Второе Домашнее задание "Архитектура вычислительных систем"

Задача: Языки программирования, Вариант 148 (8, 11)

Автор: Туракулов Исломбек Улугбекович

Условие задачи

Общие для всех альтернатив переменные

- Популярность в процентах (TIOBI) — действительное
- Год создания — целое

Общие для всех альтернатив функции

Частное от деления года создания на количество символов в названии (действительное число)

Базовые альтернативы

Процедурные

- наличие, отсутствие абстрактных типов данных – булевская величина

Объектно-ориентированные

- наследование: одинарное, множественное, интерфейса – перечислимый тип

Функциональные

- типизация – перечислимый тип = строгая, динамическая
- поддержка «ленивых» вычислений – булевский тип

Функционал

После размещения данных в контейнер необходимо осуществить их обработку в соответствии с вариантом задания. Обработанные данные после этого заносятся в отдельный файл результатов.

Упорядочить элементы контейнера по возрастанию используя сортировку Сортировка с помощью прямого выбора (Straight Selection). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.

Сборка полученной программы.

```
cmake -B <build_dir> -DCMAKE_BUILD_TYPE=Release
cmake --build <build_dir> --target main
```

Исполняемый файл программы будет расположен в папке `bin`.

Тестирование

Исходные данные для тестирования содержатся в каталоге `tests`.

Файл с результатами прогонов тестов `tests/report.txt`.

Примеры использования программы без тестирующего скрипта. Предполагается, что сейчас мы в корне проекта. Использовался компилятор GCC в ОС Linux

```
# Parse input file with data
./bin/main tests/mixed.txt tests/mixed.out
# Parse input file with random generator setup for the test
./bin/main -n 100 tests/data/random.out.txt tests/random_sorted.out.txt
```

Время работы программы на разных размерах входных данных:

Количество языков программирования	Время работы, seconds	Потребляемая память, KB
7	< 0.001	~2600
100	< 0.01	~2800
1000	0.01	~3500
5000	0.14	~4150
10000	0.90	~4625

Разница между процедурной и объектно-ориентированной реализацией

Если сравнить тест процедурного подхода с объектно-ориентированным лучше сравнить по времени выполнения и потребления памяти. У процедурной реализации код потребляет в 1,1 раза больше памяти и в 1,2 раза больше времени выполнения. Следовательно, объектно-ориентированный подход показывает хороший результат по времени и памяти чем процедурный.

Метрики, определяющие характеристики программы:

Метрика	Значение
Число интерфейсных модулей	6
Число модулей с реализацией	5 + 1 (дополнительные методы для рандомизации)
Общий размер исходных текстов программы	17.451 KB
Размер исполняемого файла релизной сборки (GCC, Linux)*	34.902 KB

***** Версии подробнее:

```
 $ g++ --version
g++ (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

$ lsb_release -a
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.2 LTS
Release:        20.04
Codename:       focal

$ uname -a
Linux DESKTOP-250N7G3 5.10.16.3-microsoft-standard-WSL2 #1 SMP Fri Apr 2 22:23:49 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
```

Описание структуры вычислительной системы:

Размеры фундаментальных типов на машине amd64:

Тип	Размер в байтах
int	4
char	1
double	8
bool	1

Базовые Классы

Таблица классов Базовые Классы	Container [80004 байт]
Поля	Static: Отсутствует Local: int length [4 байта] Language *cont[8 * 10000 байт]

Методы	Static: Отсутствует
	
	Local:
	void In(FILE *file);
	void InRnd(int size);
	void Out(FILE *file);
	void StraightSelectionSort();

Enum	Отсутствует
Таблица классов Базовые Классы	Languages [0 байт]
Поля	Static: Отсутствует Local: Отсутствует

Методы	Static: Languages *StaticIn(FILE *file);
	Languages *StaticInRnd();
	
	Local:
	~Languages();
	void In(FILE *file);
	void InRnd();
	void Out(FILE *file);
	double Quotinent();

Enum	Отсутствует

Производные классы

Таблица классов Производные классы	Functional : Languages [26 байт + 0 байт]

Таблица классов Производные классы	Functional : Languages [26 байт + 0 байт]
Поля	Static: Отсутствует Local: const char *name_[5 байт]; int age_ [4 байт]; double popularity_ [8 байт]; bool lazy_calculation_ [1 байт]; int type_ [4 байт];

Методы	Static:
	
	Local:
	~Functional();
	void In(FILE *file);
	void InRnd();
	void Out(FILE *file);
	double Quotinent();

Enum	Enum MAX_LENGTH -> 4 байта
Производный класс	ObjectOriented : Languages [26 байт + 0 байт]
Поля	Static: Отсутствует Local: const char *name_[5 байт]; int age_ [4 байт]; double popularity_ [8 байт]; bool has_abstract_variables_ [1 байт]; int type_ [4 байт];

Методы	Static:
	
	Local:
	~ObjectOriented();
	void In(FILE *file);
	void InRnd();
	void Out(FILE *file);
	double Quotinent();

Enum	Enum MAX_LENGTH -> 4 байта
Производный класс	Procedural : Languages [18 байт + 0 байт]
Поля	Static: Отсутствует Local: const char *name_[5 байт]; int age_ [4 байт]; double popularity_ [8 байт]; bool has_abstract_variables_ [1 байт];

Таблица классов Производные классы	Functional : Languages [26 байт + 0 байт]

Методы	Static:
	
	Local:
	~ObjectOriented();
	void In(FILE *file);
	void InRnd();
	void Out(FILE *file);
	double Quotinent();

Enum	Отсутствует

Глобальная память

Глобальная память (ГП)	Базового класса (БК) Container	Производного класса (ПК) Language
Поля	Все статические поля	Все статические поля

Локальные методы	void In(Container *self, FILE *file);
	void InRnd(Container *self, int size);
	void Out(Container *self, FILE *file);
	void StraightSelectionSort(Container *self);
		~Languages();
		void In(FILE *file);
		void InRnd();
		void Out(FILE *file);

Таблица Виртуальных методов

Глобальная память (ГП)	Базового класса (БК)	Производных классов (ПК) Functional, ObjectOriented, Procedural

Таблица Виртуальных методов
(ТВМ)
	~Languages(Languages *self);
	void In(FILE *file);
	void InRnd(Languages *self);
	void Out(Languages *self, FILE *file);
		~InheritedClassed(Languages *self);
		void In(Languages *self, FILE *file);
		void InRnd(Languages *self);
		void Out(Languages *self, FILE *file);

Программная память

Программная память (ПП)	Container	Languages	Производные классы (ПК) Functional, ObjectOriented,Procedural
Статические	-	char *name_ [4 байт] int age_ [4 байт] int, file : Languages *StaticIn(FILE *file) -> 12 байт self : Languages *StaticInRnd() -> 16 байт	-
Локальные	file : void In(FILE *file) -> 8 байт size : void InRnd(int size) -> 4 байта file : void Out(FILE *file) -> 8 байт self : void StraightSelectionSort() -> 8 байт	-	-

Программная память (ПП)	Container	Languages	Производные классы (ПК) Functional, ObjectOriented,Procedural
Виртуальные	-	self : ~Languages() -> 8 байт file : void In(FILE *file) -> 8 байт self : void InRnd() -> 8 байт file : void Out(FILE *file) -> 8 байт double, self : double Quotient() -> 16 байт	self : ~Languages() -> 8 байт file : void In(FILE *file) -> 24 или 25 байт self : void InRnd() -> 24 или 25 байт file : void Out(FILE *file) -> 24 или 25 байт + Quotient() размер double, name_, age_ ,self : double Quotient() -> 24 байт

Куча

Куча
Container *cont -> 80000 байт
Объекты ПК в зависимости от количества (от 0 до 10000)
ПК_1 : Functional or ObjectOriented or Procedural
ПК_2 : Functional or ObjectOriented or Procedural
и т.д до i диапазона
ПК_(i - 1) : Functional or ObjectOriented or Procedural
ПК_i : Functional or ObjectOriented or Procedural