HW4 ACS V218

ZhengWu Pan

November 2021

1 Полученное задание

Реализовать обобщённый массив языки программирование:

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)
4. Реализовать для них функцию вычисление частное от деления года создания на количество символов в названии.
5. Упорядочить элементы массива по убыванию используя сортировку методом деления пополам (Binary Insertion). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.

2 Формат входа

Программа получает на вход либо строку вида:
-f <inputfile> <outputfile> <outputfile2> (получение данных о фигур произойдёт считыванием <inputfile>)
либо: -n <count> <outputfile> <outputfile2> (случайная генерация <count> фигур).

2.1 Формат входа файла

Сначала количество языки в данной файле. С второго строк идет ыид языки от 1 до 3 (1 - Процедурный, 2 - Объектно-ориентированный, 3 - Функциональный), TIOBI, год создания, и свое индивидуальное поле:

1. Процедурные
   1 - наличие, 0 - отсутствие абстрактных типов данных
2. Объектно-ориентированные
   наследование: 1 - одинарное, 2 - множественное, 3 - интерфейса
3. Функциональные
   типизация – перечислимый тип: 1 - строгая, 2 - динамическая; поддержка «ленивых» вычислений – 1, 0 - нет.

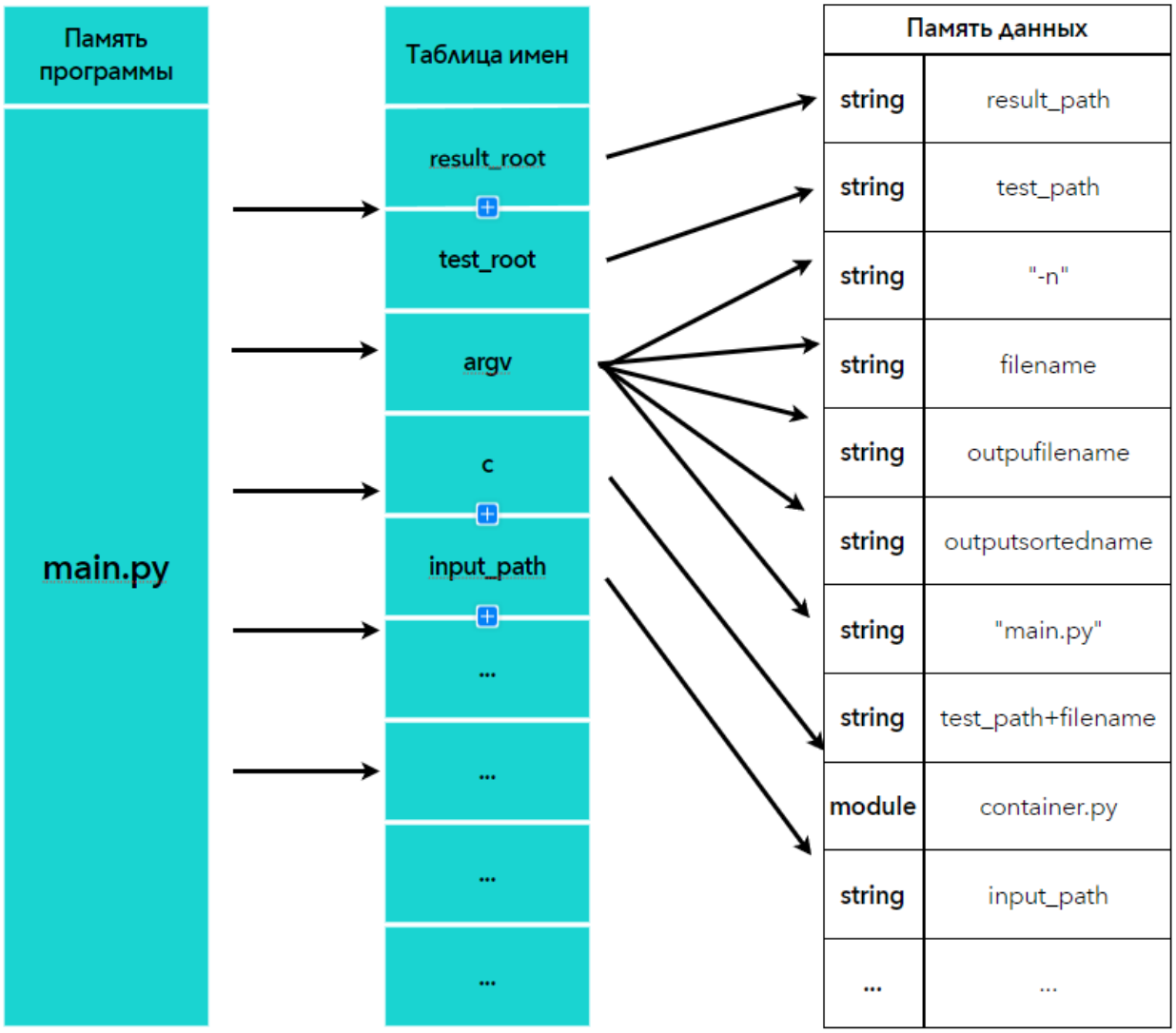

Рис. 1: An image of memory map

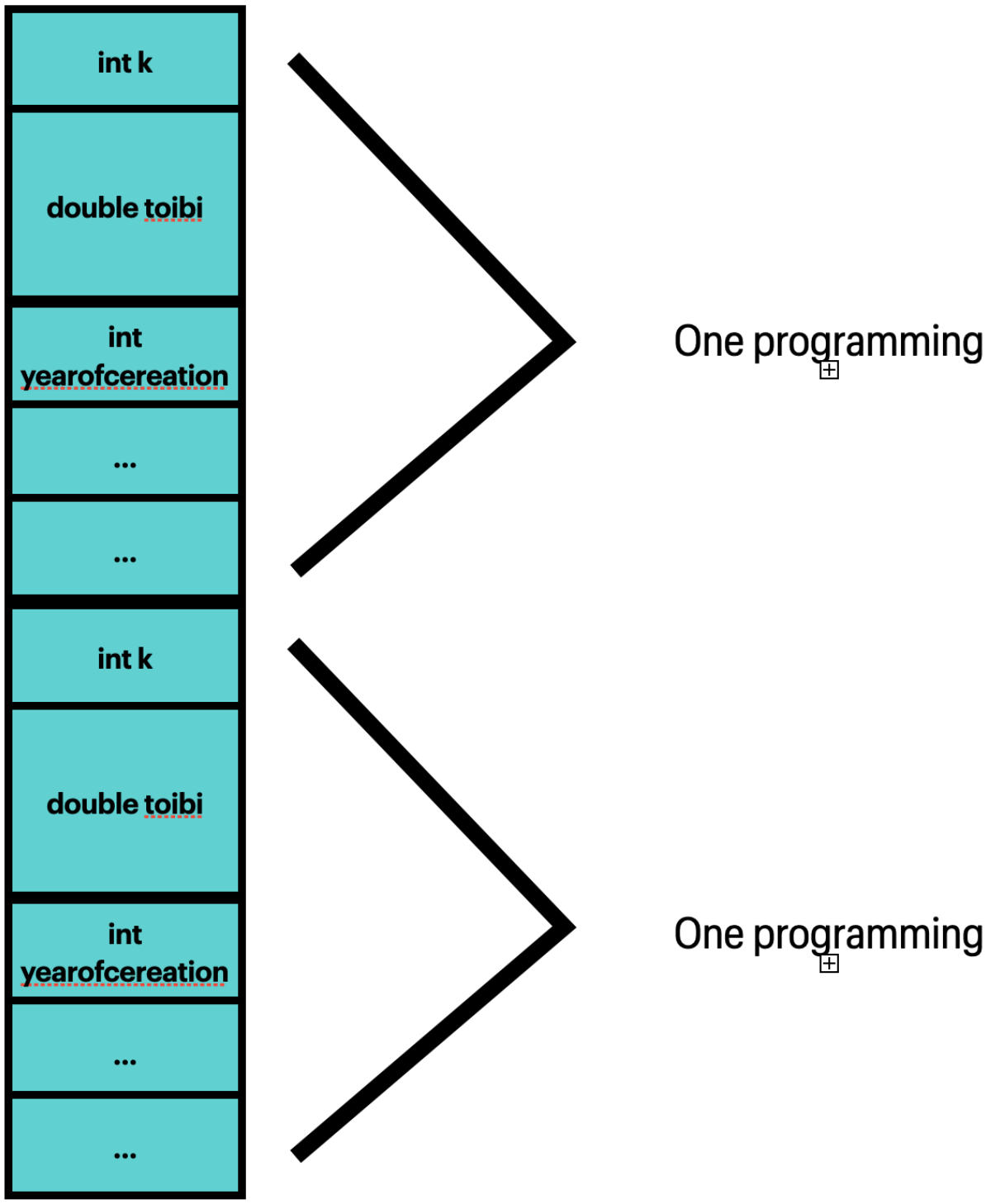

Рис. 2: unsigned char[maxSize]

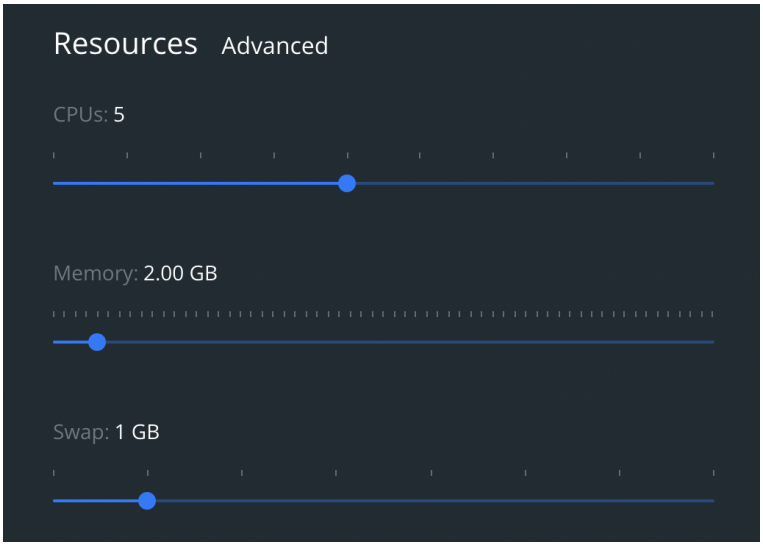

Рис. 3: container info

3 При запуске:

Данный программа собирается через Докер-компоуз:

1. Сначала зайти терминал, зайдите в папке HW4_ACS_ V218 с docker-compose.yml
2. Запускайте команда: docker-compose up
3. Зайти в shell docker контейнер (Attach to shell)
4. Зайти в папке с проектами
5. Собираете проект через make, запускайте его через ./task <параметры по описание в пункт 2 и 2.1>

4 Время

time/programm	HW1	HW2	HW3	HW4
test01.txt	*	*	0.023932	0.010471
test02.txt	*	*	0.029932	0.01186
test03.txt	*	*	0.429932	0.323745
test04.txt	*	*	0.229932	0.218221
test05.txt	*	*	0.009932	0.012177
test06.txt	*	*	0.008932	0.014299

Время тестировано в новом среде (контейнере, см Рис. 3)
Можем заметить, скорости выполнение повышается в 2-3 раза благодаря язык nasm. Любое преобразование языка высокого уровня в машинный код приводит к издержкам. Ассемблер работает быстрее, потому что программист не пишет ничего лишнего, а чтобы обеспечить универсальность применения языковых конструкций, все машинные коды, созданные трансляторами, избыточны. Также помогли void pointer: самый оригинальный способ реализации ООП.

5 Разме файла

time/programm	Size
data.h	2.5K
extdata.h	3.1K
function_sort.asm	7.6K
input.c	2.2K
inrnd.c	2.8K
main.c	3.7K
makefile	256B
output.c	5.4K
task	exexcutabel: 33K