## ПКШ2021\_ЛР7. Использование статической библиотеки для выполнения заданий Проектно-технологической практики

## Цель работы.

## На примере «Выдача книг студенту» научиться использовать созданную в ЛР6 библиотеку при выполнении заданий проектно-технологической практики.

В демонстрационной программе ПТП2 должны быть реализованы варианты заданий всех членов бригады.

Для выполнения варианта задания «Выдача книг студенту» расширенный в ЛР6 интерфейс класса DBTableTxt должен содержать переопределённые виртуальные функции IndexOfRecord(), SelfRows(), CreateRow(), AddRow() и operator[]().

//----------menu( )---------------------------------

int menu(dbmsLib::DBDate today)

{

int action;

cout<<"\nСегодня "<<today<<endl;

cout<<"======= Практикум по программированию. Вариант 0 =======\n";

cout<<"1 - чтение БД из файла\t\t 4 - тестирование \n";

cout<<"2 - печать всех таблиц БД\t 5 - запись таблиц БД в файл\n";

cout<<"3 - запись таблицы в файл\t 6 - список книг студента\n";

cout<<"7 - выдача книг студенту\t 9 - чтение таблицы из файла\n";

cout<<"10 - печать таблицы\t\t11 - изменение цены товара Hot Dog\n";

cout<<"12 - отпуск товаров со склада\t13 - запись .txt-таблицы в .bin-файл\n";

cout<<"14 - чтение в .txt-таблицу из .bin-файла\n";

cout<<"\t\t\t8 - выход\n";

cout<<"===========================================================\n";

cout<<("Введите номер операции\n");

cin>>action;

while(cin.fail())

{

cin.clear();

cin.ignore(10,'\n');

cout<<("Ошибка формата. Повторите ввод номера операции\n");

cin>>action;

}

return action;

}

//=====================================================

//-------------------------------------------------------------

void DeliveryBooksToStudents(dbmsLib::DBTableSet &db,int screenWidth){

/\*Выдача книг студенту в библиотеке (словесное описание алгоритма):

1. Записать студента в библиотеку, если он ещё не записан (табл. Students).

2. Проверить наличие книги в библиотеке и на руках у студента. У него не должно быть

более одного экземпляра книги(табл. Books, Students и Abonements: BookId, StudentId ).

3. Перед выдачей сделать проверку наличия экземпляра книги в книгохранилище (табл. Books).

4. Внести изменения в таблицы Books и Abonements. (Можно сначала создать запись в буфере,

проверить выполнение условий, а затем, если все проверки ОК!, внести созданные записи в БД и

выдать сообщение о результате выполнения операции.)

\*/

//Выдача нескольких книг студенту

// string tabName = "Students";

cout<<"Введите имя студента\n";

string name;

cin>>name;

string dbName = db.GetDBName();

string path="..\\"+dbName+"\\";

//Проверка наличия абонента в БД библиотеки

if((db["Students"]->IndexOfRecord(&name, "Name").size()==0)){

//замените"Name" на "StudentID"(номер читательского билета),

//чтобы можно было добавить однофамильца

cout<<"В БД нет данных о студенте "<<name<< endl;

cout<<"Добавить студента в список абонентов Библиотеки? ('y'/'n')"<<endl;

char choice;

cin>>choice;

if(choice=='y'){

dbmsLib::Row row=db["Students"]->CreateRow();

db["Students"]->AddRow(row,db["Students"]->GetSize());

}else return;

}

db["Students"]->PrintTable(78);

int studId;

cout<<"Выберите и введите StudentID"<<endl;//так как могут быть однофамильцы

cin>>studId;

bool isContinue=false;

do{

db["Books"]->PrintTable(78);

int bookId;

cout<<"Выберите и введите BookID"<<endl;

cin>>bookId;

// Проверка наличия экземпляра книги в библиотеке и отсутствия её у студента

vector<int> indexes=db["Books"]->IndexOfRecord(&bookId,"BookID");

if(indexes.size()==0){

cout<<"В библиотеке нет книг с индексом "<<bookId<<endl;

return;

}

if((\*(int\*)(\*db["Books"])[indexes[0]]["Quantity"])<=

(\*(int\*)(\*db["Books"])[indexes[0]]["Number"])){

cout<<"В библиотеки нет свободных экземпляров книг с индексом "<<bookId<<endl;

return;

}

// Проверка отсутствия экземпляра книги у студента

// Студент не брал книгу или брал, но вернул её

dbmsLib::DBTable\* studIdBooks=

db["Abonements"]->SelfRows("StudentID",dbmsLib::Equal,&studId);

cout<<"Распечатка книг, которые выдавались студенту StudentID="<<studId<<endl;

studIdBooks->PrintTable(screenWidth);

dbmsLib::DBTable\* studIdBookId=

studIdBooks->SelfRows("BookID",dbmsLib::Equal,&bookId);

vector<int> bookIdIndexes=studIdBookId->IndexOfRecord(&bookId,"BookID");

if(studIdBookId->GetSize()!=0)

{

cout<<"Распечатка книг BookID="<<bookId<<" у студента StudentID="<<studId<<endl;

studIdBookId->PrintTable(screenWidth);

dbmsLib::DBDate returnDate("1.1.1");

//проверка возврата книги bookId (студент может брать несколько раз и возвращать

//одну и ту же книгу).

for(int i=0; i<bookIdIndexes.size();i++)

if(\*((dbmsLib::DBDate\*)(\*studIdBookId)[bookIdIndexes[i]]["DateOut"])==

returnDate){

cout<<"Книга не может быть выдана, т.к. она есть у студента StudentID"<<endl;

return;

}

}

//Выдача книги и внесение соответствующих изменений в БД

cout<<"Добавление записи в таблицу Abonements и внесение изменений в таблицу Books"<<endl;

dbmsLib::Row row=db["Abonements"]->CreateRow();

db["Abonements"]->AddRow(row,db["Abonements"]->GetSize());

vector<int> bookIndexes=db["Books"]->IndexOfRecord(&bookId,"BookID");

(\*((int\*)((\*db["Books"])[bookIndexes[0]]["Number"])))++;//!!!

cout<<"Информация о результатах оформления выдачи книги"<<endl;

db["Abonements"]->PrintTable(screenWidth);

db["Books"]->PrintTable(screenWidth);

cout<<"Продолжить заказ? 1/0\n";

cin>>isContinue;

}while(isContinue);

//Записываем в БД изменения в таблицах Abonements,Books и Students

//db["Abonements"]->WriteDBTable(path+"Abonements.txt");

//db["Books"]->WriteDBTable(path+"Books.txt");

db["Students"]->WriteDBTable(path+"Students.txt");

// db.WriteDB();

}

//--------------------------------------------------------

void main()

{

system("chcp 1251 > nul");

cout<<"Введите имя БД\n";//обратить внимание на удобство тестирования

string bName ;//LibraryBin","CompanyBin","LibraryTxt",CompanyTxt);

cin>>dbName;

cout<<dbName<<endl;

dbmsLib::DBTableSet db(dbName);

string path="..\\"+dbName+"\\";

// cout<<"Введите текущую дату dd.mm.yyyy\n";

string str("04.04.2021");

// cin>>str;

dbmsLib::DBDate today(str);

// cout<<"Введите ширину экрана в символах\n";

int screenWidth=78;

// cin>>screenWidth;

cout<<"Введите имя таблицы\n";

string tabName/\*"Students"\*/;

cin>>tabName;//"Students";"TableRelations";"Abonements";"Books"

//"Products";"OrderDetails";"Orders";"Customers"

//string pathBin="..\\LibraryBin\\";//CompanyBin, LibraryBin

//cout<<"pathBin=..\\LibraryBin\\";

string pathBin="..\\CompanyBin\\";//CompanyBin, LibraryBin

cout<<"pathBin=..\\CompanyBin\\";

while(true)

{

switch (menu(today))

{

case 1: db.ReadDB();//Чтение таблиц БД.

//БД хранится в папке "..\\"+dbName+".

//Имя БД оканчивается или на Txt, если таблицы БД хранятся в текстовых файлах,

//или на Bin, если таблицы БД хранятся в бинарных файлах. Например,

//"LibraryBin","CompanyBin","LibraryTxt","CompanyTxt".

//Каждая таблица хранится в отдельном файле с именем = <имя таблицы>+<.txt | .bin>

//Первая строка файла содержит имя таблицы и имя столбца с первичным ключем

//Вторая строка файла - заголовок таблицы.

//Все последующие строки - записи таблицы.

//В БД список имен файлов с таблицами хранится в файле с именем DBTables.txt,

//а описание связей между таблицами - в файле с именем DataRelations.txt

break;

case 2: cout<<setfill('-')<<setw(screenWidth-1)<<'-'<<setfill(' ')<<endl;

cout<<"\t"<< today;

db.PrintDB(screenWidth);//Печать всех таблиц БД

break;

case 3: db[tabName]->WriteDBTable(path+tabName+TabNameExtension( dbName));break;//pathBin

case 4: Testing(dbName,tabName); break;

case 5: db.WriteDB(); break;

//case 6: StudentBooksList(db,screenWidth); break;

case 7: DeliveryBooksToStudents(db,screenWidth); break;

case 9: db[tabName]->ReadDBTable(path+tabName+TabNameExtension( dbName)); break;

case 10:db[tabName]->PrintTable(screenWidth);

//db.GetDBName() - для универсальности, чтобы можно было бы

//распечатывать DBTableTxt и DBTableBin.

break;

case 11:ChaingePrice(db,tabName); break;

case 12: ExecuteOrder(db,screenWidth);break;

case 8: return;

default: cout<<(" Надо вводить число от 1 до 12\n"); break;

}

}

system("pause");

}