Архитектура вычислительных систем.

Вариант 250 (12,18)

Выполнила: Караваева Екатерина Aндреевна

БПИ207

1. **Описание задания**

### **Общие для всех альтернатив переменные:**

1. Название – строка символов,
2. Вес в граммах (целое)

### **Общие для всех альтернатив функции:**

Частное от деления суммы кодов незашифрованной строки на вес (действительное число)

### **Обобщенный артефакт** - животные

### **Базовые альтернативы**

1. Рыбы (место проживания – перечислимый тип: река, море, озеро…) 2. Птицы (отношение к перелету: перелетные, остающиеся на зимовку – булевская величина) 3. Звери (хищники, травоядные, насекомоядные… – перечислимый тип)

1. **Описание структуры ВС**

|  |  |
| --- | --- |
| **Типы данных** | |
| **Тип** | **Размер** |
| Int | 4 |
| Double | 8 |
| Bool | 1 |
| Char | 1 |
| FILE | 4 |

**Базовые Классы**

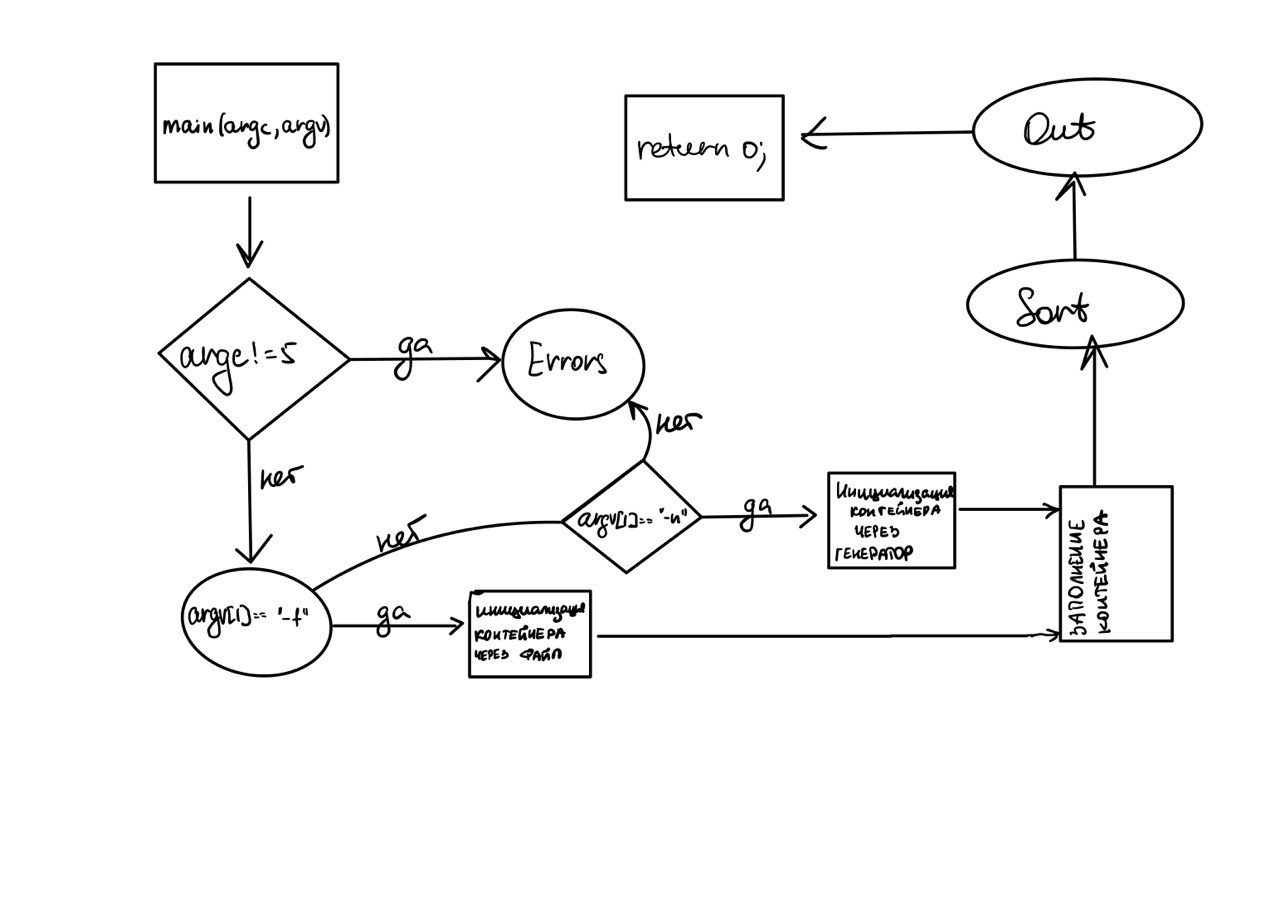
|  |  |
| --- | --- |
| **Базовые Классы** | Container [80004 байт] |
| Поля | Static: Отсутствует  Local: int length [4 байта]  Animal \*cont[80000] байт |
| Методы | Static: Отсутствует  Local: void In(FILE \*file);  void InRnd(int size);  void Out(FILE \*file);  void Sort(Container \*c, int l, int r) ;  void Merge(Container \*c, int l, int m, int r); |
| Enum | **-** |
| **Базовые Классы** | Animals [0 байт] |
| Поля | Static: Отсутствует Local: Отсутствует |
| Методы | Static: Animal \*StaticIn(FILE \*file);  Animal \*StaticInRnd();  Local: ~Movies(); |
| Enum | **-** |

**Производные классы**

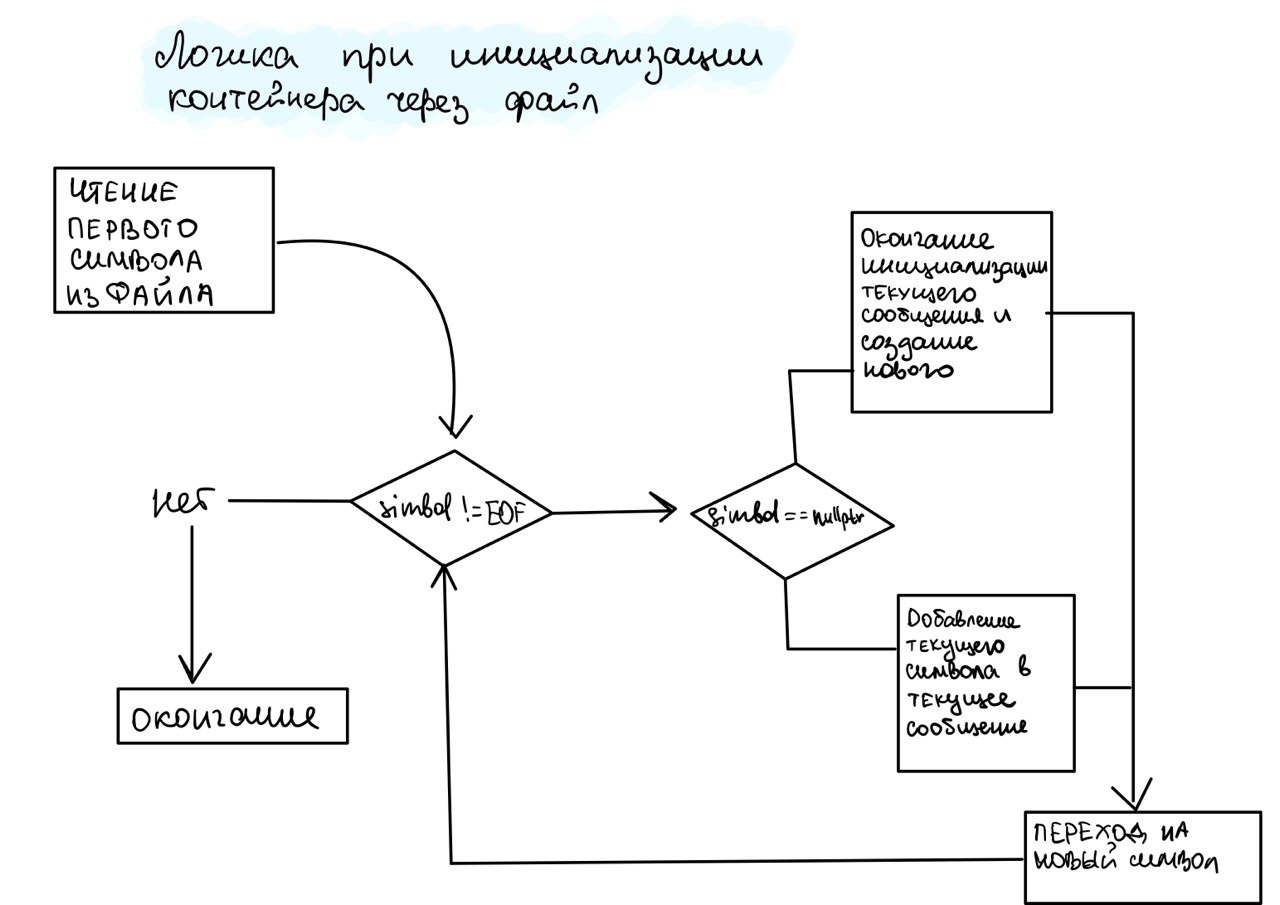
|  |  |
| --- | --- |
| **Производный класс** | Beast: Animal[17 байт] |
| Поля | Static: - Local: const char \*name [5 байт];  int weight [4 байт];  type [4 байт]; |
| Методы | Static: - Local: ~Beast();  void In(FILE \*file);  void InRnd();  void Out(FILE \*file);  double Quotient(); |
| Enum | Enum max\_length -> 4 байта |
| **Производный класс** | Bird: Animal[10 байт] |
| Поля | Static: - Local: const char \*name [5 байт];  int weight [4 байт];  bool flyOrStay [1 байт]; |
| Методы | Static: - Local: ~ Bird ();  void In(FILE \*file);  void InRnd();  void Out(FILE \*file);  double Quotinent(); |
| Enum | - |
| **Производный класс** | Fish: Animal[17 байт] |
| Поля | Static: - Local: const char \*name [5 байт];  int weight [4 байт];  type [4 байт]; |
| Методы | Static: - Local: ~ Fish();  void In(FILE \*file);  void InRnd();  void Out(FILE \*file);  double Quotient(); |
| Enum | Enum max\_length -> 4 байта |

|  |  |
| --- | --- |
| Main (4248008 байт) | |
| Container container | 4135000 |
| Int size | 4[4135000] |
| FILE input | 4[4248004] |
| FILE first\_output | 4[4248008] |

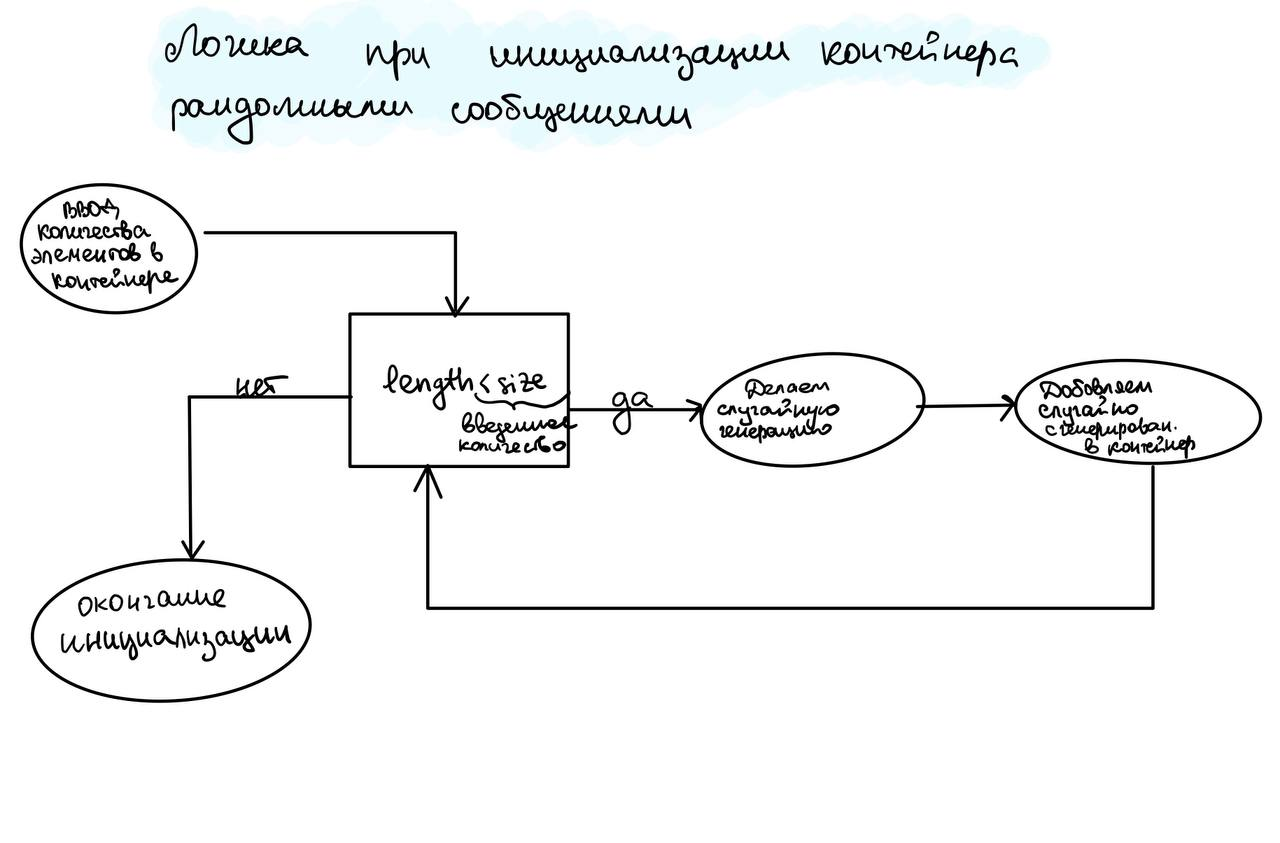
**Схемы:**



**Рис.1**



**Рис.2**



**Рис.3**

1. **Основные характеристики программы**

1) число интерфейсных модулей (заголовочных файлов) – 6

**animal.h, beast.h, bird.h, Container.h, fish.h, includes.h**

2) число модулей реализации (фалов с определением программных объектов) – 6

**animal.cpp beast.cpp bird.cpp Container.cpp fish.cpp includes.cpp**

3) общий размер исходных текстов – 2+1+2+1+1+1+1+3+1+2+1+2+2 = 20 КГ

4) полученный размер исполняемого кода – 60КГ

5) время выполнения программы для различных тестовых наборов данных:

1 тест:

elapsed time: 0.006021 seconds

2 тест:

elapsed time: 0.004013 seconds

3 тест:

elapsed time: 0.005101 seconds

4 тест:

elapsed time: 0.004120 seconds

5 тест:

elapsed time: 0.004011 seconds

1. **Расположение данных**

Входные данные лежат в:

… \HomeWork2AVC\cmake-build-debug\Input\_Tests

Выходные лежат в:

…\HomeWork2AVC\cmake-build-debug\Output\_Tests

1. **Сравнение с процедурным подходом**

* При выполнении ООП программы необходимо зачастую рисовать схему, отражающую композицию программы (из каких сущностей она состоит, есть ли между ними какие-то отношения). При процедурном подходе всего этого не требуется, что является несомненно достоинством данного подхода.
* При процедурном подходе программа выполняется немного быстрее, чем ООП программа (результаты в таблице ниже).
* Процедурный подход используется больше в небольших программах, где нет сложной и массивной архитектуры, так как он быстрее по времени. Если же требуется написать большую программу, то это лучше делать в стиле ООП, потому что читаемость кода в данном случае значительно повышается, что можно отнести к достоинству данного подхода.
* Также взаимодействие между объектами в процедурном подходе происходило через обобщенную структуру, что не совсем удобно, в этом плане ООП

|  |  |  |
| --- | --- | --- |
|  | **Первая работа** | **Вторая работа** |
| **Первый тест** | 0.0069192s | 0.006021 seconds |
| **Второй тест** | 0.0080333s | 0.004013 seconds |
| **Третий тест** | 0.0036296s | 0.005101 seconds |
| **Четвертый тест** | 0.0034514s | 0.004120 seconds |
| **Пятый тест** | 0.007765s | * 1. seconds |

1. **Ввод данных**

**0 Woodcock 24 1  
1 Hen 34 1  
1 Horse 45 0  
2 Сapelin 7 0  
0 Sparrow 25 2  
1 Donkey 14 2  
2 Vobla 56 0**

Первая переменная - вид животного (0 – птица, 2 – рыба, 3 – зверь), вторая переменная – название животного, третья переменная – за его вес в граммах, четвертая переменная – за дополнительные параметры.

Параметры для рыбы – это место обитания:

0 - River, 1 - sea, 2 - lake

Параметр для птицы – переменная, которая отвечает за то, является ли птица перелетной

0 – нет

1 – да

Параметры для зверя – тип животного:

0 - Predators, 1 - herbivores, 3 - insectivores