

### <fstream>

ifstream

ofstream

### <cstring>

strcpy(destination, source) incl. sentinel char

strcat(destination, source)

strcmp(str1, str2); if = 0 then equal, if >0 then str1 is greater else str1 is smaller

strlen(char\* str) returns int; not include sentinel

strncpy(dest,src, number of characters)

strncat(d,s,n); strncmp(d,s,n)

strchr(char str, char) returns a pointer (char) to the first occurrence of char in str else returns null

strstr(str1, str2) returns pointer to first occurrence of str2 in str1 (e.g. ptr = strstr(str1, "hello")

### <cmath>

abs(int) returns the absolute of the int

pow(base, exponent) returns the power

sqrt(number) returns the square root

ceil(number) rounds up; floor(number) rounds down

### <cstdlib>

atoi(str) converts str to int; atof(str) converts str to double; use with c strings (char \*)

exit(1)

x = rand() % number; srand(time(NULL)) requires <time.h>

itoa(value, array, base); value = str to be converted; array = c string array; base = 10, 2, 16, etc.

### Pointers

- Pointer is the memory address of a variable (i.e. int \* num\_ptr)

- Use typedef to declare variable type (i.e. typedef int\* num\_ptr -> num\_ptr X;)

- Create dynamic var: ptr = new type; delete dynamic var: delete ptr

- Deleting dynamic var does not delete the ptr just what is stored inside it

- Make sure to assign null to ptr after delete so it is not dangling

```
ptr_a = new (nothrow) int;
```

```
if (ptr_a == NULL) {
```

```
cout << "sorry";
```

```
exit(1); } <- ptr set to null if allocation fails
```

- array identifiers are pointers

```
int hours[5];
```

```
int* ptr;
```

```
ptr = hours; <- both point to first index
```

- hours[1] is equivalent to \*(hours+1)

- char phrase[] is equivalent to char\* phrase

```
int* number_ptr;
```

```
number_ptr = new int[10] <- dynamic array
```

```
delete [] number_ptr;
```

### Strings

- Sentinel character is '\0' and marks the end of string

- char phrase[5] = {'A', 'B'} or char phrase[] = "Hello"

- Using >> to input strings is limited because it ends at white space

- cin.getline(string\_name, 80) is often used instead

### <algorithm>

swap(var1, var2) swaps the 2 variables; also works arrays/vector

```
min(val1,val2), max(val1,val2);
```

```
find(begin, end, val); if no found return end
```

```
replace(begin,end,oldval,newval)
```

```
sort(begin, end)
```

### <cctype>

tolower(char) converts to lowercase;

toupper(char) converts to upper case

isalpha(char) returns true if char is alphabetic (0 = false)

isalnum(char) returns true if char is decimal, upper/lower

isblank(char), isspace(char) return true if char is either ' ' or \n

isupper(char), islower(char), isdigit(char) are all boolean

ispunct(char) returns true if char is punctuation

### letter to number / number to letter

```
int -> char
```

```
int number = 40;
```

```
char letter = number;
```

```
char -> int
```

```
char letter = 'c';
```

```
int number = letter;
```

notes:

ASCII for 0 = 48

ASCII for space = 32

ASCII for A = 65

ASCII for a = 97



By **zreich**

[cheatography.com/zreich/](http://cheatography.com/zreich/)

Not published yet.

Last updated 8th January, 2018.

Page 1 of 3.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

### Streams

- connect: stream.open("name"); <- connects to the beginning of the file
- disconnect: stream.close();
- stream.open("name")
- if (stream.fail()) {
- sorry + exit }
- in\_stream.get(ch) -> assigns ch the next char in the file and repositions file
- out\_stream.put(ch) -> puts ch in next position in file
- in\_stream.putback(ch) -> puts back in the file but does not alter file
- char = in\_stream.peek() assigns char the next char in file but does not move it forward
- in\_stream.get(ch)
- while (! in\_stream.eof()) {
- cout << ch;
- out\_stream.put(ch);
- in\_stream.get(ch) }
- streams must be reference & arguments in functions only

### GDB

- 'gdb' to start gdb
- Start with 'run' or 'r'
- Set breakpoint with 'break'
- Print variable\_name, &variable\_address
- 'Watch' variable\_name
- 'C' for continue
- Execute next line using 'step' or 's'
- 'Next' or 'n'
- Pressing enter repeats last command
- 'q' to quit
- 'finish' to stop execution of current function

### Loops

- Do { ... } While (bool)
- switch (selector) {
- case label1:
- statement;
- break;
- default:
- statement;
- break; }

### <array>

array\_name.begin() / array\_name.end = first and last elements in the array; also .front/.back

array\_name.size() = number of elements in the array vs. sizeof(expression/type) = bytes

array\_name.max\_size() returns max number of elements allowed in array

array\_name.empty() returns true if array size is 0 meaning no elements

array\_name.at(int) = array at position int like array\_name[int]

### <string>

getline(input\_stream, str, optional\_limit)

begin(array); end(array); .begin/.end

### Makefiles

```
OBJ = MovieList.o Movie.o NameList.o Name.o Iterator.o
CC = g++
DEBUG = -g
CFLAGS = -Wall -c $(DEBUG)
LFLAGS = -Wall $(DEBUG)

p1 : $(OBJ)
$(CC) $(LFLAGS) $(OBJ) -o p1

MovieList.o : MovieList.h MovieList.cpp Movie.h NameList.h Name.h Iterator.h
$(CC) $(CFLAGS) MovieList.cpp

Movie.o : Movie.h Movie.cpp NameList.h Name.h
$(CC) $(CFLAGS) Movie.cpp

NameList.o : NameList.h NameList.cpp Name.h
$(CC) $(CFLAGS) NameList.cpp

Name.o : Name.h Name.cpp
$(CC) $(CFLAGS) Name.cpp

Iterator.o : Iterator.h Iterator.cpp MovieList.h
$(CC) $(CFLAGS) Iterator.cpp

clean:
rm *.o *~ p1
```

exe: main.o exe.o  
(tab) g++ -g main.o exe.o -o exe

main.o: main.cpp exe.h  
(tab) g++ -Wall -c main.cpp

exe.o: exe.cpp exe.h  
(tab) g++ -Wall -c exe.cpp

### Input/Output with >> & <<

- out\_stream << 34 << ' ' would result in '3', '4', ' ' as characters in output file

- doing input\_stream >> number would take all characters together until reaching a blank space

- >> skips over blank space no matter the data type

### Arrays

- When declaring a function that passes an array use []

- When passing actual array to function no need to use []

- Array parameters are essentially reference parameters but not need to use &

- Pass the size of the array in the function as well

- Add const before array in function if you don't want it to change the array e.g. const int array\_name[]

- 2D arrays -> int array\_name[row][column]

- function(array[][column])

### Header file (function declarations)

```
#ifndef HEADERNAME_H
#define HEADERNAME_H
#include directives
Using directives
statements
#endif
```

### Another Makefile

```
OBJ = main.o tube.o helper.o
EXE = tube
CXX = g++
CXXFLAGS = -Wall -g
$(EXE) : $(OBJ)
$(CXX) $(OBJ) -o $@
%.o : %.cpp
$(CXX) $(CXXFLAGS) -c $<
main.o: tube.h
tube.o: tube.h helper.h
```

### Another Makefile (cont)

```
helper.o: helper.h  
clean:  
    rm -f $(OBJ) $(EXE)
```



By **zreich**

[cheatography.com/zreich/](http://cheatography.com/zreich/)

Not published yet.

Last updated 8th January, 2018.

Page 3 of 3.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>