Разработка консольных приложений на языке Си (Часть 2). Лабораторный практикум для дисциплины "Проектирование программного обеспечения" для студентов специальности 113 Прикладная математика согласно НБ-14-113/16 (Лекции 34 часов, практические занятия 68 часов, самостоятельная работа 108 часов)

А.Г. Пискунов

27 июня 2019 г.

Содержание

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Целью преподавания дисциплины "Проектирование программного обеспечения" является ознакомление студентов с основными концепциями, понятиями, методами и технологямм объектно-ориентированного программирования; выработкой у них практических навыков, достаточных для самостоятельной разработки програмного обеспечения;

Данная дисциплина состоит из двух тематических модулей, которые завершается сдачей модульной работы, курс завершается дифференциальным зачетом.

Кроме того, считается, что читатель освоил материал курса [16]. А сам курс является основой для курса [15]. Последнюю версию документа можно найти по адресу [17].

Основную литературу для курса по вопросам, связанным с языком C, см. [11, 6]; по вопросам, связанным с языком C++, см. [20, 8]. В качестве дополнительной литературы см. [10, 5, 21, 22]. И очень рекомендуется по каждому вопросу заглядывать в [3].

Сдача каждой работы (все равно лабораторной или модульной) предполагает передачу файлов исходного кода приложения, файлов с тестами, файлов командного интерпретатора для выполнения тестов и файла с отчетом о работе. Кроме того, отчеты по модульной работе сдаются в еще и в напечатанном виде.

Успешное выступление в любом отборочном туре любого соревнования на сайте codeforces.com тоже засчитывается как выполнение модульной работы.

1.1 Требования к приложениям и данным

Любое приложение должно:

- небольшие порции (одно, два, три значения) входных данных брать из аргументов командной строки.
- большие порции данных (текст или координаты перемещения транспортного средства) брать из текстового файла, если не оговорено противное, то через стандартный ввод.
- по ключам '-?', '/?', '-h','/h', '-help', '/help' выдавать подсказку для использования.
- если не оговорено противное, результаты работы выводить в стандартный вывод.
- по ключу '-v' в стандартный вывод ошибок выводить дополнительную информацию о ходе работы приложения, например, время работы приложения.
- должно состоять более чем из одного файла и иметь глобальную переменную.

- Во всех текстовых файлах с данными символ '#' является комментарием и сам символ и текст справа от него до конца строки должен игнорироваться.
- в случае преобразования чего либо в что либо предоставлять возможность выполнить его в обе стороны. Например, при требовании преобразования из десятичного представления целого числа в двоичное, приложение должно мочь выполнять обратное - из двоичного в десятичное.

2 МОДУЛЬ 1: ОСНОВЫ ПРОЦЕДУРНОГО ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ C++

2.1 ЛАБОРАТОРНАЯ РАБОТА: ТЕКСТОВОЕ ПРЕДСТАВЛЕНИЕ ВЕЩЕСТВЕННЫХ ЧИСЕЛ

Тема:

Анализ текстов.

Цель:

Разработать функцию для решения вопроса: является ли текст вещественой константой (то есть, тест содержит не пустую последовательность цифр, с, возможно, более одним символом '.', которая, возможно, начинается с символа '-' или '+'). Прототип функции:

```
int isDouble  (const char *str); // текстовое представление вещественного числа.
```

Задание:

- Познакомиться с обработкой аргументов командной строки.
- Научиться записывать формальные требования к лексемам в виде функции.

2.2 ЛАБОРАТОРНАЯ РАБОТА: СВЕТОФОР

Тема:

Перечисление и оператор switch.

Цель:

Разработать приложение для выбора действий пешехода на светофоре. Приложение обязано сообщать три вида диагностики:

- о отсутствии данных. Пример ошибки оператора:

  ```
  a.exe -v
  ```

- о не правильном типе данных (введенное данное не является целым числом). Пример ошибки оператора:

a.exe hello

- о не правильных данных (введенное данное не является кодом цвета) Пример ошибки оператора (при условии, что 57 не является одним из трех цветов):

  a.exe 57

Подсказка приложения:

a.exe [-?] [-v] number
где
number - код цвета (100 - красный, 101 - желтый, 102 - зеленый)

Задание:

- Научиться обрабатывать обязательные аргументы командной строки.
- Использовать оператор множественного выбора switch.
- Использовать константы перечисления enum.

2.3 ЛАБОРАТОРНАЯ РАБОТА: ИЗМЕНЕНИЕ ЗАДАННОГО БАЙТА ФАЙЛА

Тема:

Файлы с прямым доступом.

Цель:

Разработать приложение для изменения в заданном файле заданного байта на заданное значение.

Подсказка приложения:

Приложение предназначено для замены заданного байта в заданном файле.
Использование:
a.exe [-?] [-v] -f flNm -n number -b bite
где
-f flNm - имя файл для редактирования
-n number - номер байта для замены значения
-b byte - значение байта на которое надо заменить (число от 0 до 255).

Три ключа -f, -n и -b являются обязательными. Требуется выполнять проверку их параметров flNm, number и byte на наличие в массиве аргументов командной строки (argv), два последних на то, что параметры являются целочисленными значениями и byte является числом от 0 до 255.

Задание:

- Познакомиться с типом данных FILE *.
- Познакомиться с функциями fopen, fclose.
- Научиться работать с файлами при помощи функций fread и fwrite.

2.3.1 Усложненный вариант

В усложненном варианте лабораторной предлагается зашифровать заданный файл побитовой операцией исключающее ИЛИ (XOR - символ операции: ^) заданным значением байта. Для смещения в конец файла и в начало можно использовать функцию fseek (FILE *, int , int) ;

```
fseek(f, 0, SEEK_END ); // сместиться в конец файла
fseek(f, 0, SEEK_SET ); // сместиться в начало файла
```

2.4 ЛАБОРАТОРНАЯ РАБОТА: СОЗДАНИЕ ТАБЛИЦЫ СЛОВ

Тема:

Структуры.

Цель:

Разработать приложение для построения таблицы частоты вхождения слов в некоторм файле заданного размера. При переполнении таблицы сообщить оператору об ошибке.

Подсказка приложения:

```
Приложение для составления таблицы вхождения слов в файл.
a.exe [-?] [-v]   [-c NUM]   [ -desc | -asc ]  -m  MAX
где
-m  MAX    - максимальный размер таблицы слов
-c NUM     - номер  колонки по которой сортировать таблицу (1 или 2)
-desc      - задать убывающий порядок сортировки  (от z до a)
-asc       - задать возрастающий порядок сортировки  (от a до z)
-v         - показывать время работы приложения
```

Как видно из подсказки, обязательным ключем является только -m. Наличие параметра, тип параметра и корректность значений параметра у ключей -c и -m выполнять как в лабораторной 2.3 .

Задание:

- Требуемая таблица должна имееть вид вроде:

```
#   слово      кол-во вхождений в файл
hello      1
world      1
Smith      3
...
```

- Познакомиться с типом данных структура, ввести структуру wrd.
- Разработать функции для работы со структурами:

```
void swap1 (wrd *a, wrd *b);          // обмен значениями между параметрами a и b
int   wrdcmp(const wrd *a, const wrd *b, int txtNmbr);
                          // сравнения структур по слову или количеству вхождений
                                  // значение возвращать аналогично strcmp
void  sort  (wrd *tbl,  size_t tblSz);    // сортировка таблицы
void  print (const char * title,
             const wrd *tbl,  size_t tblSz);  // вывод таблицы
```

2.4.1 Варианты разработки приложения

В простом варианте структура wrd имеет вид:

```
#define   WSZ  50
typedef   unsigned int  uint;
struct  wrd {
  uint cnt;     // счетчик вхождений слова
  char w[WSZ];  // место для хранения слова
};
```

При такой структуре необходимо один раз захватить память под массив структур и, в конце работы, один раз освободить.

Более сложный вариант:

```
typedef   unsigned int  uint;
struct  wrd {
  uint cnt;     // счетчик вхождений слова
  char *w;      // указатель на место для хранения слова
};
```

В этом варианте, при внесении каждого нового слова в таблицу дополнительно требуется захватить память для хранения слова. Кроме того, после выполнения работы приложения и перед освобождением памяти массива структур, освободить всю память захваченную для хранения каждого слова.

2.5 ЛАБОРАТОРНАЯ РАБОТА: ВЫВОД ЛОКАЛЬНОГО ВРЕМЕНИ

Тема:

Ссылки

Цель:

Разработать набор функций для вывода структуры локального времени struct tm при помощи указателей в виде '2019/02/02 22:07:02'

---- File:./cpp/dttmtoa/dttmto.h

```
#ifndef _INC_DTTMTO_H
#define _INC_DTTMTO_H

#include <stdio.h>
#include <string.h>
#include <time.h>

//
//  функции для преобразования времен в текстовое представление
//

char * dttoa  ( time_t dtTm, char *b  = 0);
char * tmtoa  ( time_t dtTm, char *b  = 0);
char * dttmtoa( time_t dtTm, char *b  = 0);


char * dtToA  ( const struct tm * dtTm, char *b = 0);
char * tmToA  ( const struct tm * dtTm, char *b = 0);
char * dtTmToA( const struct tm * dtTm, char *b = 0);

#endif

---- End Of File:./cpp/dttmtoa/dttmto.h
```

Задание:

- Познакомиться с указателем на структуры.
- Познакомиться с передачей указателей на структуры в функции.
- Использовать параметр по умолчанию.
- Использовать массив символов (в котором будет храниться текстовое представление даты) закрытый от доступа из функций, расположенных вне файла, содержащего разрабатываемые функции.
- Использовать спецификаторы вывода целых чисел для правильного вывода лидирующих нулей.

Замечание

Все приложения в последующих работах по ключу -v должны выводить сколько секунд потрачено на выполнение приложения и дату запуска приложения.

2.6 ЛАБОРАТОРНАЯ РАБОТА: СПИСКИ

Тема:

Однонаправленные списки.

Цель:

Разработать функции (

```
line *mkLine  (const char * str);  // создание элемента списка по строчке
```

и

```
void delLine  (line * list);       // удаление списка
```

) при помощи структуры

```
struct  line{
     static uint   total; // статическая переменная для хранения кол-ва
                          // созданных элементов списка.
     char *        text;
     line *        next;
};
```

Используя их, переписать функции

```
void addStr  ( const char *str); //  добавления строчки к памяти
void freeMem  ( );               //  освобожденние памяти
void priMem  ();                 // вывод содержимого введенного файла
```

из [16, лаб. 2.14], ЛАБОРАТОРНАЯ РАБОТА: ДИНАМИЧЕСКАЯ ПАМЯТЬ.

Задание:

- Использовать библиотеку слежения за утечками памяти crtdbg.h.

```
#include <crtdbg.h>
void main()
    _CrtSetDbgFlag(
    _CrtSetDbgFlag(_CRTDBG_REPORT_FLAG)
                | _CRTDBG_CHECK_ALWAYS_DF
                | _CRTDBG_LEAK_CHECK_DF
      );
    _CrtSetReportMode(_CRT_ASSERT, _CRTDBG_MODE_FILE);
    _CrtSetReportFile(_CRT_ASSERT, _CRTDBG_FILE_STDERR);
    _CrtSetReportMode(_CRT_WARN, _CRTDBG_MODE_FILE);
    _CrtSetReportFile(_CRT_WARN, _CRTDBG_FILE_STDERR);
    _CrtSetReportMode(_CRT_ERROR, _CRTDBG_MODE_FILE);
    _CrtSetReportFile(_CRT_ERROR, _CRTDBG_FILE_STDERR);
 //
 //   тут код для выполнения задания
 //
}
```

- в случае использования компилятора из командной строки компилировать следующим образом:

```
cl /MTd  /D_DEBUG /oa.exe *.cpp
```

- Выполнить построчное чтение текстового файла при помощи следующей функции ввода-вывода из stdio.h:

  char * fgets(char * str, int num, FILE * fl);

- Засечь время работы приложения при помощи функции

  time_t time (time_t * tn); // где time_t секунды прошедшие с 1 января 1970 года.

- Сравнить время работы приложения с приложением, которое работало без структур, ссылающихся на себя. (См. [16, лаб. 2.14], ЛАБОРАТОРНАЯ РАБОТА: ДИНАМИЧЕСКАЯ ПАМЯТЬ

2.7 ТЕМАТИКА МОДУЛЬНОЙ РАБОТЫ

2.7.1 Задания для модульной работы

Разработать консольное приложение для:

1. нахождения координат пересечения двух отрезков, каждый из которых, задается четырьмя вещественными числами. Данные должны лежать в текстовом файле, при этом одна пара чисел записывается в одной строке, координата x - первая, координата y - вторая. Разделяются любым количеством пробелов и табуляций. Прибилизительно в таком виде:

   ```
   1 1.0
         2.2          3.3
    3.3  5.9
      10.09  2
   ```

   Наличие и тип значений проверять аналогично лабораторной 2.3 .

2. конвертации последовательности координат транспортного средства (в каждой строчке файла содержится широта и долгота, см. предыдущее задание) из текстового файла в двоичный и наоборот:

   ```
   a.exe [-?]  [-v]  -f flNm  { -txt2bin | -bin2txt }
   где
   -txt2bin   взять координаты со stdin и вывести в двоичный файл  flNm
   -bin2txt   взять координаты из двоичного файла flNm и вывести в stdout
   ```

3. конвертации последовательности координат экрана (номер пикселя по горизонтали, номер пикселя по вертикали) из текстового файла в двоичный и наоборот.

4. вывода простых чисел до заданного алгоритмом решето Эратосфена (захватить память для целочисленного массива) и измерение времени работы программы.

5. сортировки файла с переменными среды окружения см. https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D1%8B_Windows

```
a.exe [-?]  [-v] { -desc | -asc }
где
-desc   - задать убывающий порядок сортировки  (от z до a)
-asc    - задать возрастающий порядок сортировки  (от a до z)
```

6. изменения кодировок текстового файла 1251 в 866 и наоборот.

```
a.exe  [-?]  [-v]   { -win2dos | -dos2win }
где
-win2dos   кодировку 1251 заменить на 866
-dos2win   кодировку 866 заменить на 1251
```

2.7.2 Вопросы для модульной работы

1. Как используется объявление typedef? Примеры синонимов типов.
2. Какое средство языка C++ предназначено для замены макросов с параметрами? Пример.
3. Что такое inline - функция?
4. Формальные параметры функций по умолчанию.
5. Шаблоны функций.
6. Какие есть возможности описания составных типов в Си?
7. Оператор ветвления switch.
8. Стандартные файлы ввода-вывода консольного приложения. Способы их перенаправления. Переменные языка соответствующие файлам.
9. Функции ввода-вывода Си.
10. Что такое перечисление?
11. Оператор ветвления switch.
12. Что такое структура? Приведите пример описания структуры.
13. Что такое шаблон функции?
14. Функция из библиотеки string.h, которая используется для разделения текста на лексемы.
15. Что такое объединение (union)?
16. Дана структура для хранения координат МГС84:

```
struct coor {
   double B;
   double L;
}
```

Переделайте её в структуру для хранения списка координат МГС84.

17. Дана структура для хранения списка координат МГС84 (У последнего елемента списка указатель next равен 0):

```
struct coor {
   double B;
   double L;
   coor * next;
}
```

Напишите метод для подсчета длины списка.

18. Дана структура для хранения списка координат МГС84 (У последнего елемента списка указатель next равен 0):

```
struct coor {
   double B;
   double L;
   coor * next;
}
```

Напишите метод для вывода всего списка в стандартный вывод.

19. Дана структура для хранения списка координат МГС84 (У последнего елемента списка указатель next равен 0):

```
struct coor {
   double B;
   double L;
   coor * next;
}
```

Напишите статический метод для удаления первого элемента списка.

20. Дана структура для хранения списка координат МГС84 (У последнего елемента списка указатель next равен 0):

```
struct coor {
   double B;
   double L;
   coor * next;
}
```

Напишите метод для вставки в начало список еще одного элемента.

21. Дана структура для хранения списка координат МГС84 (У последнего елемента списка указатель next равен 0):

```
struct coor {
   double B;
   double L;
   coor * next;
}
```

Напишите деструктор для этого списка.

3 МОДУЛЬ 2: НЕКОТОРЫЕ ОСОБЕННОСТИ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ C++

3.1 ЛАБОРАТОРНАЯ РАБОТА: ПРЯМОУГОЛЬНИК, СОДЕРЖАЩИЙ ТРЕК

Тема:

Конструктор класса

Цель:

Разработать приложение для вычисления минимального прямоугольника, содержащего последовательность координат транспортного средства. В каждой строке файла записаны два вещественных числа широта и долгота. Использование ключа -v влечет вывод в стандартный файл ошибок список строчек с ошибками.

a.exe [-?] [-v] [-8]
где
-8 координаты полученного прямоугольника выдавать до 8-го знака после запятой, иначе до 6-го.

Задание:

- Делить читаемые строчки файла на отдельные координаты при помощи функции strtok;
- Создать класс для хранения двух координат coor2 с тремя перегруженными конструкторами. Один конструктор - без параметров (по умолчанию), второй - с двумя вещественными параметрами, третий - с указателем на символы.

```
coor2:: coor2();
coor2:: coor2(double B, double L);
coor2:: coor2(const char * str);
```

- Создать класс для хранения прямоугольника rectangle, с методом expande (coor2 nextPoint) - расширяющий прямоугольник со следующим прототипом:

```
void rectangle:: expande (coor2 & nextPoint);
```

- В классе rectangle предусмотреть возможность вывода координат с точностью 6 знаков или 8 знаков после запятой.

3.2 ЛАБОРАТОРНАЯ РАБОТА: ТАБЛИЦА СЛОВ

Тема:

Перегрузка конструктора и операции вывода

Цель:

Разработать приложение аналогичное 2.4 без ограничения на размер таблицы.

Задание:

- Добавить в структуру wrd рекурсивный указатель на себя, так чтобы в ней можно было хранить связный список.
- Прототипы функций swap1 и cmp из раздела 2.4 переделать на ссылки.
- функцию для вывода таблицы слов print переписать при помощи перегруженной операцию вывода одного элемента таблицы в поток cout. Пример использования перегруженной операции вывода

```
wrd a;
cout <<a;
```

- Разработать конструктор и деструктор для структуры.

3.3 ЛАБОРАТОРНАЯ РАБОТА: КОНВЕРТОР ТРЕКА

Тема:

Полиморфизм

Цель:

Разработать приложение для конвертации последовательности координат транспортного средства (широта-долгота или широта-долгота-высота) из текстового файла в двоичный и наоборот

Задание:

- Из класса coor2 (см. 3.1) наследовать класс coor3 с третьей координатой (высота).
- Для демонстрации позднего связывания в класс coor2 добавить виртуальные методы ввода- вывода из двоичных файлов, в классе coor3 переопределить их.

3.4 ЛАБОРАТОРНАЯ РАБОТА: ЕЩЕ РАЗ О ТАБЛИЦЕ СЛОВ

Тема:

Операции и классы

Цель:

Переделать функцию сортировки таблицы слов из лабораторной 3.2 так, чтобы она использовала системный шаблон swap (или собственный шаблон для обмена значений).

Задание:

- Перегрузить операцию присваивания и конструктор копирования структуры.
- в функции сортировки таблицы слов использовать системный шаблон swap.

3.5 ТЕМАТИКА МОДУЛЬНОЙ РАБОТЫ

3.5.1 Задания для модульной работы

Разработать консольное приложение для:

1. выдачи сдачи из кассы. Текущее состояние кассы берутся из стандартного ввода: в каждой строчке содержится номинал монеты (или банкноты в копейках) и количество монет указанного номинала. Величину сдачи взять из командной строки в копейках. хранятся в текстовом файле.

   a.exe [-?] [-v] -s SSS
   где
   -s SSS сумма сдачи

   Приложение должно выдать сдачу в виде списка номиналов монет (и банкнот) и их количеств которую выдают из кассы.

2. выдачи сдачи, даные хранятся в двоичном файле.

3. рисования трека (широта, долгота и высота) при помощи псевдографики.

4. преобразования CSV- файла с треком (широта, долгота и высота) в XML-формат.

5. для преобразования десятичного целого в двоичную систему исчисления.

   a.exe [-?] [-v] [-r] SSS
   где
   SSS число в десятичной или двоичной системе исчисления
   -r преобразовать из двоичной в десятичную системы исчисления.

6. для преобразования десятичного целого в восмеричную систему исчисления.

7. для преобразования десятичного целого в шеснадцатиричную систему исчисления.

8. для символьного сложения - отнимания или умножения - деления двух чисел. Действия выполнять над целыми числами произвольной длины, то есть, не использовать арифметику Си.

9. тестирования класса управляющее выводом градусов в форме

   - целые градусы, минуты с дробной частью.
   - целые градусы, целые минуты, секунды с дробной частью.

3.5.2 Вопросы для модульной работы

1. Перегрузить операцию вывода '«' для своего класса.
2. Перегрузить операцию присванивания '=' для своего класса.
3. Перегрузить конструктор копирования для своего класса.
4. Что такое тип ссылка? Примеры использования.
5. Использование ссылок в качестве аргументов функций. Отличие от использование указателей.
6. Области видимости и пространство имен, операция разрешения области видимости, создание своей области видимости. (Namespace - именованная и фигурные скобки - локальная)
7. Директива расширения умолчательной области видимости - using.
8. Для чего может использоваться ключевое слово static?
9. Что такое класс? Приведите пример описания класса.
10. Что такое наследование?
11. Что такое инкапсуляция?
12. Что такое полиморфизм?
13. Что такое метод класса? Зачем используется ключевое слово this?
14. Что такое конструктор и деструктор?
15. Какие есть модификаторы доступа? (public, private, protected)
16. Напишите основные отличия классов от структур.
17. Что такое объект?
18. Что такое перегрузка операции (оператора)?
19. Дан класс для хранения координат МГС84:

```
class coor {
  public:
  double B;
  double L;
}
```

    Наследуйте из него класс для хранения этих координат и высоты.

20. Дан класс для хранения имени человека:

```
class coor {
  public:
  char * name;
  char * surname;
}
```

Напишите для него оператор присваивания.

21. Дан класс для хранения имени человека:

```
class coor {
  public:
  char * name;
  char * surname;
}
```

Напишите для него конструктор копирования.

22. Дан класс для хранения имени человека:

```
class coor {
  public:
  char * name;
  char * surname;
}
```

Переопределите операцию вывода класса в поток вывода (cout).

4 ВОПРОСЫ ДЛЯ ЗАЧЕТА

Вопросы для зачета берутся из соответствующих разделов первого (см. 2.7.2) и второго (см. 3.5.2) модуля, а так же раздела ВОПРОСЫ ДЛЯ МОДУЛЬНОЙ РАБОТЫ из курса "Алгоритмические языки программирования".

5 ПРАКТИЧЕСКИЕ РАБОТЫ

Отчеты к работам, излагаемым в этом разделе должны удовлетворять требованиям к оформлению отчетов раздела 6 и дополнительно содержать две UML - диаграммы. Различные примеры UML - диаграмм, относящихся к потокам данных, можно посмотреть в работе [18, с. 4] или [18, с. 7]. Диаграммы строились при помощи свободно-распостраняемой утилиты graphviz (см. [13]). Примеры UML - диаграмм классов, которые описывающая структуру системы, демонстрирующая классы системы, их атрибуты, методы и зависимости между классами, можно посмотреть в пояснительной записке проекта [14] (файл args.2.10.pdf на стр.26). Эта диаграмма был сгенерирована из кода приложения при помощи утилиты doxygen (см. [12]).

5.1 ПРАКТИЧЕСКАЯ РАБОТА: СОЗДАНИЕ И ВЫПОЛНЕНИЕ НОВОГО ПРОЦЕССА

Тема:

Процессы

Цель:

Разработать приложение для выполнения машины Тьюринга ([7]).

This program use a Turing machine
app.exe [-d] [-?] {script(name.tm)} {argument1} [argument2]
Where:
script - some file with extention *.tm for MT
argumentX - some decimal numbers for Turing machine
-d - print debug
-? - help
We have this script in case:
compare.tm - это программа !!!!! для сравнения двух чисел по меньше равно
div.tm - деление 2 чисел
even_odd.tm - не четное или четное число (не четное - 1, четное - 0)
int_dev2.tm - деление нацело на 2
mul.tm - умножение 2 чисел
vid.tm - differents of two numbers

Задание:

- В случае запроса справки по приложению, произвести поиск файлов с расширением .tm и показать первую строчку коментария из каждого файла, если она есть.
- Подготовить аргументы командной строки для запуска машины Тьюринга;
- Упрощенная версия - породить процесс машины Тьюринга через функцию system;
- Нормальная версия - породить процесс машины Тьюринга через одну из функций семейства _spawn;
- Получить код завершения процесса и выдать диагностику в случае ошибок.

5.2 ПРАКТИЧЕСКАЯ РАБОТА: РИСОВАНИЕ ТРЕКА ПСЕВДОГРАФИКОЙ

Тема:

Псевдографика

Цель:

Разработать приложение для рисования трека движения транспортного средства псевдографикой, то есть символами таблицы ASCII

Задание:

- Найти прямоугольник, содержащий трек, как в лабораторной 3.1

- Предусмотреть диагностику ошибок при чтении файла с треком;
- Использовать связный список для хранения координат транспортного средства;
- Использовать две (широта и долгота) или три (широта, долгота и высота) колонки из файла;
- В случае использования трех координат отображать различные высоты различными ASCII символами. Например, '.' - очень низко, ':' - низко, '+' - высоко, '*' - очень высоко;

6 ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТОВ СДАВАЕМЫХ РАБОТ

Титульная страница должна содержать название министерства, университета, института, кафедры, темы модульной работы, исполнителя, преподавателя, город и год.

Объем работы не менее 10 листов. Текст надо набирать шрифтом Times New Roman, 12 pt, интервал между строчками - 1, форматирование по ширине, отступы: 2 см слева, 1.5 - справа, сверху и снизу страницы, абзац - 1 см. Нумерация страниц - внизу по центру.

Список литературы оформлять согласно требований ДСТУ [9].

Отчеты к модульным работам должны содержать следующие части:

- Титульный лист;
- Содержание;
- Постановка задачи;
- Теоретическая часть (описание предметной области, краткие теоретические сведения, необходимые для выполнения задачи, описание инструментов программирования и т.д.);
- Описание алгоритма программы;
- Описание использования приложения;
- Тесты для проверки работоспособности;
- Выводы;
- Список используемой литературы.

В отчетах к лабораторным некоторые части разрешается опускать.

7 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ВЫПОЛНЕНИЯ РАБОТ

7.1 ЛАБОРАТОРНАЯ РАБОТА: СПИСКИ

При выполнении лабораторной 2.6 (имитируется работа со стеком) желательно выполнять следующие действия:

- переписать с учетом структуры line (которая ссылается на себя и позволяет создавать односвязные списки) функции addStr, freeMem, priMem;
- примерная реализации функции addStr:

```
line * list = 0;  // глобальная переменная
!!wrong command: ..!!void addStr (const char * str){
  if (str) {   // обязательная проверка указателей
      line * head = 0;
      head   = mkLine (str);
//      head = new line(str); // версия с конструктором и деструктором
      head->next = list;
      list = head;
      head = 0;
  }
}
```

- примерная реализация freeMem:

```
line * list = 0;  // глобальная переменная
!!wrong command: ..!!void freeMem (){
  if (list) {      // обязательная проверка указателей
    delLine (list);  // цикл по удалению тел элементов
//    delete list;  //   версия с конструтором и деструктором
                  // списка выполнять в деструкторе
    list = 0;
  }
}
```

- написать функцию замены строки с прототипом:

```
char  * strrplc(char * oldStr, const char * str);
```

  которая, если требуется, освободит старую память oldStr. Захватит память достаточную для хранения строки из str. Скопирует туда содержимое str и отдаст её в качестве возвращаемого значения.

- описание структуры line расположить в файле line.h (естественно, с защитой от повторного включения), код необходимых методов - в line.cpp;
- создавать новый элемент списка следущим образом:

```
line * item =0;
char * str ="hello";
item = (line *)malloc(sizeof(line));    // создаем элемент
item::total++;
// line::total++; // альтернативная версия доступа к статическим полям
item->text=0;
item->next=0;
item->text = strrplc(item->text, str);  // элемент готов
```

- добавлять элемент в список следующим образом:

```
line * list =0;
// создать  элемент
item->next = list; // к элементу добавили уже готовый стек
list = item;       // вернули его в переменную списка
item = 0;          // элемент уже не нужен. можно создавать следующий
```

- освобождать начало списка следующим образом:

```
item = list->next;
list->next = 0;
if (list->text){
    free(list->text);
    item->text=0;
}
free(list);
list = 0;
list = item;
item = 0;
```

- для построчного чтения стандартного ввода использовать fgets:

```
#define   BUF_SZ 1024
 ...
 char buf [BUF_SZ+1] = {0};
 ...
 while (fgets(buf, BUF_SZ, stdin) == buf){
   // создать элемент для текущей строки
   // добавить элемент в список
 }
 ...
```

7.2 ЛАБОРАТОРНАЯ РАБОТА: ПРЯМОУГОЛЬНИК, СОДЕРЖАЩИЙ ТРЕК

При выполнении лабораторной 3.1 желательно:

- для класса coor2 создать два файла coor2.h и coor2.cpp;
- файл coor2.h содержит текст:

```
class coor2{
  public:        // два поля
    double lat;  //  широта
    double lng;  //  долгота
                 // прототипы трех конструкторов
    coor2 ();
    coor2 (double B, double L);
    coor2 (const char * str);
};
```

В отличие от структур в классах все поля по умолчанию создаются с модификатором доступа private, поэтому в нашем примере явно открывается доступ к полям класса при помощи модификатора доступа public;

- файл coor2.cpp должне содержать определения конструкторов. Например, умолчательный конструктор:

```
coor2::coor2 (){
   lat = 0.0;
   lng = 0.0;
}
```

- добавить определения остальных двух конструкторов;
- В файле, содержашем точку входа main, объявить и инициировать объект типа coor2 двумя разными способами:

```
coor2  test (1.0, 2.0);          // применить второй конструктор, нет захвата памяти
coor2 *test2  = new coor2();      // применить первый конструктор + захват памяти
 // тут  работа по подсчету прямоугольника
 // тестовый вывод координат
 if (pFlag)
 fprintf (
   stderr
   ,"first point: (%.6f Nord, %.6f East); second point: (%.6f Nord, %.6f East); "
   , test.lat, test.lng, test2->lat, test2->lng);
 delele test2; test2=0;                // освободить память
```

- создать и провертить третий конструктор;
- в функцию main добавить цикл для построчного чтения файла в объект типа coor2;
- похожим образом создаем файлы для класса rectangle;
- в методе rectangle::expande первый раз используется тип ссылка.

7.3 ЛАБОРАТОРНАЯ РАБОТА: ТАБЛИЦА СЛОВ

При выполнении лабораторной 3.2 желательно:

-

8 НЕКОТОРЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ И ПРИМЕРЫ

Для лучшего понимания тонкостей использования основополагающих понятий типа, класса и объекта дополнительно рекомендуется ознакомиться с монографией Люки Карделли и Мартина Абади 'Теория Объектов' , см. [1]. Для некоторых глав из неё существует перевод см. [19].

8.1 Получение кода завершения приложения

Код завершения приложения это целое положительное число (см. например, [4]), задается оператором

```
return X;
```

в функции main, либо оператором

```
exit(X);
```

в любой функции приложения, где X- целая переменная. При этом 0 - означает нормальную работу приложения, X > 0 - означает, что работа приложения закончилась с ошибкой. Познакомиться с Отрицательные коды возвращать нежелательно, в связи с спецификацией функций семейства spawn, которые порождают новый процесс и, в некоторых случаях, возвращают код завершения приложения. Они возвращают код -1, если не удалось найти приложения для запуска.

Ниже приводится приложение, которое возвращает целое, полученное из аргумента командной строки:

---- File:./cpp/errorlevel/source.cpp

```
//#include"Header.h"
#include <iostream>
#include <ctype.h>
#include <stdlib.h>
#include<stdio.h>
#include<string.h>


int main(int argc, char **argv)
{
  int rc = 1;
  if (argc <= 1) {
     printf ("nothing to do!");
  }
  else {
    rc = atoi(argv[1]);
  }

  //exit (rc);
  return rc ;
```

```
}
```

---- End Of File:./cpp/errorlevel/source.cpp

Если приложение выполнялось из скрипта командного интерпретатора Winsows (см. например, [2]), то код завершения приложения попадает в переменную %errorlevel% и доступен для однократного использования. После чего значение переменной сбрасывается в 0. Ниже приводится скрипт для исполнения приведенного выше приложения:

---- File:./cpp/errorlevel/u.run.cmd

```
@echo off

@echo off
if %1. == -?.   goto usage
if %1. == -h.   goto usage

rem

rem
a.exe %1
set rc=%errorlevel%



echo off
echo rc is %rc%
goto done

:usage
    @echo off
    echo utility    to show errorlevel varianle
    echo ************************************************
    echo.
    echo Usage: %0 number
    echo.
    echo example: %0  4
    echo.


:done
```

---- End Of File:./cpp/errorlevel/u.run.cmd

8.2 Перегрузка операции присваивания и конструктора копирования

Добавленные в приложение класс или структура, которые содержат указатели, могут привести к краху программы в совершенно безобидных на первый взгляд ситуациях. Обычно крахом может завершится передача объекта

по значению в какую либо функцию, либо выполнение обычной операции присвоения (а значит и попытка использовать системную функцию swap для сортировок) вроде

```
a = b;
```

Причина кроется в побитном копировании содеражния объекта и, значит, копировании указателя. После такого присвоения оба объекта содержат указатель на один и тот же участок памяти. Если деструктор должен его освобождать (что, в общем, подразумевается), то деструктор первого объекта освободит память, а деструктор второго при попытке освобождения того же самого участка памяти приведет к краху приложения. Перегрузка операции присваивания и конструктора копирования и предназначена для того, что бы избегать такую проблему. В приведенном ниже примере, после операции присвоения, каждый из объектов будет указывать на свой участок памяти. Кроме того, пример демонстрирует использование условной компиляции. При закоментаренной строчке

```
//#define ERROR 1
```

приложение проработает хорошо. При раскоментаренной

```
#define ERROR 1
```

- работа приложения должна закончится крахом.

---- File:./cpp/copy/source.cpp

```
//#include"Header.h"
#include <iostream>
#include <ctype.h>
#include<stdio.h>
#include<string.h>
//#include <crtdbg.h>
//#define ERROR 1

class word {
public:
  char * w;
  int    cnt;
  word (const char * w){
    if (w) {
       this->w=(char *)malloc(strlen(w)+1);
       strcpy(this->w, w);
       cnt=1;
    }
    else {
       w = 0;
       cnt=0;
    }
  }
  ~word(){
     printf("\n *** destructor \n");
    if(w) {
```

```
      free(w);
      w=0;
    }
  }

/*       конструктор копирования
  word (const word& b) {
    printf("\n ***constructor is here\n");
    if (b.w) {
      w= (char *)malloc(strlen(b.w)+1);
      strcpy(this->w, b.w);
      cnt = b.cnt;
    }
    else {
      w=0;
      cnt = 0;
    }
  } */

#ifndef ERROR
  word & operator=(const word& b){ // операция присвоения
    printf("\n *** = is here \n");
    if(w) {
       printf("\n *** free old \n");
       free(w);
       w=0;
       cnt = 0;
    }
    else {
       this->w=0;
       cnt = 0;
    }
    if (b.w) {
      printf("\n *** malloc new \n");

      w= (char *)malloc(strlen(b.w)+1);
      strcpy(this->w, b.w);
      cnt = b.cnt;
    }
    return *this;
  }
#endif
};




int main()
{
_CrtSetDbgFlag(
_CrtSetDbgFlag(_CRTDBG_REPORT_FLAG)
```

```
| _CRTDBG_CHECK_ALWAYS_DF
| _CRTDBG_LEAK_CHECK_DF
);
_CrtSetReportMode(_CRT_ASSERT, _CRTDBG_MODE_FILE);
_CrtSetReportFile(_CRT_ASSERT, _CRTDBG_FILE_STDERR);
_CrtSetReportMode(_CRT_WARN, _CRTDBG_MODE_FILE);
_CrtSetReportFile(_CRT_WARN, _CRTDBG_FILE_STDERR);
_CrtSetReportMode(_CRT_ERROR, _CRTDBG_MODE_FILE);
_CrtSetReportFile(_CRT_ERROR, _CRTDBG_FILE_STDERR);

   word a("aaa1"), b("bbb");
   a=b;            //  использование операции присвоения
   printf("a/b: '%s'/'%s'",a.w, b.w);
}
```

---- End Of File:./cpp/copy/source.cpp

8.3 Перегрузка операции вывода

Перегрузка операции вывода используется, как и любая другая функция, для структурирования кода приложения и сокращению усилий разработчика. Пример перегрузки приводится в следующем приложении:

---- File:./cpp/cout/source.cpp

```
//#include"Header.h"
#include <iostream>
#include <ctype.h>
#include<stdio.h>
#include<string.h>
using namespace std;
//#include <crtdbg.h>

class word {
public:
  char * w;
  int    cnt;

  word (const char * w){
    if (w) {
       this->w=(char *)malloc(strlen(w)+1);
       strcpy(this->w, w);
       cnt=1;
    }
    else {
       this->w = 0;
       cnt=0;
    }
  }
  ~word(){
     printf("\n *** destructor");
```

```
    if(w) {
      printf(":'%s'", w);
      free(w);
      w=0;
    }
  }

  friend ostream& operator<< (ostream &out, const word& o){
    if (o.w)
      out<<"word/count: '"<<o.w<< "'/"<<o.cnt;
    else
      out<<"word/count: ''/"<<o.cnt;
    return out;
  }
};

int main()
{
   word a("aaa1"), b(0);
   cout<< a<<"\n"<<b;
   return 0;
}
```

---- End Of File:./cpp/cout/source.cpp

Предметный указатель

Список литературы

[1] Cardelli, Abadi. A theory of objects, 1996, Springer. – https://drive.google.com/open?id=1mShdblP3LnooSSfk8Osh_SAtIc54Jczu. 24

[2] Microsoft. Cmd.exe — интерпретатор командной строки. – https://ru.wikipedia.org/wiki/Cmd.exe. 25

[3] Microsoft. Язык C/C++ и стандартные библиотеки. – https://docs.microsoft.com/ru-ru/cpp/cpp/c-cpp-language-and-standard-libraries?view=vs-2019. 4

[4] OpenNET. Приложение C. Коды завершения, имеющие предопределенный смысл. – https://www.opennet.ru/docs/RUS/bash_scripting_guide/a14876.html. 24

[5] Айвор Хортон. Visual C++ 2010: полный курс = Ivor Hortonś Beginning Visual C++ 2010. – М.: «Диалектика», 2010. — С. 1216. 4

[6] Болски М.И. Язык программирования Си. Справочник: Пер. с англ. - М.: Радио и связь, 1988, - с.96. 4

[7] В.Ю.Лавринович. Програмна реалізація машини Тьюринга/ Л.М. Соснєв, В.Ю. Лавринович// Тези xvi міжнародної науково-практичної конференції молодих учених та студентів «ПОЛІТ. Сучасні проблеми науки. Інформаційно-діагностичні системи». – К.:НАУ, 2016, С.159, http://er.nau.edu.ua:8080/handle/NAU/36796. 19

[8] Герберт Шилдт. Полный справочник по C++, 4-е издание = C++: The Complete Reference, 4th Edition. – М.: «Вильямс», 2011. — с. 800. 4

[9] ДСТУ ГОСТ 7.1:2006. Бібліографічний запис, бібліографічний опис. Загальні вимоги та правила складання : метод. рекомендації з впровадження / уклали: Галевич О. К., Штогрин І. М. – Львів, 2008. – 20 с. 20

[10] Дэвид Р. Мюссер, Жилмер Дж. Дердж, Атул Сейни. C++ и STL: справочное руководство, 2-е издание (серия C++ in Depth) = STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library, 2nd edition, (C++ in Depth Series). – М.: «Вильямс», 2010. — с. 432. 4

[11] Керниган Б., Ритчи Д. Язык программирования Си. – Издательство Невский Диалект, 3-е издание,-2001.- с.290. 4

[12] Пискунов А.Г. Doxygen и Graphviz: документирование проектов на C# / А.Г. Пискунов, А.С.Горбань // Журн. Алгоритм.– №4(10), – 2006. http://www.realcoding.net/dn/docs/dgIntro.pdf. 18

[13] Пискунов А.Г. Graphviz и Sed: построение схем иерархии наследования / А.Г. Пискунов, С.М. Петренко // Журн. Алгоритм.– №3(9). – 2006. http://www.softcraft.ru/design/graphvizsed/mkhrrchy.pdf. 18

[14] Пискунов А.Г. Библиотека обработки аргументов командной строки. / А.Г.Пискунов, Д.И.Гись. // LXXIV-а наукова конференція професорсько-викладацького складу, аспірантів, студентів та співробітників відокремлених структурних підрозділів університету. – К.: НТУ, 2018. – с.421, https://drive.google.com/open?id=1TMVE6PF1rRqtpM2ZwQtBdnN5MnVZsaQF. 18

[15] Пискунов А.Г. Разработка консольных и оконных приложений на языке C# / Интернет ресурс. – 2019. – http://er.nau.edu.ua:8080/handle/NAU/37048. 4

[16] Пискунов А.Г. Разработка консольных приложений на языке Си (Часть 1) / Интернет ресурс. – 2018. – http://er.nau.edu.ua:8080/handle/NAU/36797. 4, 10, 11

[17] Пискунов А.Г. Разработка консольных приложений на языке Си (Часть 2) / Интернет ресурс. – 2018. – http://er.nau.edu.ua:8080/handle/NAU/38057. 4

[18] Пискунов А.Г. проектирование и декомпозиция потоков данных интерактивного приложения / Интернет ресурс. – 2009. – http://er.nau.edu.ua:8080/bitstream/NAU/39629/1/dataFlow.pdf. 18

[19] Пискунов А.Г. (Пер. с англ.). Классы и типы в языках, основанных на классах / Интернет ресурс. – 2010. – https://drive.google.com/open?id=1M9-geni3uvVJtlC41eWmK1D0q3T70-8m. 24

[20] Прата, Стивен. Язык программирования C++. Лекции и упражнения, 5-е изд. Пер. с англ. – М. «Вильямс», 2007. — С. 1184. 4

[21] Страуструп Б. Программирование: принципы и практика использования C++, исправленное издание. = Programming: Principles and Practice Using C++ — М.: «Вильямс», 2011. — С. 1248. 4

[22] Страуструп Б. Язык программирования C++ = The C++ Programming Language / Пер. с англ . – 3-е изд. — СПб.; М.: Невский диалект — Бином, 1999. 4