

8 ЛАБОРАТОРНАЯ РАБОТА № 8

«ИССЛЕДОВАНИЕ СРЕДСТВ УПРАВЛЕНИЯ ПОТОКАМИ ВВОДА-ВЫВОДА. ИССЛЕДОВАНИЕ МЕХАНИЗМА ОБРАБОТКИ ИСКЛЮЧЕНИЙ»

8.1 Цель работы

Изучить способы реализации и особенности управления потоками ввода/вывода, исследовать способы генерации и обработки исключений.

8.2 Краткие теоретические сведения

8.2.1 Классы потокового ввода/вывода

Поток – это абстрактное понятие, относящееся к любому переносу данных от источника к приемнику. Потоки C++, в отличие от функций ввода/вывода в стандартном C, обеспечивают надежную работу как со стандартными, так и с определенными пользователем типами данных, а также единообразный и понятный синтаксис.

Пример:

```
#include <iostream>
using namespace std;
int main(){
    int i;
    cin >> i;                // ввод переменной
    cout << "Вы ввели " << i; // вывод (цепочка
вывода)
    return 0;
}
```

Чтение данных из потока называется извлечением, вывод в поток – помещением, или включением. Поток определяется как последовательность байтов и не зависит от конкретного устройства, с которым производится обмен (оперативная память, файл на диске, клавиатура или принтер). Обмен с потоком для увеличения скорости передачи данных производится, как правило, через специальную область оперативной памяти – буфер. Фактическая передача данных выполняется при выводе после заполнения буфера, а при вводе – если буфер исчерпан.

По направлению обмена потоки можно разделить на входные (данные вводятся в память), выходные (данные выводятся из памяти) и двунаправленные (допускающие как извлечение, так и включение).

По виду устройств, с которыми работает поток, можно разделить потоки на стандартные, файловые и строковые.

Стандартные потоки предназначены для передачи данных от клавиатуры и на экран дисплея, файловые потоки – для обмена информацией с файлами на внешних носителях данных (например, на магнитном диске), а строковые потоки – для работы с массивами символов в оперативной памяти.

Для поддержки потоков библиотека C++ содержит иерархию классов, построенную на основе двух базовых классов – `ios` и `streambuf`. Класс `ios` содержит общие для ввода и вывода поля и методы, класс `streambuf` обеспечивает буферизацию потоков и их взаимодействие с физическими устройствами.

Таблица 8.1 – Классы потокового ввода/вывода и буферизации потоков

<code>ios</code>	базовый класс потоков
<code>istream</code>	класс входных потоков
<code>ostream</code>	класс выходных потоков
<code>iostream</code>	класс двунаправленных потоков
<code>istringstream</code>	класс входных строковых потоков
<code>ostringstream</code>	класс выходных строковых потоков
<code>stringstream</code>	класс двунаправленных строковых потоков
<code>ifstream</code>	класс входных файловых потоков
<code>ofstream</code>	класс выходных файловых потоков
<code>fstream</code>	класс двунаправленных файловых потоков
<code>iomanip</code>	класс манипуляторов двунаправленных потоков
<code>bitset</code>	класс для работы с двоичными числами
<code>streambuf</code>	базовый класс буферизации потоков
<code>filebuf</code>	класс буферизации файловых потоков

Основным преимуществом потоков по сравнению с функциями ввода/вывода, унаследованными из библиотеки C, является контроль типов, а также расширяемость, то есть возможность работать с типами, определенными пользователем. Для этого требуется переопределить операции потоков.

К недостаткам потоков можно отнести снижение быстродействия программы, которое, в зависимости от реализации компилятора, может быть весьма значительным.

Заголовочный файл `<iostream>` содержит, кроме описания классов для ввода/вывода, четыре предопределенных объекта потокового ввода/вывода (табл. 8.2).

Эти объекты создаются при включении в программу заголовочного файла `<iostream>`, при этом становятся доступными связанные с ними средства ввода/вывода. Имена этих объектов можно переназначить на другие файлы или символьные буферы.

В классах `istream` и `ostream` операции извлечения из потока `>>` и помещения в поток `<<` определены путем перегрузки операций сдвига.

Операции извлечения и чтения в качестве результата своего выполнения формируют ссылку на объект типа `istream` для извлечения и ссылку на `ostream` – для включения. Это позволяет формировать цепочки операций, что проиллюстрировано последним оператором приведенного ранее примера. Вывод при этом выполняется слева направо.

Как и для других перегруженных операций, для вставки и извлечения невозможно изменить приоритеты, поэтому в необходимых случаях используются скобки:

```
cout << i + j;      // Скобки не требуются – приоритет сложения
                      // больше, чем <<
cout << (i < j);    // Скобки необходимы – приоритет операции
                      // отношения меньше, чем <<:
cout << (i << j);    // Правая операция << означает сдвиг
```

Таблица 8.2 – Стандартные потоки для ввода/вывода

Объект	Класс	Описание
<code>cin</code>	<code>istream</code>	Связывается с клавиатурой (стандартным буферизованным вводом)
<code>cout</code>	<code>ostream</code>	Связывается с экраном (стандартным буферизованным выводом)
<code>Cerr</code>	<code>ostream</code>	Связывается с экраном (стандартным не буферизованным выводом), куда направляются сообщения об ошибках
<code>clog</code>	<code>ostream</code>	Связывается с экраном (стандартным буферизованным выводом), куда направляются сообщения об ошибках

8.2.2 Флаги и форматирующие методы

Флаги представляют собой отдельные биты, объединенные в поле `x_flags` типа `long` класса `ios`. Флаги перечислены в таблице 8.3. Форматирующие методы приведены в таблице 8.4.

Таблица 8.3 – Флаги форматирования

Флаг	Умол- чение	Описание действия при установленном бите
1	2	3
<code>skipws</code>	+	При извлечении пробельные символы игнорируются
<code>left</code>		Выравнивание по левому краю поля
<code>right</code>	+	Выравнивание по правому краю поля
<code>internal</code>		Знак числа выводится по левому краю, число – по правому.
<code>dec</code>	+	Десятичная система счисления
<code>oct</code>		Восьмеричная система счисления
<code>hex</code>		Шестнадцатеричная система счисления
<code>showbase</code>		Выводится основание системы счисления (0x для шестнадцатеричных чисел и 0 для восьмеричных)

showpoint		При выводе вещественных чисел печатать десятичную точку и дробную часть
uppercase		При выводе использовать символы верхнего регистра
showpos		Печатать знак при выводе положительных чисел
scientific		Печатать вещественные числа в форме мантиссы с порядком
fixed		Печатать вещественные числа в форме с фиксированной точкой
unitbuf		Выгружать буферы всех потоков после каждого вывода
stdio		Выгружать буферы потоков stdout и stderr после каждого вывода

ПРИМЕЧАНИЕ:

Флаги (left, right и internal), (dec, oct и hex), а также (scientific и fixed) взаимно исключают друг друга, то есть в каждый момент может быть установлен только один флаг из каждой группы.

Таблица 8.4 – Функции форматирования

Функция	Описание действия
int width(int w);	устанавливает ширину поля вывода в соответствии со значением параметра
int width();	возвращает значение ширины поля вывода
char fill(char);	устанавливает значение текущего символа заполнения, возвращает старое значение символа
char fill();	возвращает текущий символ заполнения
int precision(int);	устанавливает значение точности представления при выводе вещественных чисел, возвращает старое значение точности
int precision();	возвращает значение точности представления при выводе вещественных чисел
long flags(long f);	установить состояния всех флагов
long flags();	вернуть состояния всех флагов
long setf (long setbits, long field);	присваивает флагам, биты которых установлены в первом параметре, значение соответствующих битов второго параметра
long setf(long);	устанавливает флаги, биты которых установлены в параметре
long unsetf(long);	сбрасывает флаги, биты которых установлены в параметре

Пример форматирования при выводе с помощью флагов и методов:

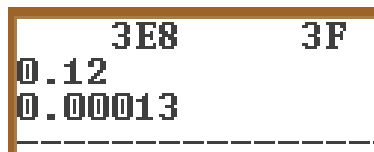
```
#include <iostream>
```

```

using namespace std;
int main() {
    long a = 1000, b = 077;
    cout.width(7);
    cout.setf(ios::uppercase);
    cout << hex<<a;
    cout.width(7);
    cout << b << endl;
    double d = 0.12, c = 1.3e-4;
    cout.setf(ios::left);
    cout << d << endl;
    cout << c;
    return 0;
}

```

В результате работы программы (рисунок 8.1) в первой строке будут прописными буквами выведены переменные *a* и *b* в шестнадцатеричном представлении, под каждую из них отводится по 7 позиций (функция *width* действует только на одно выводимое значение, поэтому ее вызов требуется повторить дважды). Значения переменных *c* и *d* прижаты к левому краю поля:



```

      3E8      3F
0.12
0.00013

```

Рисунок 8.1 – Результат работы программы

8.2.3 Манипуляторы

Манипуляторами называются функции, которые можно включать в цепочку операций помещения и извлечения для форматирования данных. Манипуляторы делятся на простые, не требующие указания аргументов, и параметризованные. Пользоваться манипуляторами более удобно, чем методами установки флагов форматирования (табл. 8.5).

Таблица 8.5 – Манипуляторы

Манипулятор	Назначение	Поток
1	2	3
dec	Вывод чисел в десятичной системе счисления	Вывод
endl	Вывод символа новой строки и флэширование	Вывод
ends	Вывод нуля (NULL)	Ввод\вывод
flush	Флэширование	Вывод
hex	Вывод чисел в шестнадцатеричной системе счисления	Вывод
oct	Вывод чисел в восьмеричной системе счисления	Вывод

<code>resetiosflags(long f)</code>	Сбросить флаги, определяемые <code>f</code>	Ввод\вывод
<code>setbase(int base)</code>	Установить основание системы счисления	Вывод
<code>setfill(char ch)</code>	Установить символ заполнения <code>ch</code>	Вывод
<code>setiosflags(long f)</code>	Установить флаги, задаваемые <code>f</code>	Ввод\вывод
<code>setprecision(int p)</code>	Установить точность, равную <code>p</code>	Вывод
<code>setw(int w)</code>	Установить ширину поля, равную <code>w</code>	Вывод
<code>ws</code>	Пропускает начальный символ-разделитель	

8.2.4 Методы обмена с потоками

В потоковых классах наряду с операциями извлечения `>>` и включения `<<` определены методы для неформатированного чтения и записи в поток (при этом преобразования данных не выполняются).

В таблице 8.6 приведены функции чтения, определенные в классе `istream`.

Таблица 8.6 – Функции чтения

Функция	Описание действия
<code>get ()</code>	возвращает код извлеченного из потока символа или EOF
<code>get (c)</code>	возвращает ссылку на поток, из которого выполнялось чтение, и записывает извлеченный символ в <code>c</code>
<code>get(buf, num, lim='\n')</code>	считывает <code>num-1</code> символов или пока не встретится символ <code>lim</code> и копирует их в символьную строку <code>buf</code> . Вместо символа <code>lim</code> в строку записывается признак конца строки <code>('\\0')</code> . Символ <code>lim</code> остается в потоке. Возвращает ссылку на текущий поток
<code>getline(buf, num, lim='\n')</code>	аналогична функции <code>get</code> , но копирует в <code>buf</code> и символ <code>lim</code>
<code>ignore(num = 1, lim = EOF)</code>	считывает и пропускает символы до тех пор, пока не будет прочитано <code>num</code> символов или не встретится разделитель, заданный параметром <code>lim</code> . Возвращает ссылку на текущий поток
<code>seekg(pos)</code>	устанавливает текущую позицию чтения в значение <code>pos</code> байт от начала файла
<code>seekg(off, org)</code>	перемещает текущую позицию чтения на <code>off</code> байтов, считая от трех позиций, определяемых параметром <code>org</code> : <code>ios::beg</code> (от начала файла),

	<code>ios::cur</code> (от текущей позиции) или <code>ios::end</code> (от конца файла)
<code>flush()</code>	записывает содержимое буфера потока вывода на физическое устройство
<code>put (c)</code>	выводит в поток символ <code>c</code> и возвращает ссылку на поток
<code>seekg(pos)</code>	устанавливает текущую позицию записи в значение <code>pos</code>

В классе `ostream` определены аналогичные функции для вывода (табл. 8.7).

Таблица 8.7 – Функции вывода

Функция	Описание действия
<code>seekg (offs, org)</code>	перемещает текущую позицию записи на <code>offs</code> байтов, считая от трех позиций, определяемых параметром <code>org</code> : <code>ios::beg</code> (от начала файла), <code>ios::cur</code> (от текущей позиции) или <code>ios::end</code> (от конца файла)
<code>write(buf, num)</code>	записывает в поток <code>num</code> символов из массива <code>buf</code> и возвращает ссылку на поток

8.2.5 Файловые потоки

Чтобы открыть для вывода файл `myfile.txt` с помощью объекта `ofstream`, необходимо создать экземпляр объекта класса `ofstream` и передать ему имя файла в качестве параметра: `ofstream fout («myfile.txt»)`.

Чтобы открыть этот файл для ввода, применяется та же методика, за исключением того, что используется объект класса `ifstream`: `ifstream fin («myfile.txt»)`.

Обратите внимание, что `fout` и `fin` не более чем имена объектов; здесь `fout` использовался для вывода в файл подобно тому, как `cout` используется для вывода на экран; `fin` аналогичен `cin`.

Очень важным методом, используемым в файловых потоках, является функция-член `close()`. Каждый раз при открытии файла для чтения или записи (или и того и другого) создается соответствующий объект файлового потока. По завершении работы файл необходимо закрыть (например: `fout.close()`); чтобы впоследствии не повредить его и записанные в нем данные. На практике нередко бывают непредвиденные случаи, когда не закрытые файлы теряют всю внесенную в них информацию.

После того как объекты потока будут ассоциированы с файлами, они используются наравне с другими объектами потока ввода и вывода. Например:

```
#include <fstream>
```

```

#include <iostream>
using namespace std;
int main() {
    char buffer[255];          // для ввода данных
    пользователь
    cout << "File name: ";
    cin.getline (buffer,255);
    ofstream fout(buffer);    // открыть для записи
    fout << "This line written directly to the
file\n";
    cin.getline (buffer,255); // получить данные от
    пользователь
    fout << buffer;           // и запись их в файл
    fout.close();            // закрыть файл
    return 0;                // уходя, гасите свет
}

```

Обычно объект класса `ofstream`, открывая файл для записи, создает новый файл, если таковой не существует, или усекает его длину до нуля, если файл с этим именем уже существует (то есть удаляет все его содержимое). Изменить стандартное поведение объекта `ofstream` можно с помощью второго аргумента конструктора, заданного явно.

Допустимыми аргументами являются:

- `ios::app` – добавляет данные в конец файла, не усекая прежнее его содержимое (append - добавлять);
- `ios::ate` – осуществляет переход в конец файла, но запись допускает в любом месте файла (at end – в конец);
- `ios::trunc` – задано по умолчанию; усекает существующий файл полностью (truncate - обрезать);
- `ios::nocreate` – открывает существующий файл, если его нет – ошибка (не создавать);
- `ios::noreplace` – открывает несуществующий файл, иначе выдаст ошибку (не заменять).

8.2.6 Ошибки потоков

Каждый поток (`istream` или `ostream`) имеет связанное с ним состояние. Установкой и соответствующей проверкой этого состояния выявляются ошибки и нестандартные ситуации. Состояние потока вводится в `basic_ios`, базовом классе класса `basic_stream`, в `<ios>` (таблице 8.8).

Таблица 8.8 – Функции работы с ошибками потоков

Функция	Описание действия
<code>bool good() const;</code>	следующая операция может выполняться
<code>bool fail() const;</code>	следующая операция не выполнится
<code>bool eof() const;</code>	виден конец ввода

<code>bool bad() const;</code>	поток испорчен
<code>iosstate rdstate() const;</code>	получение флагов состояния ввода/вывода
<code>void clear(iosstate f=goodbit);</code>	сбрасываются флаги ошибок (по умолчанию - все). Обычно после возникновения ошибки нужно сбросить ошибочное состояние, чтобы дальше пользоваться потоком, но этого недостаточно – для восстановления работоспособности еще нужно вручную очистить буфер ввода/вывода.
<code>void setstate(iosstate f) {clear(rdstate() f);}</code>	добавление f к флагам состояния
<code>operator void*() const;</code>	не ноль, если !fail()
<code>bool operator!() const {return fail();}</code>	не good()

Состояние потока представляется набором флагов. Эти флаги определены в базовом классе `ios`:

```
enum io_state{
    goodbit = 0x00, // если бит не установлен, то ошибок нет
    eofbit = 0x01, // обнаружен конец файла
    failbit = 0x02, // сбой в последней операции ввода-вывода
    badbit = 0x04, // попытка недопустимой операции
    hardfail = 0x80 // в потоке невозстанавливаемая ошибка
};
```

Пример: проще всего состояние потока можно проверить как обычное логическое выражение. Обычно нужно проверить, ввел ли пользователь то, что ожидает программа:

```
double d; cin>>d;
/* 1) 2 - преобразуется к d=2.0
```

2) 2А - d=2.0 - значение сформировано, но <А> еще осталось в буфере потока и ждет приема `char`

3) 3,3 вместо 3.3 - d=3.0, но запятая и вторая 3 осталась в буфере ввода и ждет ввода

4) вводим ААА - поток испорчен, вырабатывается флаг ошибки, пользоваться d нельзя! Значение d не изменилось!!! Весь последующий ввод (cin>>) будет проигнорирован!*/

if(!cin) { /* Сюда попадем, если только поток ввода Ваш ввод никаким образом проинтерпретировать не может (случай 4). Для того, чтобы программу можно было выполнять дальше необходимо:

1) сбросить ошибки */

```
cin.clear(); // иначе весь последующий ввод будет проигнорирован
```

```
// 2) очистить буфер ввода
cin.ignore(MAXINT, '\n');
cin>>d; } // теперь можно вводить снова
```

8.2.7 Обработка исключительных ситуаций

Исключительная ситуация, или **исключение** – это возникновение непредвиденного или аварийного события, которое может порождаться некорректным использованием аппаратуры.

Например, это деление на ноль или обращение по несуществующему адресу памяти. Обычно эти события приводят к завершению программы с системным сообщением об ошибке. C++ дает программисту возможность восстанавливать программу и продолжать ее выполнение.

Исключения C++ **не поддерживают** обработку асинхронных событий, таких, как ошибки оборудования или обработку прерываний, например, нажатие клавиш Ctrl+C. Механизм исключений **предназначен** только для событий, которые происходят в результате работы самой программы и указываются явным образом.

Для управления исключениями в C++ используются три ключевых слова: `try`, `catch` и `throw`.

Ключевое слово `try` служит для обозначения блока кода, который может генерировать исключение. При этом соответствующий блок заключается в фигурные скобки и называется защищенным или `try`-блоком:

```
try {
    //Защищенный блок кода
}
```

Тело всякой функции, вызываемой из `try`-блока, также принадлежит `try`-блоку. Предполагается, что одна или несколько функций или инструкций из защищенного блока могут выбрасывать (или генерировать) исключение. Если выброшено исключение, выполнение соответствующей функции или инструкции приостанавливается, все оставшиеся инструкции `try`-блока игнорируются, а управление передается вовне блока `try`.

Ключевое слово `catch` следует непосредственно за `try`-блоком и обозначает секцию кода, в которую передается управление, если произойдет исключение. За ключевым словом `catch` следует описание исключения, заключенное в круглые скобки. Описание исключения состоит из имени типа исключения и необязательной переменной:

```
catch (<имя типа>[<переменная>])
{
    //Обработчик исключения
}
```

Имя типа исключения идентифицирует обслуживаемый тип исключений. Блок кода, обрабатывающего исключение, заключается в фигурные скобки и называется `catch`-блоком или обработчиком исключения.

При этом говорят, что данный catch-блок перехватывает исключения описанного в нем типа. Если исключение перехвачено, переменная получает его значение. Если вам не нужен доступ к самому исключению, указывать эту переменную не обязательно. Переменная исключения может иметь любой тип данных, включая созданные пользователем типы классов.

За одним try-блоком могут следовать несколько catch-блоков. Оператор catch, с указанным вместо типа исключения многоточием, перехватывает исключения любого типа и должен быть последним из операторов catch, следующих за try-блоком. Рассмотрим простой пример обработки исключения:

```
int main()
{   double x, y;
    cout << "Введите x: ";
    try {
        cin >> x;
        if (!cin)      throw '1';
        if (x == 0)    throw 1;
        if (1/x < 0)   throw 1.01;
        cout << "y =sqrt(1/x) ";
        y=sqrt(1/x);
        cout << "y=" << y << endl;
    }
    catch(char) {cout << "Ошибка ввода" << endl;}
    catch(int)   {cout << "Делить на 0 нельзя" <<
endl;}
    catch(double d) {cout << "Извлекать корень из отр.
числа нельзя d=" <<d<< endl;}
    catch(...) {cout << "Ошибка" << endl;}
    cout << "продолжение работы программы";
    return 0;
}
```

В try-блоке располагается код, который потенциально может вызвать ошибку в работе программы, а именно ошибку в 3-х случаях (вводе символа, деления на 0, извлечения корня из отрицательного числа).

Задаем условие если – `if(!cin)` – поток испорчен, то будет сгенерировано исключение типа `char`. В этом случае try-блок сразу прекращает выполнение дальнейших команд, а `'1'` «падает» в `catch`. В нашем примере он выводит "Ошибка ввода". При этом программа продолжает работать и выполнять команды, размещенные ниже.

Если же `x` будет равен нулю, то в try-блоке будет сгенерировано исключение типа `int`, этом случае try-блок прекращает выполнение дальнейших команд, а `1` «падает» в `catch` и на экран будет выведено "Делить на 0 нельзя".

Если же x будет меньше нуля, то в `try`-блоке будет сгенерировано исключение типа `double`, этом случае `try`-блок прекращает выполнение дальнейших команд, а `1.0` «падает» в `catch` и на экран будет выведено "Извлекать корень из отр. числа нельзя $d=1.0$ ".

Иначе выполнится команда `y=sqrt(1/x)`, а `catch` не сработает.

Часто вложенные `try/catch` блоки возникают неявно, в результате создания защищенного блока в функции, которая сама находится в защищенном блоке.

К вложенным `try/catch`-блокам приходится прибегать потому, что какая-то функция может непредвиденно привести к исключению. В этом случае простейший выход – заключить весь код в функции `main()` в такой блок, причем для перехвата исключения использовать обработчик вида `catch(...)`. Затем можно использовать средства, предоставляемые компилятором для определения места в программе, вызвавшего появление исключения.

8.2.8 Перехватывание исключений

Когда с помощью `throw` генерируется исключение, функции исполнительной библиотеки C++ выполняют следующие действия:

- 1) создают копию параметра `throw` в виде статического объекта, который существует до тех пор, пока исключение не будет обработано;
- 2) в поисках подходящего обработчика раскручивают стек, вызывая деструкторы локальных объектов, выходящих из области действия;
- 3) передают объект и управление обработчику, имеющему параметр, совместимый по типу с этим объектом.

При раскручивании стека все обработчики на каждом уровне просматриваются последовательно, от внутреннего блока к внешнему, пока не будет найден подходящий обработчик.

Обработчик считается найденным, если тип объекта, указанного после **throw**:

- тот же, что и указанный в параметре `catch` (параметр может быть записан в форме `T`, `const T`, `T&` или `const T&`, где `T`- тип исключения);
- является производным от указанного в параметре `catch` (если наследование производилось с ключом доступа `public`);
- является указателем, который может быть преобразован по стандартным правилам преобразования указателей к типу указателя в параметре `catch`.

Обработчики производных классов следует размещать до обработчиков базовых, поскольку в противном случае им никогда не будет передано управление. Обработчик указателя типа `void` автоматически скрывает указатель любого другого типа, поэтому его также следует размещать после обработчиков указателей конкретного типа.

8.2.9 Возможности механизма исключений

Язык C++ не позволяет возвращать значение из конструктора и деструктора. Механизм исключений дает возможность сообщить об ошибке, возникшей в конструкторе или деструкторе объекта.

Если в конструкторе объекта генерируется исключение, автоматически вызываются деструкторы для полностью созданных в этом блоке к текущему моменту объектов, а также для полей данных текущего объекта, являющихся объектами, и для его базовых классов. Например, если исключение возникло при создании массива объектов, деструкторы будут вызваны только для успешно созданных элементов. Если объект создается в динамической памяти с помощью операции `new` и в конструкторе возникнет исключение, память из-под объекта корректно освобождается.

8.3 Порядок выполнения лабораторной работы

8.3.1. В ходе самостоятельной подготовки изучить основы работы с потоковым вводом/выводом и исключениями.

8.3.2. Разработать согласно варианту программу на языке C++, состоящую из двух частей: первая демонстрирует умение управлять потоками ввода-вывода, вторая демонстрирует умение генерировать и перехватывать исключения.

8.3.3. Разработать тестовые примеры и выполнить отладку программы.

8.3.4. Получить результаты работы программы и исследовать её свойства для различных режимов работы, сформулировать выводы.

8.3.5. Оформить отчет по проделанной работе.

8.4 Варианты заданий

Вариант 1

Описать класс геометрическая фигура **прямоугольник**, содержащий следующие поля: ширина, высота, цвет прямоугольника.

Написать следующие методы:

- метод заполнения частных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади прямоугольника по формуле:

$S = a \times b$, где a , b – ширина и высота прямоугольника. Результат вычисления вывести в 17-ти позициях с точностью 5 знаков, пробелы необходимо заменить знаком «&», выравнивание выполнить по левому краю.

Вариант 2

Описать класс геометрическая фигура **ромб**, содержащий следующие поля: сторона, диагональ, угол между сторонами, цвет ромба.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади ромба по формуле:

$S = a^2 \times \sin(\alpha)$, где a – известная сторона, α – угол между сторонами. Результат вычисления вывести в 20-ти позициях с точностью 3 знака, пробелы необходимо заменить знаком «@», выравнивание выполнить по правому краю.

Вариант 3

Описать класс геометрическая фигура **круг**, содержащий следующие поля: d – диаметр, цвет круга.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади круга по формуле:

$S = \pi \times d^2 : 4$, где d – это диаметр, π – это константа, равная 3.14. Результат вычисления вывести в 17-ти позициях с точностью 6 знаков, пробелы необходимо заменить знаком «^», выравнивание выполнить по левому краю.

Вариант 4

Описать класс геометрическая фигура **треугольник**, содержащий следующие поля: a , b и c – стороны треугольника, R – радиус описанной окружности, цвет треугольника.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади треугольника по формуле:

$S = (a \times b \times c) : 4 \times R$, где a , b и c – стороны треугольника, R – радиус описанной окружности. Результат вычисления вывести в 12-ти позициях с точностью 3 знака, пробелы необходимо заменить знаком «~», выравнивание выполнить по правому краю.

Вариант 5

Описать класс геометрическая фигура **прямоугольник**, содержащий следующие поля: диагональ, угол между диагоналями, цвет прямоугольника.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади прямоугольника по формуле:

$$S = 0,5 \times d^2 \times \sin(a), \text{ где } d - \text{ диагональ прямоугольника.}$$

Результат вычисления вывести в 12-ти позициях с точностью 3 знака, пробелы необходимо заменить знаком «*», выравнивание выполнить по левому краю.

Вариант 6

Описать класс геометрическая фигура **треугольник**, содержащий следующие поля: *a*, *b* и *c* – стороны треугольника, *r* – радиус вписанной окружности, цвет треугольника.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади треугольника по формуле:

$S = p \times r$, где *p* – полупериметр треугольника, *r* – радиус вписанной окружности. Результат вычисления вывести в 22-х позициях с точностью 5 знаков, пробелы необходимо заменить знаком «+», выравнивание выполнить по правому краю.

Вариант 7

Описать класс геометрическая фигура **равнобедренная трапеция**, содержащий следующие поля: *a*, *b* – параллельные стороны, *c* – длина одинаковых сторон, цвет трапеции.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления периметра трапеции по формуле:

$P = a + b + 2 \times c$, где *a*, *b* – параллельные стороны, *c* – две длины одинаковых сторон. Результат вычисления вывести в 12-ти позициях с точностью 3 знака, пробелы необходимо заменить знаком , выравнивание выполнить по левому краю.

Вариант 8

Описать класс геометрическая фигура **круг**, содержащий следующие поля: *L* – длина окружности, цвет круга.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади круга по формуле:

$S = L^2 : (4 \times \pi)$, где L – это длина окружности, π – это константа, равная 3.14. Результат вычисления вывести в 14-ти позициях с точностью 3 знака, пробелы необходимо заменить знаком «/», выравнивание выполнить по правому краю.

Вариант 9

Описать класс геометрическая фигура **прямоугольник**, содержащий следующие поля: ширина, высота, диагональ, цвет прямоугольника.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления периметра прямоугольника по формуле:

$P = 2 \times (a + b)$, где a – ширина, b – высота. Результат вычисления вывести в 18-ти позициях с точностью 5 знаков, пробелы необходимо заменить знаком «!», выравнивание выполнить по левому краю.

Вариант 10

Описать класс геометрическая фигура **треугольник**, содержащий следующие поля: длина основания, высота, цвет треугольника.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади треугольника по формуле:

$S = 0,5 \times a \times h$, где a – длина основания, h – высота, проведенная к основанию. Результат вычисления вывести в 19-ти позициях с точностью 5 знаков, пробелы необходимо заменить знаком «&», выравнивание выполнить по правому краю.

Вариант 11

Описать класс геометрическая фигура **ромб**, содержащий следующие поля: диагональ1, диагональ2, цвет ромба.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода

«некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади ромба по формуле:

$S = 0,5 \times (d1 \times d2)$, где $d1$, $d2$ – две диагонали. Результат вычисления вывести в 19-ти позициях с точностью 4 знака, пробелы необходимо заменить знаком «?», выравнивание выполнить по левому краю.

Вариант 12

Описать класс геометрическая фигура **круг**, содержащий следующие поля: d – диаметр, цвет круга.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления длины окружности по формуле:

$L = 2 \times r \times \pi$, где r – радиус, π – это константа, равная 3.14. Результат вычисления вывести в 14-ти позициях с точностью 8 знаков, пробелы необходимо заменить знаком «[», выравнивание выполнить по правому краю.

Вариант 13

Описать класс геометрическая фигура **треугольник**, содержащий следующие поля: a и b – две стороны, α – угол между ними, цвет треугольника.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади треугольника по формуле:

$S = 0,5 \times a \times b \times \sin(\alpha)$, где a и b – две стороны, α – угол между ними. Результат вычисления вывести в 12-ти позициях с точностью 3 знака, пробелы необходимо заменить знаком «%», выравнивание выполнить по левому краю.

Вариант 14

Описать класс геометрическая фигура **параллелограмм**, содержащий следующие поля: a и b – две стороны, α – угол между ними, цвет параллелограмма.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода

«некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади параллелограмма по формуле:

$S = a \times b \times \sin(\alpha)$, где a и b – две стороны, α – угол между ними.

Результат вычисления вывести в 17-ти позициях с точностью 5 знаков, пробелы необходимо заменить знаком «\», выравнивание выполнить по правому краю.

Вариант 15

Описать класс геометрическая фигура **трапеция**, содержащий следующие поля: a , b – два разных основания, h – высота трапеции, цвет трапеции.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади трапеции по формуле:

$S = (a + b) : 2 \times h$, где a , b – два разных основания, h – высота трапеции. Результат вычисления вывести в 17-ти позициях с точностью 4 знака, пробелы необходимо заменить знаком «.», выравнивание выполнить по левому краю.

Вариант 16

Описать класс геометрическая фигура **круг**, содержащий следующие поля: r – радиус, цвет круга.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади круга по формуле:

$S = \pi \times r^2$, где r – это радиус, π – это константа, равная 3.14. Результат вычисления вывести в 17-ти позициях с точностью 6 знаков, пробелы необходимо заменить знаком «-», выравнивание выполнить по правому краю.

Вариант 17

Описать класс геометрическая фигура **параллелограмм**, содержащий следующие поля: $d1$, $d2$ – диагонали, β – угол между диагоналями, цвет параллелограмма.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода

«некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади параллелограмма по формуле:

$S = 0,5 \times (d1 \times d2) \times \sin(\beta)$, где $d1$, $d2$ – диагонали, β – угол между диагоналями. Результат вычисления вывести в 18-ти позициях с точностью 5 знаков, пробелы необходимо заменить знаком «'», выравнивание выполнить по левому краю.

Вариант 18

Описать класс геометрическая фигура **треугольник**, содержащий следующие поля: a , b и c – стороны треугольника, цвет треугольника.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления периметра треугольника по формуле:

$P = a + b + c$, где a , b , c – длина стороны. Результат вычисления вывести в 15-ти позициях с точностью 4 знака, пробелы необходимо заменить знаком «_», выравнивание выполнить по правому краю.

Вариант 19

Описать класс геометрическая фигура **параллелограмм**, содержащий следующие поля: a – сторона, h – высота, цвет параллелограмма.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади параллелограмма по формуле:

$S = a \times h$, где a – сторона, h – высота. Результат вычисления вывести в 18-ти позициях с точностью 5 знаков, пробелы необходимо заменить знаком «'», выравнивание выполнить по левому краю.

Вариант 20

Описать класс геометрическая фигура **эллипс**, содержащий следующие поля: a – длина большей полуоси эллипса, b – длина меньшей полуоси эллипса, цвет эллипса.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления периметра эллипса по формуле:

$P = 4 \times (\pi \times a \times b + (a-b)) / (a+b)$, где a - длина большей полуоси эллипса, b - длина меньшей полуоси эллипса, π – это константа, равная 3.14. Результат вычисления вывести в 17-ти позициях с точностью 5 знаков, пробелы необходимо заменить знаком «:», выравнивание выполнить по правому краю.

Вариант 21

Описать класс геометрическая фигура **параллелограмм**, содержащий следующие поля: ширина, высота, цвет параллелограмма.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления периметра параллелограмма по формуле:

$P = 2 \times (a + b)$, где a – ширина, b – высота. Результат вычисления вывести в 12-ти позициях с точностью 3 знака, пробелы необходимо заменить знаком «@», выравнивание выполнить по левому краю.

Вариант 22

Описать класс геометрическая фигура **трапеция**, содержащий следующие поля: $d1$, $d2$ – диагонали, β – угол между диагоналями, цвет трапеции.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади трапеции по формуле:

$S = 1/2 \times d1 \times d2 \times \sin(\beta)$, где: $d1$, $d2$ – диагонали, β – угол между диагоналями. Результат вычисления вывести в 12-ти позициях с точностью 3 знака, пробелы необходимо заменить знаком «#», выравнивание выполнить по правому краю.

Вариант 23

Описать класс геометрическая фигура **эллипс**, содержащий следующие поля: a – длина большей полуоси эллипса, b – длина меньшей полуоси эллипса, цвет эллипса.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади эллипса по формуле:

$S = \pi \times a \times b$, где a - длина большей полуоси эллипса, b - длина меньшей полуоси эллипса, π - это константа, равная 3.14. Результат вычисления вывести в 14-ти позициях с точностью 8 знаков, пробелы необходимо заменить знаком «,», выравнивание выполнить по левому краю.

Вариант 24

Описать класс геометрическая фигура прямоугольный **треугольник**, содержащий следующие поля: a , b - катеты, цвет треугольника.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади треугольника по формуле:

$S = 0,5 \times a \times b$, где a , b - катеты. Результат вычисления вывести в 19-ти позициях с точностью 5 знаков, пробелы необходимо заменить знаком «}», выравнивание выполнить по правому краю.

Вариант 25

Описать класс геометрическая фигура **прямоугольник**, содержащий следующие поля: ширина, высота, диагональ, цвет прямоугольника.

Написать следующие методы:

- метод заполнения приватных полей класса считанными с клавиатуры данными; состояние потока после ввода каждого поля нужно проверять и генерировать исключения в случае ошибки в потоке и в случае ввода «некорректных» данных (не положительное число, несоответствие радиуса вписанной окружности длинам сторон треугольника и т.д.);

- метод вычисления площади прямоугольника по формуле:

$S = a * \sqrt{(d^2 - a^2)}$, где a - известная сторона, d - диагональ прямоугольника. Результат вычисления вывести в 12-ти позициях с точностью 3 знака, пробелы необходимо заменить знаком «@», выравнивание выполнить по левому краю.

8.5 Содержание отчета о выполнении лабораторной работы

Титульный лист, цель работы, вариант задания, текст программы с комментариями, описание тестовых примеров и выводы по проделанной работе.

8.6 Контрольные вопросы

8.6.1 Объясните понятие «поток ввода/вывода».

8.6.2 Поясните механизм буферизации потоков ввода/вывода.

8.6.3 Приведите примеры работы потоков ввода/вывода с различными типами данных.

8.6.4 Перечислите виды средств форматирования потоков.

8.6.5 Расскажите о функциях и флагах форматирования потоков ввода/вывода.

8.6.6 Расскажите о манипуляторах.

8.6.7 Опишите файловые потоки и режимы работы с файлами.

8.6.8 Дайте определение термину «исключение», опишите ситуации возникновения исключений.

8.6.9 Опишите работу механизма обработки исключений.

8.6.10 Расскажите о вложенных конструкциях `try/catch`, о правилах оформления `catch`-блоков.