1. **References**. You can pass arguments as pointers (**int** \*), references (**int** &), or as copies (**int**). References are dangerous because it's easy to mistake them for copied parameters in function bodies. You modify the variable within the routine to keep from allocating and initializing a copy, only to find that your caller is buggered on exit. These bugs can be very difficult to find.
2. **Const**. Declaring arguments and values as **const** is very powerful for enforcing code discipline. However, it can be very expensive when you realize something needs to be mutable far down the call stack. And, it can be confusing working with **const** pointers: is the pointer or the value const? Or both?
3. **Overloaded operators**.  You can create implementations of operators like '+' and '\*' in your classes. It makes code look beautiful and intuitive when done right. Unfortunately, that intuition can be different for different people - wrong for some of them. It can lead to unintended behaviors which are very difficult to find.
4. **Templates**. Templates are an incredibly convenient way to generalize code and operations. Instead of having one class for binary string trees and another for binary int trees, you can have a template and specialize it. Beautiful! But oh dear, what about templates with templates as arguments which define more templates? Or implicitly specialized templates? I got my fill of these working in Flume and related technologies at Google. Template can make code behavior unintuitive, difficult to debug, and painful to maintain.

|  |  |
| --- | --- |
| int main ()  {  double jungle;  jungle = 8 \* 7 - 3 + 4 / 3;  cout << jungle;    return 0;  } | 54  Read from Right to Left PEMDAS style   1. 4/3 2. + 3. 8\*7 4. -3 5. = 6. 54 |
| // setprecision example  #include <iostream> // cout, fixed  #include <iomanip> // setprecision  int main () {  double f =3.14159;  cout << setprecision(5) << f << '\n';  cout << setprecision(9) << f << '\n';  cout << fixed;  cout << setprecision(5) << f << '\n';  cout << setprecision(9) << f << '\n';  return 0;  } | **EXCLUDES DECIMAL POINT!!!!!!!!**  3.1416 -----------------------**5 in total**  3.14159 ----------------------**9 in total**  **ADDS 0s IN BLANK SPACES!!!!!!!!!!!**  3.14159 ----------------------**5 after decimal**  3.141590000 ----------------**9 after decimal** |
| // setw example  #include <iostream> // cout, endl  #include <iomanip> // setw  int main () {  cout << setw(10);  cout << 77 << endl;  return 0;  } | --------77  “----------” 8 blank characters |
| **<< fixed << showpoint << setprecision(4);**  // modify showpoint flag  #include <iostream> // cout, showpoint, noshowpoint  int main () {  double a = 30;  double b = 10000.0;  double pi = 3.1416;  cout.precision (5);  cout << showpoint << a << '\t' << b << '\t' << pi << '\n';  cout << noshowpoint << a << '\t' << b << '\t' << pi << '\n';  return 0;  } | Numbers of 0s after the decimal point---4 in this case  **SHOWPOINT SHOWS (.) and 0s IF IT FITS WITHIN THE PRECISION**  30.000 10000. 3.1416  **NOSHOWPOINT DOESN’T SHOW (.) and 0s AFTER**  30 10000 3.1416 |
| **y += x;**  **++x**  **X = 3**  **Y = ++x**  **== | != | < | > | >= | <=**  **&& | || | !** | y = y + x;  x = x + 1  x = 3; y = 4  equal to | not equal to | less | greater | GT or = | LT or =  Truth Table and or negation |
| **DATA TYPES:**  **Character---char**  **Integer-------int**  **Float---------float**  **Boolean-----bool**  **Void---------void**  **String-------string** | Char--- ‘a’  Int---7  Float; double; long double;----3.564  Bool---true/false  Void  String--- “this is a string” |
| STATIC\_CAST (static\_cast<data type>(value)  =static\_cast<int>(area/500);  =static\_cast<double>(sum)/count;  #include <iostream>  using namespace std;  int main()  {      float f = 3.5;      int a = f; // this is how you do in C      int b = static\_cast<int>(f);      cout << b;  } | Since the inside of the parenthesis will give an *int* value  the data type inside of the brackets must be an *int* value  Since only the *sum* that is a double value, is inside the parenthesis, the *double* must exist inside the brackets  3 |
| **NAMED CONSTANTS, (CONSTANT VARIABLES)**  #include <iostream>  using namespace std;  const double pi = 3.14159;  const char newline = '\n';  int main ()  {  double r=5.0; // radius  double circle;  circle = 2 \* pi \* r;  cout << circle;  cout << newline;  } | 31.4159  -  Stays the same throughout the entire code |
| **cin>>x** | getline. (cin, x) |
| IMPORTANT HEADERS WITH EACH FUNCTION USED IN CLASS |  |
| #include <fstream>  test for end of file:  while (inFile >> score)  sum += score; | Used when using files from a specified folder and importing/exporting it to/from a file   1. Use Header   |🡪input: ifstream inFile;   1. Create object==   |🡪output: ofstream outFile;   1. Open file=🡪 in/outFile.open(file.txt)   |🡪inFile>> partnum; (LIKE cin)   1. Use file==   |🡪outFile<< “Report” (LIKE cout)   1. Close file ==out/inFile.close() |
| #include <string>  string.length()  string.assign() | #include <string>  #include <iostream>  int main()  {  string s = "a string";  cout << s << endl;  } |
| #include <iomanip> | Setw();  Setprecision()  Fixed;  Showpoint(): |
| #include <cmath> | Pow(r,2)  Max(x,y)  Min(x,y)  Sqrt(64)  Round(2.6)  Log(2)  Abs(x) |
| #include <vector> | 1. Use header 2. Declare vector --- vector<data type> X; 3. Use like an array |
| DATA VALIDATION  int num;  cout << “Enter a number”;  while (!!!!!!!!!(cin >> number))  {  cout << “ERROR: Please enter a number”;  cin.clear();  cin.ignore(123,’\n’);  }  cout << “Favorite number is:” << number;  return 0; | Enter a number:  If not a number, then  ERROR: Please enter a number  If it is a number, then  Favorite number is: X |
| SWITCH CASES EXAMPLE #1  1)  #include <iostream>  #include <cmath>  using namespace std;  int main() {  double radius, base, height, length, width, area;  const double pi = 3.14159;  int choice;  cout << (" Geometry Calculator\n\n");  cout << ("1. Calculate the Area of a Circle.\n");  cout << ("2. Calculate the Area of a Rectangle.\n");  cout << ("3. Calculate the Area of a Triangle.\n");  cout << ("4. Quit.\n\n");  cout << ("Enter Your Choice: ");  cin >> choice;  switch (choice) {  case 1:  cout << "What is the radius of the circle?";  cin >> radius;  area = pi \* pow(radius, 2);  cout << "The area of the circle is " << area << ".";  break;  case 2:  cout << "What is the length of the rectangle?";  cin >> length;  cout << "What is the width of the rectangle?";  cin >> width;  area = length \* width;  cout << "The area of the rectangle is " << area << ".";  break;  case 3:  cout << "What is the base of the triangle?";  cin >> base;  cout << "What is the height of the triangle?";  cin >> height;  area = 0.5 \* base \* height;  cout << "The area of the triangle is " << area << ".";  break;  case 4:  cout << "You have chosen to quit the program. :(";  break;  default:  cout << "Please enter a valid option from the menu!!!\n";  cout << "Valid options include 1 - 4.";  }  system("pause>nul");  return 0;  }  SWITCH CASES EXAMPLE #2  2)  #include <iostream>  #include <iomanip>  using namespace std;  int main() {  int choice;  double feet, time;  cout << " Mediums Available:\n\n";  cout << "1. Air\n";  cout << "2. Water\n";  cout << "3. Steel\n\n";  cout << "Please choose an option from the menu: ";  cin >> choice;  cout << endl;  cout << "Enter the number of feet the sound wave will travel: ";  cin >> feet;  cout << endl;  switch (choice) {  case 1:  time = feet \* 1100;  cout << "A soundwave traveling through Air at a distance of " << feet;  cout << " feet, will take about " << fixed << showpoint << setprecision(4);  cout << time << "seconds.";  break;  case 2:  time = feet \* 4900;  cout << "A soundwave traveling through Water at a distance of " << feet;  cout << " feet, will take about " << fixed << showpoint << setprecision(4);  cout << time << "seconds.";  break;  case 3:  time = feet \* 16400;  cout << "A soundwave traveling through Steel at a distance of " << feet;  cout << " feet, will take about " << fixed << showpoint << setprecision(4);  cout << time << "seconds.";  break;  default:  cout << "Invalid choice!!!\n Please enter a valid number.";  }  system("pause>nul");  return 0;  } | Geometry Calculator  1. Calculate the Area of a Circle.  2. Calculate the Area of a Rectangle.  3. Calculate the Area of a Triangle.  4. Quit.  Enter Your Choice: X  1  What is the radius of your circle? X  The area of your circle is (pi \* X^2)  2  What is the length of your rectangle? X  What is the width of your rectangle? Y  The area of your rectangle is (X \* Y)  3  What is the length of your triangle? X  What is the base of your triangle? Y  The area of your triangle is (1/2 \* X \* Y)  4  You have chosen to quit the program |
| #include <iostream>  **CHAPTER 2**  #include <string>  using namespace std;  int main() {  string response;  int quit, days, total;  double dailyRate, chargeService, chargeMedication;  cout << "Were you an inpatient or outpatient during your last stay? "  << "(or 'q' to quit): ";  while ((response != "Inpatient" && response != "inpatient") || (response != "Outpatient" && response != "outpatient"))  {  cin >> response;  if (response == "q")  {  cout << "You have quit the program!";  }  else if (response == "Inpatient" || response == "inpatient")  {  cout << "How many days did you spend at the Hospital?: ";  cin >> days;  cout << "What was the daily rate for your stay?: $";  cin >> dailyRate;  cout << "What were the charges for the services in the Hospital?: $";  cin >> chargeService;  cout << "What were the charges for the medications in the Hospital?: $";  cin >> chargeMedication;  total = (days \* dailyRate) + chargeService + chargeMedication;  cout << "The total for your stay is $" << total;  }  else if (response == "Outpatient" || response == "outpatient") {  cout << "What were the charges for the services in the Hospital?: $";  cin >> chargeService;  cout << "What were the charges for the medications in the Hospital?: $";  cin >> chargeMedication;  total = chargeService + chargeMedication;  cout << "The total for your stay is $" << total;  }  else  cout << "Please enter a valid input!";  }  system("pause>nul");  return 0;  } |  |

## Fill in the blank 1-20

## CHAPTER 5

1. To **Increment** a value means to increase it by one.
2. To **Decrement** a value means to decrease it by one.
3. When the increment or decrement operator is placed before the operand (or to operand’s right), the operator is being used in **Prefix** mode.
4. When the increment or decrement operator is placed after the operand (or to operand’s right), the operator is being used in **Postfix** mode.
5. The statement or block that is repeated is known as the **Body** of the loop.
6. Each repetition of a loop is known as a(n) **Iteration**.
7. A loop that evaluates its test expression before each repetition is a(n) **Pretest** loop.
8. A loop that evaluates its test expression after each repetition is a(n) **Posttest** loop.
9. A loop that does not have a way of stopping is a(n) **Infinite** loop.
10. A(n) **Counter** is a variable that “counts” the number of times a loop repeats.
11. A(n) **Running** **Total** is a sum of numbers that accumulates with each iteration of a loop.
12. A(n) **Accumulator** is a variable that is initialized to some starting value, usually zero, and then has numbers added to it in each iteration of a loop.
13. A(n) **Sentinel** is a special value that marks the end of a series of values.
14. The **For** **Loop** is ideal for situations that require a counter.
15. The **Do**-**While** **Loop** always iterates at least once.
16. The **While** **Loop** and **For** **Loops** will not iterate at all if their test expressions are false to start with.
17. Inside the for loop’s parenthesis, the first expression is the **Initialization**, the second expression is the **Test**, and the third expression is the **Update**.
18. A loop inside another is called a(n) **Nested** **Loop**.
19. The **Break** statement causes a loop to terminate immediately.
20. The **Continue** statement causes a loop to skip the remaining statements in the current iteration.

## Algorithm Workbench: 33-34-35-36

1. while 🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪 do-while

|  |  |
| --- | --- |
| char doAgain = 'y';  int sum = 0;  cout << "This code will increment sum 1 or more times.\n";  while ((doAgain == 'y') || (doAgain == 'Y'))  {  sum++;  cout << "Sum has been incremented. Increment it again (y/n)? ";  cin >> doAgain;  }  cout << "Sum was incremented " << sum << " times.\n"; | char doAgain = 'y';  int sum = 0;  cout << "This code will increment sum 1 or more times.\n";  do {  sum++;  cout << "Sum has been incremented. Increment it again (y/n)? ";  cin >> doAgain;  } while ((doAgain == 'y') || (doAgain == 'Y'));  cout << "Sum was incremented " << sum << " times.\n"; |

Int time = 20; (if statement) ? if statement : else statement;

**Shorthand if-else** string result **=** (time > 18) ? “Good Morning.” : “Good Evening.”;

cout << result;

1. do-while 🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪 while (no *if* statement)

|  |  |
| --- | --- |
| int number;  cout << "Enter an even number: ";  do  {  cin >> number;  if (number % 2 != 0)  cout << "Number must be even. Reenter number: ";  } while (number % 2 != 0); | int number;  cout << "Enter an even number: ";  cin >> number;  while (number % 2 != 0)  {  cout << "Number must be even. Reenter number: ";  cin >> number;  } |

1. while 🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪 for

|  |  |
| --- | --- |
| int count = 0;  while (count < 50)  {  cout << "count is " << count << endl;  count++;  } | for (int count = 0; count < 50; count++)  {  cout << "count is " << count << endl;  } |

1. for 🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪 while

|  |  |
| --- | --- |
| for (int x = 50; x > 0; x--)  {  cout << x << " seconds to go.\n";  } | int x = 50;  while (x > 0)  {  cout << x << " seconds to go.\n";  x--;  } |

## Predict the Output: 39-42

1. input 🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪 output

|  |  |
| --- | --- |
| int x = 1;  while (x < 10);  x++;  cout << x; | [blank]  while (x < 10);🡨  Because of the semi colon after the while condition. |

1. input 🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪 output

|  |  |
| --- | --- |
| int x = 1;  while (x < 10)  x++;  cout << x; | [10]  { statements }  Only displays 10 because there are no braces before the statements and after. It only runs the last line because with no braces, it only reads one line where it is true. |

1. input 🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪 output

|  |  |
| --- | --- |
| for (int count = 1; count <= 10; count++)  { cout << ++count << " "; //This is a bad thing to do!  } | [2 4 6 8 10]  With the initialization being 1, the statement adds 1 so then it becomes even, then the update adds another so now it is odd until the loop iterates again so it only displays even integers. |

1. input 🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪🡪 output

|  |  |
| --- | --- |
| for (int row = 1; row <= 3; row++)  {  cout << "\n$";  for (int digit = 1; digit <= 4; digit++)  cout << '9';  } | [$9999  $9999  $9999]  The outer loop iterates first because its statement is outside of the inner loop and inner loop iterates next. |

**CHAPTER 2**

|  |  |
| --- | --- |
| 1 | * Whole program is a comment; \*/→/\* to /\*→\*/ * iostreamis missing *open, closed brackets;* < > * int main( ); does not need the semicolon ; * Wrong side of *open, close braces;* } to { * Lines after *main* are not indented * Statements 6,*7,8,9,10* do not have semicolons ; * *Open, close braces* should be flipped; >> to << * Cout should not be capitalized, so cout * C; does not exist, so change it c; * Wrong side of *open, close braces;* { to } |
| 2 | * It will display the value of *freeze,* which is 0. Also, the value of *boil,* which is 100.   0  100   * It will display the value of *x,* which is 8.   Also, the value of *y*, which is 2.  8  2   * It will display, *I am the incredible computing machine and I will amaze you.*   I am the incredible computing  machine  and I will  amaze  you. |
| 3. Rewrite the follow statement to use the newline escape character, instead of an endl, each time subsequent output is to be displayed on a new line.  cout << "L" << endl  << "E" << endl  << "A" << endl  << "F" << endl; | cout << "L\nE\nA\nF”; |
| 4. One acre of land is equivalent to 43,450 square feet. Write a program that calculates and displays the number of acres in a tract of land whose size is 869 X 360 feet. | 1. #include <iostream> 2. using namespace std; 3. int main() 4. { 5. int l = 869; //One side of the tract of land. 6. int w = 360; //Other side of the tract of land. 7. int acre = 43450; //The size of one acre in square feet. 8. int size\_of\_land = l \* w; //Calculates 869 \* 360 for total in square feet. 9. int acres\_in\_land = size\_of\_land / acre; //Calculates 312,840 / 43,450 for number of acres. 10. cout << "There exists " << acres\_in\_land << " acres."; //Displays the number of acres in the tract of land. 11. return 0; 12. } |
| 5. A particular employee earns $39,000 annually. Write a program that determines and displays what the amount of his gross pay will be for each pay period if he is paid twice a month (24 pay checks per year) and if he is paid bi-weekly (26 checks per year). | 1. #include <iostream> 2. using namespace std; 3. int main() 4. { 5. int earned = 39000; //How much he is paid yearly. 6. int x = 24; //Number of weeks if he is paid twice a month. 7. int y = 26; //Number of weeks if he is paid bi-weekly. 8. int scenario1 = earned / x; //Amount earned, 39,000 / 24 weeks that he is paid. 9. int scenario2 = earned / y; //Amount earned, 39,000 / 26 weeks that he is paid. 10. cout << scenario1 << " dollars every paycheck for 24 weeks.\n" << scenario2 << " dollars every paycheck for 26 weeks."; // Displays amount that he is payed in each scenario respectively. 11. return 0; 12. } |
| #include <iostream>  using namespace std;  int main() {  int numBooks;  double bookPoints;    cout << "How many books have you purchased this month?";  cin >> numBooks;  if (numBooks >= 4)  bookPoints = 50;  else if (numBooks == 3)  bookPoints = 30;  else if (numBooks == 2)  bookPoints = 15;  else if (numBooks == 1)  bookPoints = 5;  else if (numBooks == 0)  bookPoints = 0;  else  bookPoints = 0;  cout << "Since you purchased " << numBooks << " books this month,”;  cout << “you earned you a total of " << bookPoints << " points.";  system("pause>nul");  return 0;  } | How many books have you purchased this month? X  If X == >4 then  You earned 50 points  If X == 3 then  You earned 30 points  If X == 2 then  You earned 15 points  If X == 1 then  You earned 5 points  If X == 0 then  You earned 0 points  If X == ANTHING ELSE (!!!!!(0 <= X >= 4) then  You earned 0 points |

**CHAPTER 6**

1. The **function header** is the part of a function definition that shows the function name, return type, and parameter list.
2. If a function doesn’t return a value, the keyword **void** will appear as its return value.
3. If function showValue has the following header: void showValue(int quantity) you would use the statement **showValue(5)** to call it with the argument 5.
4. Either a function’s **definition** or its **prototype** must precede all calls to the function.
5. Values that are sent into a function are called **arguments**.
6. Special variables that hold copies of function arguments are called **parameters**/
7. When only a copy of an argument is passed to a function, it is said to be passed by **value**.
8. A(n) **function prototype** eliminates the need to place a function definition before all calls to the function.
9. A(n) **local** variable is defined inside a function and is not accessible outside the function.
10. **Global** variables are defined outside all functions and are accessible to any function within their scope.
11. **Global** variables provide an easy way to share large amounts of data among all the functions in a program.
12. Unless you explicitly initialize numeric global variables, they are automatically initialized to **zero**.
13. If a function has a local variable with the same names as a global variable, only the **local** variable can be seen by the function.
14. **Static** local variables retain their value between function calls.
15. The **return** statement causes a function to end immediately.
16. **Default** arguments are passed to parameters automatically if no argument is provided in the function call.
17. When a function uses a mixture of parameters with and without default arguments, the parameters with default arguments must be defined **last**.
18. The value of a default argument must be a(n) **literal or constant**.
19. When used as parameters, **reference** variables allow a function to access the parameter’s original argument.
20. Reference variables are defined like regular variables, except there is a(n) **ampersand(&)** in front of the name.
21. Reference variables allow arguments to be passed by **reference**.
22. The **exit()** function causes a program to terminate.
23. Two or more functions may have the same name, as long as their **parameter lists** are different.

|  |  |
| --- | --- |
| // function example  #include <iostream> || using namespace std;  int addition (int a, int b)  {  int r;  r=a+b;  return r;  }  int main ()  {  int z;  z = addition (5,3);  cout << "The result is " << z;  } | The result is 8 |
| // function example  #include <iostream> || using namespace std;  int subtraction (int a, int b)  {  int r;  r=a-b;  return r;  }  int main ()  {  int x=5, y=3, z;  z = subtraction (7,2);  cout << "The first result is " << z << '\n';  cout << "The second result is " << subtraction (7,2) << '\n';  cout << "The third result is " << subtraction (x,y) << '\n';  z= 4 + subtraction (x,y);  cout << "The fourth result is " << z << '\n';  } | The first result is 5  The second result is 5  The third result is 2  The fourth result is 6 |
| // void function example  #include <iostream> || using namespace std;  void printmessage ()  {  cout << "I'm a function!";  }  int main ()  {  printmessage ();  } | I'm a function! |
| // default values in functions  #include <iostream> || using namespace std;  int divide (int a, int b=2)  {  int r;  r=a/b;  return (r);  }  int main ()  {  cout << divide (12) << '\n';  cout << divide (20,4) << '\n';  return 0;  } | 6  5 |
| // passing parameters by reference  #include <iostream> || using namespace std;  void duplicate (int& a, int& b, int& c)  {  a\*=2;  b\*=2;  c\*=2;  }  int main ()  {  int x=1, y=3, z=7;  duplicate (x, y, z);  cout << "x=" << x << ", y=" << y << ", z=" << z;  return 0;  } | x=2, y=6, z=14 |
| // declaring functions prototypes  #include <iostream> || using namespace std;  void odd (int);  void even (int);  int main()  {  int i;  do {  cout << "Please, enter number (0 to exit): ";  cin >> i;  odd (i);  } while (i!=0);  return 0;  }  void odd (int x)  {  if ((x%2)!=0) cout << "It is odd.\n";  else even (x);  }  void even (int x)  {  if ((x%2)==0) cout << "It is even.\n";  else odd (x);  } | Please, enter number (0 to exit): 9  It is odd.  Please, enter number (0 to exit): 6  It is even.  Please, enter number (0 to exit): 1030  It is even.  Please, enter number (0 to exit): 0  It is even. |

Pseudocode Flowchart

**CHAPTER 1**

|  |  |
| --- | --- |
| 1.   1. Ask user for cost of item 2. Prompt user with amount of item cost being discounted 3. Calculate discount times item cost=Savings 4. Display “Savings” on item 5. Subtract “Savings” from item cost 6. Display Grand Total | 1. |
| 2.   1. Ask user for Bowling Game 1 score 2. Ask user for Bowling Game 2 score 3. Ask user for Bowling Game 3 score 4. Add “Bowling Game 1 + Bowling Game 2 + Bowling Game 3” = Total 5. Total divided by 3 (amount of games) = Average Score 6. Display “Average Score” | 2. |
| 3.   1. Prompt user with “My Credit” 2. Ask user for “Maximum Credit” 3. Ask user for “Amount of Credit Used” 4. Calculate- “Available Balance” = Maximum Credit – Amount of Credit Used 5. Display “Available Balance” | 3. |

**CHAPTER 5**

|  |  |
| --- | --- |
| #include <iostream>  #include <string>  #include <fstream>  #include <iomanip>  using namespace std;  int main() {  ifstream myFile;  myFile.open("Rainfall.txt");  string monthStart, monthEnd;  double average, avgTotal, x = 0, total = 0;  if (myFile.fail())  {  cout << "Input file did not open. Make sure it is in folder!!!\n";  }  myFile >> monthStart;  myFile >> monthEnd;  cout << fixed << setprecision(2);  while (myFile >> average)  {  total += average;  x++;  }  avgTotal = total / x;  cout << "During the months of " << monthStart << "-"  << monthEnd << " the total\nrainfall was "  << total << " inches and the average\n"  << "monthly rainfall was " << avgTotal  << " inches.";  myFile.close();  system("pause>nul");  return 0;  } | If file does NOT open then  Input file did not open. Make sure it’s in folder  If file DOES open then  *Grabs first line---*monthStart---X  *Grabs second line---*monthEnd---Y  *Grabs third line----*monthStart average---  *Grabs fourth line---*nextMonth average--  :  :  :  *Grabs final line---*monthEnd average---    Calculates #OFAVERAGES (%) and DIVIDES  by #OFLINES (&)  DISPLAYS  During the months of X – Y the total  rainfall was % inches and the average  monthly rainfall was & inches |

**IMPORTANT DEFINITIONS**

WHILE LOOP -------------------PRE-TEST LOOP

FOR LOOP

--------------BOTH POST TEST LOOP

DO-WHILE LOOP

^-----------RUNS ONCE EVEN IF FALSE

SENTINEL

ACCUMULATOR  
ITERATION

RUNNING TOTAL

USE “cin” TWICE TO VALIDATE INPUT (BEFORE LOOP & INSIDE LOOP)

NESTED LOOP---INNER FIRST AND SECOND LAST

USE break; ----FOR SWITCH CASES and TO TERMINATE LOOPS

PUSH\_BACK()----- PUSHES AN ELEMENT INTO A VECTOR FROM THE BACK

POP\_BACK()----POP OR REMOVE ELEMENTS FROM A VECTOR FROM THE BACK

CHAPTER 8

|  |  |
| --- | --- |
| #include <iostream>  using namespace std;  void arrayFunction(int [], int, int); // Prototype for arrayFunction  int main()  {  int n;  cout << "Enter a value between 1 to 20 :" << endl;  cin >> n;  const int size = 21;  int arrayNumbers[size] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20};  arrayFunction(arrayNumbers, size, n); // Call function  return 0;  }  void arrayFunction(int A[], int sizeA, int number) // Function Definition  {  for (int i = 0; i < sizeA; i++)  {  if (A[i] > number)  {  cout << A[i] << " ";  cout << endl;  }  }  } | Enter a number: X  Will then display all values greater than the value in the array:  X  Y  Z  W  E  :  :  :  :  A |