Sorting

Knox Game Design

March 2021

Levi D. Smith

Overview

- Basics of comparing
- Run time
- Sorting algorithms

- Implementations
  - Unity
    - C#
  - GameMaker
    - GML
  - Godot
    - GDScript
  - Unreal Engine
    - C++
  - Pico-8
    - Lua

Why is Sorting Useful?

- Display information to user in a way that makes information easier to find
  - By item count
  - By name
  - By type
  - By cost
  - By oldest to newest
  - By race/lap time
  - By rarity

Comparing Items

- By integer value
  - 1, 3, 8, 8, 25, 64...
- By character value
  - A, G, O, P, c, f, m, n, z
  - Usually, each character is converted to its ASCII integer value
    - A = 65, a = 97, 0 = 48
- By (character) string value
  - Case sensitive
    - APPLE, Cat, Elephant, aardvark, dog, zebra
  - Case insensitive - convert to same case before sorting
    - aardvark, APPLE, Cat, Dog, elephant, zebra
- Time
  - Convert to lowest significant unit (seconds, milliseconds, etc)
- Dates
  - Convert to time since a starting point (Epoch time, since January 1, 1970)

032	{ }	052	4	072	H	092	\
033	!	053	5	073	I	093	]
034	"	054	6	074	J	094	^
035	#	055	7	075	K	095	_
036	$	056	8	076	L	096	`
037	%	057	9	077	M	097	a
038	&	058	:	078	N	098	b
039	'	059	;	079	O	099	c
040	(	060	<	080	P	100	d
041	)	061	=	081	Q	101	e
042	*	062	>	082	R	102	f
043	+	063	?	083	S	103	g
044	,	064	@	084	T	104	h
045	-	065	A	085	U	105	i
046	.	066	B	086	V	106	j
047	/	067	C	087	W	107	k
048	0	068	D	088	X	108	l
049	1	069	E	089	Y	109	m
050	2	070	F	090	Z	110	n
051	3	071	G	091	[	111	o

KNOX GAME DESIGN

Getting a character ASCII value

- C/C++
  - (int) c
- Java
  - (int) str.charAt(i)
- C#
  - (int) str[0]
- Ruby
  - str.ord
- Python
  - ord(str)
- GameMaker GML
  - ord(str)

String comparison

- Most languages have a string comparison operator that returns the relative position value between two strings (-1 less than, 0 equal, 1 greater than)
- C/C++
  - strcmp(char *str1, char * str2) > 0
- Java
  - str1.compareTo(str2) > 0
- C#
  - string.Compare(str1, str2)
- Ruby
  - str1 <=> str2
- Python
  - str1 > str2
  - str1 == str2
  - str1 < str2
- GameMaker GML
  - str1 = str2

Asymptotic Notation

How to measure computation time as the number of items to be sorted increases?

- **Θ - Theta or Big Theta**
  - Asymptotic upper and lower bounds
- **O - Big O**
  - Asymptotic upper bound only
- **Ω - Big Omega**
  - Asymptotic lower bound only
- Each can be applied to best, worst, and average case run times
- Also o (Little O) and ω (Little Omega) for not asymptotically tight

Running Time

Most common, from shortest to longest run time

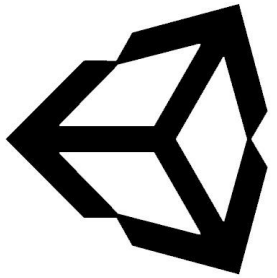
- $\Theta(1)$ - constant
- $\Theta(n)$ - linear
- $\Theta(\lg n)$ - logarithmic
- $\Theta(n^2)$ - quadratic
- $\Theta(n^c)$ - polynomial
- $\Theta(2^n)$ - exponential

n - number of items
c - constant

Sorting Algorithms

- Bubble Sort
  - For each position, compare the current item with the next item; Switch if larger
- Insertion Sort
  - For each item, compare with all previous items and insert into correct position
- Heapsort
  - Running time: $O(\mathrm{n}\ lg\ \mathrm{n})$
- Merge Sort
- Quicksort
  - Average case: $\Theta(\mathrm{n}\ \mathrm{lg}\ \mathrm{n})$
  - Worst Case: $\Theta(\mathrm{n}^2)$
- Comparison sorts
  - Counting sort
  - Radix sort
  - Bucket sort

Sorting C# (Unity) integer array



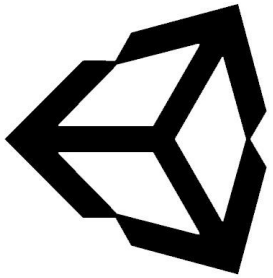
- Add **using System;** to import section
- Use **Array.Sort(MyArray);**

```
using System;
```

```
private void displaySortedInts() {
    int[] iArray = { 75, 59, 11, 35, 3, 10, 2021 };

    textOutputColumns[0].text = "";
    textOutputColumns[1].text = "";
    textOutputColumns[2].text = "";
    foreach (int iValue in iArray) {
        textOutputColumns[0].text += string.Format("{0}\n", iValue);
    }

    Array.Sort(iArray);

    foreach (int iValue in iArray) {
        textOutputColumns[1].text += string.Format("{0}\n", iValue);
    }

}
```

Unsorted	Sorted
75	3
59	10
11	11
35	35
3	59
10	75
2021	2021

Sorting C# (Unity) string array



- Same as integer array
  - Add **using System;** to import section
  - Use **Array.Sort(MyArray);**
  - C# uses case insensitive comparison
    - 'a' == 'A'

```
private void displaySortedStrings() {
    string[] strArray = { "Orange", "Banana", "grapes",
        "WATERMELON", "LEMON", "apple"};

    textOutputColumns[0].text = "";
    textOutputColumns[1].text = "";
    textOutputColumns[2].text = "";
    foreach (string strValue in strArray) {
        textOutputColumns[0].text += strValue + "\n";
    }

    Array.Sort(strArray);

    foreach (string strValue in strArray) {
        textOutputColumns[1].text += strValue + "\n";
    }

}
```

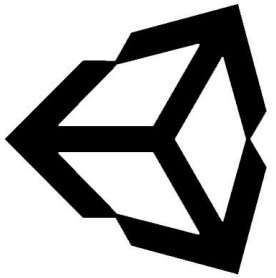
Unsorted	Sorted
Orange	apple
Banana	Banana
grapes	grapes
WATERMELON	LEMON
LEMON	Orange
apple	WATERMELON

Sorting C# (Unity) object List

- Have class extend IComparable<*MyClass*>
- Add **using System;** to import section
- Implement **public int CompareTo(*MyClass*)**
- Example: Weapon class with name (string), cost (int), and attack (int) values

```
using System;

public class Weapon : IComparable<Weapon> {

public int CompareTo(Weapon weapon) {
    int iReturn = 0;
    int iType = 1;

    switch (iType) {
        case 0:
            //Compare by cost
            if (iCost > weapon.iCost) {
                iReturn = 1;
            } else if (iCost < weapon.iCost) {
                iReturn = -1;
            }
            break;
        case 1:
            //Compare by name (C# Compare handles case conversion)
            iReturn = string.Compare(strName, weapon.strName);
            break;
        case 2:
            //Compare by attack
            if (iAttack > weapon.iAttack) {
                iReturn = 1;
            } else if (iAttack < weapon.iAttack) {
                iReturn = -1;
            }
            break;
    }
    return iReturn;
}
```

Sorted C# (Unity) object List



Unsorted

Name	Cost	Attack
Club	640	24
Bamboo Pole	100	8
Hand Axe	980	15
Broad Sword	150	40
Flame Sword	560	35
Copper Sword	180	15

Sort by Name

Name	Cost	Attack
Bamboo Pole	100	8
Broad Sword	150	40
Club	640	24
Copper Sword	180	15
Flame Sword	560	35
Hand Axe	980	15

Sort by Cost

Name	Cost	Attack
Bamboo Pole	100	8
Broad Sword	150	40
Copper Sword	180	15
Flame Sword	560	35
Club	640	24
Hand Axe	980	15

Sort by Attack

Name	Cost	Attack
Bamboo Pole	100	8
Hand Axe	980	15
Copper Sword	180	15
Club	640	24
Flame Sword	560	35
Broad Sword	150	40

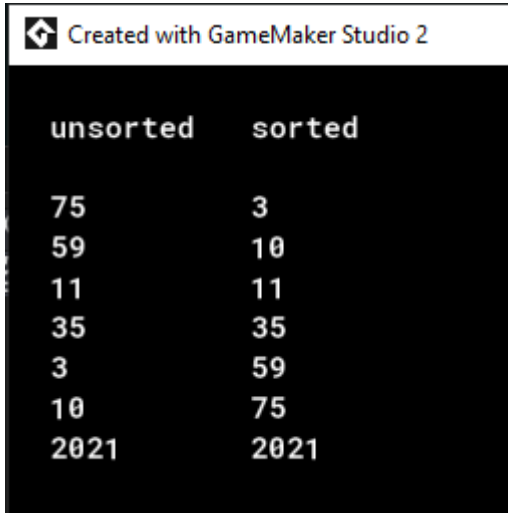
Sorting GameMaker (GML) integer array

- Use array_sort to sort an array
- Using array_copy for display purposes
  - The unsorted array is lost after it is sorted

Create

```
4  myArray = array_create(0)
5  myArray[0] = 75
6  myArray[1] = 59
7  myArray[2] = 11
8  myArray[3] = 35
9  myArray[4] = 3
10 myArray[5] = 10
11 myArray[6] = 2021
12
13 myArraySorted = array_create(0)
14
15 array_copy(myArraySorted,0,myArray,0,array_length(myArray))
16
17 array_sort(myArraySorted, true)
```

Output



Draw GUI Begin

```
3  draw_set_color(c_white)
4
5  draw_text(20, 20, "unsorted")
6  for (i = 0; i < array_length(myArray); i += 1) {
7      draw_text(20, (i + 3) * 20 , myArray[i])
8  }
9
10 draw_text(120, 20, "sorted")
11 for (i = 0; i < array_length(myArraySorted); i += 1) {
12     draw_text(120, (i + 3) * 20 , myArraySorted[i])
13 }
```

KNOX GAME DESIGN

Sorting GameMaker (GML) string array

- array_sort is case sensitive
  - Capital letters will be sorted before lowercase letters
  - Can define a sort function to do case insensitive sort

Create

```
myArray = array_create(0)
myArray[0] = "Orange"
myArray[1] = "Banana"
myArray[2] = "grapes"
myArray[3] = "WATERMELON"
myArray[4] = "LEMON"
myArray[5] = "apple"

myArraySorted = array_create(0)
array_copy(myArraySorted,0,myArray,0,array_length(myArray))
array_sort(myArraySorted, true)

case_insensitive_sort = function(str1, str2) {
    if (string_upper(str1) < string_upper(str2)) {
        return -1
    } else if (string_upper(str1) > string_upper(str2)) {
        return 1
    } else {
        return 0
    }

}

myArraySortedInsensitive = array_create(0)
array_copy(myArraySortedInsensitive,0,myArray,0,array_length(myArray))
array_sort(myArraySortedInsensitive, case_insensitive_sort)
```

Draw GUI Begin

```
draw_set_color(c_white)

draw_text(220, 20, "unsorted")
for (i = 0; i < array_length(myArray); i += 1) {
    draw_text(220, (i + 3) * 20 , myArray[i])
}

draw_text(320, 20, "sorted")
for (i = 0; i < array_length(myArraySorted); i += 1) {
    draw_text(320, (i + 3) * 20 , myArraySorted[i])
}

draw_text(420, 20, "sorted insensitive")
for (i = 0; i < array_length(myArraySortedInsensitive); i += 1) {
    draw_text(420, (i + 3) * 20 , myArraySortedInsensitive[i])
}
```

Output

unsorted	sorted	sorted insensitive
Orange	Banana	apple
Banana	LEMON	Banana
grapes	Orange	grapes
WATERMELON	WATERMELON	LEMON
LEMON	apple	Orange
apple	grapes	WATERMELON

Sorting GameMaker (GML) objects

- Use custom sort function for each sort category

Create array of Weapon objects

```
myArray = array_create(0)

weapon = instance_create_layer(0,0,0,Weapon)
weapon.strName = "Club"
weapon.iCost = 640
weapon.iAttack = 24
myArray[0] = weapon

weapon = instance_create_layer(0,0,0,Weapon)
weapon.strName = "Bamboo Pole"
weapon.iCost = 100
weapon.iAttack = 8
myArray[1] = weapon

weapon = instance_create_layer(0,0,0,Weapon)
weapon.strName = "Hand Axe"
weapon.iCost = 980
weapon.iAttack = 15
myArray[2] = weapon
```

Sort by cost

```
myArraySorted1 = array_create(0)
array_copy(myArraySorted1,0,myArray,0,array_length(myArray))

weapon_cost_sort = function(weapon1, weapon2) {
    if (weapon1.iCost < weapon2.iCost) {
        return -1
    } else if (weapon1.iCost > weapon2.iCost) {
        return 1
    } else {
        return 0
    }

}

array_sort(myArraySorted1, weapon_cost_sort)
```

Sort by attack

```
myArraySorted2 = array_create(0)
array_copy(myArraySorted2,0,myArray,0,array_length(myArray))

weapon_attack_sort = function(weapon1, weapon2) {
    if (weapon1.iAttack < weapon2.iAttack) {
        return -1
    } else if (weapon1.iAttack > weapon2.iAttack) {
        return 1
    } else {
        return 0
    }

}

array_sort(myArraySorted2, weapon_attack_sort)
```

Draw GUI Begin

```
draw_set_color(c_white)

draw_text(20, 320, "unsorted")
for (i = 0; i < array_length(myArray); i += 1) {
    draw_text(20, (i + 3) * 20 + 300 , myArray[i].strName + "," + string(myArray[i].iCost) + ", " +  + string(my
}

draw_text(220, 320, "sorted by cost")
for (i = 0; i < array_length(myArraySorted1); i += 1) {
    draw_text(220, (i + 3) * 20 + 300 , myArraySorted1[i].strName + "," + string(myArraySorted1[i].iCost) + ", "
}

draw_text(420, 320, "sorted by attack")
for (i = 0; i < array_length(myArraySorted2); i += 1) {
    draw_text(420, (i + 3) * 20 + 300 , myArraySorted2[i].strName + "," + string(myArraySorted2[i].iCost) + ", "
}
```

Output

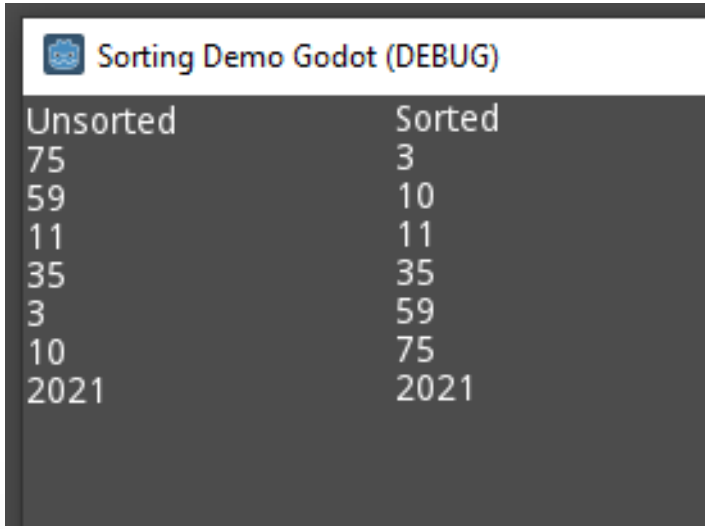
```
unsorted              sorted by cost        sorted by attack

Club,640, 24          Bamboo Pole,100, 8    Bamboo Pole,100, 8
Bamboo Pole,100, 8    Board Sword,150, 40   Hand Axe,980, 15
Hand Axe,980, 15      Copper Sword,180, 15  Copper Sword,180, 15
Board Sword,150, 40   Flame Sword,560, 35   Club,640, 24
Flame Sword,560, 35   Club,640, 24          Flame Sword,560, 35
Copper Sword,180, 15  Hand Axe,980, 15      Board Sword,150, 40
```

Sorting Godot (GDScript) integer array

- Use MyArray.sort()
- MyArray.duplicate() used for display purposes

```gdscript
extends Node

var unsorted_label
var sorted_label
var myArray
var myArraySorted

# Called when the node enters the scene tree for the fi
func _ready():
    unsorted_label = get_node("Control/UnsortedLabel")
    sorted_label = get_node("Control/SortedLabel")

    sortIntegers()
    displayArrays()

func sortIntegers():
    myArray = [75, 59, 11, 35, 3, 10, 2021]

    myArraySorted = myArray.duplicate()
    myArraySorted.sort()

func displayArrays():
    var strUnsorted = "Unsorted\n"
    for item in myArray:
        strUnsorted += str(item) + "\n"
    unsorted_label.set_text(strUnsorted)

    var strSorted = "Sorted\n"
    for item in myArraySorted:
        strSorted += str(item) + "\n"
    sorted_label.set_text(strSorted)

```

Output



KNOX GAME DESIGN

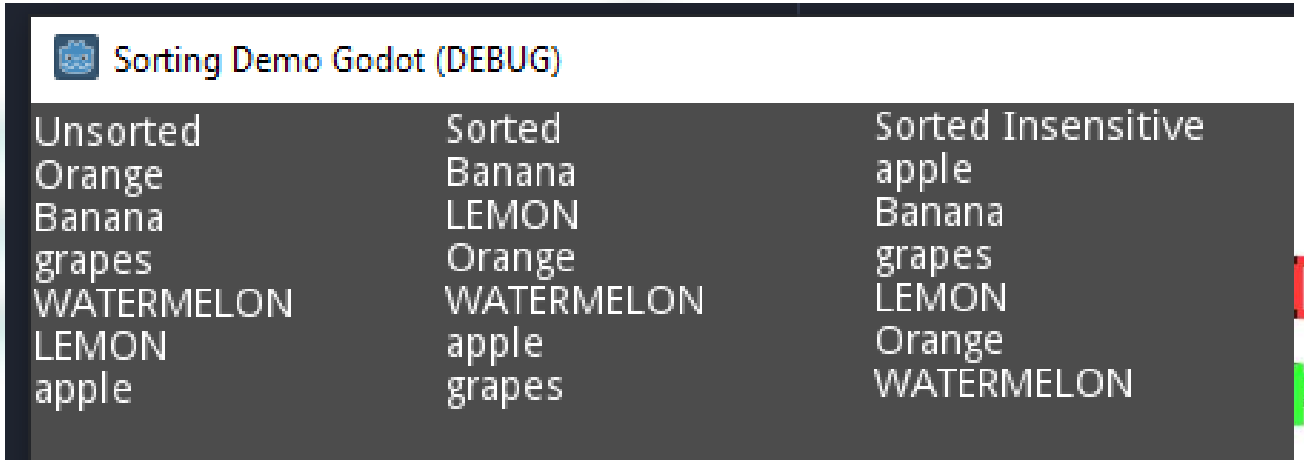
Sorting Godot (GDScript) string array

- Case sensitive sort
- Use **sort_custom(MyArray, sort_func)** to define custom sort function
  - Custom sort function must return true (value is less) or false (value is greater)

```gdscript
func sortStrings():
    myArray = ["Orange", "Banana", "grapes", "WATERMELON", "LEMON", "apple"]

    myArraySorted = myArray.duplicate()
    myArraySorted.sort()

    myArraySortedInsensitive = myArray.duplicate()
    myArraySortedInsensitive.sort_custom(self, "sort_insensitive")

func sort_insensitive(str1, str2):
    if (str1.to_upper() < str2.to_upper()):
        return true
    else:
        return false
```

```gdscript
func displayArrays():
    var strUnsorted = "Unsorted\n"
    for item in myArray:
        strUnsorted += str(item) + "\n"
    unsorted_label.set_text(strUnsorted)

    var strSorted = "Sorted\n"
    for item in myArraySorted:
        strSorted += str(item) + "\n"
    sorted_label.set_text(strSorted)

    var strSortedInsensitive = "Sorted Insensitive\n"
    for item in myArraySortedInsensitive:
        strSortedInsensitive += str(item) + "\n"
    sorted_insensitive_label.set_text(strSortedInsensitive)
```

```gdscript
extends Node

var unsorted_label
var sorted_label
var sorted_insensitive_label
var myArray
var myArraySorted
var myArraySortedInsensitive

# Called when the node enters the scene tree for the first time.
func _ready():
    unsorted_label = get_node("Control/UnsortedLabel")
    sorted_label = get_node("Control/SortedLabel")
    sorted_insensitive_label = get_node("Control/SortedInsensitiveLabel")

#   sortIntegers()
    sortStrings()
    displayArrays()
```



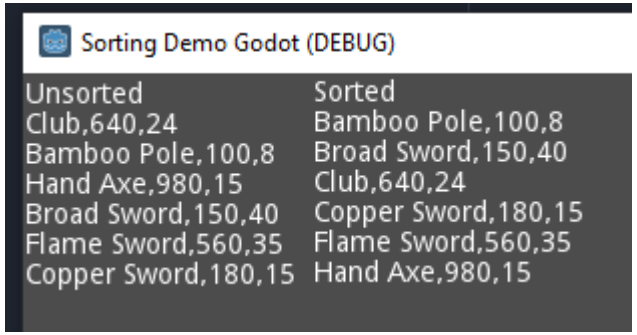
Sorting Godot (GDScript) objects

- Use **sort_custom(self, "<function>")** and define sorting function

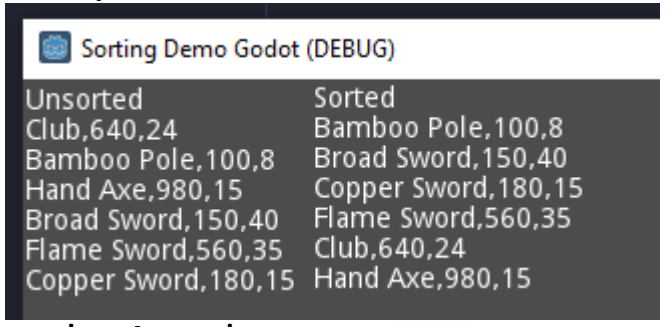
```gdscript
func createWeapons():
    var weapon
    
    myArray = []

    #create the Weapon.tscn scene by making a new no
    #"Save Branch as Scene"
    var weaponScene = load("res://Weapon.tscn")

    weapon = weaponScene.instance()
    weapon.strName = "Club"
    weapon.iCost = 640
    weapon.iAttack = 24
    myArray.append(weapon)

    weapon = weaponScene.instance()
    weapon.strName = "Bamboo Pole"
    weapon.iCost = 100
    weapon.iAttack = 8
    myArray.append(weapon)

    weapon = weaponScene.instance()
    weapon.strName = "Hand Axe"
    weapon.iCost = 980
    weapon.iAttack = 15
    myArray.append(weapon)
```

```gdscript
func sortWeapons():
    myArraySorted = myArray.duplicate()
    myArraySorted.sort_custom(self, "sort_weapon_name")
    #myArraySorted.sort_custom(self, "sort_weapon_cost")
    #myArraySorted.sort_custom(self, "sort_weapon_attack")


func sort_weapon_name(weapon1, weapon2):
    if (weapon1.strName.to_upper() < weapon2.strName.to_upper()):
        return true
    else:
        return false

func sort_weapon_cost(weapon1, weapon2):
    if (weapon1.iCost < weapon2.iCost):
        return true
    else:
        return false

func sort_weapon_attack(weapon1, weapon2):
    if (weapon1.iAttack < weapon2.iAttack):
        return true
    else:
        return false
```

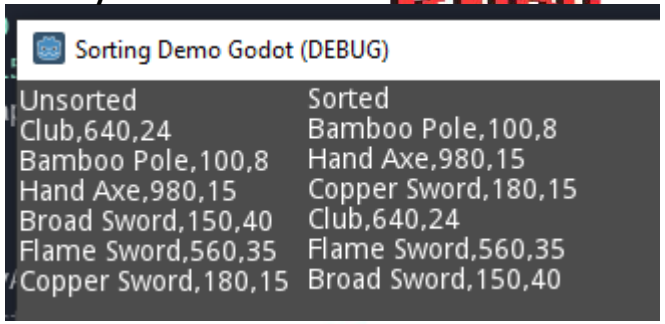
Sort by Name



Sort by Cost

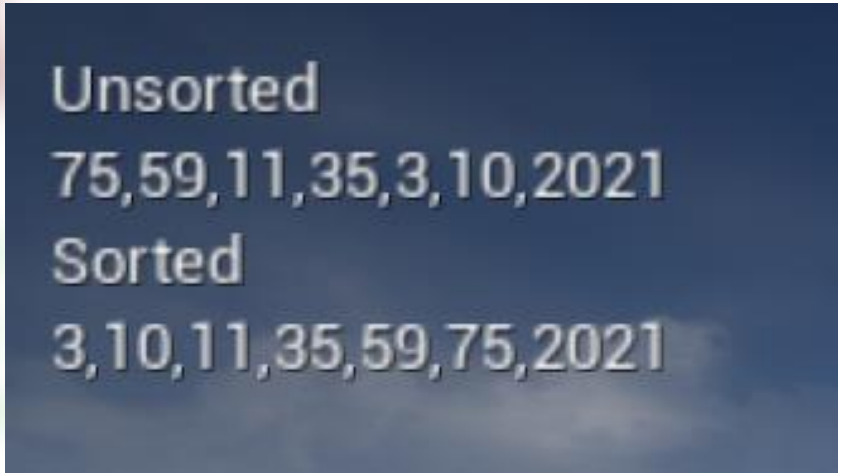


Sort by Attack



Sorting C++ (Unreal) integer array

- Use TArray instead of std vector
- Use TArray.Sort() to sort integers




```cpp
void USortingDemo::sortIntegers() {
    TArray<int> MyArray = { 75, 59, 11, 35, 3, 10, 2021 };

    displayString("Unsorted");
    displayArray(MyArray);

    displayString("Sorted");
    MyArray.Sort();
    displayArray(MyArray);
}
```

```cpp
void ADemoManager::displayArray(TArray<int> array) {
    FString strOutput = "";

    int i;
    for (i = 0; i < array.Num(); i++) {
        strOutput += FString::FromInt(array[i]);
        if (i < array.Num() - 1) {
            strOutput += ",";
        }
    }

    GEngine->AddOnScreenDebugMessage(-1, 60.0f, FColor::White, strOutput, false, FVector2D
      (2.0f, 2.0f));
}
```

```cpp
void ADemoManager::displayString(FString str) {
    GEngine->AddOnScreenDebugMessage(-1, 60.0f, FColor::White, str, false, FVector2D(2.0f,
      2.0f));
}
```

Sorting C++ (Unreal) string array

- Use FString instead of string
- Case insensitive sort by default
- Use TArray.Sort();
- Overload display method to display TArray of FStrings

```
void USortingDemo::sortStrings() {
    TArray<FString> MyArray = { "Orange", "Banana", "grapes", "WATERMELON", "LEMON", "apple" };

    displayString("Unsorted");
    displayArray(MyArray);

    displayString("Sorted");
    MyArray.Sort();
    displayArray(MyArray);
}
```

Sorting C++ (Unreal) object array

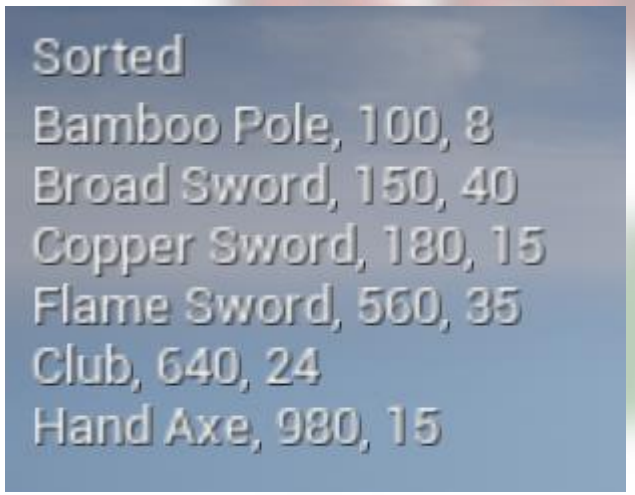
- Store pointers to objects in **TArray <MyClass *>MyArray**
- Use **Algo::SortBy(MyArray, &MyClass::myvariable);**
  - Change the variable parameter to change the sorting value
  - Can specify a third parameter with operator function, such as sort backwards
  - MyArray.Sort() *may* be possible with overloading the < operator
- Remember to #include the header file of the class to be sorted. Class declaration may also be needed

```
void ADemoManager::sortObjects() {
    TArray <AWeapon *>MyArray;

    MyArray.Add(createWeapon("Club", 640, 24));
    MyArray.Add(createWeapon("Bamboo Pole", 100, 8));
    MyArray.Add(createWeapon("Hand Axe", 980, 15));
    MyArray.Add(createWeapon("Broad Sword", 150, 40));
    MyArray.Add(createWeapon("Flame Sword", 560, 35));
    MyArray.Add(createWeapon("Copper Sword", 180, 15));

    displayString("Unsorted");
    displayArray(MyArray);

    Algo::SortBy(MyArray, &AWeapon::strName); //sort by name
//  Algo::SortBy(MyArray, &AWeapon::iCost); //sort by cost
//  Algo::SortBy(MyArray, &AWeapon::iAttack); //sort by attack

    //Sort in reverse
    //Algo::SortBy(MyArray, &AWeapon::iCost, TGreater<>());

    displayString("Sorted");
    displayArray(MyArray);
}
```

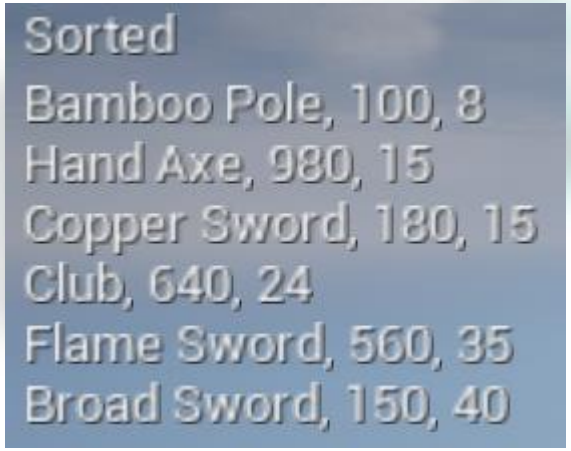
```
AWeapon *ADemoManager::createWeapon(FString in_strName, int in_iCost, int in_iAttack) {
    AWeapon *weapon;
    FActorSpawnParameters spawninfo;

    weapon = GetWorld()->SpawnActor<AWeapon>(FVector(0.0f), FRotator(0, 0, 0), spawninfo);
    weapon->strName = in_strName;
    weapon->iCost = in_iCost;
    weapon->iAttack = in_iAttack;

    return weapon;
}
```

Sorting C++ (Unreal) object array

Sort by Cost



Sort by Attack



Sorting Lua (Pico-8) integer table

- No built in method to sort a table
- Custom insertion sort function

```
MYTABLE = { 75, 59, 11, 35, 3,
10, 2021 };

PRINT "UNSORTED"
PRINTTABLE(MYTABLE)
PRINT ""

PRINT "SORTED"
SORT(MYTABLE)
PRINTTABLE(MYTABLE)
```

```
FUNCTION PRINTTABLE(T)
  FOR V IN ALL(T) DO
    PRINT(V)
  END
END

FUNCTION SORT(T)
  FOR I=1, #T DO
    LOCAL J = I
    WHILE J > 1 AND T[J-1] > T[J] DO
      T[J],T[J-1] = T[J-1],T[J]
      J = J - 1
    END
  END
END
```

SORTINTS.P8 (PICO-8)

```
> RUN
UNSORTED
75
59
11
35
3
10
2021

SORTED
3
10
11
35
59
75
2021
```

KNOX GAME DESIGN

Sorting Lua (Pico-8) string table

- Same custom insertion sort function works
- Case doesn't matter since everything in Pico-8 is one case

```
MYTABLE = { 'ORANGE', 'BANANA',
'GRAPES', 'WATERMELON', 'LEMON',
'APPLE' };

PRINT 'UNSORTED'
PRINTTABLE(MYTABLE)
PRINT ''

PRINT 'SORTED'
SORT(MYTABLE)
PRINTTABLE(MYTABLE)
```

```
> RUN
UNSORTED
ORANGE
BANANA
GRAPES
WATERMELON
LEMON
APPLE

SORTED
APPLE
BANANA
GRAPES
LEMON
ORANGE
WATERMELON
```

KNOX GAME DESIGN

Sorting Lua (Pico-8) object table

- Change the custom sort function to compare object values (name/cost/attack)

```
FUNCTION SORT(T)
  FOR I=1, #T DO
    LOCAL J = I
    WHILE J > 1 AND
      T[J-1].NAME > T[J].NAME DO
      T[J],T[J-1] = T[J-1],T[J]
      J = J - 1
    END
  END
END
```

```
FUNCTION PRINTTABLE(T)
  FOR V IN ALL(T) DO
    PRINT(V.NAME .. ',' ..
          V.COST .. ',' ..
          V.ATTACK)
  END
END
```

```
MYTABLE = { }

WEAPON = { NAME = 'CLUB',
  COST = 640, ATTACK = 24 }
ADD(MYTABLE, WEAPON)
WEAPON = { NAME = 'BAMBOO POLE',
  COST = 100, ATTACK = 8 }
ADD(MYTABLE, WEAPON)
WEAPON = { NAME = 'HAND AXE',
  COST = 980, ATTACK = 15 }
ADD(MYTABLE, WEAPON)
WEAPON = { NAME = 'BROAD SWORD',
  COST = 150, ATTACK = 40 }
ADD(MYTABLE, WEAPON)
WEAPON = { NAME = 'FLAME SWORD',
  COST = 560, ATTACK = 35 }
ADD(MYTABLE, WEAPON)
WEAPON = { NAME = 'COPPER SWORD'
  COST = 180, ATTACK = 15 }
```

```
> RUN
UNSORTED
CLUB,640,24
BAMBOO POLE,100,8
HAND AXE,980,15
BROAD SWORD,150,40
FLAME SWORD,560,35
COPPER SWORD,180,15

SORTED BY NAME
BAMBOO POLE,100,8
BROAD SWORD,150,40
CLUB,640,24
COPPER SWORD,180,15
FLAME SWORD,560,35
HAND AXE,980,15
```

```
SORTED BY COST
BAMBOO POLE,100,8
BROAD SWORD,150,40
COPPER SWORD,180,15
FLAME SWORD,560,35
CLUB,640,24
HAND AXE,980,15
```

```
SORTED BY ATTACK
BAMBOO POLE,100,8
HAND AXE,980,15
COPPER SWORD,180,15
CLUB,640,24
FLAME SWORD,560,35
BROAD SWORD,150,40
```

References

- Sound of sorting
  - https://panthema.net/2013/sound-of-sorting/
  - https://www.youtube.com/watch?v=kPRA0W1kECg
- Big Theta explained
  - https://stackoverflow.com/questions/10376740/what-exactly-does-big-%d3%a8-notation-represent#12338937
- IComparable from Microsoft
  - https://docs.microsoft.com/en-us/dotnet/api/system.icomparable-1?view=net-5.0
- GameMaker array_sort
  - https://manual.yoyogames.com/GameMaker_Language/GML_Reference/Variable_Functions/array_sort.htm
- Godot Array
  - https://docs.godotengine.org/en/latest/classes/class_array.html?highlight=array
- Unreal Engine TArray sort
  - https://docs.unrealengine.com/en-US/ProgrammingAndScripting/ProgrammingWithCPP/UnrealArchitecture/TArrays/index.html
- Pico-8 table sorting
  - https://www.lexaloffle.com/bbs/?pid=50453

KNOX GAME DESIGN