**Computer language (binary)**

**Bit-**smallest unit info processed computer.

Value either 1 or 0.

A computer reads and writes in bits.

**Binary to Decimal**

Multiply by power of 2 for each digit then add. 1010 = 23+21. 20 counts.

**Decimal to Binary**

Take #, divide by two, take remainder until 0. 46/2, 23/2, etc. Read from bottom to top

**Data Coding**

ASCII, UNICODE, Image encoding-red, blue, green, 1 byte/color. Sound encoding-mp3 measure sound 44100x/sec

**Algorithms and pseudo code**

**Algorithm-**step-by-step approach for performing a task, or solving a problem.

Is not written in a programming language, can be translated easily.

May already be defined and ready to use.

**Pseudo code-**most common way to show the steps of an algorithm.Description of some algorithm.

Similar to programming language but intended for human reading

Typically omits details that are not essential for human understanding of the algorithm.

Used for sketching out the structure of a program before the actual coding takes place.

Document any assumptions made about the algorithm.

**Software lifecycle**

**Software lifecycle**-organized approach to developing software.

**Waterfall model-**began around 1970.

Based on models from the manufacturing and construction industries. Plan, Anlys, design, code, test, maint.

**Agile model-**published in 2001.

An iteration lasting from 2-4 weeks.

A daily “standup” meeting to report status and potential issues.

**Program components**

**Program**-sequence of computer instructions that: Executes very fast. Does exactly what the instructions say.

**Function**-named group of statements that perform a task.

Program-one or more functions.

**Statement-**single instruction that Consists of one or more lines.

Function-one or more statements.

**Syntax**-set of rules determining what a legal programming statement is.

**Token-**The smallest sequence of characters defined by the syntax.

The smallest element of any programming language.

End of token can’t be determined until at least one character is read past the token.

Statement-one or more **tokens**.

**Comments**

**Comment**-note that describes corresponding code.

**Header comment**-appears at the top of a program.

**Detail comments**-appears throughout the body of a program.

**Whitespace (spacing, indentation)**

Whitespace includes spaces, tabs, and blank lines. Whitespace is critical to making a program more readable.

**Preprocessor directives**

Library-collection of functions written to perform a related set of tasks.

#include statement connects a header file, that represents a library, to an application.

**Variables**

**Variable**-named spot in memory used to hold a value. 3 characteristics: Name, Data type, Value

Variable name: Begins with a letter or underscore. Is followed by zero or more letters, digits, and underscores

**Data types**

Int, char, short-stores integers

Float, double-stores real #

Max int-int\_max or 231.

Bool-true or false (1/0)

**String literal**-sequence of zero or more characters enclosed in double quotes.

**Character literal**-single character enclosed in single quotes.

**Arithmetic operators**

+ - \*/%

/-whole numbers unless one cast as double

%-takes remainder

**Casting / data type conversions**

**Casting**-value changed from one data type to another.

Automatically by C++–**implicit casting**. Coded by the programmer-**explicit casting**

String/Char can’t be cast. Int, double, char implicit. Int to char explicit (char)

**Constants**

**Constant-**identical to a variable except cannot be changed running.

Begins with a letter or underscore.

Is followed by zero or more letters, digits, and underscores.

const <data-type> <constant-name> = <value>;

cout<< <expression><<endl;

setw(length), left, right, setfill(‘char’), fixed<<setprecision(#)

cin>>(variable)

Skips all past tabs, spaces, until enter key. bTcBBTR = b

Cin.peek()

Getline(cin, <variable>)

cin.ignore scans past the next ## characters or through the next char, whichever comes first.

cin.ignore(<integer-expression>, <char-expression>)

Clears input buffer

**Conditions**

Either TRUE or FALSE.

Boolean expression.

Determines the flow of processing

Enables an application to have multiple execution pathways.

Makes app more flexible and capable.

Uses relational and logical operators.

Used in conditional (if) and iterative (while, for) statements.

And-one or all false, result false

Or-all false, result false

Not-opposite value

if (<condition-1>) { <block-1> }

else if (<condition-x>) { <block-x> }

else { <block-default> }

switch (<expression>)

{ case <value-1> <block-1> break;

case <value-x>: <block-x> break; default: <block-default> break; }

While statement-pre-test loop: condition tested *before* the block of code executes. Indefinite iteration: the number of times it loops is unknown ahead of time. Loops zero or more times. Continues to execute while condition true.

While(<cond>)

{ <block> }

**Infinite loop**.

Every loop must have these three parts:

Initialization, condition, update

Any of these missing or incorrect, infinite loop will result.

**Validation loops**

Reads and checks values entered at the keyboard. Continues to loop until the user enters a valid value.

<promt>

While(<test-for-invalid-input>)

{ <error> <prompt> }

**Sentinel loops**

reads input until a special value read from the keyboard.

Do-while-test cond at end of loop.

Never use continue.

Only use break in switch.