Cheatography

C++ 101 Midterm 3 Cheat Sheet by sadieweaver via cheatography.com/86436/cs/20098/

istream ostream

friend ostream& operator << (ostream& out, room& r) { out <<</pre> "Width:" << r.width << ", Length:" << r.length; return out; }

friend istream& operator>>(istream& in, room& r) { cout << "Enter width: " << endl; in >> r.width; cout << "Enter length: " < endl; in >> r.length; return in; }

Strings vs C-strings (ch8)		
#include	#include <cstring></cstring>	
<string></string>		
Reading in:		
	>> stops reading at the	
	first space	

	first space	
string	char input[100]	
input;	cin.getline(input,	
getline(cin,	100);	
input);		
string s1;	empty c-string: char	
	s[20] = "";	

Returning:

returning c-strings: return type of pointer. char* funct() {...return...}

for both: make sure that if you return something in a function if it's a variable defined in the function that is was defined dynamically or is static.

Operations:

s.length() or	strlen(s) - s is a c-
s.size()	string cannot use .length() or .size()
s2 = s1;	<pre>strcpy(s2, s1) - copy s1 to s2</pre>
s1 == s2;	<pre>strcmp(s1, s2) == 0; - tests for equality</pre>

Strings vs C-strings (ch8) (cont)

```
strcat(char* s1, const
s1
+=
       char* s2);
s3
string s to double stod(s);
```

char* s to double: strtod(s, nullptr);

Functions:

Random:

c-strings have a null termination character '/0'. So to define a cstring to hold 20 it's char input [21]; same c++ doesn't check indexes to

thing validate if something is within as > bounds int n = 3; cout << (char)(n + '0') << endl; prints 3. char c = 'D'; cout << (c - 'A') << endl; prints 3. char name[100]; char* get_name() { . . . return ____ } return name

Constructors

Default: foo() no arguments foo f1; or foo* f2 = new foo; **Conversion:** foo(int i) one argument to

be turned into the class object foo f1(3); or foo* f2 = new foo(3); General: foo(int x, int y) anything

with more than one

Copy: foo(foo& f) pointer argument

Move: foo (foo&& f) double pointer

Constructor and Initializer List Examples

Choose the best C++ class named person that has: A private member field for the person's first name; A private member field for the person's last name; A private member field for the person's age; and a public constructor:

```
class Person{ string first;
string last; int age;
public: person(string f,
string 1, int a) : first(f),
last(1), age(a) {} };
```

UML: Student

-name: string -gpa:double +Student(n: string, g: double) class Student {private: string name; double gpa; public: Student(string n, double g) : name(n), gpa(g) {} };

Write a single constructor that works as default, conversion, and general: UML fraction -numerator:int -denominator:int +fraction(n: int, d:int) (default values: n = 0, d = 1)fraction(int n = 0, int d = 1): numerator(n), denominator(d) {}

UML: Foo -count:int +running: bool (set running to true) +Foo(a count: int) -my_helper(arg: int) : char class Foo {private: int count: char my_helper(int arg) public: bool running = true; Foo(int a_count) : count(a_c-

Extras

ount) {} };

Only technical difference between structures and classes:

features in classes are private by default, features in structures are public by default

Constructors name is

the same as the name of the class

By sadieweaver

Published 27th July, 2019. Last updated 14th August, 2019. Page 1 of 3.

Sponsored by Readable.com

Measure your website readability! https://readable.com

cheatography.com/sadieweaver/

Cheatography

Facts

+: public

-: private

#: protected

underlined:static

Auto: Stack

Dynamic: Heap

defining functions outside of a class:return-type class::function-name(arguments) with a prototype included in class file.

Member/Friend Summary				
member/nonmbr	Implicit Args	Explicit Args		
Unary Member:	1	0		
Unary Friend:	0	1		
Binary Mbr:	1	1		
Binary Friend:	0	2		

mbr/ nonmbr in and out a class

Mbr defined in the class

foo operator+(foo f) {...}

Mbr defined out the class

```
foo foo::operator+(foo f) {...}
```

Nonmbr defined in the class

friend foo operator+(foo f1, foo f2) {...}

Nonmbr defined out the class

```
foo operator+(foo f1, foo f2) {...}
```

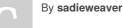
Command-line

main function definition to allow a program to access command line arguments:

```
main(int argc, char* argv[])
```

A program is named "my_program" and is executed from the command line as my_program file1 file2 file3 file4 If the program is written in C++ and the arguments are passed in to main, what is the value of argc and what is stored in argv[2]?

argc = 5, argv[2] = file2



Published 27th July, 2019. Last updated 14th August, 2019. Page 2 of 3.

cheatography.com/sadieweaver/

Sponsored by **Readable.com** Measure your website readability! https://readable.com