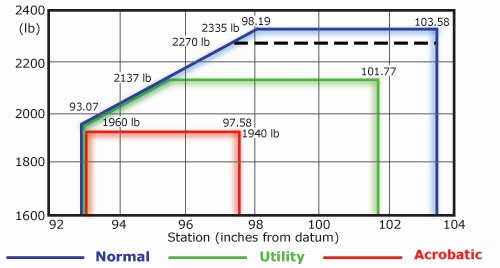
**PROJECT 12**

**SMALL AIRCRAFT CG CALCULATOR**

***For use of modifying permanent equipment records (TCDS)***

***per FAA-H-8083-1A***

**CSC-5: 42450**

**June 9th, 2014**

**Adams, Jonathan**

**TABLE OF CONTENTS**

**Page 3-4: Additions (Cargo Loader & Torque Table)**

**Introduction/Summary**

**Pages 5: Defining technical terms & their variables**

**Page 6: Other Variables/Future Features**

**Page 7-9: Flow Chart**

**Page 8+: Raw Code**

**Introduction:**

This program is to assist in performing the computations for different tasks commonly encountered in aircraft maintenance. I wanted to program this to one day provide myself with something I can actually use in my line of work. I wanted to have my program knowledge and job knowledge complement each other, given I put time into both. This way I can effectively improve both skills.

My end goal is to become an engineer, however, I am hoping as I go down that road I can use my new skills to create actual useful products, if not for the entire industry that I will be leaving, at least for the people I worked with and the specific tasks they tackle. But overall it is the entrepreneur spirit that lead me to want to program this sort of project versus a game.

**Note for ver2.0 (project 2):**

It was exciting to get new concepts into the project, but I admit it ended up a bit messy. Did have not yet re-wrote most of ver 1.0, so it does not use function, like the newer stuff does. But nevertheless, I added about 400 lines of code using functions and arrays. I don’t think there’s any lack of using all the data types and different ways to pass values.

**ADDED CARGO LOADER:**

This option allows the user to see the effect loading cargo on an aircraft has on aircraft CG, depending on location in reference to the datum. The program allows the user to input cargo on a grid in the cargo bay. User may keep adding cargo onto the already loaded floor.

**NOTE:** As of right now, once you have loaded to square, you cannot load additional cargo there and have it work properly.

Although simple, I think this program was a greay way to learn multi-dimensional arrays, and it is a great way of showing visually how they work!

**ADDED TORQUE TABLE CREATOR:**

The most useful program I created is this option. It allows the user to create a very customizable table of whatever torque values fit their needs for a reference sheet. You can choose the amount of rows, type of data, and how the table is incremented and have a table printed out for all three of the common torque units of measurement.

I used this program to demonstrate passing arrays in functions, passing by reference and by value. It is “clunkier” than it needs to be due to this. As I improve this project for next semester, I will be making it efficient to the greatest of my ability, so it is clean to add new concepts to in the future.

**FIXED BUGS:**

I noted a few instances in the original CG calculator that would create a multiplied CG above where it should be. These were corrected.

**Summary:**

A CG calculator that can be used to make the basic calculations when changing permanent equipment on an aircraft Type Certificate Data Sheet. It is this document that the input numbers for limitations on fuel, cargo and passengers is derived. So, it is only to be modified when doing a major alteration to the aircraft, such as changing an aircraft starter or adding a piece of avionics equipment.

Cargo Loader now is option to load cargo onto one aircraft profile. In the future, it should be easy to implement it for you can load your own acft profile as well as use vectors to design your own cargo bay.

**Important Terms:**

The CG calculator will be confusing to someone not familiar with basic aircraft weight and balance. So I will define a few terms so anyone could use the program to get a feel to how CG works on small aircraft.

**Center of Gravity:**

**Variable: float acftCG**

The point in which the weight of one side of a point is equal to the weight on the other side of a point. It is where the aircraft "weighs" the most.

Most of the time pilots and maintainers are generally focused on and worried about the CG point about the lateral axis, which means the point of balance of the forward and aft dimension. This is the point, if the aircraft were suspended from, where the nose and tail of the aircraft will neither dip or rise. It is important that the CG is near the center of lift. All aircraft have a specified range of allowable CG limits.

NOTE: These limits have not been included in this code, to include the C-172M profile.

Whenever any weight on the aircraft shifts aft or forward, the aircraft CG changes.

**Basic Empty Weight:**

**Variable: float acftW**

Weight of the aircraft with zero fuel, zero passengers, zero cargo. It does include equipment permanently installed on the aircraft. It is where fuel, cargo and other loading charts are derived from. So, it is not done often or lightly.

**Aircraft Moment:**

**Variable: float acftMom**

Aircraft moment is a value of force applied to an aircraft at a given station number (aircraft is divided in inches) with a given weight at that station #. Thus, whenever weight is added, removed or shifted, the aircraft moment also changes.

**Datum point:**

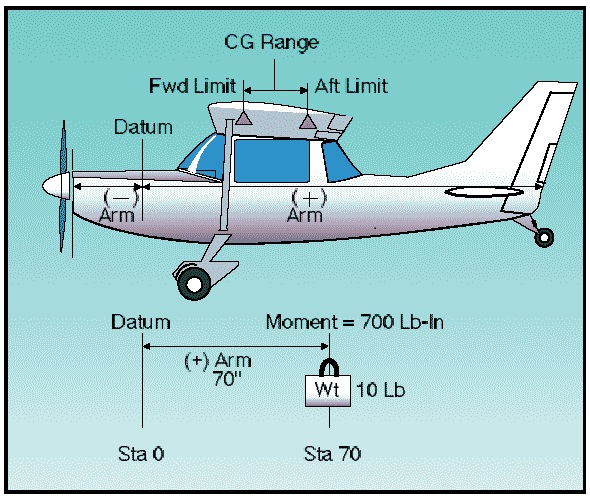
An imaginary vertical plane that all measurements of horizontal distances/ARM are taken from, for weight and balance. The reference datum is set by the aircraft manufacturer and is set a "0". On small aircraft, it is usually located at the firewall between the cockpit and engine bay. However, it can be located somewhere not even on the aircraft.

NOTE: For this program, it is assumed that the mechanic understands this and it is a requirement that he knows where the datum point is, and how far away the equipment they are changing is, from the Datum. The program will ask the user whether or not they are operating forward or aft of the Datum.

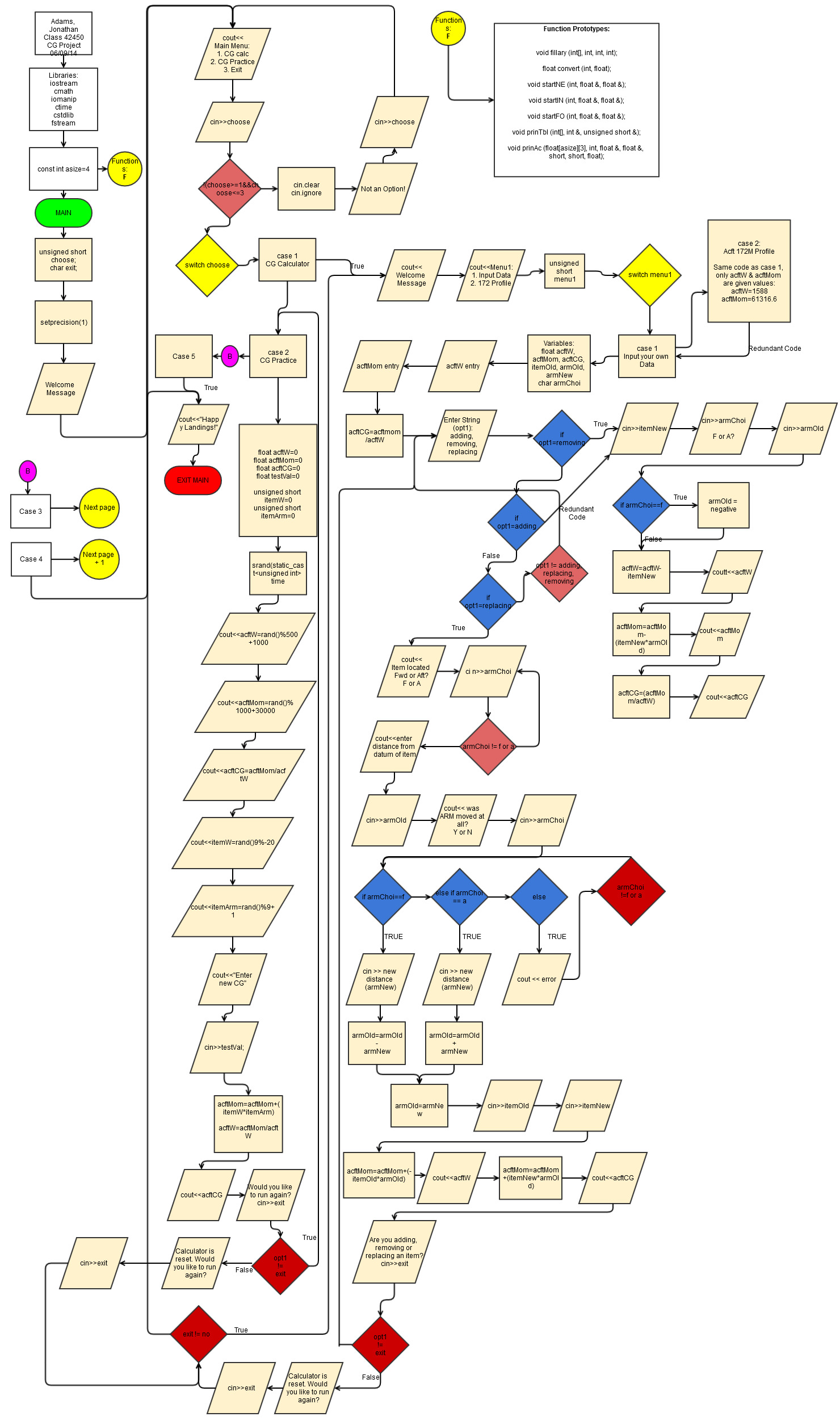
**Arm:**

**Variable float armOld & armNew**

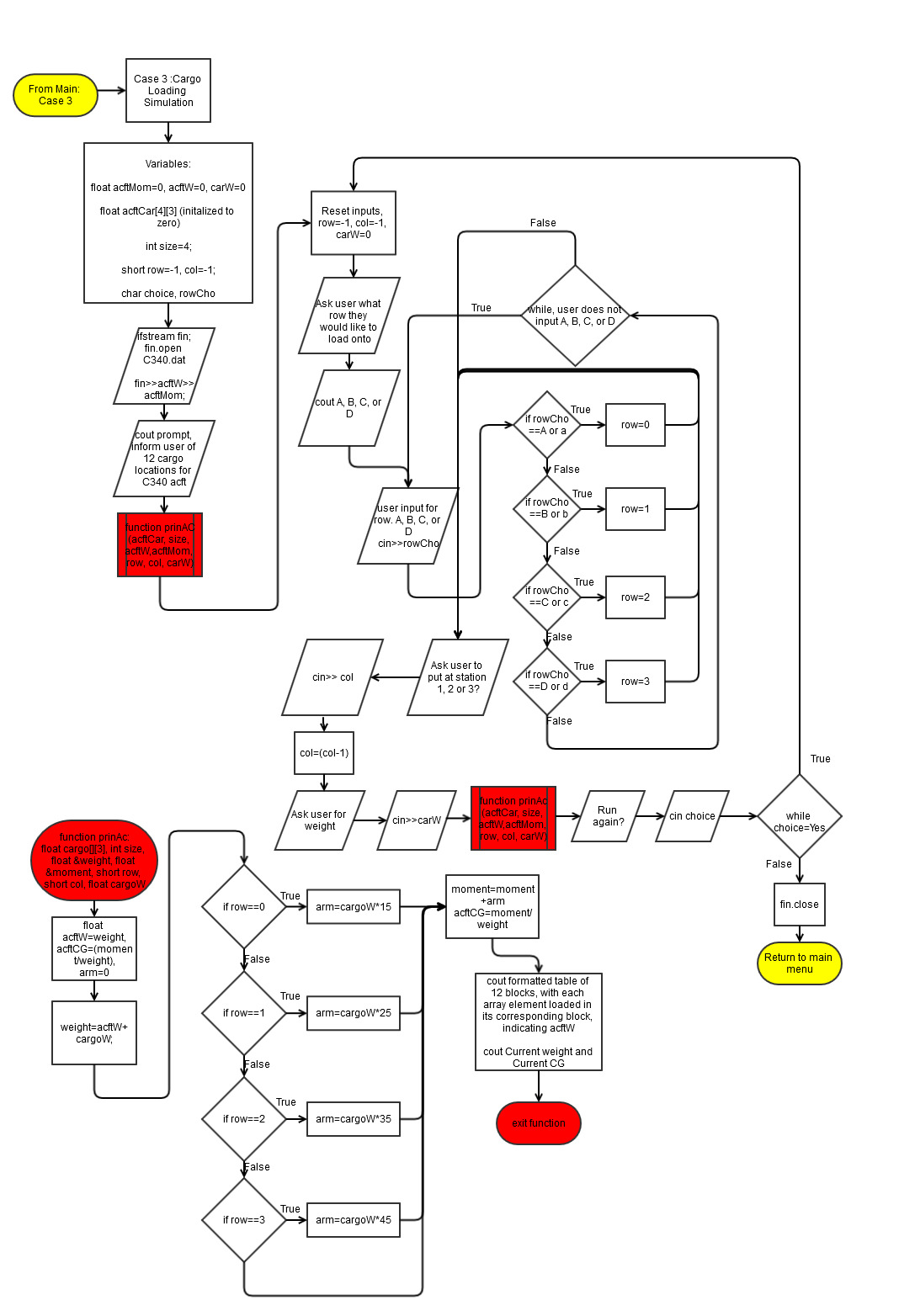
Aircraft ARM is the distance of a point on the aircraft to the datum. Arm multiplied by weight gives the moment for that piece of equipment at that station on the aircraft.

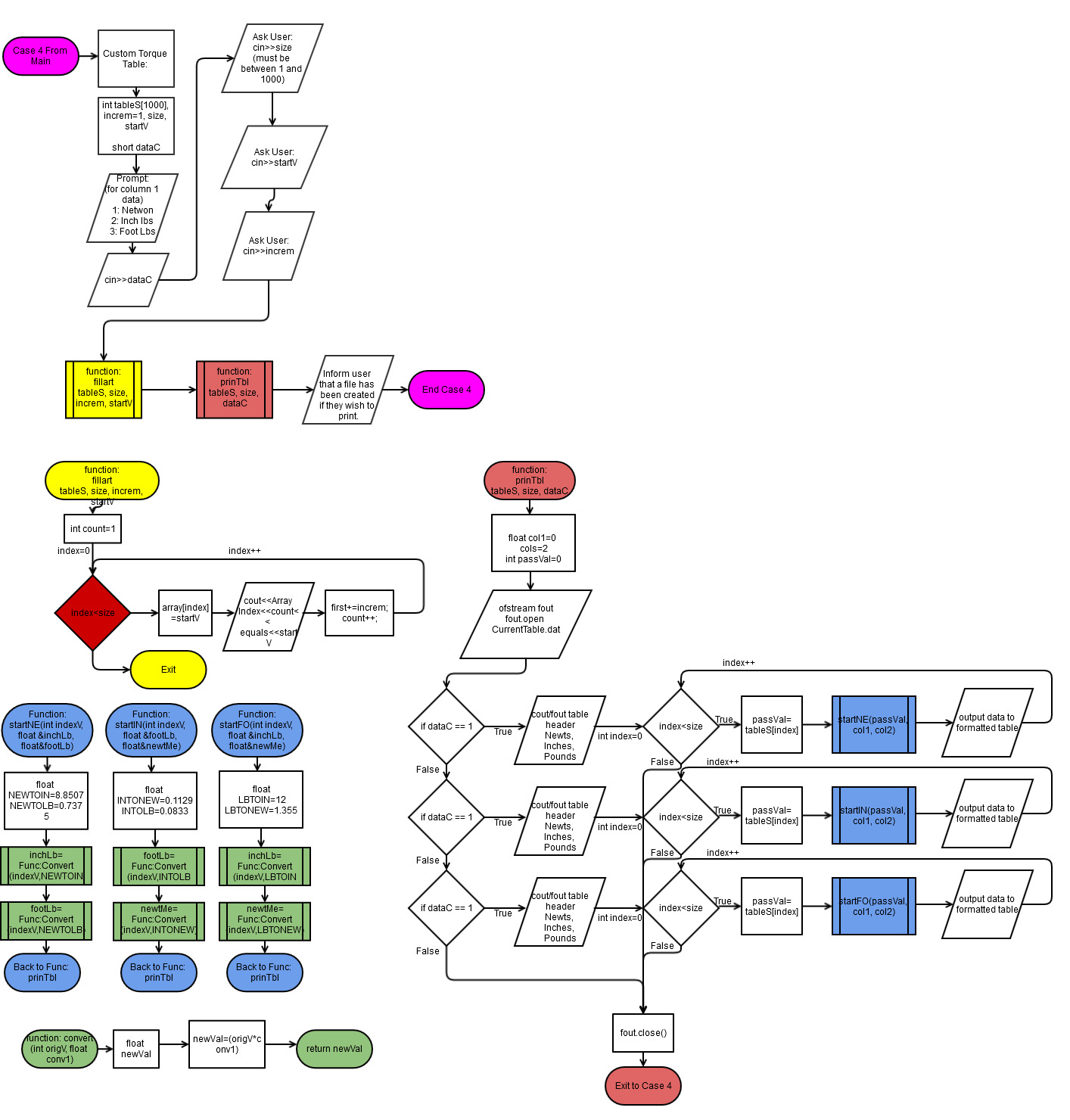


**Other variables in menus:** Various catch statements & variables used for menus and error management. Not listed due to simplicity



Continued: Case 3 Cargo Loading Simulator w/ associated functions



Continued: Case 4 (Torque Table Generator w/ associated functions)

Pseudo Code:

Main Menu: 1. CG Calculator

2. CG Practice

3. Exit

1. CG Calculator:

Calculator Menu: 1. Enter Your Own Data

2. Cessna 172M Profile

1. Ask user for acftW & acftM:

Ask the user if they would like to add, remove or replace an item

if remove:

User is prompted to enter weight of item being removed.

User is prompted to enter moment of item being removed.

User is asked if ARM is forward or aft of datum.

Calculate and display new CG

if adding:

User is prompted to enter weight of item being removed.

User is prompted to enter moment of item being removed.

User is asked if ARM is forward or aft of datum.

Calculate and display new CG

if replacing:

User is prompted on ARM location; FWD or AFT of Datum?

User is prompted on whether or not ARM was adjusted slightly?

User is prompted on removed item's weight

User is prompted on new item's weight.

Calculates new moment

Calculates and displays new CG

2. Cessna 172 Profile is the same as Input Your Own Data, but sets a value to original aircraft weight and aircraft moment.

2. CG Practice:

Set random seed

Computer generates aircraft weight, moment, CG, a new item's weight and the new item's moment.

User is prompted to enter new CG

Display new CG

3. Cargo Loader:

Create a 12 unit, two dimensional array. 4 by 3 (easy to mod later)

User is asked if they would like to add cargo, if so which row and station in the row.

The further back the row (larger the letter), the great it affects acft CG.

A graphical menu of the acft is displayed each time cargo is loaded, along with updated parameters

4. Torque Table

In aviation maintenance, torque wrenches are extremely important. Although rare, sometimes you get stuck using a torque wrench that only has one data type, where the info you’re given is another data time (like foot pounds versus inch pounds).

This is just a fun little table creator that lets you make a reference table to display different values.

User enters the size of the table (can go up to 100), then input which data type for column 1. User also chooses what increment to load the table with. (every 1, 2, 5, etc);

Program outputs a table to the screen, as well as a file the user can print out for reference.

3. Exit

Exit program

**Raw Code:**

#include <iostream>

#include <cmath>

#include <iomanip>

#include <ctime>

#include <cstdlib>

#include <fstring>

using namespace std;

const int asize=4;

//Function Prototypes

void fillary (int[], int, int, int);

float convert (int, float);

void startNE (int, float &, float &);

void startIN (int, float &, float &);

void startFO (int, float &, float &);

void prinTbl (int[], int &, unsigned short &);

void prinAc (float[asize][3], int, float &, float &, short, short, float);//Execution

int main(int argc, char\*\* argv) {

//Declare Variables

unsigned short choose;

char exit;

//-- 1 Decimal point is sufficient for all calculations

//in this program

cout<<fixed<<setprecision(1)<<showpoint;

//Welcome statement to not be looped

cout<<"Welcome to the Aerospace and Powerplant Maintenance Helper."<<endl;

//Prompt user for number of problem to execute

cout<<endl;

cout<<endl;

do { //MENU DO LOOP BEGIN

cout<<"Choose from the following list"<<endl;

cout<<"1. Re-Calculate Longitudinal CG for new component"<<endl;

cout<<"2. Longitudinal CG Practice"<<endl;

cout<<"3. Exit Program - All"<<endl;

cin>>choose;

//Catch invalid input choose

while (!(choose>=1 && choose<=4)) {

cin.clear();

cin.ignore();

cout<<"Not an option! Please re-enter: "<<endl;

cin>>choose;

}

//End catch invalid input choose

//Utilize switch to implement the menu

switch(choose) { //Begin switch choose

case 1:{

///////////////////////////////Component Calculator/////////////////////////////

cout<<"Program 1:New Component Longitudinal CG Calculator";

cout<<endl;

do {

//START CODE//////////////////////////////

//Prompt

cout<<setfill('-')<<setw(40)<<"-"<<endl;

cout<<" Welcome to Longitudinal CG Calculator!"<<endl;

cout<<setfill('-')<<setw(40)<<"-"<<endl;

cout<<setfill('-');

cout<<"This CG calculator is meant to aid you in recalculating \n";

cout<<"aircraft empty weight CG when you remove, add or replace \n";

cout<<"a permanent piece of equipment, as defined in the \n";

cout<<"Aircraft Equipment List."<<endl<<endl;

//Prompt for user to select ACFT to modify, or make their own

cout<<"Select which option you would prefer:"<<endl;

cout<<"1. Enter my own aircraft data"<<endl;

cout<<"2. Cessna 172M Profile"<<endl;

//Menu 1 Input Variable

unsigned short menu1;

//User input for menu1

cin>>menu1;

//Catch invalid input menu1

while (!(menu1>=1&&menu1<=2)) {

cin.clear();

cin.ignore();

cout<<"Not an option! Please re-enter: "<<endl;

cin>>menu1;

}

//Catch invalid input menu1

switch (menu1) { /////Menu1 Switch start

//Variables declared for CG calculations

//Planning to turn this into a function later

//Basic Aircraft Weight

float acftW;

//Aircraft moment comes from taking all aircraft longitudinal

//station numbers (in inches) and multiplying their weight by

//their ARM (distance from the DATUM)

float acftMom;

//CG is calculated by dividing aircraft moment by aircraft

//weight. The purpose here is to determine Basic Empty CG.

//So fuel, cargo and passengers are considered to be at 0.

float acftCG;

//Place holder for the weight of an item being removed

float itemOld;

//The location of the item being removed, in reference to

//manufacture specified DATUM point.

float armOld;

//Weight of new item

float itemNew;

//Location of new item, if modified. (EX. longer starter)

float armNew;

//Menu character

char armChoi;

case 1:{

//Retrieve Aircraft Weight

cout<<"Please enter aircraft empty weight, in lbs\n";

cout<<"to 1/10 of a lb accuracy. (xxxx.x)\n";

cout<<"NOTE: Info on acft Type Certificate Data Sheet\n";

cin>>acftW;

//Retrieve Aircraft Moment

cout<<"Please enter aircraft empty moment,\n";

cout<<"to 1/10 accuracy. (xxxx.x)\n";

cout<<"NOTE: Info on acft Type Certificate Data Sheet\n";

cin>>acftMom;

//Calculate CG

acftCG=(acftMom/acftW);

cout<<endl;

//Display CG

cout<<"Current CG is "<<acftCG<<"."<<endl;

//Ask what user wants to do to aircraft

cout<<"Are you adding, removing or replacing an item?\n";

cout<<"Type in option above in lowercase letters"<<endl;

cout<<"Type exit to leave or reset CG calculator."<<endl;

string opt1;

cin>>opt1;

do {

//Error catching loop opt1

while (opt1!="removing"&&opt1!="adding"&&opt1!=

"replacing"&&opt1!="exit"){

cin.clear();

cin.ignore();

cout<<"Invalid input, re-enter:"<<endl<<endl;

cout<<"Please type one of the following options: \n";

cout<<"removing \nadding \nreplacing\n";

cout<<"Type in option above in lowercase letters"<<endl;

cout<<"Or type exit to leave or reset CG calculator"<<endl;

cin>>opt1;

cout<<endl<<endl;

}

//End error catching loop opt1

//Removing option will only remove an item.

//COUT statements self explanatory

if (opt1=="removing") {

cout<<"You're removing an item"<<endl;

cout<<"Enter weight, in lbs to 1/10 lb accuracy\n";

cout<<"of item being removed"<<endl;

cin>>itemNew;

cout<<"Determining ARM: Is the item located\n";

cout<<"FWD or AFT of Acft Datum?"<<endl;

cout<<"F for FWD, A for AFT: ";

cin>>armChoi;

//Error catch statement for FWD & AWF

//Important that the user has to input this

//because it sets the ARM to positive or negative

do {

cout<<"Error! Invalid Input!"<<endl<<endl;

cout<<"Determining ARM: Is the item located\n";

cout<<"FWD or AFT of Acft Datum?"<<endl;

cout<<"F for FWD, A for AFT: ";

cin>>armChoi;

} while (armChoi!='f'&&armChoi!='F'

&&armChoi!='a'&&armChoi!='A');

//end error statement

cout<<"Enter ARM (distance from Datum in inches)\n";

cout<<"of item being removed, to 1/10th of an inch"<<endl;

cin>>armOld;

//set ARM to negative if item was FWD of datum

if (armChoi=='f'||armChoi=='F') armOld=(armOld\*-1);

//Calculations & Display

acftW=(acftW)-(itemNew)+.01;

cout<<setw(20)<<left<<"New Acft Weight: ";

cout<<setw(20)<<right<<acftW<<endl;

acftMom=acftMom-(itemNew\*armOld)+.01;

cout<<setw(20)<<left<<"New Acft Mom: ";

cout<<setw(20)<<right<<acftMom<<endl;

acftCG=(acftMom/acftW)+.01;

cout<<setw(20)<<left<<"New Acft CG is: ";

cout<<setw(20)<<right<<acftCG<<endl<<endl;

}

//Adding an item

if (opt1=="adding") {

cout<<"You're adding an item"<<endl;

cout<<"Enter weight, in lbs to 1/10 lb accuracy\n";

cout<<"of item being added"<<endl;

cin>>itemNew;

cout<<"Determining ARM: Is the item located\n";

cout<<"FWD or AFT of Acft Datum?"<<endl;

cout<<"F for FWD, A for AFT: ";

cin>>armChoi;

//Error catch statement ARM location

while (armChoi!='f'&&armChoi!='F'

&&armChoi!='a'&&armChoi!='A') {

cout<<"Error! Invalid Input!"<<endl<<endl;

cout<<"Determining ARM: Is the item located\n";

cout<<"FWD or AFT of Acft Datum?"<<endl;

cout<<"F for FWD, A for AFT: ";

cin>>armChoi;

}

//End error catch statement

cout<<"Enter ARM (distance from Datum in inches)\n";

cout<<"of item being removed, to 1/10th of an inch"<<endl;

cin>>armOld;

//Set ARM to negative if item FWD of datum

if (armChoi=='f'||armChoi=='F') armOld=(armOld\*-1);

//Calculations & display

acftW=(acftW)+(itemNew)+.01;

cout<<setw(20)<<left<<"New Acft Weight: ";

cout<<setw(20)<<right<<acftW<<endl;

acftMom=acftMom+(itemNew\*armOld)+.01;

cout<<setw(20)<<left<<"New Acft Mom: ";

cout<<setw(20)<<right<<acftMom<<endl;

acftCG=(acftMom/acftW)+.01;

cout<<setw(20)<<left<<"New Acft CG is: ";

cout<<setw(20)<<right<<acftCG<<endl<<endl;

}

//Replacing an item

if (opt1=="replacing") {

cout<<"You're replacing an item"<<endl;

cout<<"Determining ARM: Is the item located\n";

cout<<"FWD or AFT of Acft Datum?"<<endl;

cout<<"F for FWD, A for AFT: ";

cin>>armChoi;

//Error catching statement

do {

cout<<"Error! Invalid Input!"<<endl<<endl;

cout<<"Determining ARM: Is the item located\n";

cout<<"FWD or AFT of Acft Datum?"<<endl;

cout<<"F for FWD, A for AFT: ";

cin>>armChoi;

} while (armChoi!='f'&&armChoi!='F'

&&armChoi!='a'&&armChoi!='A');

//End error catching statement

cout<<"Enter ARM (distance from Datum in inches)\n";

cout<<"of item being removed, to 1/10th of an inch"<<endl;

cin>>armOld;

//Set ARM to negative if FWD of Datum

if (armChoi=='f'||armChoi=='F') armOld=(armOld\*-1);

cout<<"Was the ARM moved forward or aft?\n";

cout<<"Y for Yes, any other key for No:"<<endl;

cin>>armChoi;

armNew=armOld;

if (armChoi=='y'||armChoi=='Y') {

do {

cout<<"Forward, or Aft?"<<endl;

cout<<"F for FWD, A for AFT"<<endl;

cin>>armChoi;

if (armChoi=='f'||armChoi=='F') {

cout<<"Enter distance moved in inches\n";

cin>>armNew;

armOld=armOld-armNew;

}

else if (armChoi=='a'||armChoi=='A') {

cout<<"Enter distance moved in inches\n";

cin>>armNew;

armOld=armOld+armNew;

}

else {

cout<<"Error.";

}

} while (armChoi!='f'&&armChoi!='F'

&&armChoi!='a'&&armChoi!='A');

}

armOld=armNew;

cout<<"Enter the item's original weight"<<endl;

cout<<"Enter weight, in lbs to 1/10 lb accuracy\n";

cin>>itemOld;

acftMom=(acftMom)+(-itemOld\*armOld)+.01;

cout<<"Acft mom="<<acftMom<<endl;

cout<<"Enter weight of new item"<<endl;

cout<<"Enter weight, in lbs to 1/10 lb accuracy\n";

cin>>itemNew;

acftW=(acftW-itemOld)+(itemNew)+.01;

cout<<setw(20)<<left<<"New Acft Weight: ";

cout<<setw(20)<<right<<acftW<<endl;

acftMom=acftMom+(itemNew\*armOld)+.01;

cout<<setw(20)<<left<<"New Acft Mom: ";

cout<<setw(20)<<right<<acftMom<<endl;

acftCG=(acftMom/acftW)+.01;

cout<<setw(20)<<left<<"New Acft CG is: ";

cout<<setw(20)<<right<<acftCG<<endl<<endl;

}

cout<<"Are you adding, removing or replacing another item?"<<endl;

cout<<"Type in option above in lowercase letters"<<endl;

cout<<"Type exit to leave or reset CG calculator."<<endl;

cin>>opt1;

while (opt1!="removing"&&opt1!="adding"&&opt1!=

"replacing"&&opt1!="exit"){

cin.clear();

cin.ignore();

cout<<"Invalid input, re-enter:"<<endl<<endl;

cout<<"Please type one of the following options: \n";

cout<<"removing \nadding \nreplacing\n";

cout<<"Type in option above in lowercase letters"<<endl;

cout<<"Or type exit to leave or reset CG calculator"<<endl;

cin>>opt1;

cout<<endl<<endl; }

} while (opt1!="exit");

}

NOTE: Acft 172 Profile Left Out - Redundant Code; only difference is that it sets values into acftW & acftMom at the beginning, versus asking user

CG Calculator Test:

} ////////Menu 1 Switch End

//FINISH CODE/////////////////////////////

cout<<"\n\n";

cout<<"CG Calculator is reset."<<endl;

cout<<"Would you like to run calculator again?"<<endl;

cout<<"Type N for No and to return to main menu."<<endl;

cout<<"Press any other key to run calculator again."<<endl;

cin>>exit;

} while ((exit!='n')&&(exit!='N')); /////////do while loop

cout<<endl;

cout<<endl;

cout<<"End Program 1"<<endl;break;

} //End option 1

case 2:{

////////////////////////////////PROBLEM 2///////////////////////////////////////

cout<<"Welcome to Option 2.";

cout<<endl;

do {

//START CODE//////////////////////////////

//Randomly choose a sequence start

float acftW=0; //weight

float acftMom=0; //moment

float acftCG=0; //mom/weight

float testVal=0; //user input of new CG

unsigned short itemW=0; //weight of new item

unsigned short itemArm=0; //location of new item

srand(static\_cast<unsigned int>(time(0))); // time seed for rand

cout<<"Aircraft Weight: "<<endl; //display weight of 1000 to 1049

acftW=rand()%500+1000;

cout<<acftW<<" lbs"<<endl;

cout<<"Aircraft Moment: "<<endl; //display mom of 30000 to 30999

acftMom=rand()%1000+30000;

cout<<acftMom<<endl;

itemW=rand()%9+20; //display starting CG and create itemW 20-29

cout<<"Starting CG is: "<<endl;

cout<<acftMom/acftW<<endl<<endl;

cout<<"Item added weighs:"<<endl;

cout<<itemW<<" lbs"<<endl;

itemArm=rand()%9+1; //display item location 2-11

cout<<"Located "<<itemArm<<" inches aft of Datum"<<endl;

cout<<"Enter new aircraft CG: (to 1 decimal point)"<<endl;

//User inputs their calculation

cin>>testVal;

//Computer calculation for new CG

acftMom=acftMom+(itemW\*itemArm); //Add new items moment

acftW=acftW+(itemW); // add new items weight

acftCG=acftMom/acftW; //new cg

cout<<"Correct new CG is: "<<endl; //output to user

cout<<acftCG<<endl;

//FINISH CODE/////////////////////////////

cout<<"\n\n";

cout<<"Would you like to run again? N for No."<<endl;

cout<<"Press any other key to run again."<<endl;

cin>>exit;

} while ((exit!='n')&&(exit!='N'));

cout<<endl;

cout<<endl;

cout<<"End problem 2"<<endl;break;

} //End option 2

case 3: {

cout<<"Happy Landings!"<<endl;break;

}

} // switch statement end bracket

} while (choose!=3);

case 3: {

float acftMom=0, acftW=0, carW=0;

float acftCar[4][3]={{0,0,0},{0,0,0},{0,0,0},{0,0,0}};

int size=4;

short row=-1, col=-1;

char choice, rowCho;

ifstream fin;

fin.open("C340.dat");

fin>>acftW>>acftMom;

cout<<"Cargo Loading Simulation"<<endl;

cout<<"A Cessna 340's Data has been loaded for "<<endl;

cout<<"the simulation."<<endl;

cout<<"In the future, you will be able to load your own"<<endl;

cout<<"aircraft profile from a file"<<endl<<endl;

cout<<"There are 12 Cargo Locations on the acft: "<<endl<<endl;

prinAc(acftCar, size, acftW, acftMom, row, col, carW);

do {

//Reset inputs

row=-1, col=-1, carW=0;

//Print out current state of the array

//Fetch row #

cout<<"What row would you like to load cargo onto?"<<endl;

do {

cout<<"A, B, C or D?"<<endl;

cin>>rowCho;

if (rowCho=='A'||rowCho=='a') row=0;

else if (rowCho=='B'||rowCho=='b') row=1;

else if (rowCho=='C'||rowCho=='c') row=2;

else if (rowCho=='D'||rowCho=='d') row=3;

else cout<<"Invalid Choice!"<<endl;

} while (rowCho!='A'&&rowCho!='a'&&rowCho!='B'&&rowCho!='b'&&

rowCho!='C'&&rowCho!='c'&&rowCho!='D'&&rowCho!='d');

//Fetch column #

cout<<"Station 1, 2 or 3? Enter number for station (1-3)"<<endl;

cin>>col;

while (col>3||col<0) {

cout<<"Invalid input. Input 1, 2 or 3!"<<endl;

cin>>col;

}

//Set column # to appropriate index for array

col=(col-1);

//Now ask for weight

cout<<"Enter weight of cargo: "<<endl;

cin>>carW;

prinAc(acftCar, size, acftW, acftMom, row, col, carW);

//Run again??

cout<<"Would you like to load more cargo?"<<endl;

cout<<"Y for Yes, N for No"<<endl;

cin>>choice;

} while ((choice!='n')&&(choice!='N'));

fin.close();

//Return to menu

break;

}

////////////////////////////////////////////////////////////////////////////////

//This is primarily just to practice with arrays

case 4: {

//Declare an array that can accommodate up to 1000 units of int

//I know a vector would be better here, but I am more comfortable

//with arrays for now.

int tableS[1000], increm=1, size=1000, startV;

unsigned short dataC;

cout<<"Print A Custom Shop Use Torque Value Conversion Table"<<endl<<endl;

cout<<"User can print a table up to 1000 values"<<endl;

cout<<"User chooses Troque Data needed (column 1)"<<endl;

cout<<"---Either, Newton Meters, Foot Pounds, or Inch Pounds"<<endl<<endl;

cout<<"Table will print out with corresponding values of the"<<endl;

cout<<"other two data types, so the user can use any kind of"<<endl;

cout<<"torque wrench!"<<endl<<endl;

cout<<"User can choose how many values to print, and how to "<<endl;

cout<<"increment those values!"<<endl<<endl;

cout<<"What data type would you like column 1 to be?"<<endl;

cout<<"(this should be the type you're commonly asked"<<endl;

cout<<" for in tech data)"<<endl;

cout<<"Press 1 for Newton Meters"<<endl;

cout<<"Press 2 for Inch Pounds"<<endl;

cout<<"Press 3 for Foot Pounds"<<endl;

cin>>dataC;

//Error catch

while (dataC!=1&&dataC!=2&&dataC!=3) {

cout<<"Error! Not an option!"<<endl;

cout<<"Press 1 for Newton Meters"<<endl;

cout<<"Press 2 for Inch Pounds"<<endl;

cout<<"Press 3 for Foot Pounds"<<endl;

cin>>dataC;

}

cout<<"Enter how many values you would like the table to go up to: "<<endl;

cout<<"Max is 1000; if printing it is suggested to keep under 75"<<endl;

cin>>size;

while (size>1000) {

cout<<"Too large, re-enter under 1000"<<endl;

cin>>size;

}

while (size<1) {

cout<<"Too small, value must be a whole #"<<endl;

cout<<"above zero and below 1000"<<endl;

cin>>size;

}

cout<<"Enter first value you want the table to start at: "<<endl;

cin>>startV;

cout<<"Enter how you would like the table to be incremented: "<<endl;

cout<<"(Every 1, 2, 5, 10, etc)"<<endl;

cin>>increm;

fillary(tableS, size, increm, startV);

prinTbl(tableS, size, dataC);

cout<<"File: CurrentTable.dat is now available in this"<<endl;

cout<<"directory to print with an appropriate program."<<endl;

break;

}

////////////////////////////////////////////////////////////////////////////////

case 5: {

cout<<"Happy Landings!"<<endl;break;

}

} // switch statement end bracket

////////////////////////////////////////////////////////////////////////////////

} while (choose!=5);

////////////////////////////////////////////////////////////////////////////////

//Exit Stage Right

return 0;

}

void fillary(int array[], int size, int increm, int first) {

int count=1;

for (int index=0; index<size; index++){

array[index]=first;

//cout<<"Array Index "<<count<<" equals "<<first<<endl;

first+=increm;

count++;

}

}

float convert (int origV, float conv1) {

float newVal;

newVal=(origV\*conv1);

return newVal;

}

//Function for user choosing to start with newtons

void startNE (int indexV, float &inchLb, float &footLb){

//Multiplication factors to convert newton meters, to foot lbs and inch lbs

float NEWTOIN=8.85074579, NEWTOLB=0.737562149;

inchLb=convert(indexV, NEWTOIN);

footLb=convert(indexV, NEWTOLB);

}

//Function for user choosing inch pounds to start

void startIN (int indexV, float &footLb, float &newtMe){

//Multiplication factors to convert inch lbs, to foot lbs and newton meters

float INTONEW=0.112984829, INTOLB=0.0833;

footLb=convert(indexV, INTOLB);

newtMe=convert(indexV, INTONEW);

}

//Function for user choosing foot pounds to start

void startFO (int indexV, float &inchLb, float &newtMe){

//Multiplication factors to convert inch lbs, to foot lbs and newton meters

float LBTOIN=12, LBTONEW=1.35581795;

inchLb=convert(indexV, LBTOIN);

newtMe=convert(indexV, LBTONEW);

}

//Once completing this function, I realize that a future improvement could be

//re-writing thus function to print just one option, with more input variables.

//Then I can take the menu logic for the data type, and xfer it into main.

void prinTbl (int array[], int &size, unsigned short &choice){

//Columns

float col1=0, col2=0;

//Placeholder

int passVal=0;

//Open out stream to create a text file.

ofstream fout;

fout.open("CurrentTable.dat");

//Set precision for new out stream

fout<<fixed<<setprecision(1)<<showpoint;

//Begin printing table options

//If user selected newtons:

if (choice==1) {

cout<<setw(15)<<left<<"NewtMeters:";

fout<<setw(15)<<left<<"NewtMeters:";

cout<<setw(15)<<left<<"InchPounds:";

fout<<setw(15)<<left<<"InchPounds:";

cout<<setw(15)<<left<<"FootPounds:"<<endl;

fout<<setw(15)<<left<<"FootPounds:"<<endl;

for (int index=0; index<size; index++){

passVal=array[index];

startNE(passVal, col1, col2);

cout<<setw(15)<<left<<array[index];

fout<<setw(15)<<left<<array[index];

cout<<setw(15)<<left<<col1;

fout<<setw(15)<<left<<col1;

cout<<setw(15)<<left<<col2;

fout<<setw(15)<<left<<col2;

cout<<endl;

fout<<endl;

}

}

if (choice==2) {

cout<<setw(15)<<left<<"InchPounds:";

fout<<setw(15)<<left<<"InchPounds:";

cout<<setw(15)<<left<<"FootPounds:";

fout<<setw(15)<<left<<"FootPounds:";

cout<<setw(15)<<left<<"NewtMeters:"<<endl;

fout<<setw(15)<<left<<"NewtMeters:"<<endl;

for (int index=0; index<size; index++){

passVal=array[index];

startIN(passVal, col1, col2);

cout<<setw(15)<<left<<array[index];

fout<<setw(15)<<left<<array[index];

cout<<setw(15)<<left<<col1;

fout<<setw(15)<<left<<col1;

cout<<setw(15)<<left<<col2;

fout<<setw(15)<<left<<col2;

cout<<endl;

fout<<endl;

}

}

if (choice==3) {

cout<<setw(15)<<left<<"FootPounds:";

fout<<setw(15)<<left<<"FootPounds:";

cout<<setw(15)<<left<<"InchPounds:";

fout<<setw(15)<<left<<"InchPounds:";

cout<<setw(15)<<left<<"NewtMeters:"<<endl;

fout<<setw(15)<<left<<"NewtMeters:"<<endl;

for (int index=0; index<size; index++){

passVal=array[index];

startFO(passVal, col1, col2);

cout<<setw(15)<<left<<array[index];

fout<<setw(15)<<left<<array[index];

cout<<setw(15)<<left<<col1;

fout<<setw(15)<<left<<col1;

cout<<setw(15)<<left<<col2;

fout<<setw(15)<<left<<col2;

cout<<endl;

fout<<endl;

}

}

fout.close();

}

void prinAc(float cargo[][3], int size, float &weight, float &moment,

short row, short col, float cargoW){

//Initial declarations and calculations

float acftW=weight, acftCG=(moment/weight), arm=0;

//Tally on new weight

weight=acftW+cargoW;

//Send weight to appropriate location

cargo[row][col]=cargoW;

//Add ARM multiplyer depending on location

if (row==0) {

arm=cargoW\*15;

}

else if (row==1) {

arm=cargoW\*25;

}

else if (row==2) {

arm=cargoW\*25;

}

else if (row==3) {

arm=cargoW\*35;

}

//Final calculations

moment=moment+arm;

acftCG=(moment/weight);

//Display

cout<<setw(58)<<setfill('\*')<<"\*"<<endl;

cout<<"A"<<setfill(' ')<<setw(19)<<"\*"<<setw(19)<<"\*"<<setw(19)<<"A"<<endl;

cout<<"1"<<setw(8)<<left<<" LB= "<<setw(10)<<left<<cargo[0][0]<<right;

cout<<"2"<<setw(8)<<left<<" LB= "<<setw(10)<<left<<cargo[0][1]<<right;

cout<<"3"<<setw(8)<<left<<" LB= "<<setw(10)<<left<<cargo[0][2]<<right<<"\*"<<endl;

cout<<"\*"<<setw(19)<<"\*"<<setw(19)<<"\*"<<setw(19)<<"\*"<<endl;

cout<<setfill('\*')<<setw(58)<<"\*"<<endl;

cout<<"B"<<setfill(' ')<<setw(19)<<"\*"<<setw(19)<<"\*"<<setw(19)<<"B"<<endl;

cout<<"1"<<setw(8)<<left<<" LB= "<<setw(10)<<left<<cargo[1][0]<<right;

cout<<"2"<<setw(8)<<left<<" LB= "<<setw(10)<<left<<cargo[1][1]<<right;

cout<<"3"<<setw(8)<<left<<" LB= "<<setw(10)<<left<<cargo[1][2]<<right<<"\*"<<endl;

cout<<"\*"<<setw(19)<<"\*"<<setw(19)<<"\*"<<setw(19)<<"\*"<<endl;

cout<<setfill('\*')<<setw(58)<<"\*"<<endl;

cout<<"C"<<setfill(' ')<<setw(19)<<"\*"<<setw(19)<<"\*"<<setw(19)<<"C"<<endl;

cout<<"1"<<setw(8)<<left<<" LB= "<<setw(10)<<left<<cargo[2][0]<<right;

cout<<"2"<<setw(8)<<left<<" LB= "<<setw(10)<<left<<cargo[2][1]<<right;

cout<<"3"<<setw(8)<<left<<" LB= "<<setw(10)<<left<<cargo[2][2]<<right<<"\*"<<endl;

cout<<"\*"<<setw(19)<<"\*"<<setw(19)<<"\*"<<setw(19)<<"\*"<<endl;

cout<<setfill('\*')<<setw(58)<<"\*"<<endl;

cout<<"D"<<setfill(' ')<<setw(19)<<"\*"<<setw(19)<<"\*"<<setw(19)<<"D"<<endl;

cout<<"1"<<setw(8)<<left<<" LB= "<<setw(10)<<left<<cargo[3][0]<<right;

cout<<"2"<<setw(8)<<left<<" LB= "<<setw(10)<<left<<cargo[3][1]<<right;

cout<<"3"<<setw(8)<<left<<" LB= "<<setw(10)<<left<<cargo[3][2]<<right<<"\*"<<endl;

cout<<"\*"<<setw(19)<<"\*"<<setw(19)<<"\*"<<setw(19)<<"\*"<<endl;

cout<<setfill('\*')<<setw(58)<<"\*"<<endl;

cout<<setfill(' ');

cout<<"Current: "<<endl;

cout<<setw(29)<<left<<"Acft Gross Weight: "<<setw(29)<<left;

cout<<"CG: "<<endl<<endl;

cout<<setw(10)<<left<<weight<<setw(19)<<left<<" pounds";

cout<<setw(10)<<left<<acftCG<<setw(29)<<left<<" inches"<<endl<<endl;

}