Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»

Алгоритмы распределения памяти

Студент: Сапожков Андрей Максимович ИУ7-73Б

Научный руководитель: Строганов Юрий Владимирович

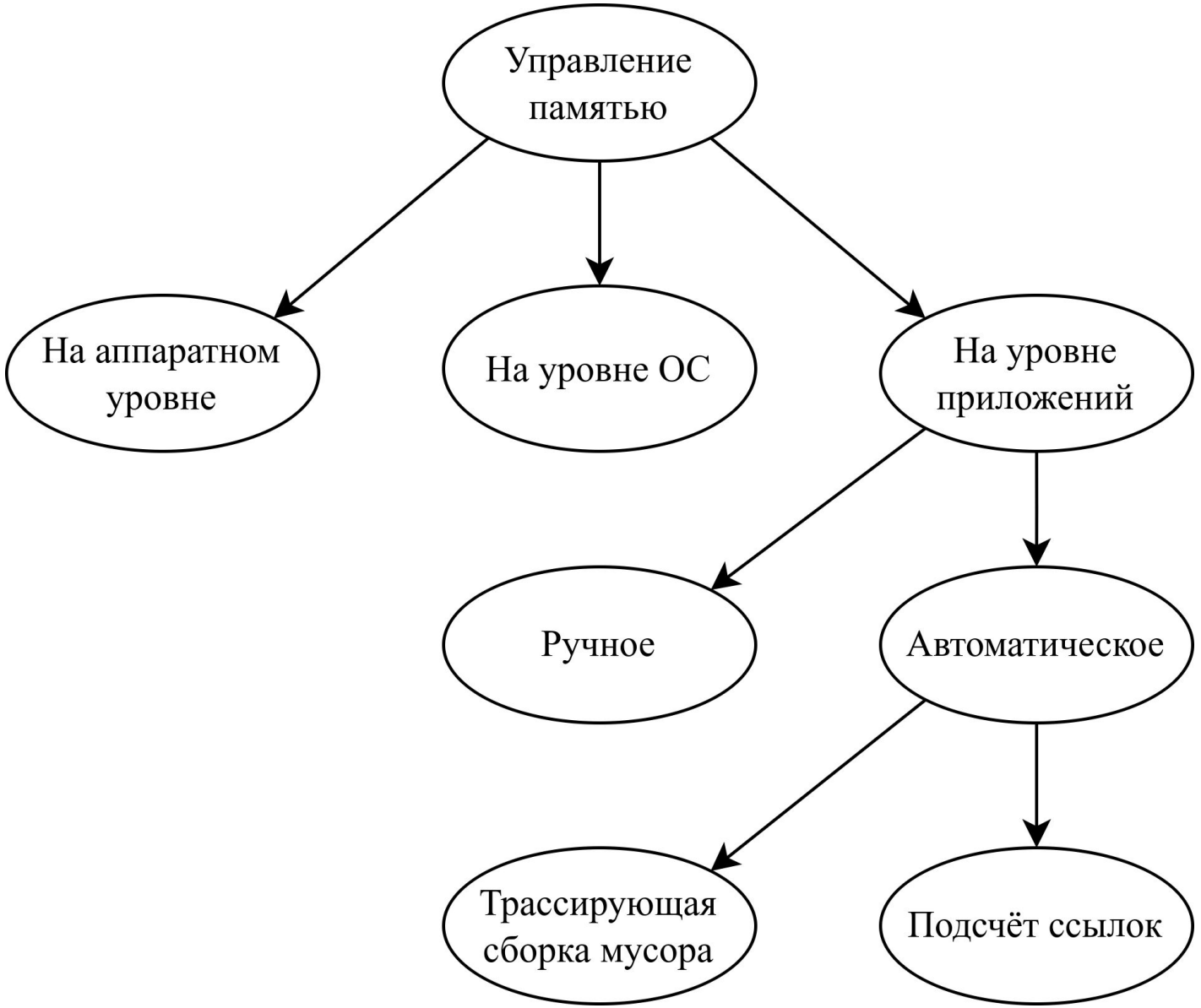
Цель и задачи

Цель – изучение алгоритмов распределения памяти в языках программирования с автоматической сборкой мусора.

Задачи:

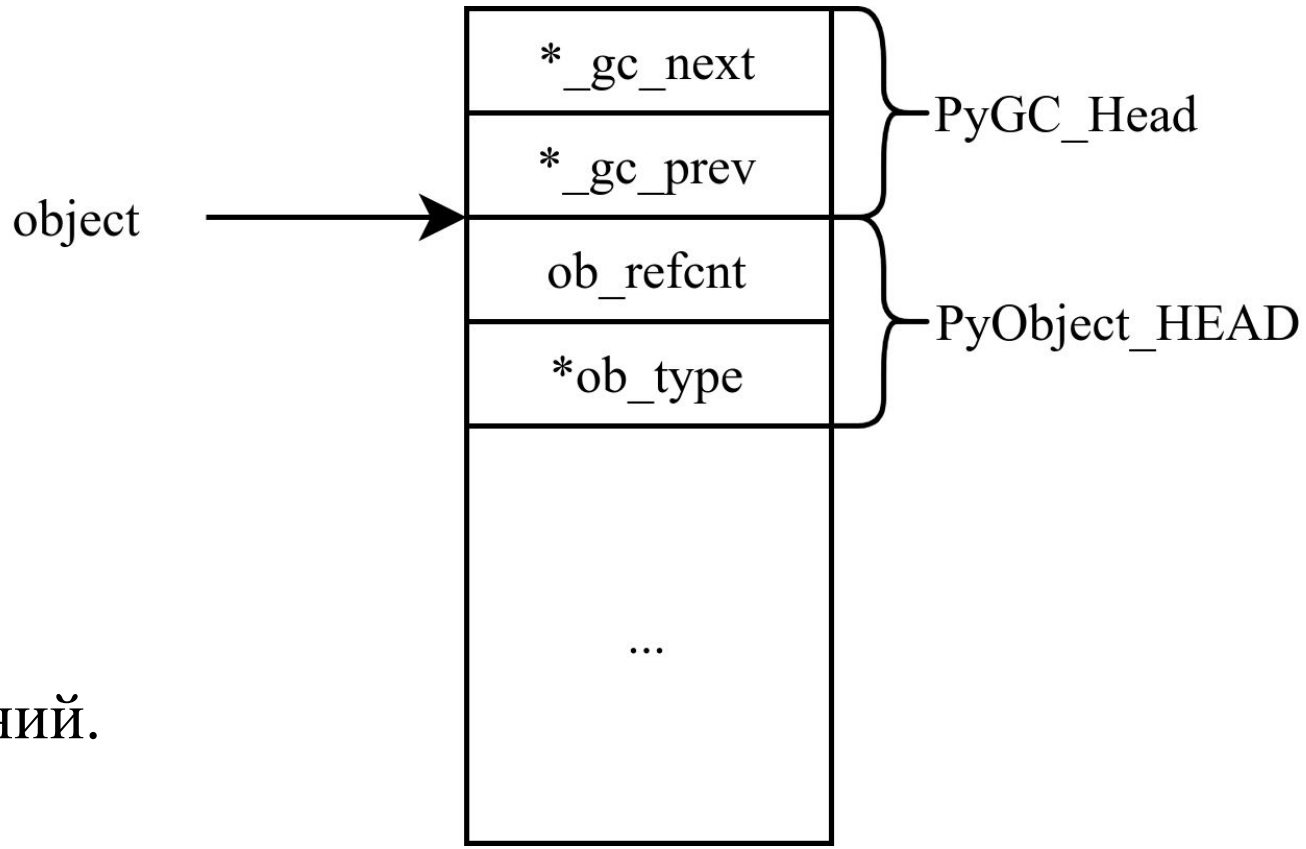
1. Проанализировать предметную область работы с памятью в языках программирования с автоматической сборкой мусора.
2. Рассмотреть существующие принципы организации работы с памятью в языках программирования с автоматической сборкой мусора на примере Python, Java, JavaScript, C# и Golang.
3. Описать алгоритмы сборки мусора в рассматриваемых языках.
4. Сформулировать критерии сравнения и оценки описанных алгоритмов.
5. Сравнить существующие решения по сформулированным критериям.

Управление памятью



Управление памятью в Python

- Использование GIL.
- Размещение объектов.
  в приватной куче.
- Домены аллокаторов:
  - Raw domain;
  - "Mem" domain;
  - Object domain.
- Подсчёт ссылок.
- Выделенный сборщик мусора.
  для циклических ссылок.
- Использование алгоритма поколений.



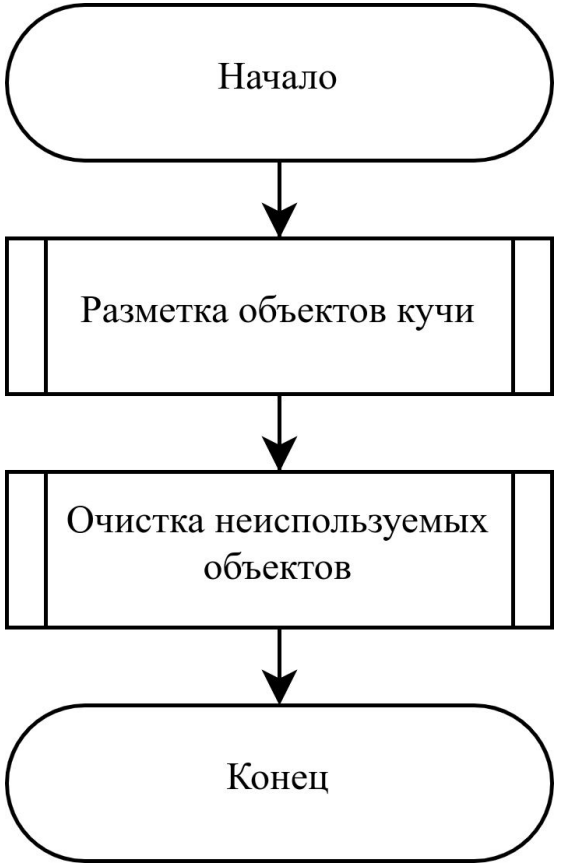
Структура объекта Python

Управление памятью в Java

- Куча управляется менеджером хранилища JVM.
- Каждый поток использует свой буфер для выделения объектов – TLAB.
- Объекты разделяются на поколения на уровне JVM.
- Предоставляется возможность выбора сборщика мусора:
  - Serial Collector;
  - Parallel Collector;
  - Garbage-First Collector;
  - Z Garbage collector.

Управление памятью в JavaScript

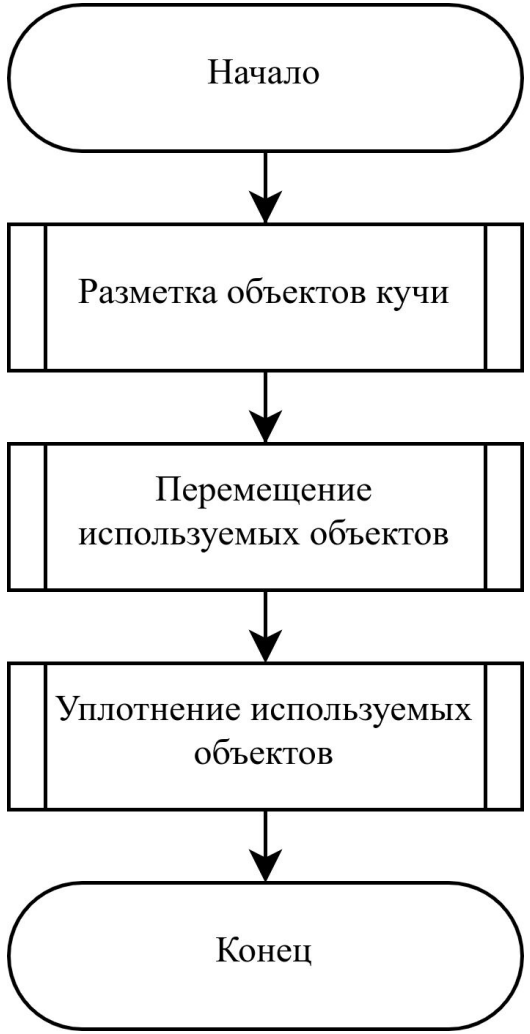
- Асинхронная однопоточная модель выполнения в рамках цикла событий.
- Реализация сборки мусора на уровне среды выполнения, как правило по алгоритму mark-sweep (разметка и очистка).
- Использование "слабых" коллекций:
  - WeakMap;
  - WeakSet.
- Разрешение циклических ссылок с помощью эфемеронов.
- Использование слабых ссылок на объекты.
- Реализация уведомлений об освобождении объектов.



Основные шаги алгоритма mark-sweep

Управление памятью в C#

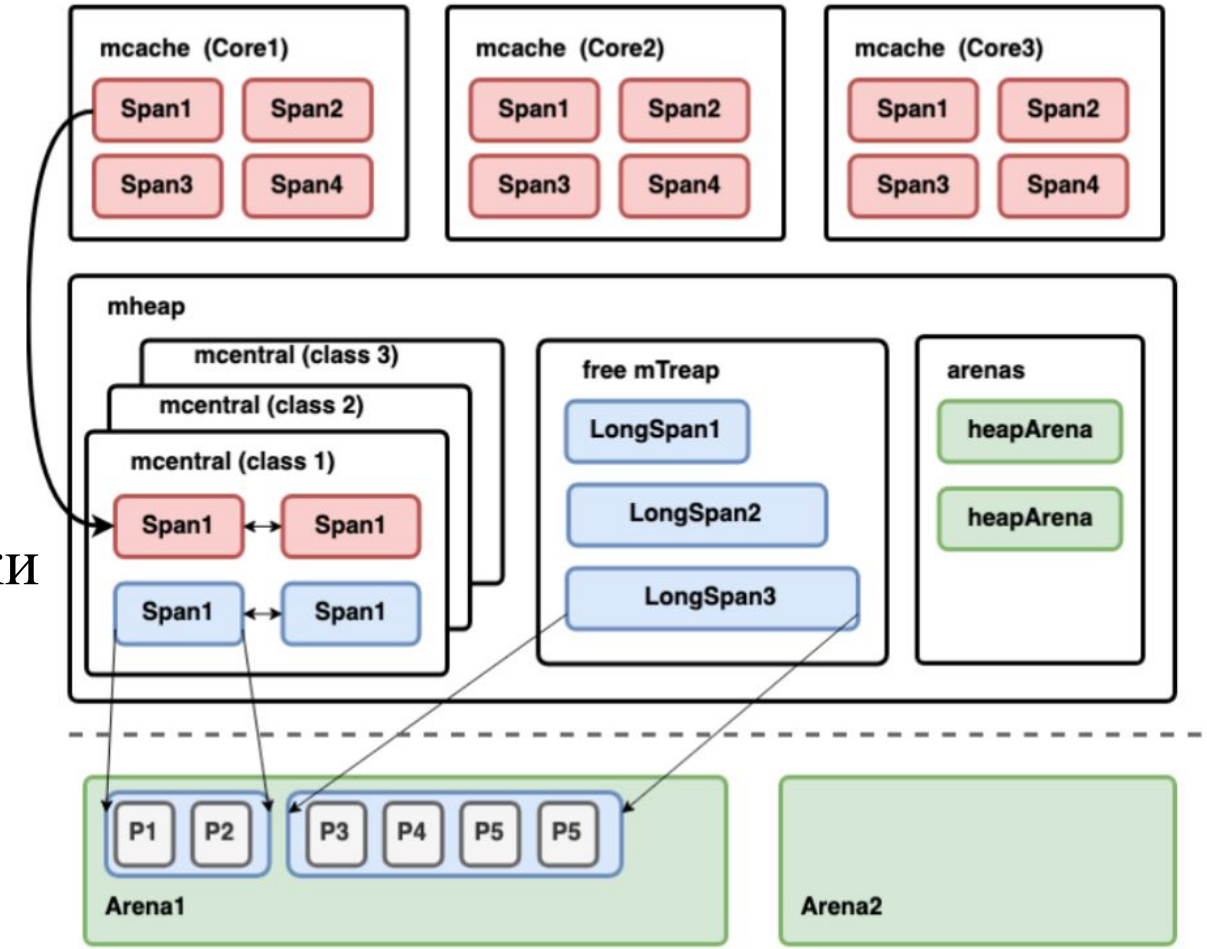
- Управление памятью на уровне среды выполнения – платформы .NET.
- Использование файла подкачки при нехватке памяти.
- Хранение объектов размером более 85000 байт в отдельном разделе – LOH.
- Сборка мусора по алгоритму mark-compact.
- Использование алгоритма поколений.
- Предоставление возможности выбора режима сборки мусора:
  - сборка мусора рабочей станции;
  - серверная сборка мусора.



Основные шаги алгоритма mark-compact

Управление памятью в Golang

- Среда выполнения встроена в исполняемый файл.
- Использование классов размеров для выделения памяти.
- Сборка мусора по алгоритму concurrent mark-sweep.
- Использование барьеров записи для конкурентной сборки мусора.
- Предоставление возможности настройки периодичности сборки мусора и ограничения доступной памяти.
- Предоставление арен памяти для реализации пользовательских аллокаторов.



Структура кучи в программе на Golang

Классификация существующих решений (1)

Язык программирования	Сборщик мусора	Использование поколений	Отсутствие хранения доп. данных в объектах	Конкурентная сборка мусора
Python	По умолчанию	+	-	+
Java	Serial	+	+	+
	Parallel	+	+	+
	Garbage-First	+	+	+
	ZGC	+	+	+
JavaScript	По умолчанию	-	+	-
C#	По умолчанию	+	+	+
Golang	По умолчанию	-	+	+

Классификация существующих решений (2)

Язык программирования	Сборщик мусора	Параллельная сборка мусора	Остановка программы не на весь цикл сборки	Количество остановок программы за один цикл сборки
Python	По умолчанию	-	+	1
Java	Serial	-	-	1
	Parallel	+	-	1
	Garbage-First	+	+	2
	ZGC	+	+	1
JavaScript	По умолчанию	-	-	1
C#	По умолчанию	+	-	1
Golang	По умолчанию	+	+	2

Результаты сравнения

В результате сравнения были выделены менеджеры памяти следующих языков программирования:

- Python за счёт применения алгоритма поколений и подсчёта ссылок;
- Java за счёт предоставления пользователю возможности выбора сборщика мусора для выполнения каждой программы, а также оптимизации времени пауз на сборку мусора.

Заключение

В рамках научно-исследовательской работы была проведено изучение алгоритмов распределения памяти в языках программирования с автоматической сборкой мусора. Для достижения этой цели были решены следующие задачи:

1. Проанализирована предметная область работы с памятью в языках программирования с автоматической сборкой мусора.
2. Рассмотрены существующие принципы организации работы с памятью в языках программирования с автоматической сборкой мусора на примере Python, Java, JavaScript, C# и Golang.
3. Описаны алгоритмы сборки мусора в рассмотренных языках.
4. Сформулированы критерии сравнения и оценки описанных алгоритмов.
5. Проведено сравнение существующих решений по выделенным критериям.