**Test review exam 1 cs250 notes**

**DEBUGGING TOOLS -**

1. Be familiar with the basic debugging tools: Breakpoints, autos/locals/watch windows, call stack.

## Breakpoints

Breakpoints are where you stop in the code when you are running it, it stops running at a breakpoint the little red dot on the side of the grey bar on visual studio.

When a breakpoint is hit in debug mode the program will pause execution and take you to the IDE. From a paused state, you can step through the code execution line by line, as well as investigate the variable values.

It paused at this line.

Step into the (the down arrow one or the first one), when at a function, step into the functions execution.

Step over(the second one or the little curved arrow going to the right so this but curved 🡪), when at a function, step over it and go to the next line at the current scope.

Step out(the third one or the upwards arrow), when inside a function, finish execution and step to the line after the functions call.

# Watch/auto/locals windows

The autos pane will show you the variables used on the current or preceding line of code. The autos pane/window will show you the variables used on the current or preceding line of code.

The locals pane will show you the variables that are currently in scope. The locals pane/window will show you the variables that are currently in scope. The locals pane will show you the variables that are currently in scope. That are currently in scope, that are currently in scope, currently in scope.

The watch pane, in the watch pane, you can type in a variable name to track it throughout the entire program execution. In the watch pane you can type in a variable and track it throughout the entire program execution.

* **CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK CALL STACK ABCDEFGHIJKLMNOPQRSTUVWXYZ NOW I KNOW MY ABC’S NEXT TIME WONT YOU SING WITH ME!!!**

When your program execution is paused, the call stack will show you the functions that have been called in order to get to where you’re currently at. The call stack can be invaluable to trace the path that your program is following. Basically what called what first, starts with main on the very bottom!! Main called the one above it, the one above it called the one above the one above, and so on.

**C++ Review –**

Know how pointers work, dynamic variables, and dynamic arrays, and how to code them!!!

* Pointers

Pointers are just a type of variable that stores addresses. No matter how were using a pointer, that’s what it all boils down to. We have:

Int somenumber = 20;

Int\* ptrNum = &someNumber;

Cout << ptrNum (prints out garbage values 0x7fff8fd721bc)

Cout << \*ptrNum (prints out 20)

ptrNum points to the memory address of somenumber.

Different ways to use pointers:

//pointing to existing variable

Int a;

Int ptr = &a;

ptr = &b;

//dynamic variable

Int \* var = new int;

delete var;

TYPE\* ptr = new TYPE;

TYPE\* ptr = new TYPE;

TYPE\* ptr = new TYPE;

//dynamic array

Int \* arr = new int[100];

delete [] arr;

TYPE\* ptr = new TYPE[size];

TYPE\* ptr = new TYPE[size];

If a pointer is pointing to an object and you want to acess the original items’ member functions and varuables you need to use the arrow operator -> (as in the case with nodes linked lists…)

For example:

ptrCurrent = ptrCurrent->ptrNext;

Node\* ptrNext;

**DESIGN –**

Be familiar with how testing is used, why writing tests prior to implementing code can be a good design decision.

* How testing is used ?

Manual testing is tedius, coming up with an idea 0f tests prior to starting the project will help you explicitely state what the program should do. It willalso give u alist of tasks that need to be handled without to try to figure out what(and how) to test after the fact.

Writing tests prior to implementing code can be a good design decisim because it will help you explicitely state what the program should do. State what the program should do explicitlu, explixitly state what the program should do, explicitly state what the program should do explicitely state what the program should do exlicitely state what the program should do explicitely state what the program, should do explicitely state what the program should do explicitely state what the progtam should do explicitely state what the program should do explicitely state what the program should do explicitely state what the program should do. Explicitely state what the program should do explicitely state what the program should do. yesh

Types of testing:

Unit tests, are meant to test small units of the program. This generally means the functions. Function takes some inputs, returns some output, expected output, actual ouput, input, test number. Actual outputs == expected outputs. Don’t change the tests to pass, fix the functionality, so don’t put the expected for false when the actual is false, when it really should expected be true. (**what we use in class is unit tests!!! )**

**Integra**tion tests- is the phase in software testing in which individual software modules are combined and tested as a group.

Regression tests- a type of software testing which verifies that software which was previousily developed and tested still preforms the same way after it was changed or interfaced with other software.

* Class design such as:
  + 1. When to use Const member functions: int getValue() const {return value}, they do not change the data fields of an object, declare as const any method that does not change the object(accessor methods).
    2. When to use const parameters in a function: passed paremeters by constant referece section of book k
    3. Private, protected, and public members: private members=This restricts access to the data field to just the class in which it is defined. makes program easier to debug and also have fewer logical errors rom the beginning, guarentees their integrity. Private members are onlu accessible whithin the class defining them.
    4. Protected= Protected members are only accesbile in the class that defines them, and in the classes that inherit from that class. I guess……when a class inherits from a parent class, the derived class can access which kinds of members from the parent-protected&public.
    5. Public- accessible by everyone!!
    6. Pros of using source control-(when we learned about git) first of all what is source control, source control saves changes to server, sc keeps track of changes over time, sc merge code together easier , sc share code easier to share , code is saved on a server so your work shouldn’t get lost git is a source control solution bitbicket hosts your repositories
    7. Keeps track of changes over time keeps track of changes over time keeps track of changes over time keeps track of changes over time keeps track of changes over time keeps track of changes over time keeps track of changes over time keeps track of changes over time keeps track of chages over time keeps track of changes over time keeps track of changes over time keeps track of changes over time keeps track of changes over time keeps track of changes over time keeps track of changes over time Madison Bethany Hubbard morgan brooke Hubbard morgan brooke hubard morgan brooke Hubbard morgan brooke Hubbard morgan brooke Hubbard morgan brooke Hubbard morgan brooke Hubbard morgan brooke Hubbard morgan brooke Hubbard soiurce control keeps track of changes over time and your work shouldn’t get lost, keeps track of changes over time and your work shouldn’t get lost, your work shouldn’t get lost your work shouldn’t get lost, keeps track of changes over time. And your work shouldn’t get lost because it is on a server and merges code together easier.

**RECURSION –**

Minimum required parts of a recursive function; base-case and recursive-case. How to tell them apart. Recursion errors, like Stack Overflow - how do you cause it, and why does it occur?

* Requited parts of a recursive function –

The required parts of a recursive function are as follows: basecase/terminating case and a recurxive case return RecurionFunctionname(n+1), term case returns some value .

* Stack overflow- stack overflow is a programming error when too much memory is used on the call stack, is a programming error when too much memory is used on the call stack, is a progtamming error when too much memory is used on the call stack. Is a prpgrammig error when too much memory is used on the call stack. Is a programming error when too much memory is used on a call stack is a programming error when too much memory is used on a call stack is a programming err or when to much memory is used on the call stack. Is a programming error when to much memory is used on the call stack. Is a programming error when too much memory is used on the call stack.
* Stack overlow is when you’ve used up more memory for the stack than your program was supposed to use.
* A way to cause stack overflow is infinite recursion, infinite recursion to get a stack overlow error.
* Sc occurs with recursion without an exit path, so when you don’t have an terminating case I guess makes sense to me whoopie!!
* Too many calls to the stack??, pushing elements when stack is full,
* Use more memory than the call stack has available.sc use too much memory than the call stack has available use too much memory than the call stack has available

Too much memory than the call stack has available too mych em9ry then the call stack has available you are using all the memory that th call stack gas and you want more but there is no more???? Using too muvh memory than the call sytack has abailable

**ARRAYS AND LINKED LISTS –**

Efficiency of the add / remove / search functions for each

* Add-

Linked lists –the list elements can be easily **be inserted or** removed without reallocation or reorginazation of the entire structure because the data items need not be stored contiguiosily in memory or on disk, so the linked lists are stored noncontinuoisily

Arrays- harder to add I guess

* Remove

Linked lists- the list elements can be easily be inserted or **removed** without reallocation or reorginazation of the entire structure because the data items need not be stored contiguiosily in memory or on disk,

Arrays-harder to do I guess

* Search – not the most efficient for linked lists because linked lists are linear and you will need to iterate from beginning one by one the nodes, do not allow [random access](https://en.wikipedia.org/wiki/Random_access) to the data, or any form of efficient indexing. Arrays- more efficient for searching because arr[i} search for the array at an index i. contiguous in memory???, its instantaneous

Linked lists- better for inserting data more often because it rezisez, wraps element data in “Nodes”, pointers are used to build “chain”, elements are non-contiguous, traversing the list is slower, duh obviousily!! :P

Arrays- better for accessing items more often1!!!, resize is costly, access is cheap

Design decisions behind how to build an array-based data structure

* Do you want to be accessing items more or inserting/deleting items more?????that would factor into your design I guess??, choose which one would best solve the problem,

Knowing random access vs. sequential access, for which structures

* Random access- used for array[i]💻🌵, binary searches
* Sequential access- used for linked list, access starting from the first node

High-level knowledge of the difference between singly, doubly, and circularly linked lists.

* Singly linked lists= singly linked lists are only point to the next node 🡪 way direction

the nodes only keep track of the next item, each node only points forward to its next item. !! 😊

* Doubly linked lists= what we have been using, each node contains, beside the next node link, a second link field pointing to the previous node in the sequence. Direction left to right -> to <- kkk . in addition, the two links may be called forwards or backwards or next and prev, the nodes of the list keep track of the previous and next items, where each node points to its next item and previous item duh
* Circularly linked lists goes from left 🡪 that way to a circle think of direction of a clock circle direction is clockwise!!! In the last node of the list, the link field often contains a null refeence a special value used to indicate the lack of further nodes. A less common convention is to make it point to the first node of the list, in that case the list is said to be circular or circularly linked, otherwise it is said to be open or linear. It is a list where the last pointer points to the first node. The last node of the list has its next pointer set to the first node of the list, and vice versa if the nodes store previous items.,, where the last node of the list points to the first node as its next item .

Be able to diagram the elements of a list after some PushBack / PushFront / PopBack / PopFront commands.

* Okay done with this I get this kkkkkkkk :P think logically breathe you will do great!!

<https://fall14cs.files.wordpress.com/2016/03/data-abstraction-and-problem-soving-with-c.pdf>