1 Computers, People and Programming

1.1 Review

1 What is software?

Language that computers understand

2 Why is software important?

That's the only way computers can perform their function

3 Where is software important?

Wherever a computer is

4 What could go wrong if some software fails? List some examples.

- Satellites can get lost in space
- Unable to do your homework
- Unable to move a vehicle

5 Where does software play an important role? List some examples.

- Interact with HMI
- Code/Decode TCP stream packets
- Blink turn signals on a vehicle

6 What are some jobs related to software development? List some.

- Design
- Analyst
- Coder
- Tester

7 What's the difference between computer science and programming?

8 Where in the design, construction, and use of a ship is software used?

During concept, design, simulation, testing

9 What is a server farm?

Shared space where clusters serve a finite number of companies

10 What kinds of queries do you ask online? List some.

11 What are some uses of software in science? List some.

simulation, analysis, prediction

12 What are some uses of software in medicine? List some.

DICOM, image analysis, triage, clinical diagnosis

13 What are some uses of software in entertainment? List some.

games, 3D, sci-fi

14 What general properties do we expect from good software?

scalable, understandable, portable

15 What does a software developer look like?

human being

16 What are the stages of software development?

analysis, design, code, test

17 Why can software development be difficult? List some reasons.

obscure programming, time constraints

18 What are some uses of software that make your life easier?

internet, fuel injection

19 What are some uses of software that make your life more difficult?

driving asistance

1.2 Exercises

20 Pick an activity you do most days (such as going to class, eating dinner, or watching television) . Make a list of ways computers are directly or indirectly involved.

eating dinner: microwave (whenever was deisned and what cad software was used, define HMI and its controls), dishes (test durability among other functions), cutlery (determine the material to manufacture and its proportions to make it light enough but durable)

21 Pick a profession, preferably one that you have some interest in or some knowledge of. Make a list of activities done by people in that profession that involve computers.

hospital premises: set an appointment and get a confirmation, laboratory equipment (clinical diagnosis and their results, HMI), imaging diagnosis (CT Scan, MRI)

22 Swap your list from exercise 2 with a friend who picked a different profession and improve his or her list. When you have both done that, compare your results. Remember: There is no perfect solution to an open-ended exercise; improvements are always possible.

23 From your own experience, describe an activity that would not have been possible without computers.

computed imaging diagnosis

24 Make a list of programs (software applications) that you have directly used. List only examples where you obviously interact with a program (such as when selecting a new song on an MP3 player) and not cases where there just might happen to be a computer involved (such as turning the steering wheel of your car).

change TV channel, select radio station, washing machine program

25 Make a list of ten activities that people do that do not involve computers in any way, even indirectly. This may be harder than you think!

outdoor activities, pay with cash, make fire, drink plain water, talk, hunt, sex

26 Identify five tasks for which computers are not used today, but for which you think they will be used for at some time in the future. Write a few sentences to elaborate on each one that you choose.

hire/fire personnel, open/close a business, triage systems within hospital premises

27 Write an explanation (at least 100 words, but fewer than 500) of why you would like to be a computer programmer. If, on the other hand, you are convinced that you would not like to be a programmer, explain that. In either case, present well-thought-out, logical arguments.

If related to any technical position, even for managers, programming should be a a must. It clearly helps to make better decisions, make the complex easier to digest, great ability to make you better, and more confident about your capabilities.

28 Write an explanation (at least 100 words, but fewer than 500) of what role other than programmer you'd like to play in the computer industry (independently of whetl1er "programmer" is your first choice).

29 Do you think computers will ever develop to be conscious, thinking beings, capable of competing with humans ? Write a short paragraph (at least 100 words) supporting your position.

Definetely, feasible enough to see a computer building another computer, or programming. Hard to see though a computer solving human kind problems, but computer problems.

30 List some characteristics that most successful programmers share. Then list some characteristics that programmers are popularly assumed to have.

Open source related projects

31 Identify at least five kinds of applications for computer programs mentioned in this chapter and pick the one that you find the most interesting and that you would most likely want to participate in someday. Write a short paragraph (at least 100 words) explaining why you chose the one you did.

Medical related applications. How to solve common problems with a software related discipline

32 How much memory would it take to store (a) this page of text, (b) this chapter, (c) all of Shakespeare's work? Assume one byte of memory holds one character and just try to be precise to about 20%.

$Store_{line} = 80\frac{char}{line} * \frac{1\ byte}{1\ char} = 80\frac{byte}{line}$
$Store_{page} = 80\frac{byte}{line} * 40\frac{line}{page} = 3200\frac{byte}{page}$

$Store_{book} = 3200\frac{byte}{page} * 1300\frac{page}{book} = 4160000\frac{byte}{book}$

33 How much memory does your computer have? Main memory? Disk?

8GB ram memory, 500GB ssd

2 Hello, World!

2.1 Review

34 What is the purpose of the ”Hello, World!” program?

Understand the compiling process, linker and output generated.

35 Name the four parts of a function.

name, parameter list, function body, return type

36 Name a function that must appear in every C++ program.

int main()

37 In the ”Hello, World !” program, what is the purpose of the line return 0; ?

error code

38 What is the purpose of the compiler?

transalte human readable text to machine executable

39 What is the purpose of the #include directive?

tell the compiler that some functionalities to be used are defined in the header file

40 What does a .h suffix at the end of a file name signify in C++?

header file

41 What does the linker do for your program?

merge the declaration of header files and code provided

42 What is the difference between a source file and an object file?

depends who interprets the file, a human or a machine

43 What is an IDE and what does it do for you?

leverage the process of compiling and make ypur life easier

44 If you understand everything in the textbook, why is it necessary to practice?

the best way to learn

2.2 Exercises

45 Write a definition for each of the terms from "Terms."

- //: used to place comments
- C++: programming language
- comment: non-machine readable code
- compiler: transalte human code to machine code
- compile-time error: error that occurs at the time of compiling
- cout: standard output, i.e. displat, screen,...
- executable: object code that can be interpreted by a computer
- function: piece of code that performs some action
- header: files where function definitions are expected
- IDE: integrated development environment application
- #include: tells the compiler where some functionalities are defined
- library: set of files packed togheter to perform a specific action
- linker: helps the compiler merge off-the-shel code with our code
- main(): function where everything begins
- object code: code readble for a computer
- output: desired action performed by a function
- program: another terminaology for object code
- source code: human readble code
- statement: every line of a source code

3 Objects, Types and Values

3.1 Review

46 What is meant by the term prompt?

in a terminal, where you write statements

47 Which operator do you use to read into a variable?

assignement = operator

48 If you want the user to input an integer value into your program for a variable named number, what are two lines of code you could write to ask the user to do it and to input the value into your program?

```
cout << "Enter integer value:\n";
int value;
cin >> value;
```

49 What is
n called and what purpose does it serve?

newline onto cout

50 What terminates input into a string?

blank space

51 What terminates input into an integer?

blank space

52 How would you write:
cout << "Hello, ";
cout << first_name;
cout << "!n";
as a single line o code?

```
cout << "Hello, " << fist_name << "!\n";
```

53 What is an object?

place where variables are stored in memory

54 What is a literal?

a number

55 What kinds of literals are there?

real, complex

56 What is a variable?

type of object in memory to store values

57 What are typical sizes for a char, an int, and a double?

char 1 byte, int 4byte, double 8 byte

58 What measures do we use for the size of small entities in memory, such as ints and strings?

59 What is the difference between = and =+

'=' is a read operator, whereas '==' is compare operator

60 What is a definition?

where a variable is declared

61 What is an initialization and how does it differ from an assignment?

at compile time, an value is initialized; at run time assignemnts can be done

62 What is string concatenation and how do you make it work in C++?

merge of strings in a single one, with the operator '+'

63 Which of the following are legal names in C++? If a name is not legal, why not?
This_little_pig This_1_is fine 2_For_1_special
latest thing the_$12_method _this_is_ok
MiniMineMine number correctl

legal means no compile error. Not allowed numbers or special chars as the character in a variable name; no special chars also in the name

64 Give five examples of legal names that you shouldn't use because they are likely to cause confusion.

int string_value, char int_value, char _foo

65 What are some good rules for choosing names?

self describe their meaning, not long names, consistency

66 What is type safety and why is it important?

implicit conversion of variables from one type to another, likely to miss information

67 Why can conversion from double to int be a bad thing?

missing some precision

68 Define a rule to help decide if a conversion from one type to another is safe or unsafe.

if performing a second conversion gives the original value back, assume the conversion is safe

3.2 Exercises

4 Computation

4.1 Review

69 What is a computation?

reading inputs to perform some actions and deliver ouptuts

70 What do we mean by inputs and outputs to a computation? Give exam ples.

read temperature sensor, compute a PID algorithm to update ssr output

71 What are the three requirements a progranuner should keep in mind. when expressing computations?

a software must be simple, efficient and work as expected

72 What does an expression do?

evaluate as true or false

73 What is the difference between a statement and an expression, as described in this chapter?

statament ends with semicolon ';', whereeas an expression does not

74 What is an lvalue? List the operators that require an lvalue. Why do these operators, and not the others, require an lvalue?

lvalue is the left inside of an assignement, an object type. Assignment '=' and compound assignement '+=','-=','*=','/=' and '%=' require and lvalue. Because they do appear in the left inside of the assignemnt, and thus must update the value type they are holding.

75 What is a constant expression?

that is assigned to a value that it's not expected to change at runtime

76 What is a literal?

represent values of varouis types

77 What is a symbolic constant and why do we use them?

a value that is well defined and universal

78 What is a magic constant? Give examples.

non-obvious constant literals

79 What are some operators that we can use for integers and floating-point values?

operators: +,-,*,/

80 What operators can be used on integers but not on floating-point numbers?

modulo operator '%'

81 What are some operators that can be used for strings?

+, *

82 When would a programmer prefer a switch-statement to an if-statement?

working with enums

83 What are some common problems with switch-statements?

only work for const integer values

84 What is the function of each part of the header line in a for-loop, and in what sequence are they executed?

initializer, expression, loop number

85 When should the for-loop be used and when should the while-loop be used?

the control variable needs to be initialized beforehand in a while-loop

86 How do you print the numeric value of a char?

assign the char to an integer variable and print it

87 Describe what the line char foo(int x) means in a function definition.

return type as char, name function to be called foo with an integer as argument.

88 When should you define a separate function for part of a program? List reasons.

one function performs a single action at a time, assuming that the action make happen a few times in our code

89 What can you do to an int that you cannot do to a string?

most arithmetic operations: /, -, %, use as case switch

90 What can you do to a string that you cannot do to an int?

as a string or stream of charecters, uppercase, lowercase, ...

91 What is the index of the third element of a vector?

number 2

92 How do you write a for-loop that prints every element of a vector? vector<char >alphabet(26); do?

```
for(int =0;i< alphabet.size();i++)
    cout << alphabet[i] << '\n'
```

93 Describe what push_back() does to a vector.

member function that lets add a value to vector type

94 What do vector's member functions begin(), end(), and size() do?

begin identifies the first element of a vector, end for the last element and size keeps track of the number of elements sotored in the vector

95 What makes vector so popular/useful?

dynamic allocation, optimized versus an array

96 How do you sort the elements of a vector?

using function sort(begin(),end())

5 Errors

5.1 Review

97 Name four major types of errors and briefly define each one.

compile time errors: detected by compilers dur to syntax error
linker time errors: missing defintions when merging libraries and source code
run-time errors: fails to execute due to bad code behavior
logic errors: fail to perform as expected due to wrong interpretation

98 What kinds of errors can we ignore in student programs?

hardware related, up to some extent

99 What guarantees should every completed project offer?

should produce expected output, give reasonable errors when something goes wrong

100 List three approaches we can take to eliminate errors in programs and produce acceptable software.

organize software to minimize errors
check on errors through debugging and testing
reamining errors are not serious

101 Why do we hate debugging?

teodious task,

102 What is a syntax error? Give five examples.

not defined as C++ standard

103 What is a type error? Give five examples.

mismatch between a type variable and its assignment or initialization

104 What is a linker error? Give three examples.

mismatch between defined function and its definition, i.e. arguments, output,..

105 What is a logic error? Give three exan1ples.

a source code that does not provide the expected output

106 List four potential sources of program errors discussed in the text.

poor specification: undefined uses of a program provoke undesired result
incomplete programs: missing source code to handle expected/unexpected inputs
unexpected arguments: take care of unexpected inputs of a function unexpected input: user typed input to a program that don't follow the rules
unexpected state: data that is either incomplete or wrong
logical error: code that doesn't do what's expected to do

107 How do you know if a result is plausible? What techniques do you have to answer such questions?

double check with other sources, test on known outputs as per expected inputs, and make simple estimations

108 Compare and contrast having the caller of a function handle a run-time error vs. the called function's handling the run-time error.

withtin the called function, only in a single place the code is easy to maintain, whereas to very caller perform the same code n times

109 Why is using exceptions a better idea than returning an "error value"?

not every single returned value is interpreted, think of Linux and Windows.

110 How do you test if an input operation succeeded?

111 Describe the process of how exceptions are thrown and caught.

within the code to an unwanted variable values, you throw an exception statemant, that later in a try-catch block, the exception in processed and shows information to the user.

112 Why, with a vector called v, is v[v.size()] a range error? What would be the result of calling this?

access to an unknown memory location, as the vector exactly allocated space for n variables, not n+1!

113 Define pre-condition and post-condition; give an example (that is not the area() function from this chapter), preferably a computation that requires a loop.

pre-condition, tests the input arguments in a function; in a post-condition, the output is tested before returing a value.

114 When would you not test a pre-condition?

nobody would give bad arguments
slows down performance
too complicated to check

115 When would you not test a post-condition?

when a comment suffices a trivial operation

116 What are the steps in debugging a program ?

compile, link and run... over thousand times

117 Why does commenting help when debugging?

makes the code easier to read

118 How does testing differ from debugging?

testing is a systematic way to search for errors

6 Writing a program

119 What do we mean by ”Programming is understanding”?

understand what the problem we're trying to solve

120 The chapter details the creation of a calculator program. Write a short analysis of what the calculator should be able to do.

work with basic operators: +, -, *, / with double types. generate errors upon bad inputs, use of parenthesis for the sake of clarity, handle division by zero.

121 How do you break a problem up into smaller manageable parts?

trust on usage of tools o libraries that help, experience that sort parts of the solution

122 Why is creating a small, limited version of a program a good idea?

focus on the key problem helps understand the problem or tools, break down problem statement to manageable parts

123 Why is feature creep a bad idea?

start building alimed version taht solves the problem, at a later stage build full-scale by working on parts

124 What are the three main phases of software development?

analysis, design and implemtentation

125 What is a ”use case”?

a scenario where a software receives an external request and responds to it

126 What is the purpose of testing?

use cases to minimze logical errors, build upon software requirements

127 According to the outline in the chapter, describe the difference between a Term, an Expression, a Number, and a Primary.

Primary stands for operators and operands
Number converted primary input text to double

Expression, handles +,- operations
Term, handles *,// operations

128 In the chapter, an input was broken down into its component Terms, Expressions, Primarys, and Numbers. Do this for (17+4)/(5-1).

Primarys: operators and operands that fit into stream: key, value
Numbers: doubles as 17, 4, 5, 1
Expressions (+,-): 17+4, 5-1
Terms (*,/): (17+4)/(5-1)

129 Why does the program not have a function called number()?

once the key is read, cin interprets the pair value as double

130 What is a token?

a key pair value taht holds the input stream

131 What is a grammar? A grammar rule?

we write grammar defining the syntax of our input and then write a program that implements the rules of the grammar

132 What is a class? What do we use classes for?

a class is a user defined type, useful when none of existing types statisfies our needs

133 What is a constructor?

instance of how to generate a user defined type in our code

134 In the expression function, why is the default for the switch-statement to "put back" the token?

because we read char at once, and we need to convert numbers bigger than 9, otherwise the number will show as cropped

135 What is "look-ahead"?

136 What does putback() do and why is it useful?

for numbers, let's store more that a single digit number

137 Why is the remainder (modulus) operation,% , difficult to implement in the term()?

as long as we're working with doubles, modulo only applies to integer

138 What do we use the two data members of the Token class for?

when reading the token stream, identifies which token is an operator or number

139 Why do we (sometimes) split a class's members into private and public members?

user has access to public members about how the class, not the implementation itself

140 What happens in the Token_stream class when there is a token in the buffer and the get() function is called?

we inmediately return the stored token

141 Why were the ';' and 'q' characters added to the switch-statement in the get() function of the Token_stream class?

non operators by defintion, rather exit code (q) and solve operation (;)

142 When should we start testing our program?

as soon as possible

143 What is a "user-defined type"? Why would we want one?

also called class, when the standard library does not provide a type that suit our needs

144 What is the interface to a C++ "user-defined type"?

members functions defined inside the class

145 Why do we want to rely on libraries of code?

to focus on the real problem, helps to leviate the implementation whenever a library suits our needs

7 Completing a Program

7.1 Review

146 What is the purpose of working on the program after the first version works? Give a list of reasons.

keep adding new features
make the code cleaner, simpler and more efficient

147 Why does "1+2; q" typed into the calculator not quit after it receives an error?

because the main loop handles execptions and cleans cin for keep running

148 Why did we choose to make a constant character called number?

to make the code fre from magic constants

149 Why do we split code into multiple functions? State principles.

make the code cleaner, as states one function for a single action. splitting helps to keep code easier to maintain

150 We split main() into two separate functions. What does the new function do and why did we split main()?

one function holds the original code to perform 'expressions' and the other keeps variable assignment to read pseudo-code

151 What is the purpose of commenting and how should it be done?

comments should clarify what's not easy to explain in code

152 What does narrow_cast do?

type of cast that checks the casted value with the original one and is able to throw an exception if the results are different

153 What is the use of symbolic constants?

avoid magic constants in the code that are difficult to trace

154 Why do we care about code layout?

easier to read and maintain, keep structured layout helps scalability

155 How do we handle % (remainder) of floating-point numbers?

because % modulo operator does not hold double operands

156 What does is_declared() do and how does it work?

reads through the vector of variables returning true when the variable already exists, false otherwise

157 The input representation for let is more than one character. How is it accepted as a single token in the modified code?

for 'let' it's acknowledge as 'L' in the token kind

158 What are the rules for what names can and cannot be in the calculator program?

variables must start with alpha characters, no symbols nor numbers

159 Why is it a good idea to build a program incrementally?

easier to find bugs while testing as you go on

160 When do you start to test?

everytime is a good practice to keep testing, do it asap

161 When do you retest?

when the code has changed to be sure the old code still works as expected

162 How do you decide what should be a separate function?

stick to a rule of a function holds a single action

163 Why do you add comments?

let the reader understand what's not so easier to follow on the code

164 What should be in comments and what should not?

a developer should read through the grammar what the code does or its implemented,

165 When do we consider a program finished?

hard to say that a code is free from errors, what we can to is to minimize through test as much as possible

8 Technicalities: Functions, Etc

8.1 Review

166 What is the difference between a declaration and a definition?

declaration shows the way to use a function, rather than how it's done through its declaration

167 How do we syntactically distinguish between a function declaration and a function definition?

function declaration ends with semicolon ';', whereas a definition uses curly brackets '' and ''

168 How do we syntactically distinguish between a variable declaration and a variable definition?

a declaration sets the type adn name, and a definition includes its initialization

169 Why can't you use the functions in the calculator program from Chapter 6 without declaring them first?

at least the compiler needs to know how to use the function or description, before it can be used elsewhere

170 Is int a; a definition or just a declaration?

declaration

171 Why is it a good idea to initialize variables as they are declared?

to reduce the risk of using a variable of an unknown value

172 What can a function declaration consist of?

declaration must include return value as return, name of the fucntion and argument list if any with their type

173 What good does indentation do?

makes code reading more pleasent

174 What are header files used for?

header files hold classes, enums and function declaration

175 What is the scope of a declaration?

as long as the declaration is included in a header file, the scope is local

176 What kinds of scope are there? Give an example of each.

namespace: named scope inside a global scope or any
global: area outside any scope
class: area within the class
local: between ',' of a function definition or block
statement: for loop , do-while,..

177 What is the difference between a class scope and local scope?

local scope refers to functions, whereas class has own space

178 Why should a programmer minimize the number of global variables?

tend to be problematic on the long run

179 What is the difference between pass-by-value and pass-by-reference?

by value keeps a copy of the original value, where by reference does not

180 What is the difference between pass-by-reference and pass-by-const-reference?

by const means that the value will not change its value, otherwise by-reference means that the value will be updated

181 What is a swap()?

swap functions takes two arguments to exchange its value one another

182 Would you ever define a function with a vector <double >-by-value parameter?

for large amount of data, use by-reference its a preferred solution. Only by value if the amount of data its small

183 Give an exam ple of undefined order of evaluation. Why can undefined order of evaluation be a problem?

f(++i,++i): its upon the compiler, system the value passed onto function f

184 What do x&& y and x||y, respectively, mean?

logical operators to evaluate two expressions, AND or OR

185 Which of the following is standard-conforming C++: functions within functions, functions within classes, classes within classes, classes within functions?

functions within classes, known as member functions. The functions wihtin functions its no allowed, the rest can be used with caution

186 What is a call stack and why do we need one?

stack hold the local space, or activation records for a specific function. And keeps track of nested calls that when complete serves back the original value as a LIFO queue

187 What is the purpose of a namespace?

set an isolated environment for a set of entities that won't clash with other definitions of other namespaces, providing a unique identifier for such namespace

188 How does a namespace differ from a class?

class helps to organize data and functions; a namespace holds classes, functions, into an identifiable and named part of a program without defining a type

189 What is a using declaration?

tells the complier which namespace uses a specific function, or data member; avoids keep using the fully qualified name

190 Why should you avoid using directives in a header?

helps to loose track of which namespace you're using

191 What is namespace std?

the standard library namespace

9 Technicalities: Classes, Etc

9.1 Review

192 What are the two parts of a class, as described in the chapter?

public act as the interface to a class, and private implementation details hidden from public access

193 What is the difference between the interface and the implementation in a class?

interface is public, and implementation is declared as private

194 What are the limitations and problems of the original Date struct that is created in the chapter?

how to catch errros at compile-time, instead run-time. Ability to change date month, year or day without control

195 Why is a constructor used for the Date type instead of an init_day() function?

performs the same as the helper function, but checks the validity of the data at the time of creation of the Date type.

196 What is an invariant? Give examples.

invariant are rules to be valid. for instance is not expected a year to be below zero, but can be.

197 When should functions be put in the class definition, and when should they be defined outside the class? Why?

outside the class, called helper functions, remain for those functions that provide by itself an elegant and efficient solution, shpould be kept outside the class. After all, the class must be kept simple, also helps debugging by limiting the usual suspects of the list

198 When should operator overloading be used in a program? Give a list of operators that you might want to overload (each with a reason).

by defualt the compiler provide the copy construnctor and copy assignment, others such '==', '!=', '¡¡' or '¿¿' are not provided, so must be implemented as helper fucntions

199 Why should the public interface to a class be as small as possible?

keep understanding of the class simple, whitout misunderstanding. Helps to keep track of bugs

200 What does adding const to a member function do?

making a memeber const, makes mandatory to be used by constant references

201 Why are "helper functions" best placed outside the class defintion?

helper functions are design concept rather than programming language concept

10 Input and Output Streams

10.1 Review

202 When dealing with input and output, how is the variety of devices dealt with in most modern computers?

we're taking about keyboard, or audio stream, stream or http...

203 What, fundamentally, does an istream do?

handles how an input stream from a device driver, writes data stream to file

204 What, fundamentally, does an ostream do?

also uses a device driver to communicate to outside world, i.e. hdd, ssd, audio,..

205 What, fundamentally, is a file?

handles all input and output streams to and from the computer

206 What is a file format?

how the data stored in a file is to be handled

207 Name four different types of devices that can require I/O for a program.

audio, disk drive, http, wacom digitizer

208 What are the four steps for reading a file?

name the file, create ifstream to file, read data, close ifstream

209 What are the four steps for writing a file?

name the file, create ofstream to file, write data, close ofstream

210 Name and define the four stream states.

good(), eof(), fail(), bad()

211 Discuss how the following input problems can be resolved:
a. The user typing an out-of-range value
b. Getting no value (end of file)
c. The user typing something of the wrong type

a. check validity upon receiving values
b. eof() indicates no more data pending
c. sets fail() to give chance to recover if needed, bad() must abort operation

212 In what way is input usually harder than output?

we must check for a higher likelihood of mistyped data

213 In what way is output usually harder than input?

in a way to use graphical environments to interact with user

214 Why do we (often) want to separate input and output from computation?

computation takes input data to perform operations and shows its results to the output

215 What are the two most common uses of the istream member function clear()?

clear() removes fail bit to reinitialize the cin, accepting more inputs

216 What are the usual function declarations for ¡¡ and ¿¿ for a user-defmed type X?

using self defined <<or >>, handles the input and output streams for our defined types

11 Customizing Input and Output

11.1 Review

217 Why is I/O tricky for a programmer?

desired input among all the possibilities

218 What does the notation ¡¡ hex do?

display the following number in hex notation

219 What are hexadecimal numbers used for in computer science? Why?

hardware, it matches 4bit with a single representation

220 Name some of the options you may want to implement for formatting integer output.

decimal, hexadecimal and octal, maybe binary

221 What is a manipulator?

adapts or changes its value to show on ostream

222 What is the prefix for decimal? For octal? For hexadecimal?

dec, hex, and oct

223 What is the default output format for floating-point values?

general form stands for 6 digit representation

224 What is a field?

how to represent a number using different bases

225 Explain what setprecision() and setw() do.

setprecision sets number of output deciamls for double values, whereas setw means the width to use to representa a value on screen

226 What is the purpose of file open modes?

upon on our needs, and the type of file to write or read, changes where and how is written to it

227 Which of the following manipulators does not ”stick” : hex, scientific, setprecision, showbase, setw?

setw, setprecision

228 What is the difference between character I/O and binary I/O?

character representation codes every input as a single byte, binary uses 4 bytes to code a integer

229 Give an example of when it would probably be beneficial to use a binary flle instead of a text flle.

limited amount of space

230 Give two examples where a stringstream can be useful.

when parsing lots of text

231 What is a file position?

pointer to a file where to read or write

232 What happens if you position a flle position beyond the end of flle?

depends on the os

233 When would you prefer line-oriented input to type-specific input?

as long as is documented is fine. line oriented seems more tidy

234 What does isalnum(c) do?

function that returns non-zero value if 'c' is digit or alpha (uppercase or lowercase), according to locales

12 A Display Model

12.1 Review

235 Why do we use graphics?

one of the best ways to learn about object oroented programming

236 When do we try not to use graphics?

237 Why is graphics interesting for a programmer?

display information with in more pleasent way, that text

238 What is a window?

window as an object where other shapes or objects are attached

239 In which namespace do we keep our graphics interface classes (our graphics library)?

Graph_lib::

240 What header files do you need to do basic graphics using our graphics library?

simple_window.h, window.h, gui.h, point.h,

241 What is the simplest window to use?

simple_window, is a window with an embedded exit button

242 What is the minimal window?

Window obbject w/o any attachments, defining its width and height

243 What's a window label?

window's name at located at the toolbar

244 How do you label a window?

win.set_label("text here")

245 How do screen coordinates work? Window coordinates? Mathematical coordinates?

upper left corner is (0,0), that corresponds to x,y coordinates. The coordinates represents the number of pixel

246 What are examples of simple "shapes" that we can display?

circle, rectangle, axis,..

247 What command attaches a shape to a window?

win.attach(object_to_attach)

248 Which basic shape would you use to draw a hexagon?

Polygon and member function add(Point(x,y))

249 How do you write text somewhere in a window?

using Text object, with its corresponding coordinates

250 How would you put a photo of your best friend in a window (using a program you wrote yourself) ?

Image image(Point(x,y),"file_name")

251 You made a Window object, but nothing appears on your screen. What are some possible reasons for that?

if missing a wait_on_key(), the window shows up and closes down very fast

252 You have made a shape, but it doesn't appear in the window. What are some possible reasons for that?

probably missing an atachment of the desired object to the Window object

13 Graphics Classes

13.1 Review

253 Why don't we 'just" use a commercial or open-source graphics library directly?

creating a wrapper around the library helps the learning curve of OOP, no matter if its commercial or open source. Also makes it simpler to use

254 About how many classes from our graphics interface library do you need to do simple graphic output?

Simple_window class

255 What are the header files needed to use the graphics interface library?

graph.c, simple_window.h

256 What classes define closed shapes?

closed_polyline

257 Why don't we just use Line for every shape?

that will be more cumbersome, and grouping objects lets apply atributes to the whole group, or object

258 What do the arguments to Point indicate?

x and y coordinates

259 What are the components of Line_style?

according to an enum, lets use dashed lines for instance and also change the width of the line

260 What are the components of Color?

color enum, has predefined for us a palette of colors

261 What is RBG?

red, blue and green, represented a 8 bits each, depending on the color space

262 What are the differences between two Lines and a Lines containing two lines?

Lines inherint from Line. altough similar, Lines let add more Line as you go, which helps for creating grids for instance

263 What properties can you set for every Shape?

set_color, set_fill_color

264 How many sides does a Closed_polyline defined by five Points have?

4 points, as the last one is generated internally

265 What do you see if you define a Shape but don't attach it to a Window?

if it's not attached, it won't be drawn

266 How does a Rectangle differ from a Polygon with four Points (comers)?

constructors are different, Rectangles let you create a rectangle using width and height

267 How does a Polygon differ from a Closed_polyline?

268 What's on top: fill or outline?

outline

269 Why didn't we bother defining a Triangle class (after all, we did define Rectangle)?

that responds as a matter of use, because rectangle are the most used polygon out there

270 How do you move a Shape to another place in a Window?

member function move(dx,dy)

271 How do you label a Shape with a line of text?

use a floating Text object

272 What properties can you set for a text string in a Text?

set_color, set_font, set_font_size

273 What is a font and why do we care?

graphical representation of text, design matters and fonts also does

274 What is Vector_ref for and how do we usc it?

vector_ref holds unnamed objects, use as a vector type

275 What is the difference between a Circle and an Ellipse?

differs that an ellipse are defined by two radius and one for circles

276 What happens if you try to display an Image given a file name that doesn't refer to a file containing an image?

it will display no_image graph in the canvas, plus file not found

277 How do you display part of an image?

using a mask, and specifying which part of the image won't be shown

14 Graphics Class Design

14.1 Review

278 What is an application domain?

graphics is an application domain, where to solve a problem domain to meet client' requirements

279 What are ideals for naming?

describe logical operations

280 What can we name?

public function mmebers that describe operations with objects

281 What services does a Shape offer?

for 'free' we get set_color, draw, draw_lines and set_fill_color

282 How does an abstract class differ from a class that is not abstract?

you can't create a class directly from an abstract, rather than inherit from it

283 How can you make a class abstract?

declare all function members as pure virtual and no member variables

284 What is controlled by access control?

who can access public and private member functions or variables

285 What good can it do to make a data member private?

no direct access to update data members, secures infraudulent changes

286 What is a virtual function and how does it differ from a non-virtual function?

with virtual functions, a function can be overriden by its derived class

287 What is a base class?

also called superclass, class where derived classes inherit from

288 What makes a class derived?

by its declaration as struct derived_class : base_class {}

289 What do we mean by object layout?

how members of a class are stored in memory

290 What can you do to make a class easier to test?

think about it from the beginning, that is as part of the design

291 What is an inheritance diagram?

a diagram that shows all the relationships between base and derived classes

292 What is the difference between a protected member and a private one?

protected members can be used from derived classes, whereas private classes only from the original class or base class

293 What members of a class can be accessed from a class derived from it?

that include public and protected memebers

294 How does a pure virtual function differ from other virtual functions?

pure virtual functions asks the derived class to define a new function, but virtual can be defined if needed, not mandatory

295 Why would you make a member function virtual?

if for a derived class makes sense to redefine a fucntion that differs from the original

296 Why would you make a virtual member function pure?

when you ask for any derived class to define its own function, to keep consistency

297 What does overriding mean?

in a virtual table of a class, any function declared as virtual in the base class has new definitions and so overrides the original function in the virtual table

298 How does interface inheritance differ from implementation inheritance?

when a interface is inherit, means the name of members. An implementation inherits the functionality

299 What is object-oriented programming?

oop stands for encapsulation, inheritance and runtime poly-morphism

15 Graphing functions and data

15.1 Review

300 What is a function of one argument?

301 When would you use a (continuous) line to represent data? When do you use (discrete) points?

continuos lines matches x axis for continuous variables, such as time. Discrete variables get represented by x,y points

302 What function (mathematical formula) defines a slope?

ax+b, where a is slope and b the offset

303 What is a parabola?

$$f(x) = x^2$$

304 How do you make an x axis? A y axis?

Axis x(Axis::x,...)

305 What is a default argument and when would you use one?

when calling a constructor, and a parameter has no value defined, a default value can ease the creation of that object

306 How do you add functions together?

as long as both functions share x variable f(x)= f1(x)+f2(x)

307 How do you color and label a graphed function?

f.set_color, f.set_label

308 What do we mean when we say that a series approximates a function?

taylor series defines that any function can be represented as a sum of finite terms

309 Why would you sketch out the layout of a graph before writing the code to draw it?

focus on the result rather than the implementation

310 How would you scale your graph so that the input will fit?

shrink the y values by a y_scale factor

311 How would you scale the input without trial and error?

set the max value and normalize the output

312 Why would you format your input rather than just having the file contain ”the numbers”?

humans are prone to type wrong characters, where a file does not

313 How do you plan the general layout of a graph? How do you reflect that layout in your code?

define constants that define offsets,...

16 Graphical user Interface

16.1 Review

314 Why would you want a graphical user interface?

to interact with users in a more friendly way

315 When would you want a non-graphical user interface?

interact at low level interfaces

316 What is a software layer?

code that lays in between the user code and hardware, the ease the use of our application

317 Why would you want to layer software?

to focus in the application layer, not bothering with implementation details related to hardware interface

318 What is the fundamental problem when communicating with an operating system from C++?

the os handles calls that might be invisible to our application

319 What is a callback?

in case of an event, the code that gets executed

320 What is a widget?

in a gui, any form of interaction, i.e. button, input box,...

321 What is another name for widget?

control

322 What does the acronym FLTK mean?

fast light toolkit

323 How do you pronounce FLTK?

fulltick

324 What other GUI toolkits have you heard of?

Qt, ultimate++, Kigs, Juce, Noesis GUI, ImGUI, SFML, gtkmm, nana,...

325 Which systems use the term widget and which prefer control?

326 What are examples of widgets ?

buttons, sliders, inboxes, menu,...

327 What is the type of the value stored in an inbox?

input text

328 When would you use a button?

enabling or disabling properties

329 When would you use an inbox?

get input from user

330 When would you use a menu?

allocate more properties in a compact style

331 What is the basic strategy for debugging a GUI program?

test as much as possible on the go, at a later stage, more complicated

332 What is control inversion?

instead of relying on user top-bottom control, the inversion control states that the widget at the bottom, is the one that controls the flow of the user program in a bottom-up style

333 Why is debugging a GUI program harder than debugging an "ordinary program using streams for I/O"?

many calls may be handled by the os itself, making invisible to the user

17 Vector and Free Store

17.1 Review

334 Why do we need data structures with varying numbers of elements?

dynamic space allocation is efficient

335 What four kinds of storage do we have for a typical program?

stack, heap, code, static

336 What is free store? What other name is commonly used for it? What operators support it?

free store is heap memory, new keyword returns a pointer to free store

337 What is a dereference operator and why do we need one?

dereference a pointer *p, means the value stored at address stored in p

338 What is an address? How are memory addresses manipulated in C++?

location where an object is stored. in c++ new and & return addressess

339 What information about a pointed·to object does a pointer have? What useful information does it lack?

a pointer knows which type of object is pointing to, but lacks number of objects ther are

340 What can a pointer point to?

a pointer is an 8-byte, so anything that fits in a memory address

341 What is a leak?

when allocated resources are not free to the system

342 What is a resource?

can be memory, audio channels,...

343 How can we initialize a pointer?

using 'new' operator or '&' operator

344 What is a null pointer? When do we need to use one?

nullptr is zero, so if a pointer does not point to an object, should be NULL

345 When do we need a pointer (instead of a reference or a named object)?

whenever an object is expected to change

346 What is a destructor? When do we want one?

when the object initialized by a constructor gets of scope, a destructor frees resources. Should be defined otherwise a default one is generated

347 When do we want a virtual destructor?

to let derived classes handle therir own resources

348 How are destructors for members called?

they get called when the object is destroyed, automatically

349 What is a cast? When do we need to use one?

rarely need to use cast. reinterpret_cast, tells compiler to identify a variable witha new type; const_cast, removes consta from a variable and static_cast does implicit conversions

350 How do we access a member of a class through a pointer?

using arrow operator

351 What is a doubly-linked list?

knows the previous and successor objects; able to move in both directions

352 What is this and when do we need to use it?

this refers to the object is called

18 Vectors and Arrays

18.1 Review

353 What does "Caveat emptor!" mean?

354 What is the default meaning of copying for class objects?

shallow copy

355 When is the default meaning of copying of class objects appropriate?When is it inappropriate?

appropriate when only reading is fine, but for changes that apply to new copied class, a deep copy is need

356 What is a copy constructor?

by defintion class_name A=B, where B is also class_name

357 What is a copy assignment?

as before but, A=B, assign all elements from B to A

358 What is the difference between copy assignment and copy initialization?

when initialize means always with same values, predefined. assign ant values

359 What is shallow copy? What is deep copy?

shallow only copies the pointer to the object, not the object itself, which is a deep copy

360 How does the copy of a vector compare to its source?

deep copy

361 What are the five ”essential operations” for a class?

constructor(default, one or more arguemts), destructor, copy constructor, copy assignment

362 What is an explicit constructor? Where would you prefer one over the (default) alternative?

when its stated that for a new object, the arguments are forced to follow its constructor(no conversions allowed). To avoid implicit conversions.

363 What operations may be invoked implicitly for a class object?

constructors

364 What is an array?

old c-style for allocating elements

365 How do you copy an array?

element by element

366 How do you initialize an array?

element by element

367 When should you prefer a pointer argument over a reference argument? Why?

when changing the object the pointer points to a pointer, otherwise a reference will suffice

368 What is a C-style string?

array of chars terminated with escape character 0x00

369 What is a palindrome?

read from left to right or viceversa, the result is the same

19 Vectors, Templates, Exceptions

19.1 Review

370 Why would we want to change the size of a vector?

hard to know how much input data we need to store for further processing

371 Why would we want to have different element types for different vectors?

as a vector can handle a single type in the same vector, define different vector for input types is much needed

372 Why don't we just always define a vector with a large enough size for all eventualities?

we could, but we have to share finite resources

373 How much spare space do we allocate for a new vector?

usually, 2 times the size of initialization

374 When must we copy vector elements to a new location?

when resizing a vector, copy old data to new destination

375 Which vector operations can change the size of a vector after construction?

for vector type, we have member functions: push_back( ), resize ( )

376 What is the value of a vector after a copy?

we have to delete old vector to allocate old space back to the system

377 Which two operations define copy for vector?

copy constructor(T¡¿(elem)) and copy assignment (T& operator=(elem))

378 What is the default meaning of copy for class objects?

deep copy, that is copy data members and pointers

379 What is a template?

template is a mechanism that let use types as parameters for classes and functions

380 What are the two most useful types of template arguments?

1. use templates where performance is essential (i.e. hard real time, numerics)
2. use tempaltes where flexibility in combining information forom several types is essential (i.e. C++ STL,..)

381 What is generic programming?

write code that works with a variety of input types as parameters, as long as those types meet specific syntatic and semantic requirements

382 How does generic programming differ from object-oriented programming?

oop relates to class hierarchies and virtual functions, and generic programming to templates. Also generic programming is determined at compile time, and oop at run time

20 Containers and Iterators

20.1 Review

383 Why does code written by different people look different? Give examples.

although different techniques can be used to handle or store data, the core of the program must be similar; differences come from effiency, i.e. using different set of containers, memory allocators,..

384 What are simple questions we ask of data?

type, size, constant or volatile, operations to perform with that data,..

385 What are a few different ways of storing data?

using containers such as: vector, list, array, map,...

386 What basic operations can we do to a collection of data items?

operators depends on conatiner's type, but for basic operators: */-¿,[],++/−

387 What is an STL sequence?

388 What is an STL iterator? What operations does it support?

the standard library provides ways of accessing data with iterators, which is a pointer with enhanced capabilities; provides dereference, increment and decrement

389 What are some ideals for the way we store our data?

constant size with constant values, that lat for the entire program execution

390 How do you move an iterator to the next element?

pre/post increment '++', advance(iterator,+positions), next(interator,+positions)

391 How do you move an iterator to the previous element?

pre/post decrement '–', advance(iterator,-positions), next(interator,-positions)

392 What happens if you try to move an iterator past the end of a sequence?

iterator will move, no problem, problem is when dereference that iterator

393 What kinds of iterators can you move to the previous element?

only bidirectional and random iterators

394 Why is it useful to separate data from algorithms?

might be interesting to build a library or reuse at some point, but if data is held in tha algorithm, much harder to reuse after

395 What is the STL?

The acronym of standard library, where containers, types,... are defined and ready to use

396 What is a linked list? How does it fundamentally differ from a vector?

linked list uses more memory that standard array because stores pointers to previous and successor elements, provides iterator to access elements, not subscripting is allowed

397 What is a link (in a linked list)?

a pointer to previous or successor, if bidirectional linked list

398 What does insert() do? What does erase() do?

in alink provides an efficient method function to add a new element to the list, so resulting list size increments by one element; the oposite to erase

399 How do you know if a sequence is empty?

when begin() matches end()

400 What operations does an iterator for a list provide?

+,++,-,–,*,-¿

401 How do you iterate over a container using the STL?

start from begin() until end() and increment iterator++

402 When would you use a string rather than a vector?

deal with words massively, provides concatenation by default

403 When would you use a list rather than a vector?

when a big number of elements are to be accessed and operated

404 What is a container?

in the free memory, store data that contains data

405 What should begin() and end() do for a container?

provide methods to access the first and last element of the container

406 What containers does the STL provide?

vector, list, deque, map, multimap, unordered_map, unordereed_multimap, set, multiset, unordered_set, unordered_multiset, array

407 What is an iterator category? What kinds of iterators does the STL offer?

categories in the sense of how to access data; input iterator, output iterator, forward iterator, bidirectional iterator and random access iterator

408 What operations are provided by a random-access iterator, but not a bidirectional iterator?

subscripting and add/substract integer to iterators

21 Algorithms and Maps

21.1 Review

409 What are examples of useful STL algorithms?

copy, find, sort, min, max, accumulate, inner_product,...

410 What does find() do? Give at least five examples.

find first occurence in a given container of a satiisfying criteria: equals a value, satisfies a predicate or negated predicate

411 What does count_if() do?

given a container, counts elements for a satisfied criteria

412 What does sort(b,e) use as its sorting criterion?

less '¡': b ¡ e

413 How does an STL algorithm take a container as an input argument?

uses iterators to container

414 How does an STL algorithm take a container as an output argument?

using p* or -¿ and p++, to move through iterator container

415 How does an SIL algorithm usually indicate "not found" or "failure"?

returns end() iterator, which is last+1 element location

416 In which ways does a function object differ from a function?

function objects lets store partial summs or accumulators, inner_product

417 What is a function object?

class that implements the operator(), and can be used as predicate for several algorithms

418 What is a predicate?

a conditional that satisfies as bool

419 What does accumulate() do?

performs sum of a container elements

420 What does inner_product() do?

given two input containers, performs the element-wise product for each element of the conatiners, and adds them up

421 What is an associative container? Give at least three examples.

refers to pair values stored as key: value. i.e. map container, set container, unordered_map, ...; sorted data structure with complexity O(log(n)) with balanced binary tree representation

422 Is list an associative container? Why not?

list refers to linked elements with predecessor and successor; that is a sequence container

423 What is the basic ordering property of binary tree?

set to balanced

424 What (roughly) does it mean for a tree to be balanced?

the amount of time required to find elements is roughly the same for every object

425 How much space per element does a map take up?

8 byte left + 8 byte right + 8 byte key + 8 byte datum

426 How much space per element does a vector take up?

8 byte datum

427 Why would anyone use an unordered_map when an (ordered) map is available?

access time for unordered_map is constant for every element in an ordered map, is a log(n)

428 How does a set differ from a map?

set key are unique and elements are stores as sorted

429 How does a multi_map differ from a map?

multi refers to multiple entries with the same key

430 Why use a copy() algorithm when we could ”just write a simple loop”?

it implements range checking and efficient / performance techniques

431 What is a binary search?

tests to be true or false according to a predicate

22 Embedded Systems Programming

22.1 Review

432 What is an embedded system? Give ten examples, out of which at least three should not be among those mentioned in this chapter.

system with limited resources, i.e. memory, data storage, cpu,... tv remote, spectrum analyzer, automatic gate opener

433 What is special about embedded systems? Give five concerns that are common.

runs in remote places, often unaccessible, runs 24/7, harsh environments, resource limitations, cpu w/o mmu,

434 Define predictability in the context of embedded systems.

predictabilty for executing code that takes the some amount of time, for hard real time, where time is a constrint, it’s not an option

435 Why can it be hard to maintain and repair an embedded system?

depends on where the system is placed, so little maintainance is possible

436 Why can it be a poor idea to optimize a system for performance?

if performace optimization means low-level programming, hard to maintain

437 Why do we prefer higher levels of abstraction to low-level code?

prefered dur to better maintainance, portability, reusability

438 What are transient errors? Why do we particularly fear them?

errors that occur from time to time, hard to find and debug

439 How can we design a system to survive failure?

replicate some parts of the system to ensure if a safe state is possible

440 Why can't we prevent every failure?

perfection is a true enemy

441 What is domain knowledge? Give examples of application domains.

area of expertise, where subtle differences apply. medical, harsh industrial domains, aerospacial, automotive,..

442 Why do we need domain knowledge to program embedded systems?

every domain has diffferent requirements and constraints

443 What is a subsystem? Give examples .

parts of a system when combined as a whole, becomes the system. memory storage, communications, GUI,...

444 From a C++ language point of view, what are the three kinds of storage?

heap, stack or static memory

445 When would you like to use free store?

better not to use it

446 Why is it often infeasible to use free store in an embedded system?

when predictability is a concern

447 When can you safely use new in an embedded system?

best at the startup of the system, allocating resources but don't use delete to prevent fragmentation

448 What is the potential problem with std::vector in the context of embedded systems?

for hard real time, vector uses indirectly free store, that is new and delete

449 What is the potential problem with exceptions in the context of embedded systems?

no predictability is possible, due to how long does it take to catch the error, and besides that, who's going to read the error?

450 What is a recursive function call? Why do some embedded systems programmers avoid them? What do they use instead?

recusive function calls itself during a loop, if the loop is to large may show as a stack overflow, better to use a traditional for-loop.

451 What is memory fragmentation?

in the free store, using new and delete, may show holes that belong at some point to an object, but once deleted, the space becomes available. If a new object asks for new space, may allocate new space further than the recently deleted one, so leaves behind a hole of memeory and in the long run, causing a memory fragmentation hard to solve

452 What is a garbage collector (in the context of programming)?

tnds to solve the fragmentation issue, but needs more meory to alocate the system of garbage collector, often not an option in an embedded system

453 What is a memory leak? Why can it be a problem?

in a program taht runs indefinetely, a leak may cause the memory to grow upto unsustainable limits, causing the program to crash

454 What is a resource? Give examples.

shared subsystems in a embedded system that must be shared along the pogram execution. memory, processor cycles(time) , power

455 What is a resource leak and how can we systematically prevent it?

unmanaged code that uses memory, better not to use it

456 Why can't we easily move objects from one place in memory to another?

pointers refers to objects in memory directly, so moving objects leds to those pointers become invalid, thus we should move the pointers too, which is a garbage collector

457 What is a stack?

data structure that allocates an arbitrary amount of memory and deallocate the last allocation only; that prevents from memory fragmentation

458 What is a pool?

collection of objects of the same size, we can allocate and deallocate as much elements as we need but limited by the pool size. no fragmentation occurs as the objects are of the same size

459 Why doesn't the use of stacks and pools lead to memory fragmentation?

preventing holes between the allocated objects

460 What is reinterpret_cast necessary? Why is it nasty?

when dealing with low-level hardware defintion addresses, an unchecked way to link between the application and specific hardware resources is to cast. That needs to be done with the manual open and manually point to specific addresses

461 Why are pointers dangerous as function arguments ? Give examples.

if dealig with objects, when a subclass size is of different size, a pointer might not point to the beginning of the object as expected. let's say a circle and polygon are subclasses of shape class; defining a pointer to an array of shape objects, using oop means that pointers for polygon and circle are of the same size as shape, that is barely true. better use of references or some defined array class that prevents mislading pointers

462 What problems can arise from using pointers and arrays? Give examples.

along with the pointer we must explicitely supply a size, which might cause a potential problem. also leds the opportunity to pass a pointer as a result of an implicit conversion from an array of the derived class to a pointer of the base class

463 What are alternatives to using pointers (to arrays) in interfaces?

use an interface class that reinterprets an array

464 What is "the first law of computer science"?

don't do it

465 What is a bit'?

smallest representation in a memory, only to states 0 or 1

466 What is a byte?

a collection of 8 bits

467 What is the usual number of bits in a byte?

1 byte

468 What operations do we have on sets of bits?

&,|,^,~, <<, >>

469 What is an exclusive or and why is it useful?

bit operator that outputs 0 when a and b bitwise are equal, 1 otherwise

470 How can we represent a set (sequence, whatever) of bits?

bitset

471 How do we conventionally number bits in a word?

LSB is bit 0 and MSB is far right bit

472 How do we conventionally number bytes in a word?

LSB byte is right hand side, anf MSB left hand side

473 What is a word?

depending on a system is equivalent to two bytes or 4 bytes

474 What is the usual number of bits in a word?

16 bits

475 What is the decimal value of Oxf7?

16+32+64+128+7=247

476 What sequence of bits is Oxab?

0b10101011

477 What is a bitset and when would you need one?

bit representation, for outputting numbers with bits

478 How does an unsigned int differ from a signed int?

msb is dedicated to indicate +(0) or -(1) in a signed variable

479 When would you prefer an unsigned int to a signed int?

allocate more space, no need of negative numbers

480 How would you write a loop if the number of elements to be looped over was very high?

ensure the for-loop variable iterator is appropriate

481 What is the value of an unsigned int after you assign -3 to it?

the same as max int representation -3

482 Why would we want to manipulate bits and bytes (rather than higher level types) '?

when accessing low-level resources, i.e. registers, must set or unset bits

483 What is a bitfield?

specific representation of bits whitin a int number, specified with its location and numner of bits

484 For what are bitfields used?

register manipulation