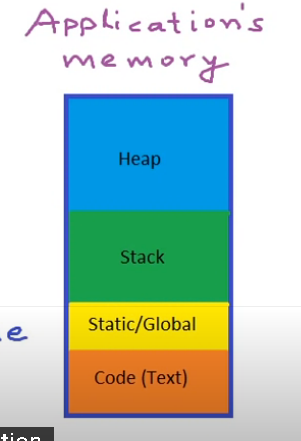
MemoryAllocationExample Description

Order of Operations



1. All functions are loaded into virtual memory into the Code(Text) area
2. All Static and Globals are allocated to memory and initialized
3. Next the stack is allocated as functions get called
4. As functions call the new functions memory is allocated to the Heap

# Code

#include <stdio.h>

#include <string>

class Animal {

public:

static int AnimalCount; // static attribute is one per class definition and is really global with class scope

int Age = 1; // attribute is only created when an object of animal type is created

Animal() { printf("Animal::Animal() - Default Constructor - At address %p - AnimalCount = %i\n", this, ++AnimalCount);}

Animal(const Animal& p2) { printf("Animal::Animal() - Copy Constructor - At address %p - AnimalCount = %i\n", this, ++AnimalCount); Age = p2.Age; }

~Animal() { printf("Animal::~Animal() - Default Destructor - At address %p - AnimalCount = %i\n", this, --AnimalCount); }

virtual void Speak() { printf("Animal::Speak - Animal said something\n"); }

};

int Animal::AnimalCount = 0; // Initialize class static attribute

class Dog: public Animal {

public:

int NumberOfTreats = 0;

virtual void Speak() { printf("Dog::Speak - Dog said Bark!\n"); }

};

class Cat: public Animal {

public:

int Lives = 9;

int SleepHours = 20;

virtual void Speak() { printf("Cat::Speak - Cat said Meow!\n"); }

};

Cat Pablo;

void PrintAllStaticAndGlobalAddresses() {

printf("\*\*\*\*\* Starting PrintAllStaticAndGlobalAddresses - Printing all static and global addresseses \n");

printf("%p is the address of %s with size of %i\n", &Animal::AnimalCount, "int Animal::AnimalCount", sizeof(Animal::AnimalCount));

printf("%p is the address of %s with size of %i\n", &Pablo, "Cat Pablo", sizeof(Pablo));

printf("\*\*\*\*\* Exiting PrintAllStaticAndGlobalAddresses \n");

}

// Forward function delarations so we can print thier addresses in the PrintAllFunctionAddresses

void AnotherFunc();

int main();

Animal\* PassingParameters(

int MyIntParam, // Pass by value

Animal\* AnyAnimalParam, // Passes a pointer

Cat CatOneParam, // Passes by value so a copy is created

int MyMiddleParam, // Pass by value

int MyMiddleParam2, // Pass by value

Cat& CatTwoParam, // Passes by reference so only the pointer value is passed under the covers

int MyLastParam // Pass by value

);

void PrintAllFunctionAddresses() {

printf("\*\*\*\*\* Starting PrintAllFunctionAddresses - Printing all function addresses \n");

printf("%p is the address of function Animal::Speak\n", &Animal::Speak);

printf("%p is the address of function Dog::Speak\n", &Dog::Speak);

printf("%p is the address of function Cat::Speak\n", &Cat::Speak);

printf("%p is the address of function PrintAllStaticAndGlobalAddresses\n", &PrintAllStaticAndGlobalAddresses);

printf("%p is the address of function PrintAllFunctionAddresses\n", &PrintAllFunctionAddresses);

printf("%p is the address of function AnotherFunc\n", &AnotherFunc);

printf("%p is the address of function PassingParameters\n", &PassingParameters);

printf("%p is the address of function main\n", &main);

printf("\*\*\*\*\* Exiting PrintAllFunctionAddresses \n");

}

void AnotherFunc() {

int myNewInt = 5;

printf("\*\*\*\*\* Starting AnotherFunc\n");

printf("%p is the address of %s with size of %i\n", &myNewInt, "int myNewInt", sizeof(myNewInt));

printf("\*\*\*\*\* Exiting AnotherFunc\n");

}

Animal\* PassingParameters(

int MyIntParam, // Pass by value

Animal\* AnyAnimalParam, // Passes a pointer

Cat CatOneParam, // Passes by value so a copy is created

int MyMiddleParam, // Pass by value

int MyMiddleParam2, // Pass by value

Cat& CatTwoParam, // Passes by reference so only the pointer value is passed under the covers

int MyLastParam // Pass by value

)

{

int LocalInt = 2;

int\* myPtr = &MyMiddleParam2 + 1; // Adds by Size of pointer type in this case by 4 because the integer has a sizeof 4

Dog\* myDogPtr = nullptr;

printf("\*\*\*\*\* Starting PassingParameters\n");

printf("\*\* Printing out parameters on the stack\n"); // print out Parameters

printf("%p is the address of %s with size of %i\n", &MyIntParam, "int MyIntParam", sizeof(MyIntParam));

printf("%p is the address of %s with size of %i with value %p\n", &AnyAnimalParam, "Animal\* AnyAnimalParam", sizeof(AnyAnimalParam), AnyAnimalParam);

printf("%p is the address of %s with size of %i\n", &CatOneParam, "Cat CatOneParam", sizeof(CatOneParam));

printf("%p is the address of %s with size of %i\n", &MyMiddleParam, "int MyMiddleParam", sizeof(MyMiddleParam));

printf("%p is the address of %s with size of %i\n", &MyMiddleParam2, "int MyMiddleParam2", sizeof(MyMiddleParam2));

printf("%p is the address of %s with size of %i - Not showing actual values but the object being referenced\n", std::addressof(CatTwoParam), "Cat& CatTwoParam", sizeof(CatTwoParam));

printf("%p is the real address of %s with size of %i with a value of %p\n", myPtr, "Cat& CatTwoParam", sizeof(myPtr), \*myPtr);

printf("%p is the address of %s with size of %i\n", &MyLastParam, "int MyLastParam", sizeof(MyLastParam));

printf("\*\* Printing out local variables on the stack\n");// Print out Local vars

printf("%p is the address of %s with size of %i\n", &LocalInt, "int LocalInt", sizeof(LocalInt));

printf("%p is the address of %s with size of %i\n", &myPtr, "int\* myPtr", sizeof(myPtr));

printf("%p is the address of %s with size of %i\n", &myDogPtr, "Dog\* myDogPtr", sizeof(myDogPtr));

myDogPtr = new Dog();

printf("%p is the address of %s with size of %i with a value of %p which is the address in the heap\n", &myDogPtr, "Dog\* myDogPtr", sizeof(myDogPtr), myDogPtr);

printf("%s has a size of %i\n", "Dog Object that myDogPtr is pointing to", sizeof(\*myDogPtr));

printf("The first 4 bytes is the hidden Virtual Table pointer address\n");

printf("%p is the address of %s with size of %i\n", &myDogPtr->Age, "myDogPtr->Age", sizeof(myDogPtr->Age));

printf("%p is the address of %s with size of %i\n", &myDogPtr->NumberOfTreats, "myDogPtr->NumberOfTreats", sizeof(myDogPtr->NumberOfTreats));

AnyAnimalParam->Speak();

CatOneParam.Speak();

CatTwoParam.Speak();

AnotherFunc();

printf("\*\*\*\*\* Exiting PassingParameters\n");

return myDogPtr;

}

//int main\_for\_MemoryAllocationExample() { // comment this line and uncomment the next when running.

int main() {

printf("\*\*\*\*\* Starting main() function - print statements before this one represent code executed to initialize statics and globals\n");

PrintAllFunctionAddresses();

PrintAllStaticAndGlobalAddresses();

Dog Gizmo;

printf("%p is the address of %s with size of %i\n", &Gizmo, "Dog Gizmo", sizeof(Gizmo));

Animal\* myAnimalPtr = PassingParameters(2, &Gizmo, Pablo, 4, 5, Pablo, 3);

AnotherFunc();

printf("%p is the address of %s with size of %i with a value of %p which is the address in the heap\n", &myAnimalPtr, "Animal\* myAnimalPtr", sizeof(myAnimalPtr), myAnimalPtr);

myAnimalPtr->Speak();

delete myAnimalPtr;

printf("\*\*\*\*\* Exiting main() function - All printfs after this are destructors on statics and globals \n");

return 0;

}

# Output

Animal::Animal() - Default Constructor - At address 008B5388 - AnimalCount = 1

\*\*\*\*\* Starting main() function - print statements before this one represent code executed to initialize statics and globals

\*\*\*\*\* Starting PrintAllFunctionAddresses - Printing all function addresses

008A1532 is the address of function Animal::Speak

008A1492 is the address of function Dog::Speak

008A12A8 is the address of function Cat::Speak

008A1019 is the address of function PrintAllStaticAndGlobalAddresses

008A1131 is the address of function PrintAllFunctionAddresses

008A1578 is the address of function AnotherFunc

008A12EE is the address of function PassingParameters

008A14B0 is the address of function main

\*\*\*\*\* Exiting PrintAllFunctionAddresses

\*\*\*\*\* Starting PrintAllStaticAndGlobalAddresses - Printing all static and global addresseses

008B5384 is the address of int Animal::AnimalCount with size of 4

008B5388 is the address of Cat Pablo with size of 16

\*\*\*\*\* Exiting PrintAllStaticAndGlobalAddresses

Animal::Animal() - Default Constructor - At address 009BF720 - AnimalCount = 2

009BF720 is the address of Dog Gizmo with size of 12

Animal::Animal() - Copy Constructor - At address 009BF5F8 - AnimalCount = 3

\*\*\*\*\* Starting PassingParameters

\*\* Printing out parameters on the stack

009BF5F0 is the address of int MyIntParam with size of 4

009BF5F4 is the address of Animal\* AnyAnimalParam with size of 4 with value 009BF720

009BF5F8 is the address of Cat CatOneParam with size of 16

009BF608 is the address of int MyMiddleParam with size of 4

009BF60C is the address of int MyMiddleParam2 with size of 4

008B5388 is the address of Cat& CatTwoParam with size of 16 - Not showing actual values but the object being referenced

009BF610 is the real address of Cat& CatTwoParam with size of 4 with a value of 008B5388

009BF614 is the address of int MyLastParam with size of 4

\*\* Printing out local variables on the stack

009BF5D0 is the address of int LocalInt with size of 4

009BF5C4 is the address of int\* myPtr with size of 4

009BF5B8 is the address of Dog\* myDogPtr with size of 4

Animal::Animal() - Default Constructor - At address 00D33C10 - AnimalCount = 4

009BF5B8 is the address of Dog\* myDogPtr with size of 4 with a value of 00D33C10 which is the address in the heap

Dog Object that myDogPtr is pointing to has a size of 12

The first 4 bytes is the hidden Virtual Table pointer address

00D33C14 is the address of myDogPtr->Age with size of 4

00D33C18 is the address of myDogPtr->NumberOfTreats with size of 4

Dog::Speak - Dog said Bark!

Cat::Speak - Cat said Meow!

Cat::Speak - Cat said Meow!

\*\*\*\*\* Starting AnotherFunc

009BF4A8 is the address of int myNewInt with size of 4

\*\*\*\*\* Exiting AnotherFunc

\*\*\*\*\* Exiting PassingParameters

Animal::~Animal() - Default Destructor - At address 009BF5F8 - AnimalCount = 3

\*\*\*\*\* Starting AnotherFunc

009BF604 is the address of int myNewInt with size of 4

\*\*\*\*\* Exiting AnotherFunc

009BF714 is the address of Animal\* myAnimalPtr with size of 4 with a value of 00D33C10 which is the address in the heap

Dog::Speak - Dog said Bark!

Animal::~Animal() - Default Destructor - At address 00D33C10 - AnimalCount = 2

\*\*\*\*\* Exiting main() function - All printfs after this are destructors on statics and globals

Animal::~Animal() - Default Destructor - At address 009BF720 - AnimalCount = 1

Animal::~Animal() - Default Destructor - At address 008B5388 - AnimalCount = 0

# Code Compiled

.LC0:

        .string "Animal::Animal() - Default Constructor - At address %p - AnimalCount = %i\n"

Animal::Animal() [base object constructor]:

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     QWORD PTR [rbp-8], rdi

        mov     edx, OFFSET FLAT:vtable for Animal+16

        mov     rax, QWORD PTR [rbp-8]

        mov     QWORD PTR [rax], rdx

        mov     rax, QWORD PTR [rbp-8]

        mov     DWORD PTR [rax+8], 1

        mov     eax, DWORD PTR Animal::AnimalCount[rip]

        add     eax, 1

        mov     DWORD PTR Animal::AnimalCount[rip], eax

        mov     edx, DWORD PTR Animal::AnimalCount[rip]

        mov     rax, QWORD PTR [rbp-8]

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC0

        mov     eax, 0

        call    printf

        nop

        leave

        ret

.LC1:

        .string "Animal::Animal() - Copy Constructor - At address %p - AnimalCount = %i\n"

Animal::Animal(Animal const&):

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     QWORD PTR [rbp-8], rdi

        mov     QWORD PTR [rbp-16], rsi

        mov     edx, OFFSET FLAT:vtable for Animal+16

        mov     rax, QWORD PTR [rbp-8]

        mov     QWORD PTR [rax], rdx

        mov     rax, QWORD PTR [rbp-8]

        mov     DWORD PTR [rax+8], 1

        mov     eax, DWORD PTR Animal::AnimalCount[rip]

        add     eax, 1

        mov     DWORD PTR Animal::AnimalCount[rip], eax

        mov     edx, DWORD PTR Animal::AnimalCount[rip]

        mov     rax, QWORD PTR [rbp-8]

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC1

        mov     eax, 0

        call    printf

        mov     rax, QWORD PTR [rbp-16]

        mov     edx, DWORD PTR [rax+8]

        mov     rax, QWORD PTR [rbp-8]

        mov     DWORD PTR [rax+8], edx

        nop

        leave

        ret

.LC2:

        .string "Animal::~Animal() - Default Destructor - At address %p -  AnimalCount = %i\n"

Animal::~Animal() [base object destructor]:

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     QWORD PTR [rbp-8], rdi

        mov     edx, OFFSET FLAT:vtable for Animal+16

        mov     rax, QWORD PTR [rbp-8]

        mov     QWORD PTR [rax], rdx

        mov     eax, DWORD PTR Animal::AnimalCount[rip]

        sub     eax, 1

        mov     DWORD PTR Animal::AnimalCount[rip], eax

        mov     edx, DWORD PTR Animal::AnimalCount[rip]

        mov     rax, QWORD PTR [rbp-8]

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC2

        mov     eax, 0

        call    printf

        nop

        leave

        ret

.LC3:

        .string "Animal::Speak - Animal said something"

Animal::Speak():

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     QWORD PTR [rbp-8], rdi

        mov     edi, OFFSET FLAT:.LC3

        call    puts

        nop

        leave

        ret

Animal::AnimalCount:

        .zero   4

.LC4:

        .string "Dog::Speak - Dog said Bark!"

Dog::Speak():

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     QWORD PTR [rbp-8], rdi

        mov     edi, OFFSET FLAT:.LC4

        call    puts

        nop

        leave

        ret

.LC5:

        .string "Cat::Speak - Cat said Meow!"

Cat::Speak():

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     QWORD PTR [rbp-8], rdi

        mov     edi, OFFSET FLAT:.LC5

        call    puts

        nop

        leave

        ret

Cat::Cat() [base object constructor]:

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     QWORD PTR [rbp-8], rdi

        mov     rax, QWORD PTR [rbp-8]

        mov     rdi, rax

        call    Animal::Animal() [base object constructor]

        mov     edx, OFFSET FLAT:vtable for Cat+16

        mov     rax, QWORD PTR [rbp-8]

        mov     QWORD PTR [rax], rdx

        mov     rax, QWORD PTR [rbp-8]

        mov     DWORD PTR [rax+12], 9

        mov     rax, QWORD PTR [rbp-8]

        mov     DWORD PTR [rax+16], 20

        nop

        leave

        ret

Pablo:

        .zero   24

.LC6:

        .string "\*\*\*\*\* Starting PrintAllStaticAndGlobalAddresses - Printing all static and global addresseses  "

.LC7:

        .string "int Animal::AnimalCount"

.LC8:

        .string "%p is the address of %s with size of %i\n"

.LC9:

        .string "Cat Pablo"

.LC10:

        .string "\*\*\*\*\* Exiting PrintAllStaticAndGlobalAddresses "

PrintAllStaticAndGlobalAddresses():

        push    rbp

        mov     rbp, rsp

        mov     edi, OFFSET FLAT:.LC6

        call    puts

        mov     ecx, 4

        mov     edx, OFFSET FLAT:.LC7

        mov     esi, OFFSET FLAT:Animal::AnimalCount

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        mov     ecx, 24

        mov     edx, OFFSET FLAT:.LC9

        mov     esi, OFFSET FLAT:Pablo

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        mov     edi, OFFSET FLAT:.LC10

        call    puts

        nop

        pop     rbp

        ret

.LC11:

        .string "\*\*\*\*\* Starting PrintAllFunctionAddresses - Printing all function addresses  "

.LC12:

        .string "%p is the address of function Animal::Speak\n"

.LC13:

        .string "%p is the address of function Dog::Speak\n"

.LC14:

        .string "%p is the address of function Cat::Speak\n"

.LC15:

        .string "%p is the address of function PrintAllStaticAndGlobalAddresses\n"

.LC16:

        .string "%p is the address of function PrintAllFunctionAddresses\n"

.LC17:

        .string "%p is the address of function AnotherFunc\n"

.LC18:

        .string "%p is the address of function PassingParameters\n"

.LC19:

        .string "%p is the address of function main\n"

.LC20:

        .string "\*\*\*\*\* Exiting PrintAllFunctionAddresses "

PrintAllFunctionAddresses():

        push    rbp

        mov     rbp, rsp

        push    r15

        push    r14

        push    r13

        push    r12

        push    rbx

        sub     rsp, 24

        mov     edi, OFFSET FLAT:.LC11

        call    puts

        mov     QWORD PTR [rbp-64], 1

        mov     QWORD PTR [rbp-56], 0

        mov     rax, QWORD PTR [rbp-64]

        mov     rdx, QWORD PTR [rbp-56]

        mov     rcx, rax

        mov     rbx, rdx

        mov     rax, rdx

        mov     rsi, rcx

        mov     rdx, rax

        mov     edi, OFFSET FLAT:.LC12

        mov     eax, 0

        call    printf

        mov     r14d, 1

        mov     r15d, 0

        mov     rcx, r14

        mov     rbx, r15

        mov     rax, r14

        mov     rdx, r15

        mov     rax, rdx

        mov     rsi, rcx

        mov     rdx, rax

        mov     edi, OFFSET FLAT:.LC13

        mov     eax, 0

        call    printf

        mov     r12d, 1

        mov     r13d, 0

        mov     rcx, r12

        mov     rbx, r13

        mov     rax, r12

        mov     rdx, r13

        mov     rax, rdx

        mov     rsi, rcx

        mov     rdx, rax

        mov     edi, OFFSET FLAT:.LC14

        mov     eax, 0

        call    printf

        mov     esi, OFFSET FLAT:PrintAllStaticAndGlobalAddresses()

        mov     edi, OFFSET FLAT:.LC15

        mov     eax, 0

        call    printf

        mov     esi, OFFSET FLAT:PrintAllFunctionAddresses()

        mov     edi, OFFSET FLAT:.LC16

        mov     eax, 0

        call    printf

        mov     esi, OFFSET FLAT:AnotherFunc()

        mov     edi, OFFSET FLAT:.LC17

        mov     eax, 0

        call    printf

        mov     esi, OFFSET FLAT:PassingParameters(int, Animal\*, Cat, int, int, Cat&, int)

        mov     edi, OFFSET FLAT:.LC18

        mov     eax, 0

        call    printf

        mov     esi, OFFSET FLAT:main

        mov     edi, OFFSET FLAT:.LC19

        mov     eax, 0

        call    printf

        mov     edi, OFFSET FLAT:.LC20

        call    puts

        nop

        add     rsp, 24

        pop     rbx

        pop     r12

        pop     r13

        pop     r14

        pop     r15

        pop     rbp

        ret

.LC21:

        .string "\*\*\*\*\* Starting AnotherFunc"

.LC22:

        .string "int myNewInt"

.LC23:

        .string "\*\*\*\*\* Exiting AnotherFunc"

AnotherFunc():

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     DWORD PTR [rbp-4], 5

        mov     edi, OFFSET FLAT:.LC21

        call    puts

        lea     rax, [rbp-4]

        mov     ecx, 4

        mov     edx, OFFSET FLAT:.LC22

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        mov     edi, OFFSET FLAT:.LC23

        call    puts

        nop

        leave

        ret

Dog::Dog() [base object constructor]:

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     QWORD PTR [rbp-8], rdi

        mov     rax, QWORD PTR [rbp-8]

        mov     rdi, rax

        call    Animal::Animal() [base object constructor]

        mov     edx, OFFSET FLAT:vtable for Dog+16

        mov     rax, QWORD PTR [rbp-8]

        mov     QWORD PTR [rax], rdx

        mov     rax, QWORD PTR [rbp-8]

        mov     DWORD PTR [rax+12], 0

        nop

        leave

        ret

.LC24:

        .string "\*\*\*\*\* Starting PassingParameters"

.LC25:

        .string "\*\* Printing out parameters on the stack"

.LC26:

        .string "int MyIntParam"

.LC27:

        .string "Animal\* AnyAnimalParam"

.LC28:

        .string "%p is the address of %s with size of %i with value %p\n"

.LC29:

        .string "Cat CatOneParam"

.LC30:

        .string "int MyMiddleParam"

.LC31:

        .string "int MyMiddleParam2"

.LC32:

        .string "Cat& CatTwoParam"

.LC33:

        .string "%p is the address of %s with size of %i - Not showing actual values but the object being referenced\n"

.LC34:

        .string "%p is the real address of %s with size of %i with a value of %p\n"

.LC35:

        .string "int MyLastParam"

.LC36:

        .string "\*\* Printing out local variables on the stack"

.LC37:

        .string "int LocalInt"

.LC38:

        .string "int\* myPtr"

.LC39:

        .string "Dog\* myDogPtr"

.LC40:

        .string "%p is the address of %s with size of %i with a value of %p which is the address in the heap\n"

.LC41:

        .string "Dog Object that myDogPtr is pointing to"

.LC42:

        .string "%s has a size of %i\n"

.LC43:

        .string "The first 4 bytes is the hidden Virtual Table pointer address"

.LC44:

        .string "myDogPtr->Age"

.LC45:

        .string "myDogPtr->NumberOfTreats"

.LC46:

        .string "\*\*\*\*\* Exiting PassingParameters"

PassingParameters(int, Animal\*, Cat, int, int, Cat&, int):

        push    rbp

        mov     rbp, rsp

        push    r12

        push    rbx

        sub     rsp, 80

        mov     DWORD PTR [rbp-52], edi

        mov     QWORD PTR [rbp-64], rsi

        mov     QWORD PTR [rbp-72], rdx

        mov     DWORD PTR [rbp-56], ecx

        mov     DWORD PTR [rbp-76], r8d

        mov     QWORD PTR [rbp-88], r9

        mov     DWORD PTR [rbp-20], 2

        lea     rax, [rbp-76]

        add     rax, 4

        mov     QWORD PTR [rbp-32], rax

        mov     QWORD PTR [rbp-40], 0

        mov     edi, OFFSET FLAT:.LC24

        call    puts

        mov     edi, OFFSET FLAT:.LC25

        call    puts

        lea     rax, [rbp-52]

        mov     ecx, 4

        mov     edx, OFFSET FLAT:.LC26

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        mov     rdx, QWORD PTR [rbp-64]

        lea     rax, [rbp-64]

        mov     r8, rdx

        mov     ecx, 8

        mov     edx, OFFSET FLAT:.LC27

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC28

        mov     eax, 0

        call    printf

        mov     rax, QWORD PTR [rbp-72]

        mov     ecx, 24

        mov     edx, OFFSET FLAT:.LC29

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        lea     rax, [rbp-56]

        mov     ecx, 4

        mov     edx, OFFSET FLAT:.LC30

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        lea     rax, [rbp-76]

        mov     ecx, 4

        mov     edx, OFFSET FLAT:.LC31

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        mov     rax, QWORD PTR [rbp-88]

        mov     rdi, rax

        call    Cat\* std::addressof<Cat>(Cat&)

        mov     ecx, 24

        mov     edx, OFFSET FLAT:.LC32

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC33

        mov     eax, 0

        call    printf

        mov     rax, QWORD PTR [rbp-32]

        mov     edx, DWORD PTR [rax]

        mov     rax, QWORD PTR [rbp-32]

        mov     r8d, edx

        mov     ecx, 8

        mov     edx, OFFSET FLAT:.LC32

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC34

        mov     eax, 0

        call    printf

        mov     ecx, 4

        mov     edx, OFFSET FLAT:.LC35

        lea     rsi, [rbp+16]

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        mov     edi, OFFSET FLAT:.LC36

        call    puts

        lea     rax, [rbp-20]

        mov     ecx, 4

        mov     edx, OFFSET FLAT:.LC37

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        lea     rax, [rbp-32]

        mov     ecx, 8

        mov     edx, OFFSET FLAT:.LC38

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        lea     rax, [rbp-40]

        mov     ecx, 8

        mov     edx, OFFSET FLAT:.LC39

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        mov     edi, 16

        call    operator new(unsigned long)

        mov     rbx, rax

        mov     QWORD PTR [rbx], 0

        mov     DWORD PTR [rbx+8], 0

        mov     DWORD PTR [rbx+12], 0

        mov     rdi, rbx

        call    Dog::Dog() [complete object constructor]

        mov     QWORD PTR [rbp-40], rbx

        mov     rdx, QWORD PTR [rbp-40]

        lea     rax, [rbp-40]

        mov     r8, rdx

        mov     ecx, 8

        mov     edx, OFFSET FLAT:.LC39

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC40

        mov     eax, 0

        call    printf

        mov     edx, 16

        mov     esi, OFFSET FLAT:.LC41

        mov     edi, OFFSET FLAT:.LC42

        mov     eax, 0

        call    printf

        mov     edi, OFFSET FLAT:.LC43

        call    puts

        mov     rax, QWORD PTR [rbp-40]

        add     rax, 8

        mov     ecx, 4

        mov     edx, OFFSET FLAT:.LC44

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        mov     rax, QWORD PTR [rbp-40]

        add     rax, 12

        mov     ecx, 4

        mov     edx, OFFSET FLAT:.LC45

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        mov     rax, QWORD PTR [rbp-64]

        mov     rdx, QWORD PTR [rbp-64]

        mov     rdx, QWORD PTR [rdx]

        mov     rdx, QWORD PTR [rdx]

        mov     rdi, rax

        call    rdx

        mov     rax, QWORD PTR [rbp-72]

        mov     rdi, rax

        call    Cat::Speak()

        mov     rax, QWORD PTR [rbp-88]

        mov     rax, QWORD PTR [rax]

        mov     rdx, QWORD PTR [rax]

        mov     rax, QWORD PTR [rbp-88]

        mov     rdi, rax

        call    rdx

        call    AnotherFunc()

        mov     edi, OFFSET FLAT:.LC46

        call    puts

        mov     rax, QWORD PTR [rbp-40]

        jmp     .L16

        mov     r12, rax

        mov     esi, 16

        mov     rdi, rbx

        call    operator delete(void\*, unsigned long)

        mov     rax, r12

        mov     rdi, rax

        call    \_Unwind\_Resume

.L16:

        add     rsp, 80

        pop     rbx

        pop     r12

        pop     rbp

        ret

Dog::~Dog() [base object destructor]:

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     QWORD PTR [rbp-8], rdi

        mov     edx, OFFSET FLAT:vtable for Dog+16

        mov     rax, QWORD PTR [rbp-8]

        mov     QWORD PTR [rax], rdx

        mov     rax, QWORD PTR [rbp-8]

        mov     rdi, rax

        call    Animal::~Animal() [base object destructor]

        nop

        leave

        ret

Cat::Cat(Cat const&):

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     QWORD PTR [rbp-8], rdi

        mov     QWORD PTR [rbp-16], rsi

        mov     rax, QWORD PTR [rbp-8]

        mov     rdx, QWORD PTR [rbp-16]

        mov     rsi, rdx

        mov     rdi, rax

        call    Animal::Animal(Animal const&)

        mov     edx, OFFSET FLAT:vtable for Cat+16

        mov     rax, QWORD PTR [rbp-8]

        mov     QWORD PTR [rax], rdx

        mov     rax, QWORD PTR [rbp-16]

        mov     edx, DWORD PTR [rax+12]

        mov     rax, QWORD PTR [rbp-8]

        mov     DWORD PTR [rax+12], edx

        mov     rax, QWORD PTR [rbp-16]

        mov     edx, DWORD PTR [rax+16]

        mov     rax, QWORD PTR [rbp-8]

        mov     DWORD PTR [rax+16], edx

        nop

        leave

        ret

Cat::~Cat() [base object destructor]:

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     QWORD PTR [rbp-8], rdi

        mov     edx, OFFSET FLAT:vtable for Cat+16

        mov     rax, QWORD PTR [rbp-8]

        mov     QWORD PTR [rax], rdx

        mov     rax, QWORD PTR [rbp-8]

        mov     rdi, rax

        call    Animal::~Animal() [base object destructor]

        nop

        leave

        ret

.LC47:

        .string "\*\*\*\*\* Starting  main() function - print statements before this one represent code executed to initialize statics and globals"

.LC48:

        .string "Dog Gizmo"

.LC49:

        .string "Animal\* myAnimalPtr"

.LC50:

        .string "\*\*\*\*\* Exiting  main() function - All printfs after this are destructors on statics and globals "

main:

        push    rbp

        mov     rbp, rsp

        push    rbx

        sub     rsp, 72

        mov     edi, OFFSET FLAT:.LC47

        call    puts

        call    PrintAllFunctionAddresses()

        call    PrintAllStaticAndGlobalAddresses()

        lea     rax, [rbp-64]

        mov     rdi, rax

        call    Dog::Dog() [complete object constructor]

        lea     rax, [rbp-64]

        mov     ecx, 16

        mov     edx, OFFSET FLAT:.LC48

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC8

        mov     eax, 0

        call    printf

        lea     rax, [rbp-48]

        mov     esi, OFFSET FLAT:Pablo

        mov     rdi, rax

        call    Cat::Cat(Cat const&)

        lea     rdx, [rbp-48]

        lea     rax, [rbp-64]

        sub     rsp, 8

        push    3

        mov     r9d, OFFSET FLAT:Pablo

        mov     r8d, 5

        mov     ecx, 4

        mov     rsi, rax

        mov     edi, 2

        call    PassingParameters(int, Animal\*, Cat, int, int, Cat&, int)

        add     rsp, 16

        mov     QWORD PTR [rbp-72], rax

        lea     rax, [rbp-48]

        mov     rdi, rax

        call    Cat::~Cat() [complete object destructor]

        call    AnotherFunc()

        mov     rdx, QWORD PTR [rbp-72]

        lea     rax, [rbp-72]

        mov     r8, rdx

        mov     ecx, 8

        mov     edx, OFFSET FLAT:.LC49

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC40

        mov     eax, 0

        call    printf

        mov     rax, QWORD PTR [rbp-72]

        mov     rdx, QWORD PTR [rbp-72]

        mov     rdx, QWORD PTR [rdx]

        mov     rdx, QWORD PTR [rdx]

        mov     rdi, rax

        call    rdx

        mov     rbx, QWORD PTR [rbp-72]

        test    rbx, rbx

        je      .L21

        mov     rdi, rbx

        call    Animal::~Animal() [complete object destructor]

        mov     esi, 16

        mov     rdi, rbx

        call    operator delete(void\*, unsigned long)

.L21:

        mov     edi, OFFSET FLAT:.LC50

        call    puts

        mov     ebx, 0

        lea     rax, [rbp-64]

        mov     rdi, rax

        call    Dog::~Dog() [complete object destructor]

        mov     eax, ebx

        jmp     .L27

        mov     rbx, rax

        lea     rax, [rbp-48]

        mov     rdi, rax

        call    Cat::~Cat() [complete object destructor]

        jmp     .L24

        mov     rbx, rax

.L24:

        lea     rax, [rbp-64]

        mov     rdi, rax

        call    Dog::~Dog() [complete object destructor]

        mov     rax, rbx

        mov     rdi, rax

        call    \_Unwind\_Resume

.L27:

        mov     rbx, QWORD PTR [rbp-8]

        leave

        ret

Cat\* std::addressof<Cat>(Cat&):

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     QWORD PTR [rbp-8], rdi

        mov     rax, QWORD PTR [rbp-8]

        mov     rdi, rax

        call    Cat\* std::\_\_addressof<Cat>(Cat&)

        leave

        ret

Cat\* std::\_\_addressof<Cat>(Cat&):

        push    rbp

        mov     rbp, rsp

        mov     QWORD PTR [rbp-8], rdi

        mov     rax, QWORD PTR [rbp-8]

        pop     rbp

        ret

vtable for Cat:

        .quad   0

        .quad   typeinfo for Cat

        .quad   Cat::Speak()

vtable for Dog:

        .quad   0

        .quad   typeinfo for Dog

        .quad   Dog::Speak()

vtable for Animal:

        .quad   0

        .quad   typeinfo for Animal

        .quad   Animal::Speak()

typeinfo for Cat:

        .quad   vtable for \_\_cxxabiv1::\_\_si\_class\_type\_info+16

        .quad   typeinfo name for Cat

        .quad   typeinfo for Animal

typeinfo name for Cat:

        .string "3Cat"

typeinfo for Dog:

        .quad   vtable for \_\_cxxabiv1::\_\_si\_class\_type\_info+16

        .quad   typeinfo name for Dog

        .quad   typeinfo for Animal

typeinfo name for Dog:

        .string "3Dog"

typeinfo for Animal:

        .quad   vtable for \_\_cxxabiv1::\_\_class\_type\_info+16

        .quad   typeinfo name for Animal

typeinfo name for Animal:

        .string "6Animal"

\_\_static\_initialization\_and\_destruction\_0(int, int):

        push    rbp

        mov     rbp, rsp

        sub     rsp, 16

        mov     DWORD PTR [rbp-4], edi

        mov     DWORD PTR [rbp-8], esi

        cmp     DWORD PTR [rbp-4], 1

        jne     .L34

        cmp     DWORD PTR [rbp-8], 65535

        jne     .L34

        mov     edi, OFFSET FLAT:Pablo

        call    Cat::Cat() [complete object constructor]

        mov     edx, OFFSET FLAT:\_\_dso\_handle

        mov     esi, OFFSET FLAT:Pablo

        mov     edi, OFFSET FLAT:Cat::~Cat() [complete object destructor]

        call    \_\_cxa\_atexit

.L34:

        nop

        leave

        ret

\_GLOBAL\_\_sub\_I\_Animal::AnimalCount:

        push    rbp

        mov     rbp, rsp

        mov     esi, 65535

        mov     edi, 1

        call    \_\_static\_initialization\_and\_destruction\_0(int, int)

        pop     rbp

        ret