Отчет о программе, разработанной на Netwide Assembler
Вариант 2. Функция 10
Демиденко Ксения БПИ208

Спецификация

Спецификация ВС

OS: Ubuntu (64bit) 20.04

Характеристики проекта

Количество программных объектов (.asm): 7
Количество подпрограмм (функций): 28

File	Function
main.asm	global main
input.asm	global InRectangle
	global InTriangle
	global InCircle
	global InContainer
	global InShape
output.asm	global OutRectangle
	global OutTriangle
	global OutCircle
	global OutContainer
	global OutShape
inrnd.asm	global Random
	global InRndRectangle
	global InRndTriangle
	global InRndCircle
	global InRndContainer
	global InRndShape

File	Function
perimeter.asm	global PerimeterRectangle
	global PerimeterTriangle
	global PerimeterCircle
	global PerimeterSumContainer
	global PerimeterShape
sort.asm	global InsertionSort
outrnd.asm	global OutRndRectangle
	global OutRndTriangle
	global OutRndCircle
	global OutRndContainer
	global OutRndShape

Количество макроопределений (.mac): 1

File	Size (bytes)	On the disk (kilobytes)
macros.mac	4849	8

Размер исполняемых файлов (единиц компиляции):

File	Size (bytes)	On the disk (kilobytes)
main.asm	7588	8
input.asm	9900	12
output.asm	9404	12
inrnd.asm	11902	12
perimeter.asm	7842	8
sort.asm	3263	4
outrnd.asm	8824	12

Суммарный размер исполняемых файлов: 58723 Б (68 КБ на диске)
В предыдущей программе, разработанной на языке Python: 10233 Б (20 КБ на диске)

Полученный размер исполняемого кода: 34040 Б (37 КБ на диске)

Временные характеристики работы программы

Время работы в сравнении с аналогичными программами, разработанными на языках C++ и Python на тестах с тем же количеством элементов

Тест	Количество обрабатываемых элементов (фигур)	Время работы данной программы (с.)	Время работы аналогичной программы на языке C++ (с.)	Время работы аналогичной программы на языке Python (с.)
inp1.txt	3	0.00000224	0.000356	0.00125
inp2.txt	10	0.00000425	0.000401	0.00092
inp3.txt	1000	0.006238452	0.014865	0.25574
inp4.txt	5000	0.160292604	0.207818	4.988
inp5.txt	10000	0.510460129	0.775573	18.3425

Мы видим, что программа, написанная с использованием NASM выигрывает по времени по сравнению с обеими предыдущими программами (особенно программой на Python). Однако значительным минусом данного языка является сложность разработки на нем. Также размеры его исполняемых файлов гораздо больше. Разработка же на C++ и Python имеет преимущество в своей легкости, но проигрывает по времени работы программ.

Ввод данных

Для начала нужно создать исполняемый файл *task* командой *make.*

Для работы с данными из существующего файла

./task -f <path_to_input_file> <path_to_output_file> <path_to_time_file>

Для рандомной генерации данных

./task -n <num_of_objects> <path_to_output_file> <path_to_time_file>

В случае рандомной генерации в папке со всеми исполняемыми файлами проекта будет сгенерирован новый файл *RndInput.txt,* куда запишется сгенерированный набор тестовых данных.

В *time_file* запишется отдельно время работы запущенной программы.

Формат данных во входном файле

На первой строке файла:
<n - количество объектов>

На следующих n строках записаны объекты в виде:

-для прямоугольника (тип фигуры 1):
<1> <x левой верхней> <y левой верхней> <x правой нижней> <y правой нижней> <COLOR (RED, ORANGE, etc)>

-для треугольника (тип фигуры 2):
<2> <x 1-ой> <y 1-ой> <x 2-ой> <y 2-ой> <x 3-ей> <y 3-ей> <COLOR (RED, ORANGE, etc)>

-для круга (тип фигуры 3):

<3> <x центра> <y центра> <r радиус> <COLOR (RED, ORANGE, etc)>