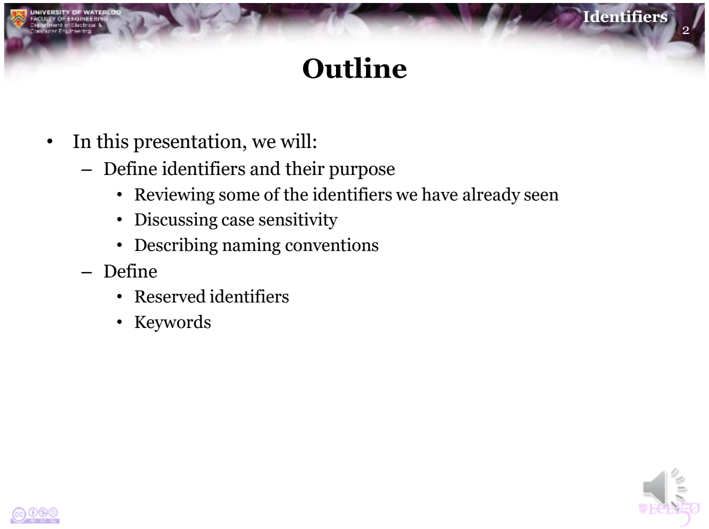
UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical &
Computer Engineering

ECE 150 *Fundamentals of Programming*

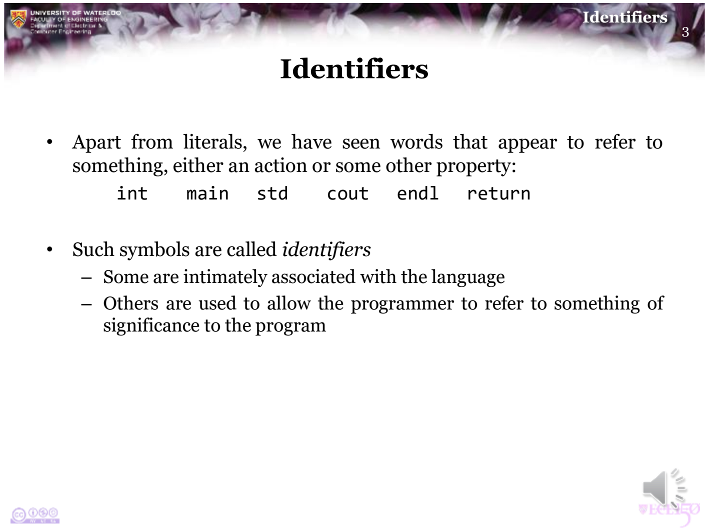
Identifiers

Douglas Wilhelm Harder, M.Math. LEL
Prof. Hiren Patel, Ph.D.
dwharder@uwaterloo.ca hiren.patel@uwaterloo.ca



Outline

- In this presentation, we will:
  - Define identifiers and their purpose
    - Reviewing some of the identifiers we have already seen
    - Discussing case sensitivity
    - Describing naming conventions
  - Define
    - Reserved identifiers
    - Keywords

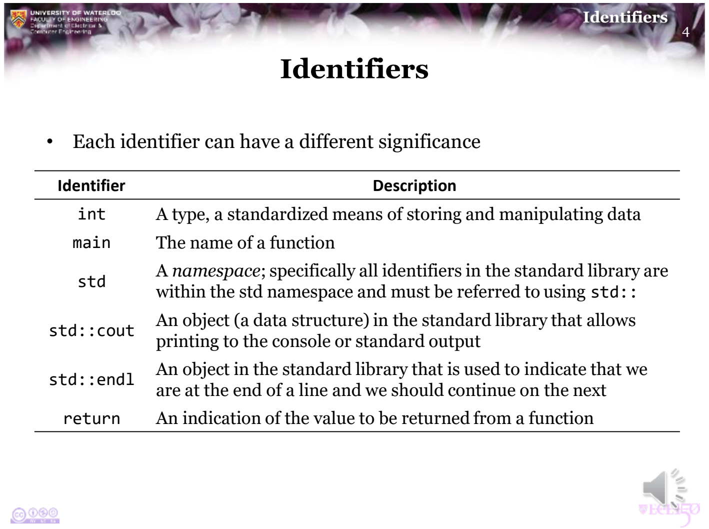


Identifiers

- Apart from literals, we have seen words that appear to refer to something, either an action or some other property:

```
int    main   std    cout   endl   return
```

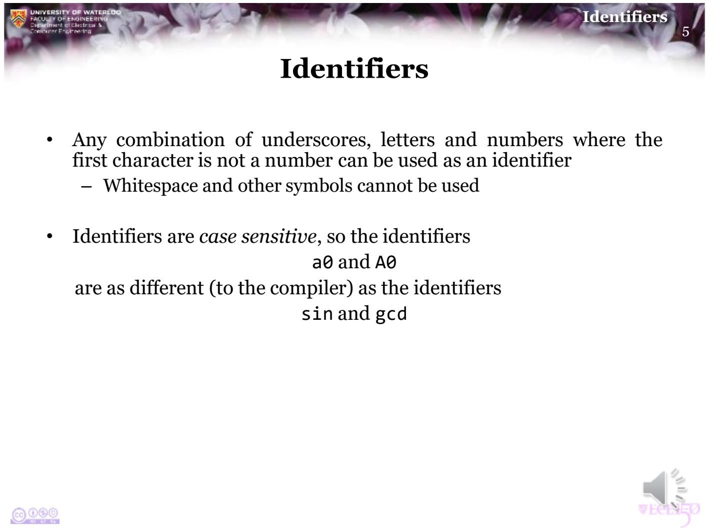
- Such symbols are called *identifiers*
  - Some are intimately associated with the language
  - Others are used to allow the programmer to refer to something of significance to the program



Identifiers

- Each identifier can have a different significance

Identifier	Description
`int`	A type, a standardized means of storing and manipulating data
`main`	The name of a function
`std`	A *namespace*; specifically all identifiers in the standard library are within the std namespace and must be referred to using `std::`
`std::cout`	An object (a data structure) in the standard library that allows printing to the console or standard output
`std::endl`	An object in the standard library that is used to indicate that we are at the end of a line and we should continue on the next
`return`	An indication of the value to be returned from a function



Identifiers 5

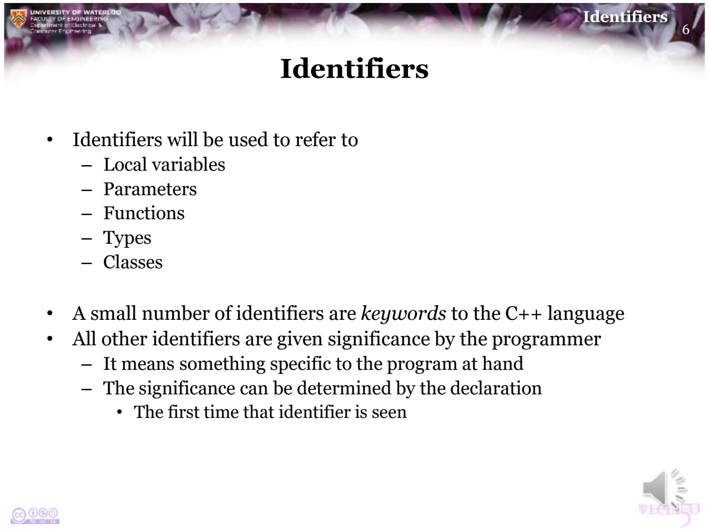
Identifiers

- Any combination of underscores, letters and numbers where the first character is not a number can be used as an identifier
  - Whitespace and other symbols cannot be used
- Identifiers are *case sensitive*, so the identifiers

  `a0` and `A0`

  are as different (to the compiler) as the identifiers

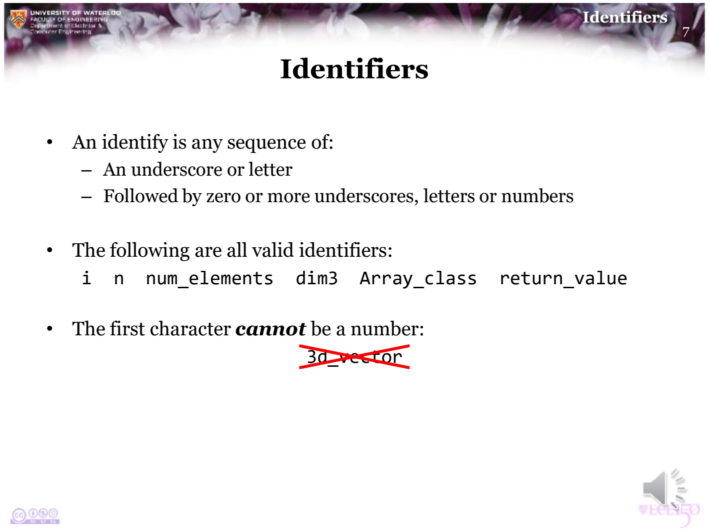
  `sin` and `gcd`



Identifiers 6

Identifiers

- Identifiers will be used to refer to
  - Local variables
  - Parameters
  - Functions
  - Types
  - Classes
- A small number of identifiers are *keywords* to the C++ language
- All other identifiers are given significance by the programmer
  - It means something specific to the program at hand
  - The significance can be determined by the declaration
    - The first time that identifier is seen



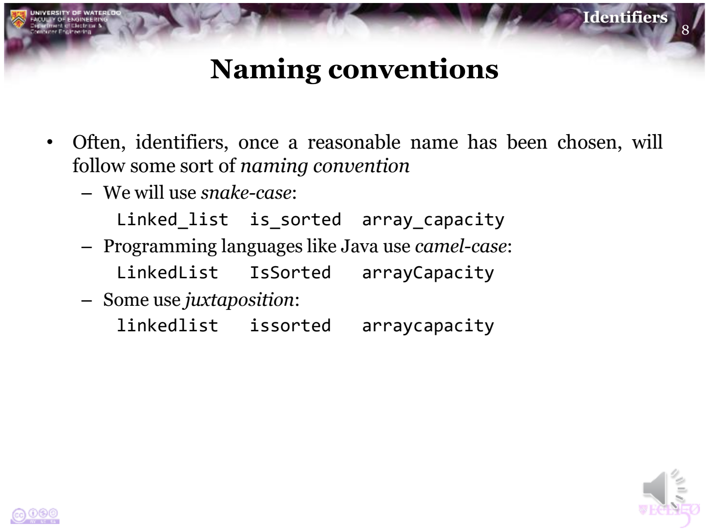
Identifiers 7

Identifiers

- An identify is any sequence of:
  - An underscore or letter
  - Followed by zero or more underscores, letters or numbers
- The following are all valid identifiers:

  `i   n   num_elements   dim3   Array_class   return_value`
- The first character ***cannot*** be a number:

  ~~`3d_vector`~~



Identifiers 8

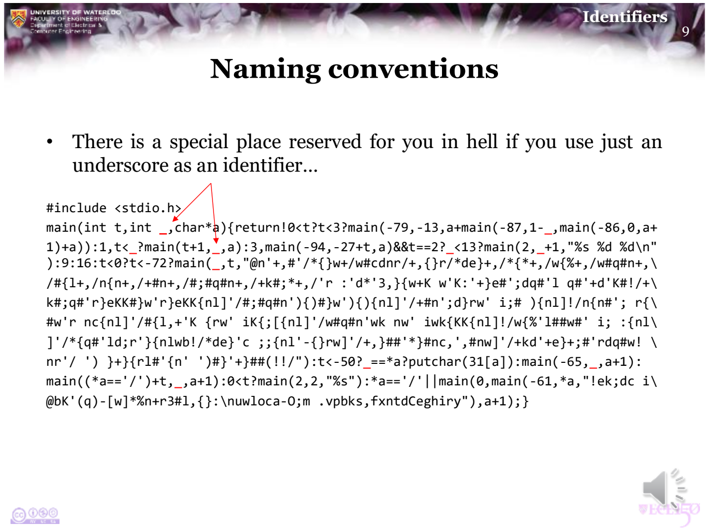
Naming conventions

- Often, identifiers, once a reasonable name has been chosen, will follow some sort of *naming convention*
  - We will use *snake-case*:

    `linked_list   is_sorted   array_capacity`
  - Programming languages like Java use *camel-case*:

    `LinkedList   IsSorted   arrayCapacity`
  - Some use *juxtaposition*:

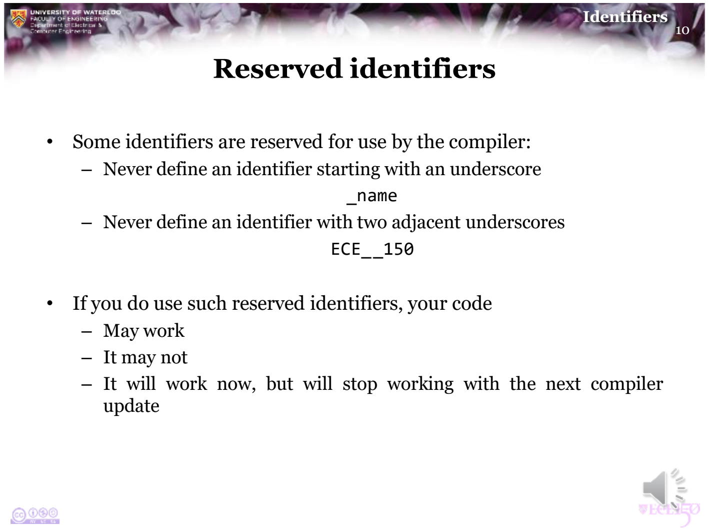
    `linkedlist   issorted   arraycapacity`



Naming conventions

- There is a special place reserved for you in hell if you use just an underscore as an identifier…

```
#include <stdio.h>
main(int t,int _,char*a){return!0<t?t<3?main(-79,-13,a+main(-87,1-_,main(-86,0,a+
1)+a)):1,t<_?main(t+1,_,a):3,main(-94,-27+t,a)&&t==2?_<13?main(2,_+1,"%s %d %d\n"
):9:16:t<0?t<-72?main(_,t,"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,\
/#{l+,/n{n+,/+#n+,/#;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K w'K:'+}e#';dq#'l q#'+d'K#!/+\
k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n'){)#}w'){){nl]'/+#n';d}rw' i;# ){nl]!/n{n#'; r{\
#w'r nc{nl]'/#{l,+'K {rw' iK{;[{nl]'/w#q#n'wk nw' iwk{KK{nl]!/w{%'l##w#' i; :{nl\
]'/*{q#'ld;r'}{nlwb!/*de}'c ;;{nl'-{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;#'rdq#w! \
nr'/ ') }+}{rl#'{n' ')#}'+}##(!!/"):t<-50?_==*a?putchar(31[a]):main(-65,_,a+1):
main((*a=='/')+t,_,a+1):0<t?main(2,2,"%s"):*a=='/'||main(0,main(-61,*a,"!ek;dc i\
@bK'(q)-[w]*%n+r3#l,{}:\nuwloca-O;m .vpbks,fxntdCeghiry"),a+1);}
```



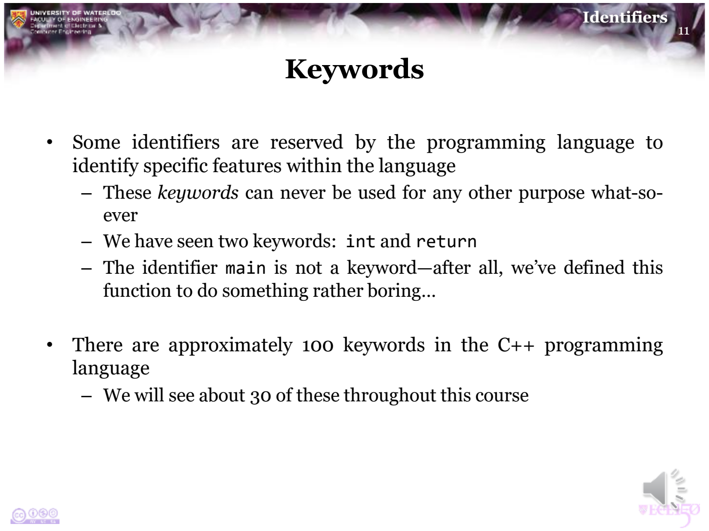
Reserved identifiers

- Some identifiers are reserved for use by the compiler:
  - Never define an identifier starting with an underscore

    `_name`
  - Never define an identifier with two adjacent underscores

    `ECE__150`

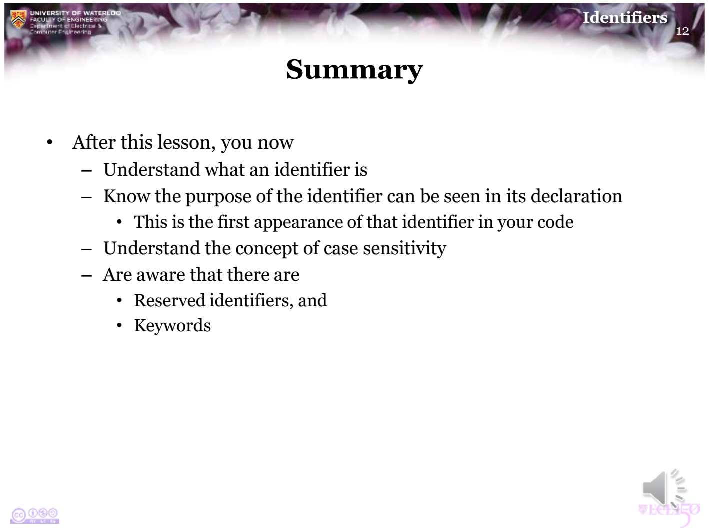
- If you do use such reserved identifiers, your code
  - May work
  - It may not
  - It will work now, but will stop working with the next compiler update



Keywords

- Some identifiers are reserved by the programming language to identify specific features within the language
  - These *keywords* can never be used for any other purpose what-so-ever
  - We have seen two keywords: `int` and `return`
  - The identifier `main` is not a keyword—after all, we've defined this function to do something rather boring…

- There are approximately 100 keywords in the C++ programming language
  - We will see about 30 of these throughout this course



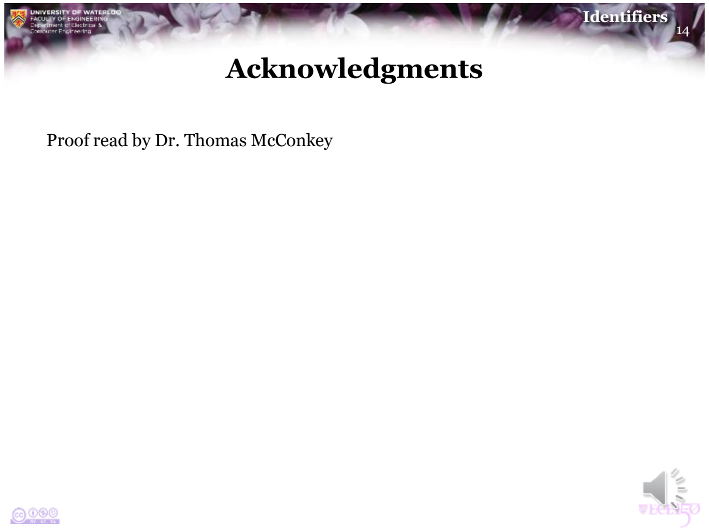
Summary

- After this lesson, you now
  - Understand what an identifier is
  - Know the purpose of the identifier can be seen in its declaration
    - This is the first appearance of that identifier in your code
  - Understand the concept of case sensitivity
  - Are aware that there are
    - Reserved identifiers, and
    - Keywords

References

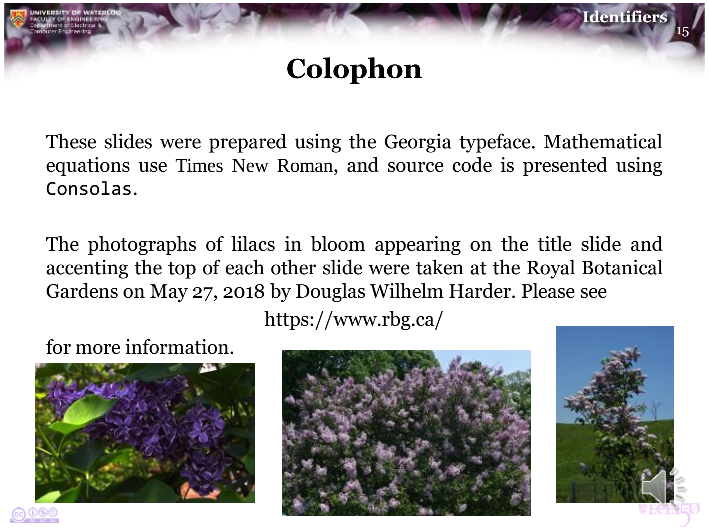
[1] Wikipedia,
https://en.wikipedia.org/wiki/Identifier#In_computer_languages

[2] C++ reference
https://en.cppreference.com/w/cpp/language/identifiers



Acknowledgments

Proof read by Dr. Thomas McConkey



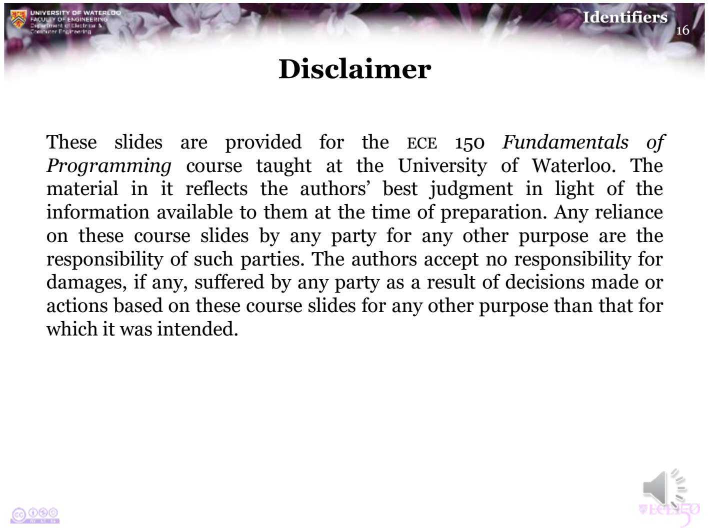
Colophon

These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`.

The photographs of lilacs in bloom appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens on May 27, 2018 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.



Disclaimer

These slides are provided for the ECE 150 *Fundamentals of Programming* course taught at the University of Waterloo. The material in it reflects the authors' best judgment in light of the information available to them at the time of preparation. Any reliance on these course slides by any party for any other purpose are the responsibility of such parties. The authors accept no responsibility for damages, if any, suffered by any party as a result of decisions made or actions based on these course slides for any other purpose than that for which it was intended.