver kurve
                                skriv(".KURVE",1);
                                skriv(++ref);
                                skriv(":",0,1);

                                // skriver objtype
                                skriv("..OBJTYPE",1);
                                skriv(ini.elem("..OBJTYPE_kurve"),0,1);

                                // skriver kommunenummer
                                skriv("..KOMM",1);
                                skriv(ini.elem("..KOMM"),0,1);

                                // skriver kvalitet
                                skriv("..KVALITET",1);
                                skriv(ini.elem("...MALEMETODE"),1);
                                skriv(csv.elem(i).coordinatePrecision(),1);
                                skriv(ini.elem("...SYNBARHET"),0,1);

                                // skriver datauttaksdato
                                skriv("..DATAUTTAKSDATO",1);
                                skriv(dato("%Y%m%d"),0,1);

                                // skriver datafangstdato
                                skriv("..DATAFANGSTDATO",1);
                                skriv(csv.elem(i).yearCollected());
                                skriv(csv.elem(i).monthCollected());
                                skriv(csv.elem(i).dayCollected(),0,1);

                                // skriver rektangel
                                skriv("..NØ",0,1);
                                skriv((csv.elem(i).utmNord()+pres)*100,1);
                                skriv((csv.elem(i).utmOst()-pres)*100,1);
                                skriv("...KP 1",0,1);

                                skriv("..NØ",0,1);
                                skriv((csv.elem(i).utmNord()+pres)*100,1);
                                skriv((csv.elem(i).utmOst()+pres)*100,0,1);
                                skriv((csv.elem(i).utmNord()-pres)*100,1);
                                skriv((csv.elem(i).utmOst()+pres)*100,0,1);
                                skriv((csv.elem(i).utmNord()-pres)*100,1);
                                skriv((csv.elem(i).utmOst()-pres)*100,0,1);
                                skriv((csv.elem(i).utmNord()+pres)*100,1);
                                skriv((csv.elem(i).utmOst()-pres)*100,1);
                                skriv("...KP 1",0,1);

                        }

                        // presisjon angitt: oppretter flateobjekt
                        if(pres){

                                skriv(".FLATE",1);
                                skriv(++ref);
                                skriv(":",0,1);

                                skriv("..OBJTYPE",1);
                                skriv(ini.elem("..OBJTYPE_flate"),0,1);
```

```
}

// presisjon ikke angitt: oppretter punktobjekt
else {

	skriv(".PUNKT",1);
	skriv(++ref);
	skriv(":",0,1);

	skriv("..OBJTYPE",1);
	skriv(ini.elem("..OBJTYPE_punkt"),0,1);

}

// skriver kommune
skriv("..KOMM",1);
skriv(ini.elem("..KOMM"),0,1);

// skriver kvalitet
skriv("..KVALITET",1);
skriv(ini.elem("...MALEMETODE"),1);
skriv(csv.elem(i).coordinatePrecision(),1);
skriv(ini.elem("...SYNBARHET"),0,1);

// skriver datauttaksdato (dagens dato)
skriv("..DATAUTTAKSDATO",1);
skriv(dato("%Y%m%d"),0,1);

// skriver datafangstdato
skriv("..DATAFANGSTDATO",1);
skriv(csv.elem(i).yearCollected());
skriv(csv.elem(i).monthCollected());
skriv(csv.elem(i).dayCollected(),0,1);

// skriver artsfunnet
skriv("..BMARTREG",0,1);

// skriver vitenskapelig navn
skriv("...BMART",1);
skriv(versaler(csv.elem(i).scientificName()));

// skriver artsfunnets dato
skriv("...BMREGDATO",1);
skriv(csv.elem(i).yearCollected());
skriv(csv.elem(i).monthCollected());
skriv(csv.elem(i).dayCollected(),0,1);

// skriver truethetskategori - nye koder
skriv("...BMTRUETKAT",1);
skriv(csv.elem(i).status(),0,1);

// skriver opphav
skriv("..OPPHAV",1);
skriv(versaler("Institusjon = "
	+ csv.elem(i).institutionName()
	+ ", Samling = "
	+ csv.elem(i).collectionName()
	+ ", Katalognr = "
	+ csv.elem(i).catalogNumber()));

// skriver tilknytninger fra ini-fil
skriv("..LINK",1);
skriv(versaler(ini.elem("..LINK")));

// skriver eventuelle tilknytninger fra csv-fil
if(!csv.elem(i).url().empty()){
	skriv("..LINK",1);
	skriv(versaler(csv.elem(i).url()));
}

// skriver prosesshistorie, erstatter <INSTITUSJON> og <DATO>
string prh = versaler(ini.elem("..PROSESS_HISTORIE"));
prh = erstatt(prh, "<INSTITUSJON>", csv.elem(i).institutionName());
prh = erstatt(prh, "<DATO>", dato("%Y-%m-%d"));

skriv("..PROSESS_HISTORIE",1);
```

```
                        skriv(prh);

                        // skriver programinfo
                        skriv("..PROSESS_HISTORIE",1);
                        skriv(versaler(ini.elem("Programinfo")));

                        // skriver bruksinfo
                        skriv("..INFORMASJON",1);
                        skriv(versaler(ini.elem("Bruksinfo")));

                        // skriver norsk navn dersom eksisterer
                        if(!csv.elem(i).norskNavn().empty()){
                                skriv("..INFORMASJON",1);
                                skriv(versaler("Norsk navn = " + csv.elem(i).norskNavn()));
                        }

                        // skriver lokalitetsbeskrivelse dersom eksisterer
                        if(!csv.elem(i).locality().empty()){
                                skriv("..INFORMASJON",1);
                                skriv(versaler("Lokalitet = " + csv.elem(i).locality()));
                        }

                        // skriver samler
                        if(!csv.elem(i).collector().empty()){
                                skriv("..INFORMASJON",1);
                                skriv(versaler("Funnet av = " + csv.elem(i).collector()));
                        }

                        // presisjon angitt: kopler flate og kurve
                        if(pres){
                                skriv("..REF :");
                                skriv(ref - 1,0,1);
                        }

                        // skriver artsfunnet posisjon
                        skriv("..NØ",0,1);
                        skriv(csv.elem(i).utmNord()*100);
                        skriv(" ");
                        skriv(csv.elem(i).utmOst()*100,0,1);

                }

        }

}

void sosi::skrivSlutt() {

        // avslutter filen
        skriv(".SLUTT",0,1);
        fil.close();

}

string sosi::versaler(const string &streng) const {

        // legger til versaler og linjeskift
        return "\"" + streng + "\"\n";

}

string sosi::erstatt(const string &streng, const string &soek, const string &erstatt) const{

        // erstatter soek med erstatt i streng
        string retur = streng;
        int pos = retur.find(soek);
        int len = soek.length();
        retur.replace(pos, len, erstatt);
        return retur;

}

string sosi::dato(char *format) const {

        // returnerer dagens dato
        time_t t;
        tm* tid;
```

```
        time(&t);
        tid = localtime(&t);
        char dfd[50];
        strftime(dfd, 50, format, tid);
        string dato(dfd);
        return dato;

}

int sosi::cos45(int radius) const {

        // reduserer hjørneradius til sideradius ved å multiplisere med cos(45)
        return static_cast<int>(radius * 0.7071);

}

unsigned int sosi::antall() const {

        // returnerer antall sosi-objekter
        return ref;

}

template <class T>
void sosi::skriv( const T &arg, bool mellomrom, bool linjeskift ) {

        fil << arg;

        if( mellomrom )
                fil << " ";

        if( linjeskift )
                fil << endl;

}


/*****************************************************************
 * Nyttefunksjoner
 *****************************************************************/

bool stigendeNord(const artsfunn &arg1, const artsfunn &arg2) {
        return ( arg1.utmNord() < arg2.utmNord() );
}

bool stigendeOst(const artsfunn &arg1, const artsfunn &arg2) {
        return ( arg1.utmOst() < arg2.utmOst() );
}

void programInfo( ini &ini) {

        // skriver info til skjerm dersom modus = konsoll
        if(ini.elem("modus") == "konsoll" ) {

                int ant = 30;

                string ver;
                ver += EXE;
                ver += " ";
                ver += __DATE__;

                fylltegn(ver.length() + ant, '*', true);
                fylltegn(ant/2, ' ', false);
                cout << ver << endl;
                fylltegn(ver.length() + ant, '*', true);

                cout << "Konsollprogram som konverterer csv-filer nedlastet fra\n";
                cout << "artskart.artsdatabanken.no til sosi 4.0\n\n";
                cout << "Dette er fri programvare gjort tilgjengelig av Nedre Eiker\n";
                cout << "kommune. Det gis ingen support utover brukermanualen!\n\n";
                cout << "Programmet krever innargumenter. Les brukermanualen!\n\n";

        }

}

void programResultat(unsigned int antall, ini &ini) {
```

```
	// skriver kjøreresultat til skjerm dersom modus = konsoll
	if(ini.elem("modus") == "konsoll" ) {

		cout << antall << " artsfunn konvertert til sosi.\n\n";
		system("pause");

	}

}

void fylltegn(int ant, char tegn, bool skift) {

	for(int i = 0; i < ant; i++)
		cout << tegn;

	if( skift )
		cout << endl;

}
```