Министерство науки и высшего образования Российской Федерации

Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

Ю.С. Белов, С.С. Гришунов

ОБРАБОТКА БИНАРНЫХ ФАЙЛОВ

Методические указания к выполнению лабораторной работы
по курсу «Типы и структуры данных»

Калуга – 2019

УДК 004.62
ББК 32.972.5
Б435

Методические указания составлены в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.04 «Программная инженерия» кафедры «Программного обеспечения ЭВМ, информационных технологий».

Методические указания рассмотрены и одобрены:

- Кафедрой «Программного обеспечения ЭВМ, информационных технологий» (ИУ4-КФ) протокол № 51.4/5 от «23» января 2019 г.

Зав. кафедрой ИУ4-КФ _____ к.т.н., доцент Ю.Е. Гагарин

- Методической комиссией факультета ИУ-КФ протокол № 7 от «28» 01 2019 г.

Председатель методической комиссии факультета ИУ-КФ _____ к.т.н., доцент М.Ю. Адкин

- Методической комиссией КФ МГТУ им.Н.Э. Баумана протокол № 4 от «5» 02 2019 г.

Председатель методической комиссии КФ МГТУ им.Н.Э. Баумана _____ д.э.н., профессор О.Л. Перерва

Рецензент:
к.т.н., доцент кафедры ИУ6-КФ _____ А.Б. Лачихина

Авторы
к.ф.-м.н., доцент кафедры ИУ4-КФ _____ Ю.С. Белов
ассистент кафедры ИУ4-КФ _____ С.С. Гришунов

Аннотация

Методические указания к выполнению лабораторной работы по курсу «Типы и структуры данных» содержат сведения о бинарных файлах, о методах их обработки.

Предназначены для студентов 2-го курса бакалавриата КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Настоящие методические указания составлены в соответствии с программой проведения лабораторных работ по курсу «Типы и структуры данных» на кафедре «Программное обеспечение ЭВМ, информационные технологии» факультета «Информатика и управление» Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания к выполнению лабораторной работы по курсу «Типы и структуры данных» содержат сведения о бинарных файлах, о методах их обработки.

Предназначены для студентов 2-го курса бакалавриата КФ МГТУ им Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ

Целью выполнения лабораторной работы является формирование практических навыков создания алгоритмов обработки бинарных файлов.

Основными задачами выполнения лабораторной работы являются:

1. Познакомиться со структурой бинарного bmp-файла.
2. Изучить способы программной обработки бинарного файла.
3. Реализовать алгоритм согласно варианту

Результатами работы являются:

- Программа, реализующая индивидуальное задание
- Подготовленный отчет

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

Основные определения

Файл — это именованная область внешней памяти, содержащая какую-либо информацию. Файл в таком понимании называют физическим файлом, т.е. существующим физически на некотором материальном носителе информации. С другой стороны, файл—это одна из многих структур данных, используемых в программировании. Файл в таком понимании называют логическим файлом, то есть существующим только в нашем логическом представлении при написании программы. Структура физического файла представляет собой простую последовательность байт памяти носителя информации жесткого магнитного диска (ЖМД) или гибкого магнитного диска (ГМД). Структура логического файла — это способ восприятия файла в программе. Образно говоря, это «шаблон», через который мы смотрим на физическую структуру файла. В языках программирования таким шаблонам соответствуют типы данных, допустимые в качестве компонент файлов.

Классификация файлов

Файлы классифицируются по двум признакам:

– по типу (логической структуре);
– по методу доступа к элементам файла.

По логической структуре файлы делятся на текстовые, типизированные и нетипизированные. По методу доступа различают файлы последовательного доступа и прямого доступа, Метод доступа представляет собой сочетание типа файла и стандартных программных средств, обеспечивающих определенный режим передачи данных между файлом и основной памятью. Практически во всех современных компьютерах основными устройствами внешней памяти являются магнитные диски с подвижными головками, и именно они служат для хранения файлов. Для произведения обмена с магнитным диском на уровне аппаратуры нужно указать номер цилиндра, номер поверхности, номер блока на соответствующей дорожке и число

байтов, которое нужно записать или прочитать от начала этого блока. Однако эта возможность обмениваться с магнитными дисками порциями меньше объема блока в настоящее время не используется в файловых системах. Это связано с двумя обстоятельствами. Во-первых, при выполнении обмена с диском аппаратура выполняет три основных действия: подвод головок к нужному цилиндру, поиск на дорожке нужного блока и собственно обмен с этим блоком. Из всех этих действий в среднем наибольшее время занимает первое. Поэтому существенный выигрыш в суммарном времени обмена за счет считывания или записывания только части блока получить практически невозможно. Во-вторых, для того чтобы работать с частями блоков, файловая система должна обеспечить соответствующего размера буфера оперативной памяти, что существенно усложняет распределение оперативной памяти. Поэтому во всех файловых системах явно или неявно выделяется некоторый базовый уровень, обеспечивающий работу с файлами, представляющими набор прямо адресуемых в адресном пространстве файла блоков. Размер этих логических блоков файла совпадает или кратен размеру физического блока диска и обычно выбирается равным размеру страницы виртуальной памяти, поддерживаемой аппаратурой компьютера совместно с операционной системой.

Файлы последовательного доступа характеризуются последовательным размещением записей в файле в порядке их поступления и доступом к записям файла в порядке их физического расположения. Примером файлов последовательного доступа являются текстовые файлы. Структура текстовых файлов обычно очень проста: это либо последовательность записей, содержащих строки текста, либо последовательность байтов, среди которых встречаются специальные символы (например, символы конца строки). В файлах прямого доступа можно позиционировать указатель файла на запись с любым номером и осуществить запись или чтение содержимого записи. Типизированные и нетипизированные файлы любой структуры являются файлами прямого доступа. Рассмотрим структуру и доступ к записям в файле растровой графики, который является нетипизированным файлом.

СТРУКТУРА BMP-ФАЙЛА

Растровый графический файл—это не просто множество точек, как следует из его названия. Это файл с весьма сложной структурой, содержащий информацию о типе, размере и цвете, а также элементы изображения—пиксели. *BMP-файлы* могут хранить не только полноцветные изображения фотографического качества, но и относительно простые картинки пиктограмм, курсоров и других фигур. Все *BMP*-файлы должны храниться в *DIB*-формате, чтобы их можно было показывать на компьютерах с различной графической аппаратурой. *DIB*-формат *BMP* файла приведен в таблице 1.

Таблица 1

BITMAPFILEHEADER Bmfh;	(14 байтов)
BITMAPFILEHEADER Brnih;	(40 байтов)
RGBQUAD aColors[];	(переменная)
BYTE abitmapBits[];	(переменная)

Структура BMP-файла состоит из следующих частей:

1) BMP – файл начинается структурой BITMAPFILEHEADER, которая идентифицирует его как растровый графический и хранит другую информацию о содержании файла.

2) Вторая структура, BITMAPINFOHEADER, содержит такие характеристики растрового изображения, как ширина и высота. Заголовок bmin и массив aColors [] входят в состав структуры BITMAPINFO, но могут быть доступны как индивидуально, так и в составе структуры BITMAPINFO.

3) Массив aColors содержит структуры типа RGBQuad - интенсивностей красного, зеленого и синего цветов. Размер массива зависит от числа используемых цветов. Например, восьмицветное растровое изображение может иметь восемь RGBQuad-элементов в

массиве aColors. В растровых изображениях с 24-разрядным представлением цвета массива aColors[] нет. Поскольку этот массив переменной длины, число его элементов влияет на расположение массива BitmapBits, описанного в следующем пункте.

4) Массив aBitmapBits содержит пиксели, которые различаются по формату в зависимости от типа растрового изображения. Кроме того, байты могут быть сжаты. Байты хранятся строками слева направо, каждая строка представляет собой линию развертки (scan line) изображения, которые могут быть дополнены до 32-разрядной границы. Линии развертки упорядочены снизу вверх, т.е. первый элемент массива содержит пикселы последней строки изображения. В растровых изображениях разрядным представлением цвета, где массив aColors отсутствует, байты в aBitmapBits непосредственно представляют 24-разрядные триадные цвета пикселей. В других растровых отражениях пикселы из aBitmapBits представляют собой индексы массива aColors.

Заголовок файла BITMAPFILEHEADER представляет собой запись состоящую из следующих полей:

```
Type
 BitMapFileHeader=Record
  bfType:Integer;
  bfSize:LongInt;
  bfReserved1:Integer;
  bfReserved2:Integer;
  bfOffBits:LongInt;
End;
```

Поля записи BITMAPFILEHEADER имеют следующую структуру:

bfType—должен содержать два ASCII-символа: «B» и «M», разумеется, означающие bitmaps. Соответствующие шестнадцатеричные значения равны 0x42 и 0x4D. При любых других значениях этого поля файл не является растровым изображением, принятым в Windows.

bfSize — равен размеру файла в байтах, как указано в соответствующем элементе оглавления дискового каталога. bfSize

используется для проверки целостности файла и для распределения памяти под весь файл. Значение bfSize не является размером растрового изображения.

bfReservedl — не документировано и не используется. Должно быть равно нулю.

bfReserved2 — не документировано и не используется. Должно быть равно нулю.

bfOffBits - специфицирует байтовое смещение до начала растрового изображения. Используйте это значение для определения местонахождения массива aBitmapBits в файле.

Структура BITMAPINFO покрывает два элемента BMP-файла: структуру заголовка и массив цветовых значений. Их можно читать и записывать индивидуально, а не в составе BITMAPINFO. BITMAPINFO — структура переменной длины. Размер массива bmiColor зависит от типа растрового изображения. При использовании этой структуры разместите bmiHeader фиксированной длины и выделите достаточное количество памяти для массива bmiColors переменной длины. Рассмотрим подробнее структуру BITMAPINFO:

1. **bmiHeader** — содержит размерность растрового изображения, формат и другую информацию и представляет собой запись, состоящую из следующих полей:

```
BitMapInfoHeader=Record biSize:Longlnt;
 biWidth:Longlnt;
 biHeight:Longlnt;
 biPlanes:Word;
 biBitCount:Word;
 biCompression:Longlnt;
 biSizelmage:Longlnt;
 biXPelsPerMeter:Longlnt;
 biYPelsPerMeter:Longlnt;
 biClrUsed:Longlnt;
 biClrlmportant:Longlnt;
 End;
```

Поля записи **bmiHeader** имеют следующую структуру:

biSize—специфицирует собственный размер структуры в байтах. Это значение должно использоваться для определения первого байта массива bmiColors в структуре BITMAPINFO. Иногда используют biSize для проверки того, является ли данный файл BMP-файлом или нет. Если biSize равно 0x28 (десятичное 40), то это, по всей вероятности, BMP-файл. Данное поле находится на расстоянии 14 байтов от начала файла. Для проверки файла позиционируйте его внутренний указатель на четырнадцатый байт, прочитайте с этого места двойное слово и сравните его значение с 0x28.

biWidth — содержит ширину изображения в пикселах.

biHeight — содержит высоту изображения в пикселах.

biPlanes — должен быть равен единице, т.к. BMP-файлы, какого бы типа они ни были, хранятся в независимом от устройства формате с одной цветовой плоскостью (color-plane).

biBitCount — содержит число битов на пиксел. Кроме того, отличает монохромное изображение от цветного. Должно быть равно 1, 4, 8 или 24. Обычно используется в конъюнкции с biCIrUsed и biCIrImportant. Значение biBitCoun = l помечает изображение как монохромное и показывает, что массив bmiColors в структуре BITMA PINFO содержит два элемента типа RGBQuad. Каждый бит изображения, хранимый в массиве aBitmapBits, служит индексом массива bmiColors. Бит, равный О, окрашен в соответствии с содержимым bmiColors[0], единичный бит—в соответствии с содержимым bmiColors[1]. Значение biBitComt-4 показывает, что массив bmiColors содержит до 17 цветовых значений типа RGBQuad. В этом самом распространенном формате каждый байт содержит два 4-битовых пиксела, а каждый пиксел есть индекс массива bmiColors, определяющего цвет. Например, байт изображения, равный 0x23, содержит два пиксела, один из которых равен 0x02, а второй— 0x03. Цвет первого пиксела равен bmiColors[0x02], а цвет другого — bmiColors[0x03]. Если biBitCount=8, то массив bmiColors содержит до 256 цветовых значений типа RGBQuad. Каждый байт в массиве aBilmapBits представляет собой отдельный пиксел, являющийся индексом массива. Цвет любого пиксела Q равен bmiColors[Q]. Этот формат наиболее удобен для обработки и быстрого отображения на

экране монитора, но занимает примерно в два раза больше места на диске, чем 17-цветный ЯЛ/Р-файл. Формат со значением biBitCount-24 может описывать изображение с более чем 17-ю миллионами цветовых оттенков. Иногда называемые полноцветными (true color), эти изображения обычно используются для представления фотографий. В данном формате массив bmiColors отсутствует. Вместо этого 24-разрядные значения массива aBitmapBits представляют каждый пиксел как красно-зелено-синюю триаду (тип RGBTRIPLE). BMP-файл этого типа может занимать огромное пространство на диске и в памяти.

biCompression—показывает, хранится ли данное растровое изображение в сжатом виде, а также метод его упаковки. Равен BI_RGB (изображение не сжато), BI_RLE8 (5-разрядное групповое кодирование) или BI_RLE4 (4-разрядное групповое кодирование).

biSizelmage - содержит размер растрового изображения в байтах. Может быть нулевым, если biCompression = BI_RGB (изображение не сжато).

biXPelsPerMeter - указывает предпочтительное разрешение по горизонтали в пикселах на метр. Используйте это значение, чтобы выбрать подходящее изображение среди многих его вариантов с разными разрешениями. Теоретически, чем ближе разрешающая способность устройства отображения соответствует этому и следующему параметрам, тем лучше будет выглядеть изображение на экране. На практике это значение используется редко.

biYPelsPerMeter—указывает предпочтительное разрешение по вертикали в пикселах на метр.

biCIrUsed— обычно содержит число цветов, используемое в растровом изображении и определяемое массивом bmiColors типа RGBQuad. Если biCIrUsed равен нулю, как это обычно и бывает, в изображении используется максимальное количество цветов, возможное для изображения данного типа. Нулевое значение biCIrUsed не указывает на то, что массив Color Array пуст.

biCIrlmportant—содержит число важных цветов изображения. Например, если это значение равно 3, первые три значения цвета в массиве bmiColors должны отображаться на экране с как можно более

точным соответствием. Другие пикселы могут отображаться с измененным цветом или безболезненно пропускаться. Если biCIrlmportant равен нулю, все цвета считаются важными. Этот параметр не имеет смысла для растровых изображений с 24-разрядным представлением цвета.

2. **bmiColors**—этот элемент структуры BITMAPINFO содержит информацию о цвете в виде массива структур типа RGBQuad. Формат и число элементов этого массива зависят от типа растрового изображения и других факторов, определяемых полями bmiHeader. Для растровых изображений с 24-разрядным представлением цвета bmiColors отсутствует. Цветовое значение типа RGBQuad представляет собой трехбайтовую композицию интенсивностей красного, зеленого и синего цветов. Четвертый байт дополняет структуру таким образом, что каждый элемент в RGBQuaduaccnBQ начинается с четного адреса. Хотя это несколько расточительно, данная схема здорово повышает производительность, т.к. процессор i86 фирмы Intel может выбирать выровненные на четную границу данные быстрее. Лишний четвертый байт объясняет (quad — четверка) название типа структуры (RGBQuad). В виде записи эту структуру можно представить следующим образом:

```
RGBQuad=Record
 RGBBlue:Byte;
 RGBGreen:Byte;
 RGBRed:Byte;
 RGBReserved:Byte;
End;
```

Значение rgbReserved предполагается равным нулю, хотя его фактическое значение не важно. Структура типа RGBQuad может представлять 17777277 уникальных значений цвета, включая черный (все элементы нулевые) и белый (все элементы равны 255). Поскольку немногие из дисплеев персональных компьютеров могут отобразить так много цветов одновременно, комбинация значений из RGBQuad будет воспроизводить соответствующий эквивалентный оттенок.

ПРИМЕР РАБОТЫ С BMP-ФАЙЛАМИ

Программа меняет интенсивность цветов в BMP-файле или выводит его на экран. Если BMP-файл содержит более 16 цветов, то для корректной работы программы необходим драйвер SVGA256.BGI. Драйвер должен находиться на диске A.

```
{$I-}
 Program Prg_17_1;
 Uses CRT,Graph;
 Type
  BitMapFileHeader=Record
 {Тип файла (для битового образа - BM)}
  BFType:Array [ 1 - . 2] Of Char;
  BFSize:LongInt; {Размер файла}
  BFReservedl:Word;{Не используется}
  BFReserved2:Word;{Не используется}
{Смещение данных битового образа от заголовка в байтах}
  BFOf f Bit s: Longlnt ;
 End;
 RGBQuad=Record
 RGBRed: Byte;{Интенсивность красного}
 RGBGreen:Byte; {Интенсивность зеленого}
 RGBBlue: Byte; {Интенсивность голубого}
 RGBReserved:Byte; {Не используется}
End;
BitMapInfoHeader=Record
{Число байт, занимаемых структурой BITMAPINFOHEADER}
BISize:LongInt;
{Ширина битового образа в пикселах}
BIWidth:Longlnt;
{Высота битового образа в пикселах}
 BIHeight: Longlnt;
 BIPlanes:Word;{Число битовых плоскостей устройства}
 BIBitCount:Word; {Число битов на пиксел}
 BiCompression:Longlnt; {Тип сжатия}
 BISizelmage:Longlnt;{Размер картинки в байтах}
{оризонтальное разрешение устройства, пиксел/м}
BIXPelsPerMeter:Longlnt;
{Вертикальное разрешение устройства, пиксел/м}
 BIYPelPerMeter:Longlnt;
 BiCIrUsed:Longlnt;{Число используемых цветов}
 BIClrlmportant:Longlnt; {Число "важных" цветов}
End;
 BitMapInfo=Record
 BMIHeader:BitMapInfoHeader;
 BMIColors:Array [1..256] Of RGBQuad;
End;
Var
 F:File;
 Info:BitMapInfo;
 Header:BitMapFileHeader;
```

```
 Bit,K,X,Y:Word;
 V:String;
 Pal:PaletteType;
{-------------}
Procedure Graphlnit(St:String);
{Инициализация графики}
Var
D,M:Integer;
Begin
D:=InstallUserDriver('svga256',Nil);
M:=2;
 InitGraph(D,M,St);
 If GraphResultoGrOk Then
 Begin
  GotoXY(30,10);
  TextColor(132);
  Write ('Графическая ошибка!');
  TextColor(132);
  GotoXY(30,11) ;
Write('Нажмите любую клавишу');
Repeat Until KeyPressed;
ReadKey;
ClrScr;
Halt(1);
 End;
End;
{---------------}
Procedure InputName;
Инициализация BMP-файла}
Var
  fff:lnteger;
  Name:String;
Begin
 Write (' Введите имя BMP-файла: ‘ ) ;
 ReadLn(Name);
 Assign(F,Name);
 Reset(F,1);
 fff:«IOResult;
 If fff<>0 Then
 Begin
  Sound(880) ;
  Delay(3000) ;
  NoSound;
  WriteLn('Ошибка чтения файла!'#13#10’ Нажмите Enter1,fff);
  ReadLn;
  Halt;
 End;
End;
{---------}
Procedure ChangeBMP;
{Изменение цветов BMP-файла}
Var
 J,X,Y:lnteger;
Begin
 Close(F);
 Reset(F,1);
```

```
 For J:=1 To Bit Do
 Begin
  X:=1;
  Y:=4;
  For I:=o To 9 Do
  Begin
   GotoXY(X,Y+I) ;
   ClrEOL;
  End;
  GotoXY(1,4);
  WriteLn('Изменяем интенсивность цвета
  ',J,':');
  Write('Ведите интенсивность красного: ');
   ReadLn(Info.BMIColors[J].RGBRed);
   Write('Ведите интенсивность зеленого: ');
   ReadLn(Info.BMIColors[J].RGBGreen);
   Write('Ведите интенсивность синего: ');
   ReadLn(Info.BMIColors[J].RGBBlue);
End;
  ReadKey;
  BlockWrite(F,Header, SizeOf(Header));
  BlockWrite(F,Info,SizeOf(Info));
 End;
{-----------}
Procedure WriteBMP;
{Чтение BMP-файла}
Var
 B:Byte;
 bb:Longlnt;
 I,J:Integer;
 ff:Longlnt;
 Is:Boolean;
Begin
Graphlnit('a:\bgi\') ;
Seek(F,0);
Seek(F,Header.BFOffBits);
With Info Do
 For K:=0 To Bit-1 Do
 SetRGBPalette(K,BMIColors[K+l].RGBBlue Div
4,BMIColors[K+l].RGBGreen Div 4,BMIColors[K+l] .RGBRed Div 4);
  ff:=0;
  Is:=True;
  If Bit=l6 Then
  Begin
 For I:=Info.BMIHeaderJBIHeight+100 Downto 100 Do
  For J:=100 To Info.BMIHeader.BIWidth +99 Do
  Begin
  BlockRead(F,B,SizeOf(Byte));
  PutPixel(J-2,I,B);
  PutPixel(J* 2+1,1,B Shr 4);
 End;
End
Else
  If Bit=256 Then
  For I:=Info.BMIHeader.BIHeight+100 Downto 100 Do
   For J:=100 Div 4
```

```
To (Info.BMIHeader.BIWidth+100) Div 4 Do
Begin
   BlockRead(F,bb,SizeOf(Longlnt)) ;
 For ff:=0 To 3 Do
 Begin
 PutPixel(J-4+ff,I,bb);
 bb:=bb Shr 8;
End;
   End;
End;
{--------------}
Begin
 ClrScr;
 InputName;
 BlockRead(F, Header,SizeOf(Header) ,K) ;
 BlockRead(F, Info,SizeOf(Info),K);
 Case Info.BMIHeader.BIBitCount Of
  l:Bit:=2;
  4:Bit:=16;
  8 :Bit:=256;
  24:Bit:=24;
 End;
 WriteLn ('Вы хотите увидить файл (1) или изменить цвета (2)?');
 ReadLn (K) ;
 Case K Of
  l:WriteBMP;
  2:ChangeBMP;
 End;
 Close(F);
 SetTextStyle (5,0,0);
 OutText ('Press any key');
 ReadKey;
 CloseGraph;
End.
{$I+}
```

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Обработать/создать/изменить файл, содержание которого предусмотрено вариантом задания.
2. Все приложение в целом должно быть написано с использованием ООП-технологии.
3. Все входные и выходные данные считать/записать из/в BMP-файл(а).
4. Все возникшие ошибки записать в файл ошибок.
5. В консольном приложении продемонстрировать работу программы.
6. Программа должна запускаться из командной строки с указанием имени исполняемого файла, имен файлов входных, выходных данных, файла ошибок.
7. Предоставить и защитить отчет.

ВАРИАНТЫ ЗАДАНИЙ

Дан BMP-файл, содержащий рисунок. Необходимо:

1. Изменить цвет фона.
2. Изменить цвет рисунка.
3. Увеличить размер рисунка в два раза.
4. Изменить интенсивность цветов.
5. Вывести содержимое полей, которое свидетельствует о том, что это файл растрового изображения.
6. Вывести размер в байтах основных элементов BMP-файла.
7. Вывести размер растрового изображения в байтах, ширину и высоту изображения в пикселях.
8. Вырезать треугольник из рисунка.
9. Получить негатив рисунка.
10. Изменить цветовую гамму BMP-файла.
11. Вырезать окружность из рисунка.

Дан чистый BMP-файл. Необходимо:

12. Нарисовать прямоугольник и закрасить красным цветом.
13. Нарисовать квадрат и закрасить зеленым цветом.
14. Нарисовать два синих прямоугольника в левом верхнем углу и нижнем правом углу соответственно.
15. Нарисовать прямоугольник, закрашенный синим цветом, и квадрат, закрашенный красным цветом.
16. Нарисовать окружность желтого цвета.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Классифицируйте файлы по типу и методу доступа элементов файла.
2. Опишите структуру типизированного/нетипизированного файлов.
3. Объясните способы записи/чтения в/из типизированный(ого) файл(а).
4. Объясните способы записи/чтения в/из нетипизированный(ого) файл(а).
5. Дайте определение BMP-файла.
6. Приведите структуру BMP-файла.
7. Расскажите какого рода информация может храниться в BMP-файле?

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 2 занятия (4 академических часа: 3 часа на выполнение и сдачу практического задания и 1 час на подготовку отчета).

Номер варианта студенту выдается преподавателем.

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания, этапы выполнения работы, результаты выполнения работы, выводы.

ОСНОВНАЯ ЛИТЕРАТУРА

1. Алексеев В.Е. Графы и алгоритмы. Структуры данных. Модели вычислений [Электронный ресурс]/ В.Е. Алексеев, В.А. Таланов. — Электрон. текстовые данные. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 153 с. — Режим доступа: http://www.iprbookshop.ru/52186.html

2. Вирт Никлаус. Алгоритмы и структуры данных [Электронный ресурс]/ Никлаус Вирт— Электрон. текстовые данные. — Саратов: Профобразование, 2017. — 272 с.— Режим доступа: http://www.iprbookshop.ru/63821.html

3. Самуйлов С.В. Алгоритмы и структуры обработки данных [Электронный ресурс]: учебное пособие/ С.В. Самуйлов. — Электрон. текстовые данные. — Саратов: Вузовское образование, 2016. — 132 с.— Режим доступа: http://www.iprbookshop.ru/47275.html

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

4. Костюкова Н.И. Графы и их применение [Электронный ресурс]/ Н.И. Костюкова. — Электрон. текстовые данные. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 147 с. — Режим доступа: http://www.iprbookshop.ru/52185.html

5. Сундукова Т.О. Структуры и алгоритмы компьютерной обработки данных [Электронный ресурс]/ Т.О. Сундукова, Г.В. Ваныкина. — Электрон. текстовые данные. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 749 с.— Режим доступа: http://www.iprbookshop.ru/57384.html

Электронные ресурсы:

6. Научная электронная библиотека http://elibrary.ru
7. Электронно-библиотечная система «ЛАНЬ» http://e.lanbook.com
8. Электронно-библиотечная система «IPRbooks» http://www.iprbookshop.ru
9. Электронно-библиотечная система «Юрайт» http://www.biblio-online.ru
10. Электронно-библиотечная система «Университетская библиотека ONLINE» http://www.biblioclub.ru