Nearest neighbour algorithm and accuracy:

```
// 1nn.cpp : définit le point d'entrée pour l'application console.
//

#include "stdafx.h"
#include "m1nn.h"

////////////////////////////////////////////////////////////////////////////
// Function to compute accuracy and 1 nearest neighbour
////////////////////////////////////////////////////////////////////////////


int main(int argc, char* argv[])
{
      if (argv[1] != NULL)
    {
            //File name to convert
            string filename = argv[1];
            cout << "File name to convert : " << filename << endl;

            //Open file
            ifstream file(filename.c_str());
            //Test if the file exist
            if (!file)
            {
                  cout << "File don't exist" << endl;
            }
            else
            {
                  //We must determined the number of field and records
                  int fields = 0, records = 0;

                  //Tempory string
                  string str_temp = "";
                  bool bField = true;
                  while (getline(file,str_temp))
                  {
                        records++;
                        if (bField)
                        {
                              for (int i = 0 ; i < str_temp.length() ; i++)
                              {
                                    if (str_temp[i] == ' ')
                                    {
                                          fields++;
                                    }
                              }
                              bField = false;
                              fields++;
                        }
                  }
                  cout << "number of fields : " << fields << endl;
                  cout << "number of records : " << records << endl;


                  //Create array to put data
                  double ** tabdata;
                  tabdata = new double * [records];
                  for (int i = 0 ; i < records ; i ++)
                  {
```

```
                        tabdata[i] = new double [fields];
                }

                //Create array to put result of 1 nearest neighbour
                double ** tab1nn;
                tab1nn = new double * [records];
                for (int i = 0 ; i < records ; i++)
                {
                        tab1nn[i] = new double [records];
                }

                //Initialisation
                for (int i = 0 ; i < records ; i++)
                {
                        for (int j = 0 ; j < records ; j++)
                        {
                                tab1nn[i][j] = 0.00;
                        }
                }

                //Come back to the beginnng of the file
                file.clear();
                file.seekg(0,ios_base::beg);
                //Copy data in the array
                int rec = 0;
                int fie = 0;
                while (getline(file,str_temp))
                {
                        for (int i = 0 ; i < str_temp.length() ; i++)
                        {
                                int j = 0;
                                while ((str_temp[i+j] != ' ')&&((i+j) !=
(str_temp.length()-1)))
                                {
                                        j++;
                                }
                                string str_temp2 = str_temp.substr(i,(j+1));
                                tabdata[rec][fie] = atof(str_temp2.c_str());
                                fie++;
                                if ( (i+j) < str_temp.length())
                                {
                                        i = i + j;
                                }
                                else
                                {
                                        break;
                                }
                        }
                        rec++;
                        fie = 0;
                }
                file.close();

                //Compute 1nn
                cout << "Compute 1 nearest neighbour" << endl;
                for (int i = 0 ; i < records ; i++)
                {
                        for (int j = i+1 ; j < records ; j++)
                        {
                                tab1nn[i][j] = nnfunction(tabdata[i],
tabdata[j],(fields-1));
```

```
                              tab1nn[j][i] = tab1nn[i][j];
                        }
                  }

                  //Compute accuracy of 1nn
                  cout << "Compute accuracy of 1 nearest neighbour" <<
endl;
                  ofstream fileoutput2("debug.data");
                  double accuracy = 0.00;
                  int index = 0;
                  double min = 0.00;
                  for (int i = 0 ; i < records ; i ++)
                  {
                        if (i == 0)
                        {
                              index = 1;
                              min = tab1nn[i][index];
                        }
                        else
                        {
                              index = 0;
                              min = tab1nn[i][index];
                        }
                        for(int j = 0 ; j < records ; j++)
                        {
                              if((j != i)&&(min > tab1nn[i][j]))
                              {
                                    index = j;
                                    min = tab1nn[i][j];
                              }
                        }
                        if (int(tabdata[i][(fields-1)]) ==
int(tabdata[index][(fields-1)]))
                        {
                              accuracy = accuracy + 1.00;
                        }
                  }
                  fileoutput2.close();

                  accuracy = (accuracy * 100)/(records);

                  cout << "Accuracy of the dataset : " << accuracy << " %"
<< endl;

                  //Save result
                  //Create file
                  cout << "Save 1 nearest neighbour" << endl;
                  ofstream fileoutput("1nn.data");
                  for (int i = 0 ; i < records ; i++)
                  {
                        for (int j = 0 ; j < records ; j++)
                        {
                              fileoutput << tab1nn[i][j];
                              fileoutput << ' ';
                        }
                        fileoutput << '\n';
                  }
                  fileoutput.close();
            }
      }
      system("PAUSE");
```

```
	return 0;
}

double nnfunction(double  * record1, double * record2, int n_fields)
{
	double result = 0;

	for (int i = 0 ; i < n_fields ; i++)
	{
		result = result + ((record1[i]-record2[i])*(record1[i]-
record2[i]));
	}

	return result;
}
```

Command line:

```
	1nn.exe examplefile.data
```

Convert comma in space:

```
// cs2ss.cpp : définit le point d'entrée pour l'application console.
//

#include "stdafx.h"


////////////////////////////////////////////////////////////////////////
// Function to replace comma by space
////////////////////////////////////////////////////////////////////////
int main(int argc, char *argv[])
{
    if (argv[1] != NULL)
    {
                //File name to convert
                string filename = argv[1];
                cout << "File name to convert : " << filename << endl;

                //Open file
                        ifstream file(filename.c_str());
                        //ifstream file("communities.data");
                //Test if the file exist
                if (!file)
                {
                          cout << "File don't exist" << endl;
                          }
                          else
                          {
                              //Create buffer to put data
                              stringstream buffer;
                              //Copy data in buffer
                              buffer << file.rdbuf();
                              //Convert buffer in string
                              string str_buf = buffer.str();
                              //Close file
                              file.close();

                              //Find size of buffer
                              int sizebuffer = buffer.str().size();
```

```
                                //Replace comma by space
                                for (int i = 0; i < sizebuffer ; i ++)
                                {
                                    //Find comma
                                                    if(str_buf[i] == ',')
                                    {
                                                          str_buf[i] = ' ';
                                                    }
                                }

                                                //Save new data
                                                ofstream fileoutput(
"dataset.ss" , std::ios_base::app );
                                                fileoutput << str_buf;
                                  }
                              fileoutput.close();

    }
    system("PAUSE");
    return 0;
}

Command line:

      cs2ss.exe examplefile.data
```

Conversion in two class for communities and crimes dataset:

```
// fiddlefield.cpp : définit le point d'entrée pour l'application console.
//

#include "stdafx.h"

////////////////////////////////////////////////////////////////////////////
// Function to convert class field in two class
////////////////////////////////////////////////////////////////////////////

int main(int argc, char* argv[])
{
      if (argv[1] != NULL)
    {
            //File name to convert
            string filename = argv[1];
            cout << "File name to convert : " << filename << endl;

            //Open file
            ifstream file(filename.c_str());
            //Test if the file exist
            if (!file)
            {
                  cout << "File don't exist" << endl;
          }
            else
            {
                  //Tempory string
                  string str_temp;
                  //Create output
                  ofstream fileoutput("dataset.data");
                  while(getline(file,str_temp))
                  {
                        int count = 0;
```

```
                        for (int i = 0 ; i < str_temp.length() ; i++)
                        {
                              if (str_temp[i] == ' ')
                              {
                                    count ++;
                              }
                              if (count == 99)
                              {
                                    int j = 0;
                                    while ((i+j) != str_temp.length())
                                    {
                                          j++;
                                    }
                                    string str = str_temp.substr(i,j);
                                    char *temp;
                                    double val = strtod(str.c_str(),&temp);
                                    int val2 = 0;
                                    if (val <= 0.4)
                                    {
                                          val2 = 0;
                                    }
                                    else
                                    {
                                          val2 = 1;
                                    }
                                    char buffer[65];
                                    _itoa( val2, buffer, 10 );
                                    string sval = buffer;
                                    str_temp.replace((i+1),j,sval.c_str());
                                    break;
                              }
                        }
                        //Write in file
                        fileoutput << str_temp;
                        fileoutput << '\n';
                  }
                  file.close();
                  fileoutput.close();
            }
      }
      system("PAUSE");
      return 0;
}
```

Command line:

```
      fiddlefield.exe examplefile.data
```

Fix communities and crimes dataset:

```
// fixcommdata.cpp : définit le point d'entrée pour l'application console.
//

#include "stdafx.h"

////////////////////////////////////////////////////////////////////////////
// Function to fix communities and crimes dataset
////////////////////////////////////////////////////////////////////////////
```

```
int main(int argc, char* argv[])
{
	if (argv[1] != NULL)
	{
		//File name to convert
		string filename = argv[1];
		cout << "File name to convert : " << filename << endl;

		//Open file
		ifstream file(filename.c_str());
		//ifstream file("communities.data");
		//Test if the file exist
		if (!file)
		{
			cout << "File don't exist" << endl;
		}
		else
		{
			//Tempory string
			string str_temp = "";
			//Creation of tempory file
			ofstream fileoutput_temp( "dataset.temp" );
			//Remove five first fields
			while ( getline(file, str_temp) )
			{
				//Detect the length of five first fields
				int i = 0;
				int j = 0;
				while (i < 5)
				{
					if (str_temp[j] == ' ')
					{
						i++;
					}
					j++;
				}

				//cout << str_temp << endl;

				//Remove the five first fields
				str_temp.erase( 0 , j );

				//Write data in a new file
			    fileoutput_temp << str_temp;
			    fileoutput_temp << '\n';
			}
			fileoutput_temp.close();
			file.close();

			//Remove fields with missing values
			ifstream file2( "dataset.temp" );

			if (!file2)
			{
				cout << "File don't exist" << endl;
			}
			else
			{
				cout << "File name to convert : " << "dataset.temp"
<< endl;
			}
```

```
                  //Initialise str_temp
                  str_temp = "";
                  //Tempory array to know fields with missing values
                  bool * tabfield;
                  tabfield = new bool [128 - 5];

                  //Initialisation of array
                  for (int i = 0 ; i < (128-5) ; i ++ )
                  {
                        tabfield[i] = false;
                  }

                  //Find field with missing data
                  while ( getline(file2, str_temp) )
                  {
                        int n_field = 0;
                        int i = 0;
                        while (i < str_temp.length())
                        {
                              if (str_temp[i] == ' ')
                              {
                                    n_field ++;
                              }
                              else
                              {
                                    if (str_temp[i] == '?')
      tabfield[n_field] = true;
                              }
                              i ++;
                        }
                  }
                  file2.close();

                  ofstream fileoutput( "communities.data" );
                  //Remove fields with missing data
                  ifstream file3( "dataset.temp" );
                  int * inds;
                  inds = new int [128-5];
                  int * inde;
                  inde = new int [128-5];
                  bool start = false;
                  while ( getline(file3, str_temp) )
                  {

                        //Find start and end index of each field
                        int n_field = 0;
                        inds[n_field] = 0;
                        start = true;
                        for (int i = 0 ; i < str_temp.length() ; i ++ )
                        {
                              if ((start == true) && (str_temp[i] == ' '))
                              {
                                    inde[n_field] = i;
                                    start = false;
                                    n_field ++;
                              }
                              else if ((start == false) && (str_temp[i-1] ==
' '))
                              {
                                    inds[n_field] = i;
```

```
                                    start = true;
                              }
                        }
                        //Create record without fields with missing data
                        string str_out = "";
                        for (int i = 0 ; i < 128-5 ; i ++)
                        {
                              if (tabfield[i] == false)
                              {
                                    str_out = str_out +
str_temp.substr(inds[i],(inde[i]-inds[i])+1);
                              }
                        }
                        //Write data in a new file
                        fileoutput << str_out;
                        fileoutput << '\n';
                  }
                  cout << "File saved" << endl;
                  fileoutput.close();
                  file3.close();
            }
      }
    system("PAUSE");
    return 0;
}
```

Command line:

```
      fixcommdata.exe examplefile.data
```

Generate distribution histogram:

```
// genhist.cpp : définit le point d'entrée pour l'application console.
//

#include "stdafx.h"

////////////////////////////////////////////////////////////////////////////
// Function to generate distribution histogram
////////////////////////////////////////////////////////////////////////////

int main(int argc, char* argv[])
{
      if (argv[1] != NULL)
    {
            //File name to convert
            string filename = argv[1];
            cout << "File name to convert : " << filename << endl;

            //Open file
            ifstream file(filename.c_str());
            //Test if the file exist
            if (!file)
            {
                  cout << "File don't exist" << endl;
          }
            else
            {
                  //We must determined the number of field and records
                  int fields = 0, records = 0;
```

```
                        //Tempory string
                        string str_temp = "";
                        bool bField = true;
                        while (getline(file,str_temp))
                        {
                              records++;
                              if (bField)
                              {
                                    for (int i = 0 ; i < str_temp.length() ; i++)
                                    {
                                          if (str_temp[i] == ' ')
                                          {
                                                fields++;
                                          }
                                    }
                                    bField = false;
                                    fields++;
                              }
                        }
                        cout << "number of fields : " << fields << endl;
                        cout << "number of records : " << records << endl;


                        cout << "creation array for data" << endl;
                        //Create array to put data
                        double ** tabdata;
                        tabdata = new double * [records];
                        for (int i = 0 ; i < records ; i ++)
                        {
                              tabdata[i] = new double [fields];
                        }

                        //Come back to the beginnng of the file
                        file.clear();
                        file.seekg(0,ios_base::beg);
                        cout << "copy data into the array" << endl;
                        //Copy data in the array
                        int rec = 0;
                        int fie = 0;
                        while (getline(file,str_temp))
                        {
                              for (int i = 0 ; i < str_temp.length() ; i++)
                              {
                                    int j = 0;
                                    while ((str_temp[i+j] != ' ')&&((i+j) !=
(str_temp.length()-1)))
                                    {
                                          j++;
                                    }
                                    string str_temp2 = str_temp.substr(i,(j+1));
                                    tabdata[rec][fie] = atof(str_temp2.c_str());
                                    fie++;
                                    if ( (i+j) < str_temp.length())
                                    {
                                          i = i + j;
                                    }
                                    else
                                    {
                                          break;
                                    }
```

```
	}
	rec++;
	fie = 0;
}
file.close();

cout << "creation two arrays for each class" << endl;
//Create two array for each class
double ** tabclass0;
double ** tabclass1;
int n_rec_class1 = 0;
int n_rec_class0 = 0;

cout << "" << endl;
for (int i = 0 ;  i < records ; i++)
{
	if (int((tabdata[i][(fields - 1)])) == 0)
	{
		n_rec_class0 ++;
	}
	else
	{
		n_rec_class1 ++;
	}
}
cout << "records of class 0 : " << n_rec_class0 << endl;
cout << "records of class 1 : " << n_rec_class1 << endl;

cout << "Allocation of both array" << endl;
//allocation new arrays
tabclass0 = new double * [n_rec_class0];
for(int i = 0 ; i < n_rec_class0 ; i++)
{
	tabclass0[i] = new double [(fields - 1)];
}
cout << "first array allocated" << endl;

tabclass1 = new double * [n_rec_class1];
for(int i = 0 ; i < n_rec_class1 ; i++)
{
	tabclass1[i] = new double [(fields - 1)];
}
cout << "second array allocated" << endl;

cout << "copy data inside the both new arrays" << endl;
//Copy data on arrays
int index0 = 0;
int index1 = 0;
for (int i = 0 ;  i < records ; i++)
{
	if ((int)(tabdata[i][(fields - 1)]) == 0)
	{
		for(int j = 0 ; j < (fields - 1) ; j++)
		{
			tabclass0[index0][j] = tabdata[i][j];
		}
		index0 ++;
	}
	else if ((int)(tabdata[i][(fields - 1)]) == 1)
	{
		for(int j = 0 ; j < (fields - 1) ; j++)
```

```
                        {
                              tabclass1[index1][j] = tabdata[i][j];
                        }
                        index1 ++;
                  }
            }
            cout << "records of class 0 : " << index0 << endl;
            cout << "records of class 1 : " << index1 << endl;

            cout << "data copied" << endl;

            cout << "compute min and max of the first class for each
field" << endl;
            //Find range of each field
            double ** minmax;
            minmax = new double * [2];
            for (int i = 0 ; i < 2 ; i++)
            {
                  minmax[i] = new double [(fields -1)];
            }
            for (int i = 0 ; i < (fields - 1) ; i++)
            {
                  minmax[0][i] = tabdata[0][i];
                  minmax[1][i] = tabdata[0][i];
                  for (int j = 0 ; j < records ; j++)
                  {
                        if (minmax[0][i] > tabdata[j][i])
                        {
                              minmax[0][i] = tabdata[j][i];
                        }
                        else if (minmax[1][i] < tabdata[j][i])
                        {
                              minmax[1][i] = tabdata[j][i];
                        }
                  }
            }

            cout << "creation of histogram" << endl;
            //Array for the generation of histogram
            double ** hist0;
            double ** hist1;
            hist0 = new double * [5];
            hist1 = new double * [5];
            for (int i = 0 ; i < 5 ; i++)
            {
                  hist0[i] = new double [(fields - 1)];
                  hist1[i] = new double [(fields - 1)];
            }

            cout << "initialization of histogram" << endl;
            //initialisation of arrays
            for(int i = 0 ; i < 5 ; i++)
            {
                  for (int j = 0 ; j < (fields - 1) ; j++)
                  {
                        hist0[i][j] = 0;
                        hist1[i][j] = 0;
                  }
            }
            cout << "histogram initialized" << endl;
```

```
		cout << "creation histogram class 0" << endl;
		//histogram class 0
		for (int i = 0 ; i < (fields - 1) ; i++)
		{
			int val2 = 0;
			for(int j = 0 ; j < n_rec_class0 ; j++)
			{
				double val = (tabclass0[j][i]-
minmax[0][i])/(minmax[1][i] - minmax[0][i]) * 5 ;
				if ( val >= 5.0 )val = 4.9;
				val2 = (int) val;
				hist0[val2][i] ++;
			}
		}
		cout << "histogram class 0 created" << endl;

		cout << "creation histogram class 1" << endl;
		//histogram class 1
		for (int i = 0 ; i < (fields - 1) ; i++)
		{
			int val2 = 0;
			for(int j = 0 ; j < n_rec_class1 ; j++)
			{
				double val = (tabclass1[j][i]-
minmax[0][i])/(minmax[1][i] - minmax[0][i]) * 5 ;
				if ( val >= 5.0 )val = 4.9;
				val2 = (int) val;
				hist1[val2][i] ++;
			}
		}

		//normalized value
		for(int i = 0 ; i < 5 ; i++)
		{
			for (int j = 0 ; j < (fields - 1) ; j++)
			{
				hist0[i][j] = hist0[i][j]/n_rec_class0;
				hist1[i][j] = hist1[i][j]/n_rec_class1;
			}
		}

		cout << "histogram class 1 created" << endl;
		//Create ouput file
		ofstream fileoutput("histogram.data");
		for (int i = 0 ; i < (fields - 1) ; i++)
		{
			for (int j = 0 ; j < 5 ; j++)
			{
				fileoutput << hist0[j][i] << " ";

			}
			fileoutput << '\n';
			for (int j = 0 ; j < 5 ; j++)
			{
				fileoutput << hist1[j][i] << " ";
			}
			fileoutput << '\n';
			fileoutput << '\n';
		}
		fileoutput.close();
	}
```

```
	}
	system("PAUSE");
	return 0;
}
```

Command line:

```
	genhist.exe examplefile.data
```

Min-max normalization :

```
// mmnorm.cpp : définit le point d'entrée pour l'application console.
//

#include "stdafx.h"

////////////////////////////////////////////////////////////////////////////
// Function to min-max normalization
////////////////////////////////////////////////////////////////////////////

int main(int argc, char* argv[])
{
	if (argv[1] != NULL)
    {
		//Store the number of class field
		int n_classfield = atoi(argv[2]);
		//File name to convert
		string filename = argv[1];
		cout << "File name to convert : " << filename << endl;

		//Open file
		ifstream file(filename.c_str());
		//Test if the file exist
		if (!file)
		{
			cout << "File don't exist" << endl;
	      }
		else
		{
			//We must determined the number of field and records
			int fields = 0, records = 0;

			//Tempory string
			string str_temp = "";
			bool bField = true;
			while (getline(file,str_temp))
			{
				records++;
				if (bField)
				{
					for (int i = 0 ; i < str_temp.length() ; i++)
					{
						if (str_temp[i] == ' ')
						{
							fields++;
						}
					}
					bField = false;
					fields++;
				}
```

```
                }
                cout << "number of fields : " << fields << endl;
                cout << "number of records : " << records << endl;

                float ** tabdata;
                tabdata = new float * [records];
                for (int i = 0 ; i < records ; i ++)
                {
                      tabdata[i] = new float [fields];
                }

                //Come back to the beginnng of the file
                file.clear();
                file.seekg(0,ios_base::beg);
                //Copy data in the array
                int rec = 0;
                int fie = 0;
                while (getline(file,str_temp))
                {
                      for (int i = 0 ; i < str_temp.length() ; i++)
                      {
                            int j = 0;
                            while ((str_temp[i+j] != ' ')&&((i+j) !=
(str_temp.length()-1)))
                            {
                                  j++;
                            }
                            string str_temp2 = str_temp.substr(i,(j+1));
                            tabdata[rec][fie] = atof(str_temp2.c_str());
                            fie++;
                            if ( (i+j) < str_temp.length())
                            {
                                  i = i + j;
                            }
                            else
                            {
                                  break;
                            }
                      }
                      rec++;
                      fie = 0;
                }
                file.close();

                float max;
                float min;

                float ** tabnorm;
                tabnorm = new float * [2];
                for (int i = 0 ; i < 2 ; i++)
                {
                      tabnorm[i] = new float [fields];
                }
                //Find min and max for each field
                for(int i = 0 ; i < fields ; i ++)
                {
                      max = tabdata[0][i];
                      min = tabdata[0][i];
                      for (int j = 0 ; j < records ; j++)
                      {
                            if (tabdata[j][i] < min) min = tabdata[j][i];
```

```
                                    else if (tabdata[j][i] > max) max =
tabdata[j][i];
                              }
                              tabnorm[0][i] = min;
                              tabnorm[1][i] = max;
                        }

                        //Normalize each field
                        for(int i = 0 ; i < fields ; i ++)
                        {
                              if(i != n_classfield)
                              {
                                    for (int j = 0 ; j < records ; j++)
                                    {
                                          tabdata[j][i] = (tabdata[j][i]-
tabnorm[0][i])/(tabnorm[1][i] - tabnorm[0][i]);
                                    }
                              }
                        }

                        ofstream fileoutput("dataset.data");
                        cout << "Save result" << endl;
                        for(int i = 0 ; i < records ; i++)
                        {
                              for(int j = 0 ; j < fields ; j++)
                              {
                                    fileoutput << tabdata[i][j];
                                    if(j < (fields-1))fileoutput << ' ';
                              }
                              fileoutput << '\n';
                        }
                        cout << "Results saved" << endl;
                  }
            }
            system("PAUSE");
            return 0;
}
```

Command line:

```
      genhist.exe examplefile.data 99
```

The second parameters (“99”) is the number of the class field.

Remove field:

```
// remfieldgen.cpp : définit le point d'entrée pour l'application console.
//

#include "stdafx.h"

////////////////////////////////////////////////////////////////////////////
// Function to remove field
////////////////////////////////////////////////////////////////////////////

int main(int argc, char* argv[])
{
      if (argv[1] != NULL)
    {
            //File name to convert
```

```
		string filename = argv[1];
		cout << "File name to convert : " << filename << endl;

		//Open file
		ifstream file(filename.c_str());
		//Test if the file exist
		if (!file)
		{
			cout << "File don't exist" << endl;
	}
		else
		{
			//Test if the number of the field exist
			if (argv[2] != NULL)
			{
				//Number of the field to remove
				int n_field_rem;
				n_field_rem = atoi(argv[2]);
				//Tempory string
				//string str_temp = "";
				//Create output file
				ofstream fileoutput( "dataset.data" );
				//We must determined the number of field and
records
				int fields = 0, records = 0;

				//Tempory string
				string str_temp = "";
				bool bField = true;
				while (getline(file,str_temp))
				{
					records++;
					if (bField)
					{
						for (int i = 0 ; i < str_temp.length() ;
i++)
						{
							if (str_temp[i] == ' ')
							{
								fields++;
							}
						}
						bField = false;
						fields++;
					}
				}
				cout << "number of fields : " << fields << endl;
				cout << "number of records : " << records << endl;


				//Create array to put data
				double ** tabdata;
				tabdata = new double * [records];
				for (int i = 0 ; i < records ; i ++)
				{
					tabdata[i] = new double [fields];
				}
				//Come back to the beginnng of the file
				file.clear();
				file.seekg(0,ios_base::beg);
				//Copy data in the array
```

```
                        int rec = 0;
                        int fie = 0;
                        while (getline(file,str_temp))
                        {
                              for (int i = 0 ; i < str_temp.length() ; i++)
                              {
                                    int j = 0;
                                    while ((str_temp[i+j] != ' ')&&((i+j) !=
(str_temp.length()-1)))
                                    {
                                          j++;
                                    }
                                    string str_temp2 =
str_temp.substr(i,(j+1));
                                    tabdata[rec][fie] =
atof(str_temp2.c_str());
                                    fie++;
                                    if ( (i+j) < str_temp.length())
                                    {
                                          i = i + j;
                                    }
                                    else
                                    {
                                          break;
                                    }
                              }
                              rec++;
                              fie = 0;
                        }
                        file.close();

                        for (int i = 0 ; i < records ; i ++)
                        {
                              for (int j = 0 ; j < fields ; j++)
                              {
                                    if (j != n_field_rem)
                                    {
                                          fileoutput << tabdata[i][j];
                                          if (j < fields - 1)
                                          {
                                                fileoutput << " ";
                                          }
                                    }
                              }
                              fileoutput << '\n';
                        }

                        fileoutput.close();
                  }
            }
      }
      system("PAUSE");
      return 0;
}
```

Command line:

```
      remfieldgen.exe examplefile.data 2
```

The second parameters (“2”) is the number of the field to remove.

Select first five fields:

```
// selec5ff.cpp : définit le point d'entrée pour l'application console.
//

#include "stdafx.h"

////////////////////////////////////////////////////////////////////////////
// Function to remove field
////////////////////////////////////////////////////////////////////////////

int main(int argc, char* argv[])
{
      if (argv[1] != NULL)
    {
            //File name to convert
            string filename = argv[1];
            cout << "File name to convert : " << filename << endl;

            //Open file
            ifstream file(filename.c_str());
            //Test if the file exist
            if (!file)
            {
                  cout << "File don't exist" << endl;
          }
            else
            {
                  //We must determined the number of field and records
                  int fields = 0, records = 0;

                  //Tempory string
                  string str_temp = "";
                  bool bField = true;
                  while (getline(file,str_temp))
                  {
                        records++;
                        if (bField)
                        {
                              for (int i = 0 ; i < str_temp.length() ; i++)
                              {
                                    if (str_temp[i] == ' ')
                                    {
                                          fields++;
                                    }
                              }
                              bField = false;
                              fields++;
                        }
                  }
                  cout << "number of fields : " << fields << endl;
                  cout << "number of records : " << records << endl;


                  //Create array to put data
                  double ** tabdata;
                  tabdata = new double * [records];
                  for (int i = 0 ; i < records ; i ++)
                  {
                        tabdata[i] = new double [fields];
```

```
                  }

                  //Come back to the beginnng of the file
                  file.clear();
                  file.seekg(0,ios_base::beg);
                  //Copy data in the array
                  int rec = 0;
                  int fie = 0;
                  while (getline(file,str_temp))
                  {
                        for (int i = 0 ; i < str_temp.length() ; i++)
                        {
                              int j = 0;
                              while ((str_temp[i+j] != ' ')&&((i+j) !=
(str_temp.length()-1)))
                              {
                                    j++;
                              }
                              string str_temp2 = str_temp.substr(i,(j+1));
                              tabdata[rec][fie] = atof(str_temp2.c_str());
                              fie++;
                              if ( (i+j) < str_temp.length())
                              {
                                    i = i + j;
                              }
                              else
                              {
                                    break;
                              }
                        }
                        rec++;
                        fie = 0;
                  }
                  file.close();

                  ofstream fileoutput("dataset.data");

                  for(int i = 0 ; i < records ; i++)
                  {
                        for (int j = 0 ; j < fields ; j ++)
                        {
                              if (j < 5)
                              {
                                    fileoutput << tabdata[i][j] << " ";
                              }
                              if (j == (fields -1))
                              {
                                    fileoutput << tabdata[i][j];
                              }
                        }
                        fileoutput << '\n' ;
                  }
            }
      }
      system("PAUSE");
      return 0;
}
```

Command line:

```
Selec5ff.exe examplefile.data
```

Select first five fields:

```
// selec5ff.cpp : définit le point d'entrée pour l'application console.
//

#include "stdafx.h"

////////////////////////////////////////////////////////////////////////////
// Function to remove field
////////////////////////////////////////////////////////////////////////////

int main(int argc, char* argv[])
{
      if (argv[1] != NULL)
    {
            //File name to convert
            string filename = argv[1];
            cout << "File name to convert : " << filename << endl;

            //Open file
            ifstream file(filename.c_str());
            //Test if the file exist
            if (!file)
            {
                  cout << "File don't exist" << endl;
          }
            else
            {
                  //We must determined the number of field and records
                  int fields = 0, records = 0;

                  //Tempory string
                  string str_temp = "";
                  bool bField = true;
                  while (getline(file,str_temp))
                  {
                        records++;
                        if (bField)
                        {
                              for (int i = 0 ; i < str_temp.length() ; i++)
                              {
                                    if (str_temp[i] == ' ')
                                    {
                                          fields++;
                                    }
                              }
                              bField = false;
                              fields++;
                        }
                  }
                  cout << "number of fields : " << fields << endl;
                  cout << "number of records : " << records << endl;


                  //Create array to put data
                  double ** tabdata;
                  tabdata = new double * [records];
                  for (int i = 0 ; i < records ; i ++)
```

```
                    {
                          tabdata[i] = new double [fields];
                    }

                    //Come back to the beginnng of the file
                    file.clear();
                    file.seekg(0,ios_base::beg);
                    //Copy data in the array
                    int rec = 0;
                    int fie = 0;
                    while (getline(file,str_temp))
                    {
                          for (int i = 0 ; i < str_temp.length() ; i++)
                          {
                                int j = 0;
                                while ((str_temp[i+j] != ' ')&&((i+j) !=
(str_temp.length()-1)))
                                {
                                      j++;
                                }
                                string str_temp2 = str_temp.substr(i,(j+1));
                                tabdata[rec][fie] = atof(str_temp2.c_str());
                                fie++;
                                if ( (i+j) < str_temp.length())
                                {
                                      i = i + j;
                                }
                                else
                                {
                                      break;
                                }
                          }
                          rec++;
                          fie = 0;
                    }
                    file.close();

                    ofstream fileoutput("dataset.data");

                    for(int i = 0 ; i < records ; i++)
                    {
                          for (int j = 0 ; j < fields ; j ++)
                          {
                                if (j < 5)
                                {
                                      fileoutput << tabdata[i][j] << " ";
                                }
                                if (j == (fields -1))
                                {
                                      fileoutput << tabdata[i][j];
                                }
                          }
                          fileoutput << '\n' ;
                    }
              }
        }
        system("PAUSE");
        return 0;
}
```

Command line:

```
Selec5ff.exe examplefile.data
```

Z-normalization:

```
// znorm.cpp : définit le point d'entrée pour l'application console.
//

#include "stdafx.h"
#include "znorm.h"

////////////////////////////////////////////////////////////////////////////
// Function to compute z-normalization
////////////////////////////////////////////////////////////////////////////

int main(int argc, char* argv[])
{
      if (argv[1] != NULL)
    {
            //Store number of class field
            int n_classfield = atoi(argv[2]);
            //File name to convert
            string filename = argv[1];
            cout << "File name to convert : " << filename << endl;

            //Open file
            ifstream file(filename.c_str());
            //Test if the file exist
            if (!file)
            {
                  cout << "File don't exist" << endl;
          }
            else
            {
                  //We must determined the number of field and records
                  int fields = 0, records = 0;

                  //Tempory string
                  string str_temp = "";
                  bool bField = true;
                  while (getline(file,str_temp))
                  {
                        records++;
                        if (bField)
                        {
                              for (int i = 0 ; i < str_temp.length() ; i++)
                              {
                                    if (str_temp[i] == ' ')
                                    {
                                          fields++;
                                    }
                              }
                              bField = false;
                              fields++;
                        }
                  }
                  cout << "number of fields : " << fields << endl;
                  cout << "number of records : " << records << endl;
```

```
                    double ** tabdata;
                    tabdata = new double * [records];
                    for (int i = 0 ; i < records ; i ++)
                    {
                          tabdata[i] = new double [fields];
                    }

                    //Come back to the beginnng of the file
                    file.clear();
                    file.seekg(0,ios_base::beg);
                    //Copy data in the array
                    int rec = 0;
                    int fie = 0;
                    while (getline(file,str_temp))
                    {
                          for (int i = 0 ; i < str_temp.length() ; i++)
                          {
                                int j = 0;
                                while ((str_temp[i+j] != ' ')&&((i+j) !=
(str_temp.length()-1)))
                                {
                                      j++;
                                }
                                string str_temp2 = str_temp.substr(i,(j+1));
                                tabdata[rec][fie] = atof(str_temp2.c_str());
                                fie++;
                                if ( (i+j) < str_temp.length())
                                {
                                      i = i + j;
                                }
                                else
                                {
                                      break;
                                }
                          }
                          rec++;
                          fie = 0;
                    }
                    file.close();

                    //Compute standard deviation and mean
                    double mean;
                    double std;
                    double ** tabnorm;
                    tabnorm = new double * [2];
                    for (int i = 0 ; i < 2 ; i++)
                    {
                          tabnorm[i] = new double [fields];
                    }
                    cout << "Compute mean" << endl;
                    for (int i = 0 ; i < fields ; i++)
                    {
                          mean = 0;
                          for (int j = 0 ; j < records ; j++)
                          {
                                mean = mean + tabdata[j][i];
                          }
                          mean = mean / ((double)records) ;
                          tabnorm[0][i] = mean;
                    }
                    cout << "compute std dev" << endl;
```

```
                  for (int i = 0 ; i < fields ; i++)
                  {
                        std = 0;
                        for (int j = 0 ; j < records ; j++)
                        {
                              std = std + ((tabdata[j][i]-
tabnorm[0][i])*(tabdata[j][i]-tabnorm[0][i]));
                        }
                        std = std / ((double)records);
                        std = sqrt(std);
                        tabnorm[1][i] = std;
                  }
                  cout << "End creation" << endl;
                  //Normalize each field
                  for(int i = 0 ; i < fields ; i ++)
                  {
                        if ( i != n_classfield)
                        {
                              for (int j = 0 ; j < records ; j++)
                              {
                                    tabdata[j][i] = (tabdata[j][i]-
tabnorm[0][i])/(tabnorm[1][i]);
                              }
                        }
                  }

                  ofstream fileoutput("dataset.data");
                  cout << "Save result" << endl;
                  for(int i = 0 ; i < records ; i++)
                  {
                        for(int j = 0 ; j < fields ; j++)
                        {
                              fileoutput << tabdata[i][j];
                              if (j < fields - 1)fileoutput << ' ';
                        }
                        fileoutput << '\n';
                  }
                  cout << "Results saved" << endl;
            }
      }
      system("PAUSE");
      return 0;
}
```

Command line:

```
      znorm.exe examplefile.data 99
```

The second parameters (“99”) is the number of the class field.