Домашнее задание 2
Рахматуллин Карим БПИ207

Описание задания(задание 2, функция 19):

Разработать программный продукт с использованием объектно-ориентированного подхода и статической типизацией. Программа должна содержать следующие структуры:

-Обобщенный артефакт, используемый в задании: Язык программирования

-Базовые альтернативы:

1. Процедурные (наличие, отсутствие абстрактных типов данных – булевская величина)

2. Объектно- ориентированные (наследование: одинарное, множественное, интерфейса – перечислимый тип)

3. Функциональные языки (типизация – перечислимый тип = строгая, динамическая; поддержка «ленивых» вычислений – булевский тип)

Для всех альтернатив общей переменной является популярность в процентах(действительное) и год создания(целое). Общей функцией всех альтернатив выступает вычисление частного от деления года создания на популярность (действительное число). В качестве дополнительной функции контейнера необходимо удалять из него элементы, отношение года создания к популярности которых меньше чем среднее арифметическое отношений всех элементов контейнера (остальные элементы передвигать к началу контейнера с сохранением порядка).

Также нужно: разработать тестовые входные данные и провести тестирование и отладку программы на этих данных; описать структуру используемой ВС с наложением на нее обобщенной схемы разработанной программы; зафиксировать количество заголовочных файлов, программных файлов, общий размер исходных текстов, полученный размер исполняемого кода (если он формируется), время выполнения программы для различных тестовых наборов данных.

Таблица типов

Название	Размер	Название	Размер
int	4	Class Procedural	13
bool	1	abstract: bool	1[0]
double	8	popularity: double	8[1]
FILE	216	founding_year: int	4[9]
char*	8	Class ObjectOriented	16
enum inheritance_type{}	4	inheritance: enum inheritance_type	4[0]
enum typization{}	4	popularity: double	8[4]
clock_t	8	founding_year: int	4[12]
long	8	Class Functional	17

Название	Размер	Название	Размер
Class Language	28	type: enum typization	4[0]
rnd3: Random	8[0]	lazy: bool	1[4]
rnd2: Random	8[8]	popularity: double	8[5]
popularity: double	8[16]	founding_year: int	4[13]
founding_year: int	4[24]	Class Container	80008
Class Random	8	len: int	4[0]
first: int	4[0]	cont: *language[max_len]	80000[4]
last: int	4[4]		

Описание работы функции main:

Память программы
int main(int argc, char* argv[]) {...}

Память данных	
argc: int	4[0]
argy: char*	8[4]
start_time: clock_t	8[12]
c: container	80008[20]
input_file: FILE	216[80028]
size: long	8[80244]
outputFile1: FILE	216[80252]
outputFile2: FILE	216[80468]

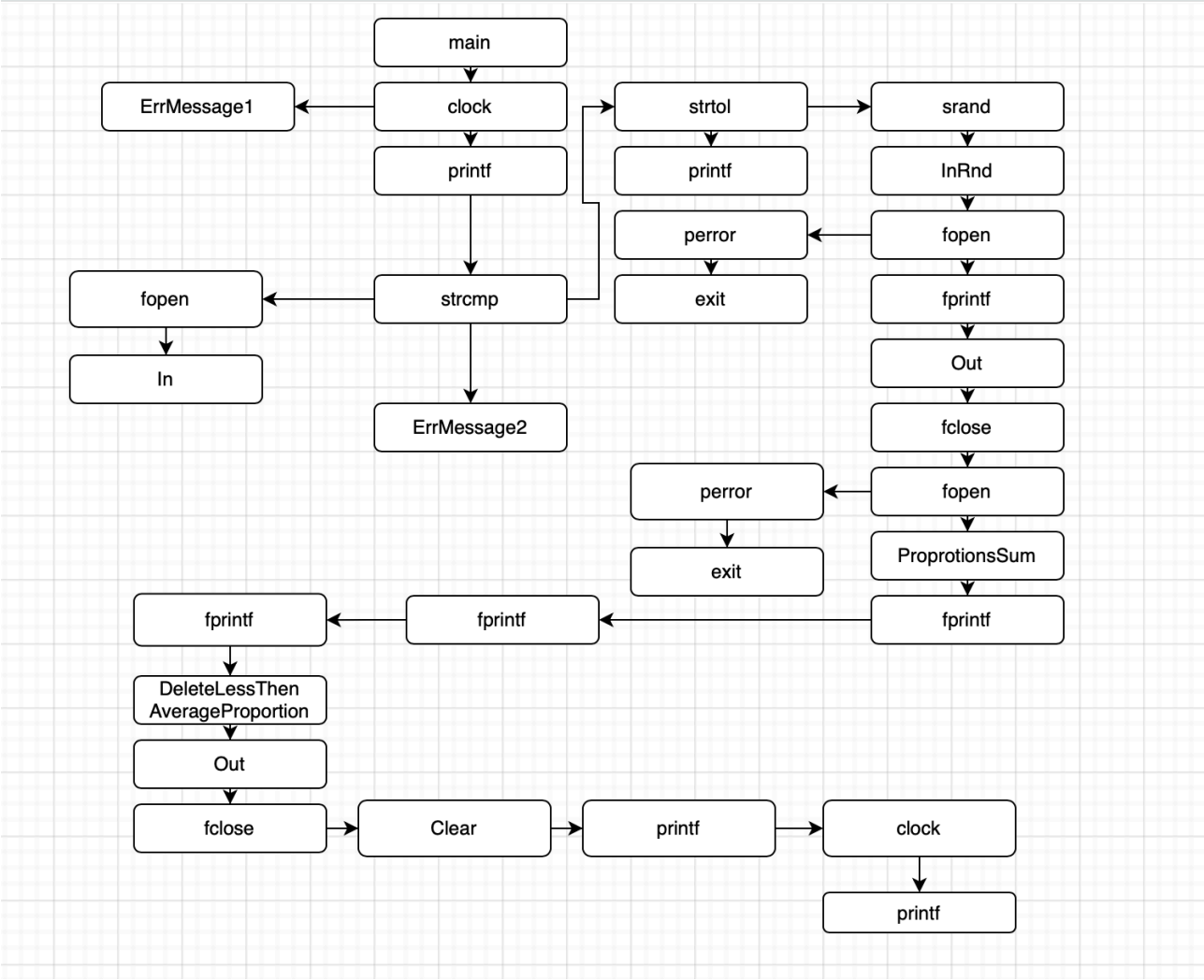
Глобальная память
0

Heap
«main» - 0
«-f» - 1
«test01.txt» - 2
«out1.txt» - 3
«out2.txt» - 4

Stack
In
Out
ProportionsSum
DeleteLessThenAverage Proportion
Out
Clear

В таблице стек представлен для частного случая (загрузка данных из файла) и содержит только созданные в процессе написания программы функции).

Блок схема стека, в результате работы функции main (глубина - 1 шаг):



Описание работы функции StaticInRnd

Stack
StaticInRnd
Get
InRnd

Память данных	
k: int	4[0]
l: Language*	8[4]

Глобальная память
0

Heap
Language[0]

Память программы

Language* Language::StaticInRnd() {
  int k = rnd3.Get();
  Language* l = nullptr;
  switch (k) {
  case 1:
  l = new Procedural;
  break;
  case 2:
  l = new ObjectOriented;
  break;
  case 3:
  l = new Functional;
  break;
  }
  l->InRnd();
  return l;
}

Описание работы функции DeleteLessThenAverageProportion

Stack
DeleteLessThenAverageProportion
ProportionsSum
Proportion
DeleteElementByIndex

Память данных	
this: Container*	8[0]
i: int	4[8]
average_value: double	8[12]
elements_cnt: int	4[20]

Глобальная память
0

Heap
Container[0]

Память программы

void Container::DeleteLessThenAverageProportion(){
  int i = 0;
  double average_value = ProportionsSum() / len;
  int elements_cnt = len;
  while((elements_cnt > 0) && (i < elements_cnt)) {
  if(storage[i]->Proportion() < average_value) {
  DeleteElementByIndex(i);
  --i;
  --elements_cnt;
  }
  ++i;
  }
}

Описание работы функции ProportionsSum

Stack
ProportionsSum
Proportion

Память данных	
this: Container*	8[0]
sum: double	8[12]
i: int	4[20]

Глобальная память
0

Heap
Container[0]

Память программы

double Container::ProportionsSum(){
  double sum = 0.0;
  for(int i = 0; i < len; i++) {
  sum += storage[i]->Proportion();
  }
  return sum;
}

Основные характеристики программы

- Число заголовочных файлов: 7
- Число модулей реализации: 7
- Общий размер исходных текстов - 679 строк кода
- Размер исполняемого кода -107 кб
- Время выполнения программы:
  - Тест 1 - 0.000460 с
  - Тест 2 - 0.001523 с
  - Тест 3 - 0.000436 с
  - Тест 4 - 0.000844 с
  - Тест 5 - 0.000627 с

	Процедурный подход и статическая типизация	Объектно-ориентированный подход и статическая типизация
Число заголовочных файлов	8	7
Число модулей реализации	7	7
Общий размер исходных текстов(строк кода)	679	654
Размер исполняемого кода(кб)	53	57
Тест 1(с)	0.000460	0.000452
Тест 2(с)	0.001523	0.000565
Тест 3(с)	0.000436	0.000701
Тест 4(с)	0.000844	0.000858
Тест 5(с)	0.000627	0.000643

Выводы

Объектно-ориентированный подход показал примерно то же время на тестах, что и процедурный.
Недостаток - увеличился размер исполняемого кода.
Преимущества: уменьшилось количество кода, код становится проще и понятнее из-за возможностей, которые открываются благодаря полиморфизму и наследованию.
ОО подход оказывается предпочтительнее, т.к. проще конструировать логику программы.