ПКШ2018\_Семинар2

# Разработка методов класса

Процесс разработки методов класса рассмотрим на примере метода   
DBTableTxt:: PrintTable(int screenWidth), который печатает на экране таблицы БД.   
Прежде чем начать разработку вспомните ЛР «Печать числовой матрицы» из курса «Основы программирования». Печать матрицы можно использовать как аналог распечатки таблицы БД.

Метод PrintTable(int screenWidth) выполняет печать таблицы типа DBTableTxt.  
Параметр screenWidth – это ширина экрана, на котором нужно распечатать таблицу.

Прежде чем писать код функции, вам нужно описать её алгоритм. Алгоритм должен быть представлен или в виде словесного описания, или в виде схемы алгоритма.

Уровень детализации алгоритма должен ограничиваться блоками, реализация которых вам понятна, но при этом в алгоритме обязательно должна быть определена связь между блоками по параметрам. Это даст вам возможность проверить реализуемость алгоритма на тестовом примере (в данном конкретном случае на таблицах тестовой БД «Библиотека»).

Приведем словесное описание алгоритма распечатки таблицы (в дальнейшем, прежде чем писать код, вы сами будете описывать алгоритм или рисовать его схему).   
- Если ширина таблицы больше ширины экрана, то таблица должна печататься в несколько полос, размещаемых друг под другом. В каждой полосе печатается последовательность столбцов, по ширине не превышающая ширину экрана.   
- В полосе сначала печатаются заголовки, а затем выводятся все строки размещенных в ней столбцов. Столбцы, не поместившиеся на экране, печатаются в последующих полосах.   
- Число столбцов, их наименование, тип и размер данных в столбцах можно определить из заголовка таблицы.

- Если предположить, что нам известно количество полос, число и ширина полей в полосе, то есть у нас есть функция, которая определяет макет распечатки таблицы, то печать таблицы можно будет выполнить с помощью трех вложенных циклов for:   
• цикл по полосам (внешний);   
• цикл по столбцам таблицы в полосе (средний);   
• цикл по строкам таблицы (внутренний).

Использование циклов поможет вам сделать программу «хорошей» (понятие «хорошей» программы рассматривалось в лекции ПКШ2018\_Л3).

В заголовочном файле библиотеки dbmsLib\_v1.h описан прототип метода DBTableTxt::CreateTableMaket(). Этот метод определяет макет распечатки таблицы и вызывается из метода PrintTable().

При разработке программы группой программистов, после того как согласован интерфейс метода CreateTableMaket(), можно выполнять параллельную разработку метода PrintTable(). Один член группы разрабатывает PrintTable(), используя CreateTableMaket() из библиотеки (или используя «заглушку» для этой функции, которая возвращает данные только для тестового примера), а другой разрабатывает CreateTableMaket().

# Задание.

Разработать алгоритм, закодировать и протестировать *глобальную* (не метод класса) функцию CreateTableMaket1(). Для этого вам нужно будет слегка задуматься и изменить список её формальных параметров.

Разработку функции выполняйте в проекте из ПКШ2018\_ЛР1, добавив в меню пункт наподобие «Макет распечатки таблицы». При выборе этого пункта должна выполняться функция, которая вызывала бы разрабатываемую функцию CreateTableMaket1() и позволяла бы её протестировать.  
Что и как тестировать? На этот вопрос вы должны давать ответ при разработке любой программы.  
 В данном случае вы должны проверить результаты следующих операций:   
- изменяя ширину экрана, убедиться в правильности изменения макета таблицы;   
- проверить случай, когда таблицу вообще нельзя распечатать;   
- проверить случай, когда заголовок столбца (он печатается в одной строке) длиннее, чем данные, которые в нем выводятся.

Ниже приведены фрагменты кодов, которые должны помочь вам в разработке методов PrintTable1() и CreateTableMaket1(...).

# 1.2 Создание макета распечатки таблицы

struct Strip{//описание полосы распечатки таблицы  
 int nField;//число полей  
 int\* fieldWidth;//ширина полей в полосе (массив)  
}

Макет распечатки таблицы – это массив из описаний полос таблицы

//screenWidth - ширина экрана (входной параметр)

//nStrip - число полос в распечатке (выходной параметр)

//Strip strips[nStrip] – массив из описаний полос таблицы   
void CreateTableMaket(...) – это функция-член класса DBTableTxt, которая вызывается из метода void DBTableTxt::PrintTable(int screenWidth). Она должна вычислить и возвратитьStrip\* &strips,int &nStrip, int screenWidth);

## 1.2.1 Пример использования итераторов для чтения заголовка таблицы (фрагмент программы)

{

Header::iterator headerIter,contHeaderIter;

int nColumn = (int)columnHeaders.size();

//Заполнение массива ширины полей для печати таблицы

int\* fieldW=new int[nColumn];

headerIter=columnHeaders.begin();

for(int k=0;k<nColumn; k++){

//что шире: заголовок или данные?

fieldW[k]=headerIter->second.length > headerIter-> first.size()?

headerIter->second.length+2 : headerIter->first.size()+2;

headerIter++;

}

int currCol=0;//порядковый номер столбца в таблице

nStrips=1;//число полос в распечатке таблицы

int sumWidth=0;//суммарная ширина столбцов в полосе

int n=0;//число столбцов в полосе

int buff[40]={0};//объявление и обнуление буфера для //временного хранения числа столбцов в полосе (n<40)

...

strips - "ступенчатый" массив из размеров столбцов strips[i].fieldWidth в полосах таблицы. Число столбцов в полосе может быть не одинаковым.   
Для его создания используется структура Strip (сущность более высокого, по сравнению со строкой, уровня, которая будет скрывать различия в числе столбцов в строках разных полос в макете таблицы.   
В результате таблица печатается в виде полос одинаковой ширины (за исключением последней полосы).

Эта функция может рассматриваться как пример "обобщенного программирования" для печати таблиц произвольного размера.

## Выделение памяти и заполнение массива strips из буфера buff. …

## …

# 1.2 Вывод таблицы на экран

void DBTableTxt::PrintTable(int screenWidth)

{

Strip \* strips;//выходной параметр функции CreateTableMaket

int nStrips;//выходной параметр функции CreateTableMaket

CreateTableMaket(strips,nStrips,screenWidth);

int nColumn = (int)columnHeaders.size();

Header::iterator headerIter,contHeaderIter;

Row::iterator rowIter, contRowIter;

cout<<"\nТаблица "<<tableName<<endl;

cout<<setfill('=')<<setw(screenWidth-1)<<'='<<setfill(' ')<<endl;

contHeaderIter=columnHeaders.begin();

for ( int r = 0; r < nStrips; r++)

{//печать заголовка

//вывод наименований столбцов в заголовке полосы

headerIter=contHeaderIter;

for (int j = 0; j < strips[r].nField; j++)

cout<<setw(strips[r].fieldWidth[j])<< headerIter++->first;

cout<<endl;

//вывод типа данных в столбцах в заголовке полосы

headerIter=contHeaderIter;

for (int j = 0; j < strips[r].nField; j++)

cout<<setw(strips[r].fieldWidth[j])<<TypeName(headerIter++-> second.colType);

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

//печать строк таблицы…  
. . .

}