Управление памятью

Подсчет ссылок. Autorelease Pool. ARC. Атрибуты свойств.

Управление памятью в C

Каждый вызов malloc() сопровождается вызовом free(). Например:

```
char *my_str = malloc(sizeof(char) * 7);
// Манипуляции со строкой
free(my_str);
```

Управление памятью в Cocoa

- Используется модель “подсчет ссылок”.
- Реализуется классом NSObject и Objective-C runtime.
- Используются методы retain, release и autorelease.

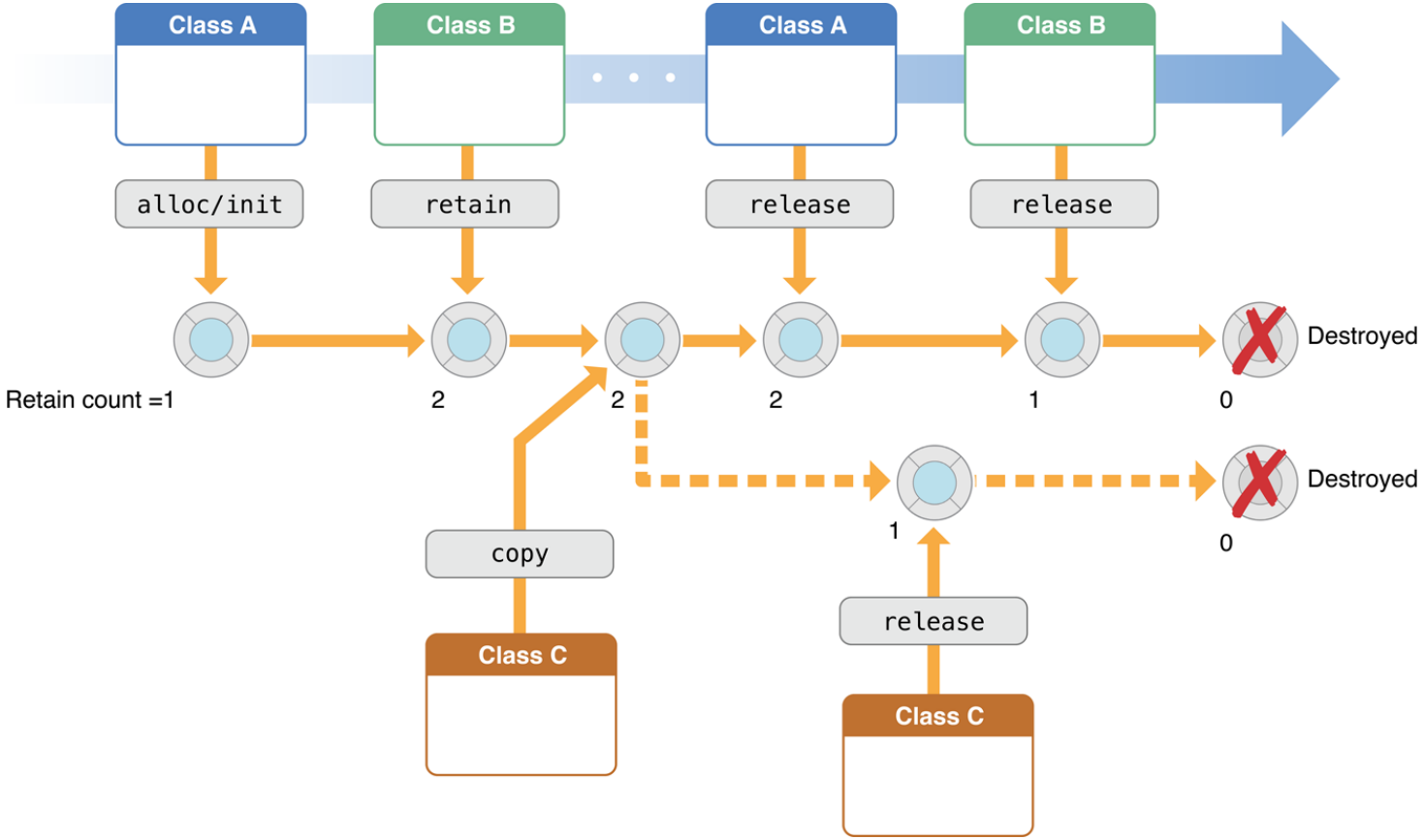
Объект хранит количество ссылок на себя.

- При создании объекта (методы alloc, new, copy, mutableCopy) количество ссылок на него равно 1. Вызывающий объект при этом “владеет” созданным объектом.
- Когда объекту посылается сообщение retain, количество ссылок *увеличивается* на 1.
- Когда объекту посылается сообщение release, количество ссылок *уменьшается* на 1.

Manual retain-release схема ч.1

- Когда объекту посылается сообщение autorelease, количество ссылок уменьшится на 1 в конце текущего блока **autorelease pool**.
- Если количество ссылок на объект становится равным нулю, он удаляется из памяти.

Manual retain-release схема ч.2



Управление памятью в Cocoa: диаграмма.

```
Person *aPerson = [[Person alloc] init]; // “владею” объектом
[aPerson release]; // “отпускаю” объект
NSLog(@”%@”, aPerson); // Обращение к освобожденному участку памяти

Person *personOne = [[Person alloc] init]; // “владею” объектом
Person *personTwo = personOne;
[personOne release]; // “отпускаю” объект
NSLog(@”%@”, personTwo); // Та же проблема

Person *personOne = [[Person alloc] init]; // “владею” объектом
Person *personTwo = [personOne retain]; // увеличиваю кол-во ссылок
[personOne release]; // “отпускаю” объект
NSLog(@”%@”, personTwo); // Все ОК, т.к. объект живет
[personTwo release];
```

Примеры ручного управления памятью

autorelease

- Вызывая autorelease, сообщаем объекту, что ему будет послано release в будущем (в итерации внутри autoreleasepool’a).
- Используется для передачи “владения” объектом.
- Зачастую используется при возвращении объекта из метода.

```
- (NSString *)fullName {
   NSString *string = [[NSString alloc] initWithFormat:@"%@ %@",
                                    self.firstName, self.lastName]; // вызван alloc-метод, я
владею объектом
   return [string autorelease];
}
```

Объект, вызвавший метод fullName успеет воспользоваться объектом, и тот удалится лишь после этого.

```
- (NSString *)fullName {
   NSString *string = [NSString stringWithFormat:@"%@ %@",
                             self.firstName, self.lastName]; // не вызван alloc-метод, я не
владею объектом
   return string;
}
```

Объектом string будет владеть объект, вызывающий метод fullName. Внутри метода завладения объектом не произошло, т.к. не он не создан с помощью соответствующих методов, и ему не послан retain.

Примеры autorelease

Autorelease Pool

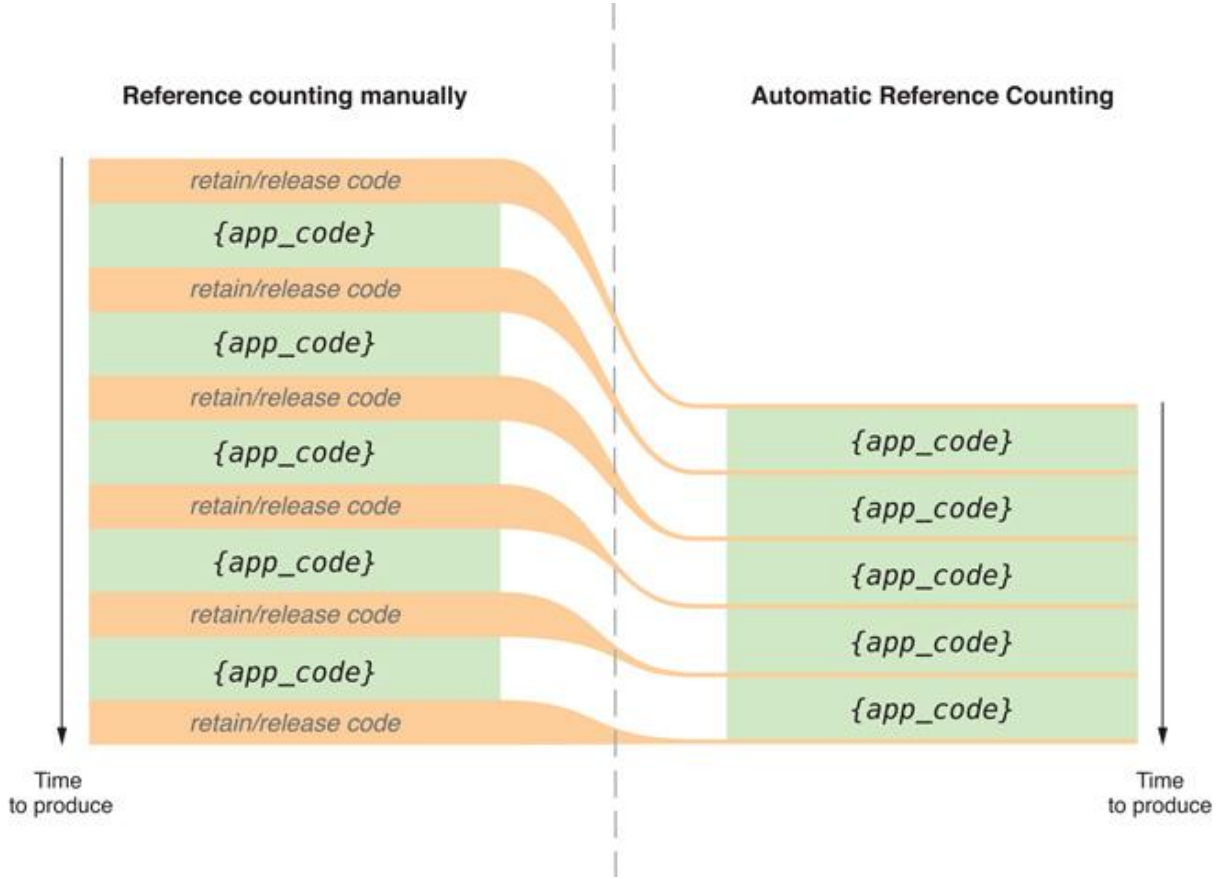
- Предоставляет механизм отложенного вызова release у объектов, которым было отправлено сообщение autorelease.
- Объявляется с помощью блока:

```
@autoreleasepool {
  // Код, использующий autoreleased объекты.
}
```

- Сообщение release посылается всем autoreleased-объектам в конце очередной итерации цикла выполнения, если эти вызовы были “обернуты” в @autoreleasepool блок.
- Если объекту послать autorelease сообщение вне этого блока, произойдет утечка (release не вызовется).

ARC (Automatic Reference Counting)

- Инструмент компилятора, который самостоятельно вставляет вызовы retain, release и autorelease.
- Анализирует исходный код.
- В режиме ARC самостоятельный вызов методов управления памятью запрещен.



Что делает ARC

Спецификаторы переменных в ARC (__strong, __weak, __autoreleasing, __unsafe_unretained)

__strong - спецификатор по умолчанию. Объект не удаляется, пока на него есть __strong ссылка (ARC вставляет retain по необходимости).

Пример:

```
Person *strongPerson = [Person defaultPerson]; // Подразу-
мевается Person * __strong aPerson
NSLog(@"%@", strongPerson.name); // Объект доступен
NSLog(@"%@", strongPerson.surname); // --||--
```

__weak - “слабая” ссылка на объект.

- retain не вызывается.
- Необходимо указывать явно.
- Объект удалится из памяти, как только на него не будет других __strong ссылок.

Пример:

```
Person * __weak weakPerson = [Person defaultPerson];
NSLog(@”%@”, weakPerson.name); // Уже не доступно, объект сразу удалился
```

Пример 2:

```
Person *strongPerson = [Person defaultPerson];
Person * __weak weakPerson = strongPerson;
NSLog(@”%@”, strongPerson.name); // OK, т.к. __strong ссылка существует
NSLog(@”%@”, weakPerson.name); // OK, т.к. __strong ссылка все еще существует
```

Спецификатор __weak

Атрибуты свойств для работы с *объектами* в ARC

- **strong** - атрибут по умолчанию. Пока жив объект, имеющий это свойство, он хранит strong-ссылку на объект, на который ссылается это свойство.

```
@property NSString *myString; //
Подразумевается @property (strong)
NSString *myString;
```

weak - свойство “не владеет” объектом.

- Память под объект освободится, если на него не будет других __strong ссылок или на него не будут ссылаться (strong) свойства.
- Когда объект уничтожится, среда выполнения выставит weak свойство в nil.
- Этот атрибут необходимо явно указывать.

```
@property (weak) id delegate; // явно указывая (weak), сообщаем компилятору о том, что не хотим вызывать retain у объекта при выставлении свойства.
```

unsafe_unretained - то же, что weak, но при удалении объекта указатель не обнулится.

Атрибуты weak и unsafe_unretained

Retain-цикл

Часто атрибут **weak** используется для избегания так называемых “retain-циклов”.
Retain-цикл (Retain Cycle) - особый случай утечки памяти. Ситуация, когда два (или более) объекта жестко ссылаются друг на друга (strongly referenced) таким образом, что ни один из них никогда не удалится из памяти.

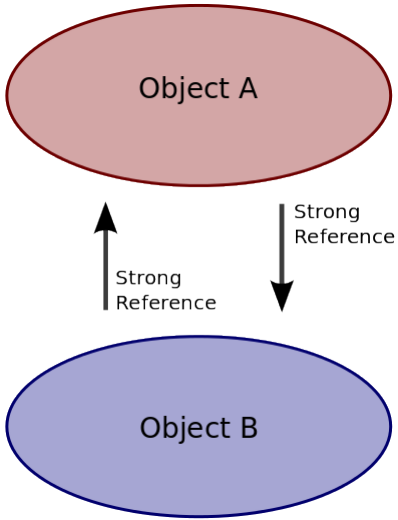


Рис. 1. Retain-цикл

```
@interface Person
@property Person *parent;
@property Person *child;
@end
```

Листинг 1. Объявление класса

```
Person *mother = [[Person alloc] init];
Person *son = [[Person alloc] init];
mother.child = son; // [son retain];
son.parent = mother; // [mother retain];
```

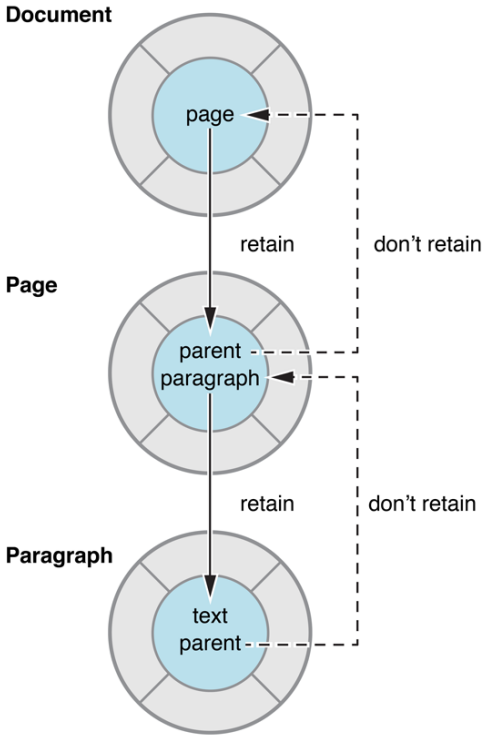
Листинг 2. иллюстрация проблемы

```
@interface Person
@property (weak) Person *parent;
@property Person *child;
@end
```

Листинг 3. Меняем атрибут свойства.

```
Person *mother = [[Person alloc] init];
Person *son = [[Person alloc] init];
mother.child = son; // [son retain];
son.parent = mother; // retain не
вызывается;
```

Листинг 4. Проблема ушла.



Пример правильной связи более двух объектов

Разрешение проблемы retain-цикла