employee.h

```
// employee.h
//=========
#include <string.h>

class employee
{
    char* m_name;
    float m_salary;

public:

    employee();

    employee(const char* name, float salary)
        :m_name(new char[strlen(name)+1]),
         m_salary(salary)
    {
        strcpy(m_name, name);
    }

    virtual ~employee()
    {
        delete[] m_name;
    }

    employee(const employee& e)
    {
        m_name = NULL;
        *this = e;
    }

    const employee& operator=(const employee& e)
    {
        if (&e != this)
        {
            delete[] m_name;
            m_name =
              new char[strlen(e.m_name)+1];
            strcpy(m_name, e.m_name);
            m_salary = e.m_salary;
        }
        return *this;
    }

    employee(ifstream& input_file);
    void SaveType(ofstream& output_file) const;
    virtual void Save(ofstream& output_file) const;

    virtual void Print() const;
};
```

employee.cpp

```
// employee.cpp
//===========
#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <typeinfo.h>
#include "employee.h"

employee::employee()
{
        char in_name[80];

        // Get all the fields
        cout<<"Name: ";
        cin>>in_name;
        m_name = new char[strlen(in_name)+1];
        strcpy(m_name, in_name);
        cout<<"Salary: ";
        cin>>m_salary;
}

employee::employee(ifstream& input_file)
{
        // read the length of the name field and the name
        int name_len;
        input_file.read((char*)&name_len, sizeof(name_len));

        m_name = new char[name_len+1];
        input_file.read(m_name, name_len);
        m_name[name_len] ='\0';

        input_file.read((char*)&m_salary, sizeof(m_salary));
}

void employee::SaveType(ofstream& output_file) const
{
        // create the type code (2 characters)
        char type[2];
        strncpy(type, typeid(*this).name()+6, 2);
        output_file.write((const char*)type, 2);
}

void employee::Save(ofstream& output_file) const
{
        // save the name field length, the name and the salary
        int name_len = strlen(m_name);
        output_file.write((const char*)&name_len, sizeof(name_len));
        output_file.write((const char*)m_name, name_len);
        output_file.write((const char*)&m_salary, sizeof(m_salary));
}

void employee::Print() const
{
        cout<<"Employee Name: "<<m_name<<endl;
        cout<<"Salary: "<<m_salary<<endl;
}
```

manager.h

```
// manager.h
//========
#include <string.h>

class manager : public employee
{
	int m_level;

public:
	manager();

	manager(const employee& em, int level)
		:employee(em), m_level(level) {}

	manager(ifstream& input_file);
	virtual void Save(ofstream& output_file) const;

	virtual void Print() const;
};
```

manager.cpp

```
// manager.cpp
//==========
#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <typeinfo.h>
#include "employee.h"
#include "manager.h"

manager::manager()
{
	//(The employee c'tor will be read automatically)
	cout<<"Level: ";
	cin>>m_level;
}

manager::manager(ifstream& input_file) :
		employee(input_file)
{
  input_file.read((char*)&m_level, sizeof(m_level));
}

void manager::Save(ofstream& output_file) const
{
	// first use the base class save
	employee::Save(output_file);
	// save the level field
	output_file.write((const char*)&m_level,
				sizeof(m_level));
}

void manager::Print() const
{
	employee::Print();
	cout<<"Level: "<<m_level<<endl;
}
```

hr_db.h

```
// hr_db.h
//======
class employee;

class hr_db
{

	int	m_list_size;
	employee** m_list;

	hr_db(const hr_db& e){}
	const hr_db& operator=(const hr_db& e){}

public:

	hr_db(int list_size) : m_list_size(list_size)
	{
		m_list = new employee*[m_list_size];
		for(int i = 0 ; i < m_list_size ; i++)
		m_list[i] = NULL;
	}

	virtual ~hr_db()
	{
		// first free all employees objects on the list
		FreeEmployees();
		delete[] m_list;
	}

	void InputEmployees();
	void PrintEmployees();

	void Load(ifstream& input_file);
	void Save(ofstream& output_file) const;

	void FreeEmployees();
};
```

hr_db.cpp

```
// hr_db.cpp
// =======
#include <iostream.h>
#include <fstream.h>
#include "employee.h"
#include "manager.h"
#include "hr_db.h"

void hr_db::InputEmployees()
{
    int in_type;

    for (int i = 0; i < m_list_size ; i++)
    {
        // Get the type
        cout<<"Please enter employee type
               (1: epmployee, 2: manger)";
        cin>>in_type;

        switch (in_type)
        {
        case 1: // employee
            m_list[i] = new employee();
            break;

        case 2: // manager
            m_list[i] = new manager();
            break;
        }
    }
}

void hr_db::PrintEmployees()
{
    cout<<"Employees List"<<endl;
    cout<<"============="<<endl;

    for(int i = 0 ; i < m_list_size ; i++)
    {
        m_list[i]->Print();
        cout<<endl;
    }
}
```

hr_db.cpp – con't

```
…

void hr_db::Load(ifstream& input_file)
{
    char type[2];

    for(int i = 0 ; i < m_list_size ; i++)
    {
        // read type to construct
        input_file.read((char*)type, 2);

        // construct the appropriate object
        if(strncmp(type,"employee", 2) == 0)
                m_list[i] = new employee(input_file);

        else if(strncmp(type,"manager", 2) == 0)
                m_list[i] = new manager(input_file);
    }
}

void hr_db::Save(ofstream& output_file) const
{
    for(int i = 0 ; i < m_list_size ; i++)
    {
        m_list[i]->SaveType(output_file);
        m_list[i]->Save(output_file);
    }
}

void hr_db::FreeEmployees()
{
    for(int i = 0 ; i < m_list_size ; i++)
    {
        delete m_list[i];
        m_list[i] = NULL;
    }
}
```

main.cpp

```
#include <iostream.h>
#include <fstream.h>

#include "hr_db.h"

void main()
{
    hr_db MyEmployees(2);
    ifstream input_file;
    ofstream output_file;

    int option;
    do
    {
        cout<<"Please enter option:"<<endl
            <<"1 - Input Employees"<<endl
            <<"2 - Print Employees"<<endl
            <<"3 - Load Employees (from db.dat)"<<endl
            <<"4 - Save Employees (to db.dat)"<<endl
            <<"0 - Exit"<<endl;

        cin>>option;

        switch (option)
        {
        case 1:
            MyEmployees.FreeEmployees();
            MyEmployees.InputEmployees();
            break;
        case 2:
            MyEmployees.PrintEmployees();
            break;
        case 3:
            input_file.open("db.dat", ios::binary|ios::in);
            MyEmployees.FreeEmployees();
            MyEmployees.Load(input_file);
            input_file.close();
            break;
        case 4:
            output_file.open("db.dat", ios::binary|ios::trunc|ios::out);
            MyEmployees.Save(output_file);
            output_file.close();
            break;
        case 0:
        default:
            option = 0;
        }

    } while (option != 0);

}
```