C++ Exam 1 Cheat Sheet

by skinker via cheatography.com/201011/cs/42497/

Vectors and Arrays

Method/Types	Vector	Array
Create	`vector <type> var(num)` or `vector <type> var{element, element…}`	`type var[num]` or `type var[] = {element, element…}`
Find number of elements	`var.size()`	`sizeof (var)/ sizeof (var[0])`
Access an element	`var.at (index)`	`var[index]`
Modify an element	`var.at (index) = element`	`var[index] = element`
Add an element	`var.push_back(element)` or `var.insert(var.begin()+index, element)`	n/a
Remove an element	`var.pop_back()` or `var.erase (var.begin() +index)`	n/a
for loop	`for (int i = 0; i < var.size(); i++) {cout << var.at (i);}`	`for (int i = 0; i < sizeof (var)/ sizeof (var[0]); i++) {cout << var[i];}`
Enhanced for loop	`for (type i : var) {cout << i}`	`for (type i : var) {cout << i}`
Common compatible types	integer, double, boolean, strings	int, double, boolean, strings

The Big Three

Copy Constructor *Used to construct an object from another, existing object*

```
className (const className &original) {
    // copy over everything from
    original to this }
```

Copy Assignment Operator *Used to copy one object into another object*

```
ClassName & operator= (const ClassName &
original) {
    //same as copy constructor
    return *this }
```

Destructor *Used when an object is destroyed—when it falls out of scope, or when delete is called on a pointer to an object*

```
~ExampleClass(){
    delete item_name
    //or more complicated code }
```

Deep and Shallow Copy

Vectors (cont)

In order to use vectors, you must include `#include <vector>` in the header of your program.

vector structure : `vector <int> vec_name(3);`

`name.push_back( data)` adds whatever is in the parantheses to the end of the vector

To add an element to a specific index in the vector: `vec_name.insert (vector.begin()+1, 50);` adds 50 to index 1

To remove an element from the end of a vector: `vec_name.pop_back()`

To erase a specific index: `vec_name.erase( vec_name.begin()+1);` erases the element at the index 1

`vec_name.at(3)` accesses the element at index 3, use this to modify specific elements

Simple (Return types, Loops, & Conditionals)

Pointers and references functions

Deep Copy

copies the data itself - allocate more space and clean it up (use destructors)

Deep copy takes two steps:
1. Allocate space for the duplicate data
`int* deep = new int[5];`
2. Copy the data values from the original location

```
for (int i = 0; i < 5; i++) {
    deep[i] = ptr[i]; }
```

Shallow Copy

copies pointers

Create the array

```
int* ptr = new int[5];
for (int i = 0; i < 5; i++) {
    ptr[i] = i * 2;
    // Set values to 0, 2, 4, 6, 8
```

Shallow copy the array
`int* shallow = ptr;`

Vectors and Arrays

What does `.push_ back()` do in a vector?

What does `.at()` do in a vector?

What does `[ ]` do in a vector?

How do you sort an array/vector?

Vectors

Vectors are dynamic, meaning you can make changes to them while the program is running. Vectors are particularly helpful when you don't know how large your collection of elements will become.

How do you create functions in C++?

> `returnType Functi on(int var_na me){}`
> *put `ClassN ame::` beforehand if working in source file*

What are the return types of these functions?

> void, string, int, unsigned int, vector<data type> (usually done with pointer), boolean, array (usually done with pointer)

Is it possible to return any data type?

> yes, as long as it is declared in the beginning of the function

How do you write if conditionals and for loops in C++?

```
if(con dit ion){}
 for(int i = 0; i < condition; i++){}
```

What is the difference between if, else if, and while loops?

What are some of the algebraic and comparison operators in C++?

> +, -, =, <, >, ==, !=, &&, ||

How do you use them with different data types?

> must be used to compare 2 of the same data type, unless you use an operator overload

What are pointers?

> Pointers tell us the location of something else. A pointer is a variable that holds the memory address of another variable. A pointer needs to be dereferenced with the * operator to access the memory location it points to. `int* name` creates a pointer to an int

What are references?

> References act as a stand-in (or alias) for another variable. A reference is the variable that it references. Any changes to a reference change the original. They don't use memory addresses, no dereferencing.

How do you pass objects by references and pointers?

> using `FuncName(int &var_name){}

What do you need to take into account when creating a function that intends to swap two integers, a and b?

> make sure to pass by reference so the values can be changed

What does `->` do in pointers?

> Indirect membership operator (For pointers to objects)

By **skinker**
cheatography.com/skinker/

Not published yet.
Last updated 23rd February, 2024.