C++ (Weeks 6–9) Cheat Sheet

by Lipsum via cheatography.com/151963/cs/40909/

Functions

```
// Function declaration
return _type functi on_ nam -
e(p ara met ers);
// Function definition
return _type functi on_ nam -
e(p ara meters) {
       // Function body
       // Code here
        return result; //
Optional
}
```

Search Algorithms

Linear Search:

Linear search is a simple algorithm that scans through an array one element at a time, comparing each element with the target value. It continues this process until it finds the target or reaches the end of the array. Linear search is straightforward but not the most efficient for large datasets.

Binary Search:

Binary search works on a sorted array and follows a divide-and-conquer approach. It starts with the middle element and compares it to the target. If they match, the search is successful. If the target is smaller, it repeats the process on the left half of the array; if the target is larger, it looks in the right half. This process continues until the target is found or the search range becomes empty.

Sorting Algorithms

Bubble Sort: Bubble sort repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. It continues to do this until no more swaps are needed, indicating that the list is sorted. Bubble sort has poor performance for large lists and is mainly used for educational purposes.

Selection Sort: Selection sort divides the list into a sorted and an unsorted region. It repeatedly selects the minimum element from the unsorted region and moves it to the end of the sorted region. The process continues until the entire list is sorted.

Arrays

```
// Declare an array
data_type array_ nam e[s ize];
// Initialize an array
data_type array_ name[] =
{value1, value2, value3};
// Access elements
element = array_ nam e[i ndex];
// Modify elements
array_ nam e[i ndex] =
new_value;
```

References

```
// Declare a reference
data_t ype & refere nce _name =
origin al_ var iable;
// Use reference
refere nce _name = new_value; //
Modifies the original variable
```

Pointers

```
// Declare a pointer
data_type* pointe r_name;
// Initialize pointer
pointe r_name = &v ari able;
// Derefe rence pointer
value = *point er_ name;
// Pointer arithmetic
pointe r_n ame++; // Moves to
the next element
// Dynamic memory allocation
data_type* dynami c_ptr = new
data_type;
delete dynami c_ptr; // Release
memory
```

Structures

```
// Declare a structure
struct StructName {
        dat a_type member1;
        dat a_type member2;
       // ...
};
// Create an instance
StructName instan ce_ name;
// Access members
instan ce_ nam e.m ember1 =
value;
// Nested structures
struct Nested Struct {
       int inner_ member;
};
struct OuterS truct {
       int outer_ member;
        Nes ted Struct nested;
};
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com