Московский Авиационный Институт
(Национальный Исследовательский Университет)

Кафедра: 806 "Вычислительная математика и программирование"
Факультет: "Информационные технологии и прикладная математика"
Дисциплина: "Объектно-ориентированное программирование"

Группа:
Студент: Пашкевич Андрей Романович
Преподаватель: Поповкин Александр Викторович

Москва, 2017

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №6

Вариант №17

ЦЕЛЬ РАБОТЫ

Целью лабораторной работы является:

- Закрепление навыков по работе с памятью в C++.
- Создание аллокаторов памяти для динамических структур данных.

ЗАДАНИЕ

Используя структуры данных, разработанные для предыдущей лабораторной работы (ЛР№5) спроектировать и разработать аллокатор памяти для динамической структуры данных.

Цель построения аллокатора – минимизация вызова операции malloc. Аллокатор должен выделять большие блоки памяти для хранения фигур и при создании новых фигур-объектов выделять место под объекты в этой памяти.

Алокатор должен хранить списки использованных/свободных блоков. Для хранения списка свободных блоков нужно применять динамическую структуру данных (контейнер 2-го уровня, согласно варианта задания).

Для вызова аллокатора должны быть переопределены оператор new и delete у классов-фигур.

Нельзя использовать:

- Стандартные контейнеры std.

Программа должна позволять:

- Вводить произвольное количество фигур и добавлять их в контейнер.
- Распечатывать содержимое контейнера.
- Удалять фигуры из контейнера.

TAllocationBlock(int size, int count)	Конструктор класса
Void *Allocate();	Выделение памяти
Void Deallocate(void *ptr);	Освобождение памяти
Bool Empty();	Проверка, пуст ли аллокатор
Int Size();	Получение количества выделенных блоков
Virtual ~TAllocationBlock();	Деструктор класса

Листинг программы:

Выводы:

При выполнении данной лабораторной работы необходимо было описать аллокатор памяти для структуры данных бинарное дерево на языке программирования C++. Аллокаторы предоставляют возможность выделить память под n-ое количество объектов класса. В отличии от вызова оператора new при создании объекта аллокатор записывает его уже в выделенном блоке памяти, что ускоряет работу программы, если не были заняты все свободные блоки.