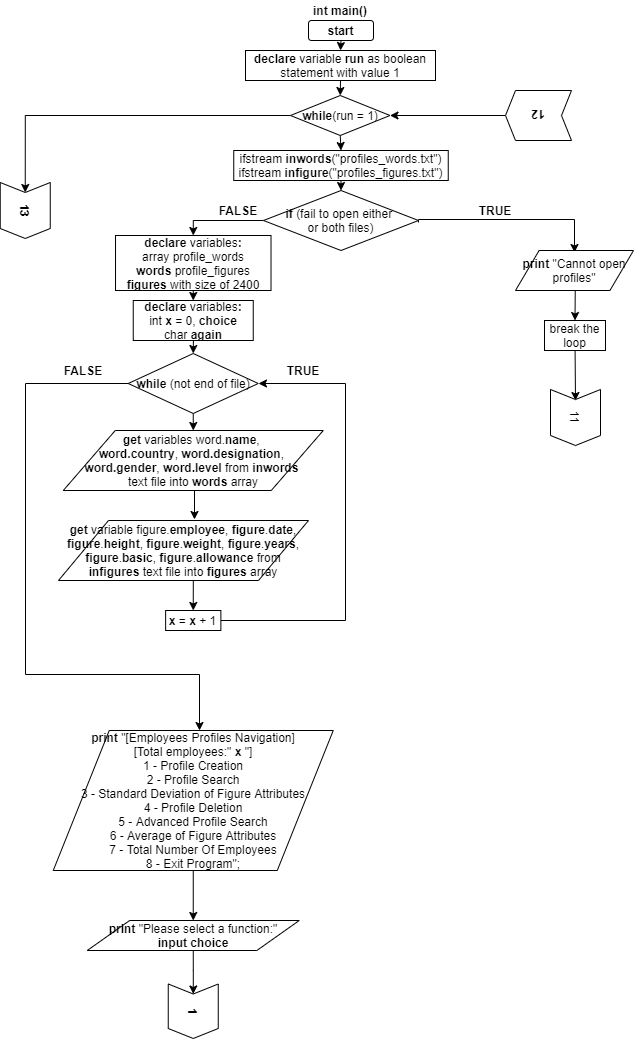
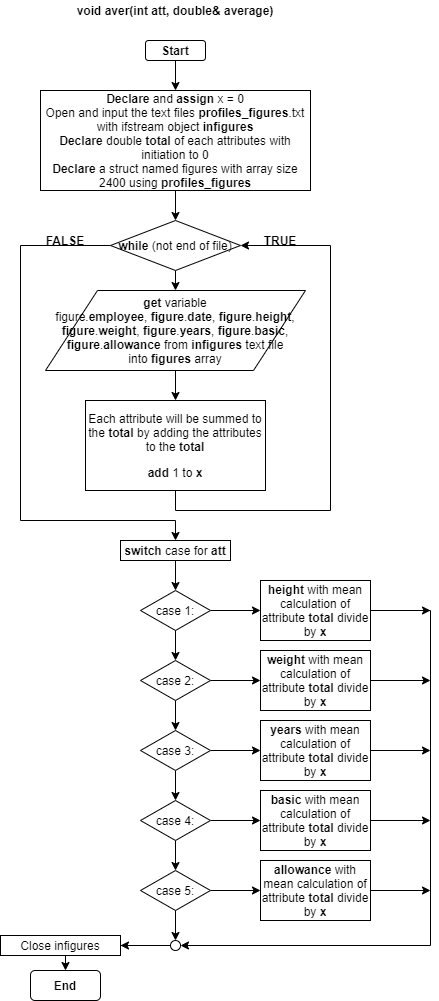
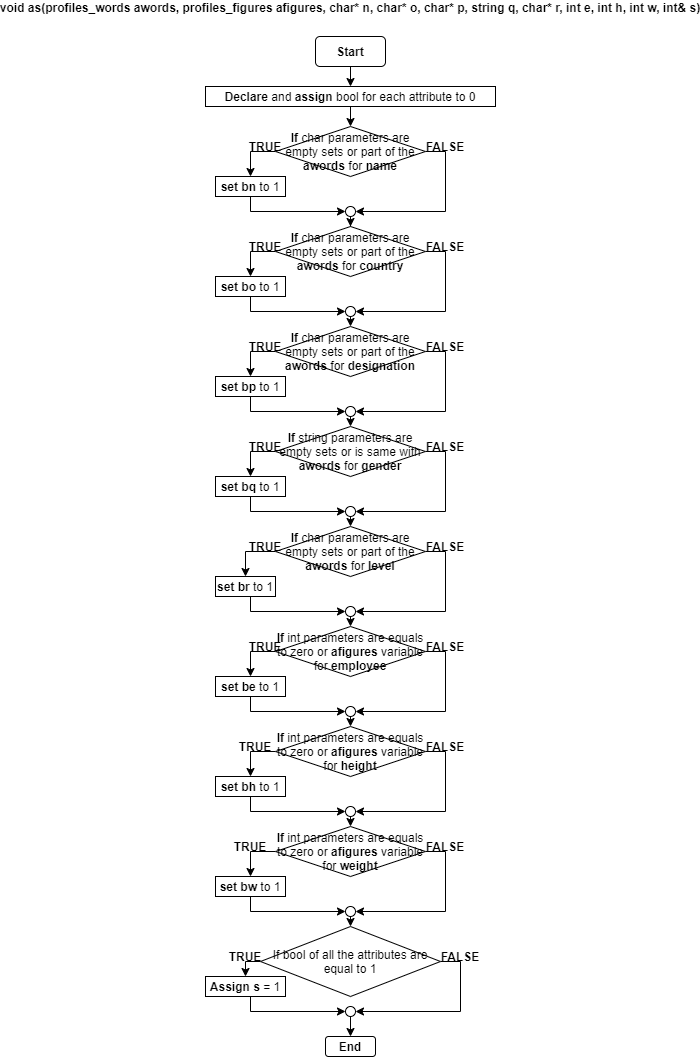
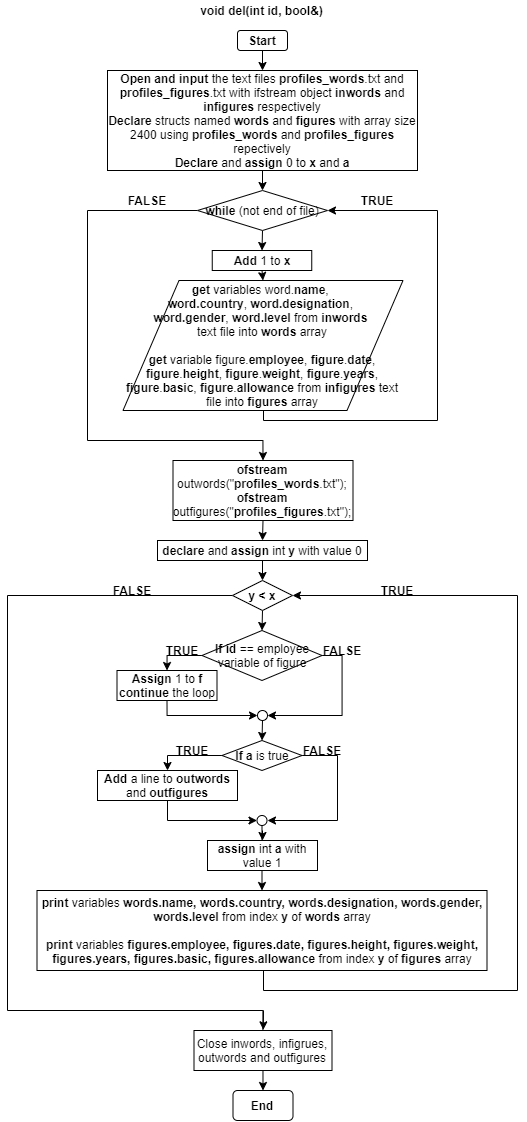
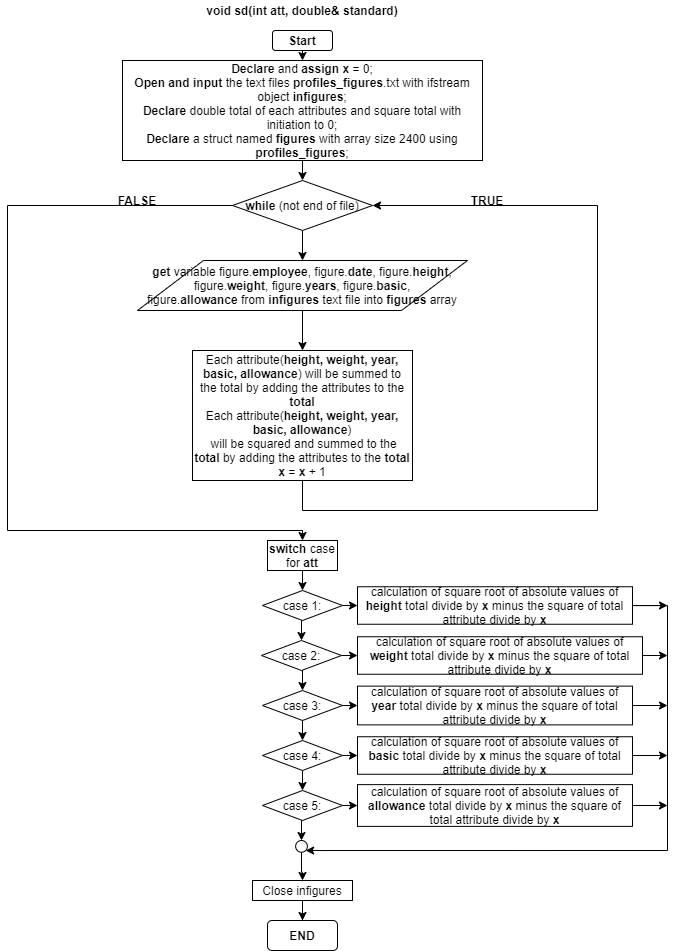
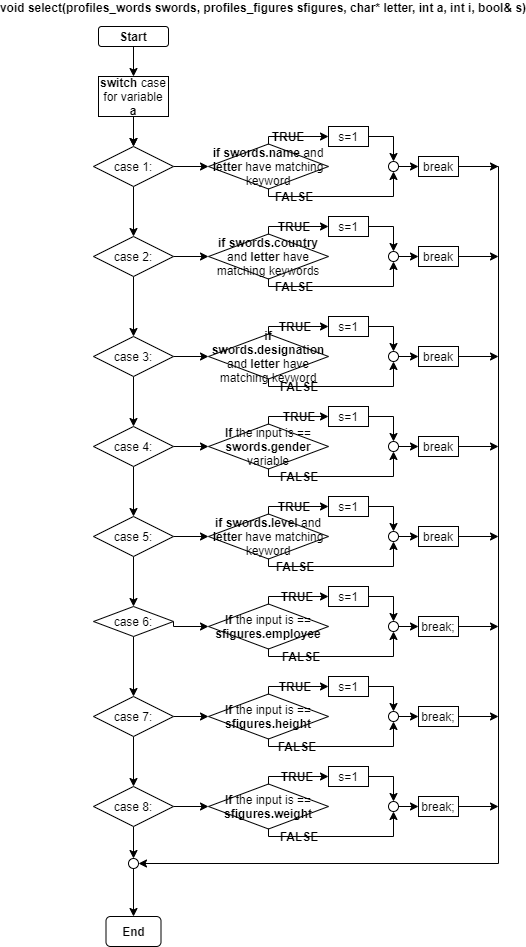
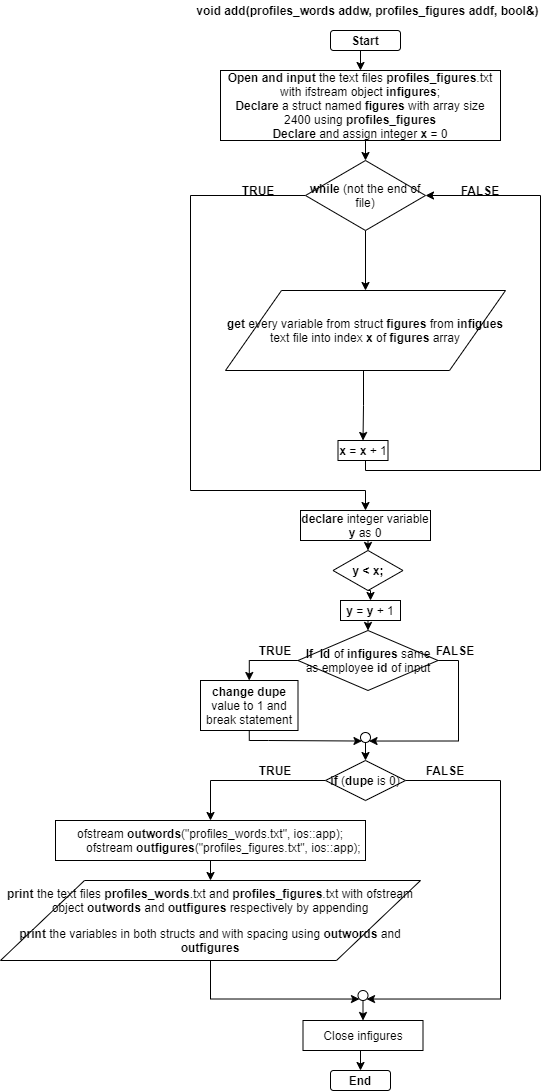
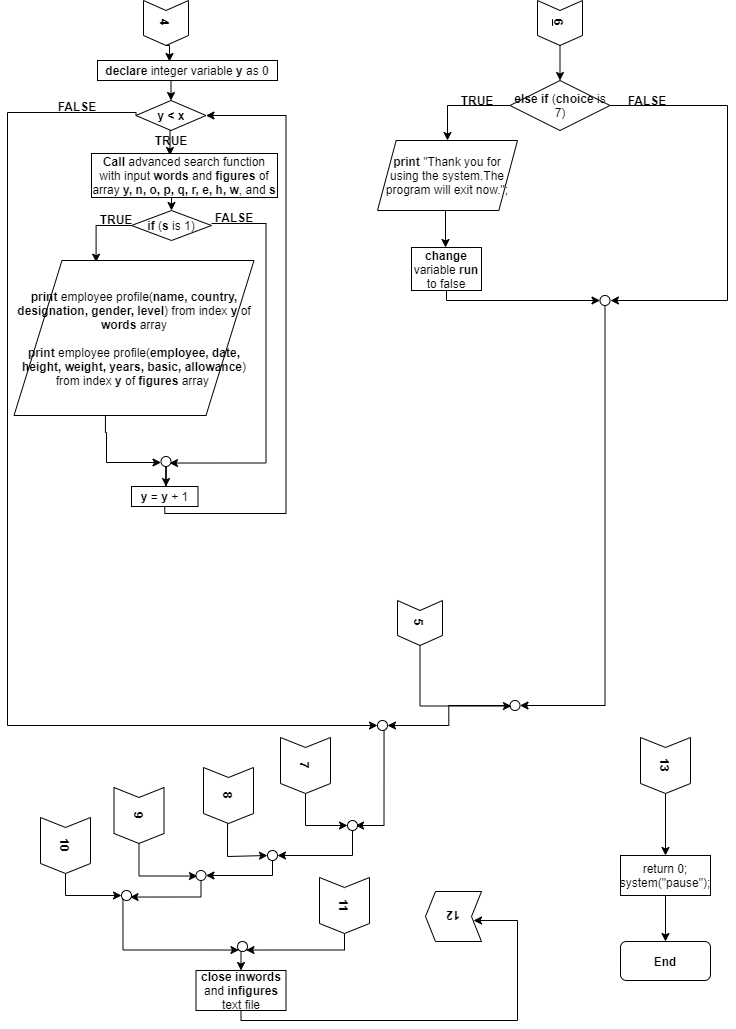
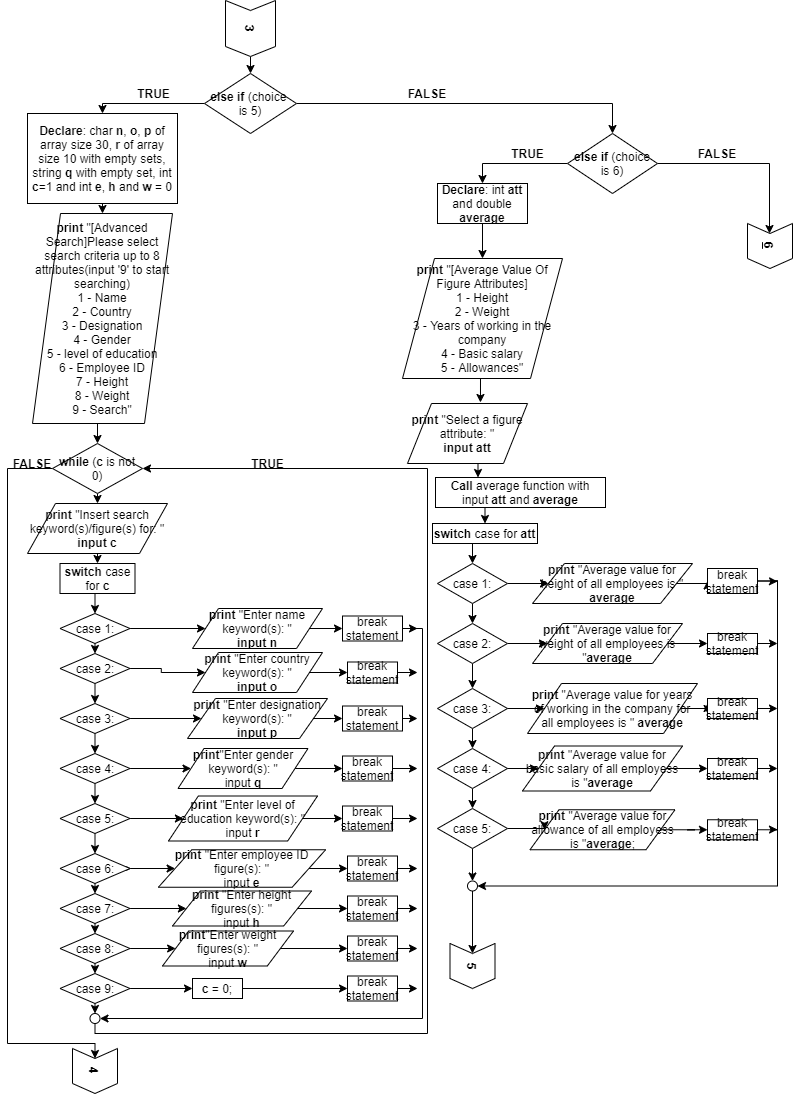
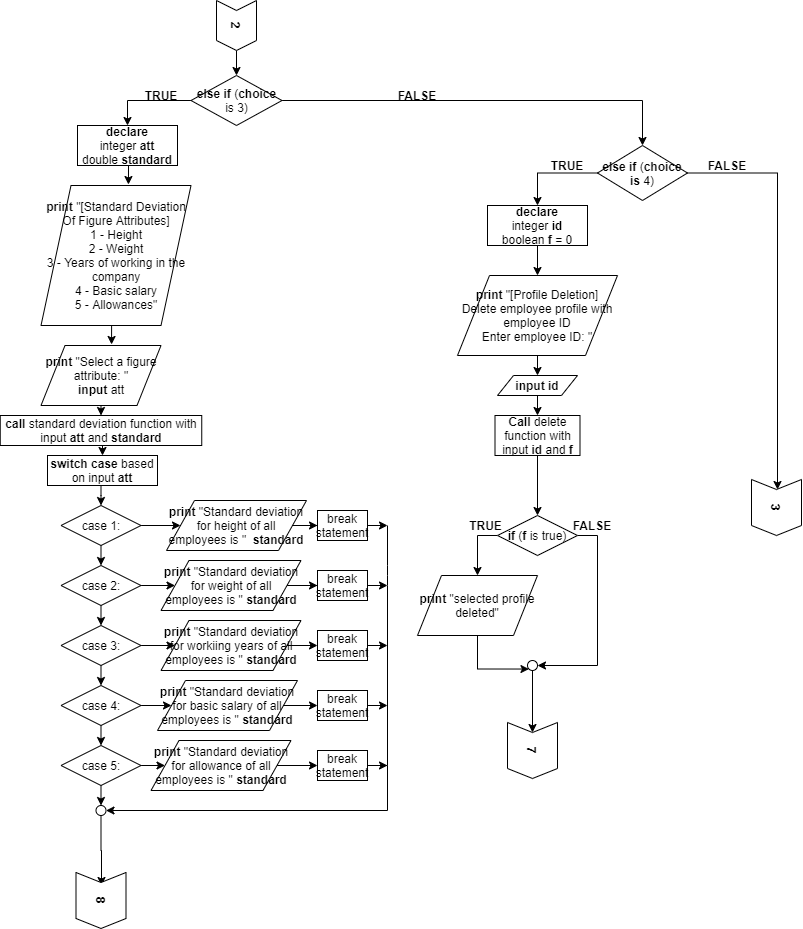
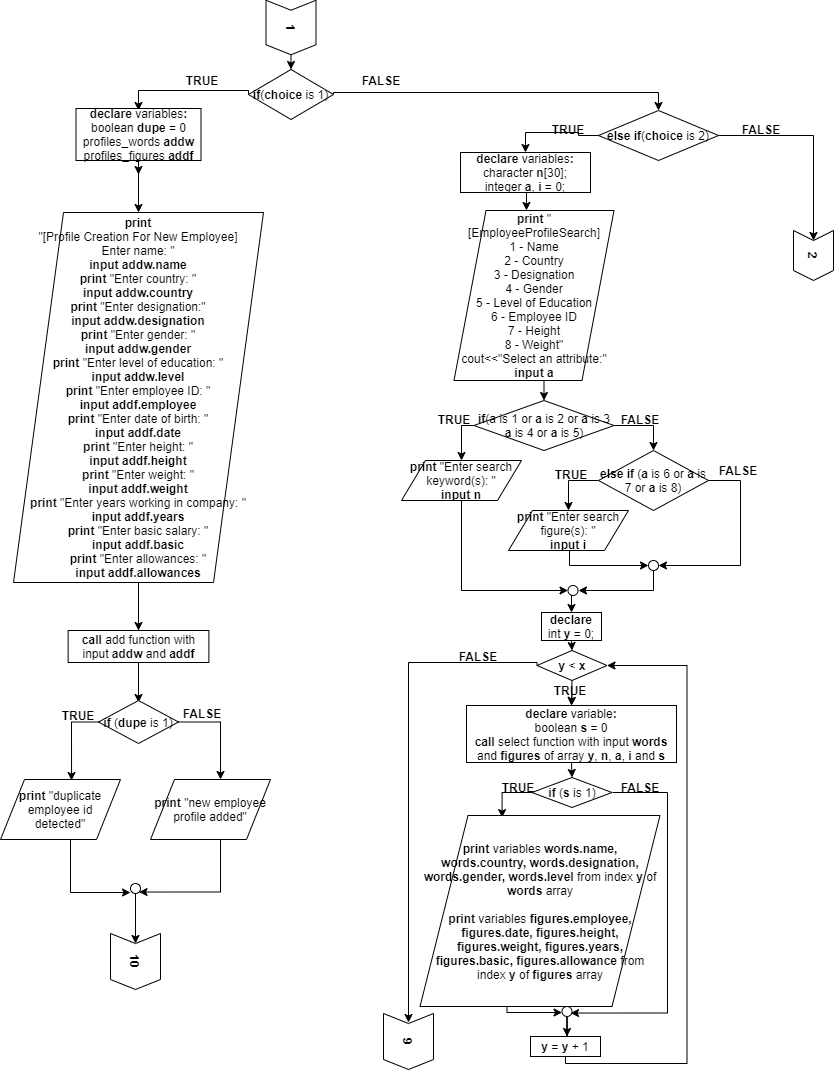
**2. Flowchart**





**3. Pseudocode**

Include header files iostream, string, fstream, cmath and iomanip

Use the namespace standard;

Create a struct called profiles\_figures that includes variables employee, date, height, weight, years, basic, allowances with data type double;

Create another struct called profiles\_words that includes name, country, designation, level with data type char and gender with data type string;

Prepare prototype function add with value parameters of data types profiles\_words, profiles\_figures and reference parameter bool;

Prepare prototype function select with value parameters of data types profiles\_words, profiles\_figures, char array, int, int and reference parameter bool;

Prepare prototype function sd with value parameters of data types int and reference parameter double;

Prepare prototype function del with value parameters of data types int and reference parameter bool;

Prepare prototype function add with value parameters of data types profiles\_words, profiles\_figures, char array, char array, char array, string, char array, int, int, int and reference parameter int;

Prepare prototype function add with value parameters of data types int and reference parameter double;

Main function,

Set a boolean called run to true let it keep running during usage;

Loop the codes while it is based on the bool run;

Open and input the text files profiles\_words.txt and profiles\_figures.txt with ifstream object inwords and infigures respectively;

If infigures or inwords fail to open,

Output profiles cannot be opened then break out of the loop;

Break out of the loop;

Else,

Declare structs named words and figures with array size 2400 using profiles\_words and profiles\_figures respectively;

Declare and assign integer x = 0, integer choice and char again;

Loop while inwords doesn’t reach end of file,

Input name, country, designation, gender and education level for words struct with inwords;

Input employee id, date of birth, height, weight, working years, salary and allowances for figure struct with infigures;

Add 1 to x;

Output the menu list of navigation to different function and total employees;

While keep looping,

Ask user to input choice function;

Input validation loop for values from 1 to 7;

If choice == 1,

Do,

Declare and assign bool dupe = 0;

Name new structs addw using profiles\_words and addf using profiles\_figures;

Ask and input name, country, designation, gender, level of education, employee id, date of birth, height, weight, working years, salary and allowances of employees;

Call the add function with input addw and addf;

If dupe is true,

Output duplicate detected;

Else,

Output employees added;

Prompt user if wanted to continue;

Input Y or N for repetition with validation;

While uppercase input is Y;

Else if choice == 2,

Do,

Declare char n to array size 30;

Declare a and initialize i = 0;

Output menu for employee search;

Input validation loop for values from 1 to 8;

If input is integer value from 1 to 5,

Prompt input search keywords;

Get line input to n with up to 30 characters;

Output search results;

Else if input is integer value from 6 to 8,

Prompt input search figures;

Input variable to i;

Output search results;

Loop for y smaller than total profiles x with initial y = 0 and y plus 1 to itself;

Declare boolean s = 0;

Call the select function with input words and figures of array y, n, a, i and s;

If s is true,

Output employee name, country with width 20 at the left, designation, gender and education level;

Output employee id, birthdate, height, weight, working years, salary and allowances;

Prompt user if wanted to continue;

Input Y or N for repetition with validation;

While uppercase input is Y;

Else if choice == 3,

Do,

Declare int att and double standard;

Output menu for attributes for standard deviation;

Input validation loop for values from 1 to 5;

Call standard deviation function with input att and standard;

Switch cases for input att,

In case 1:

Output height standard deviation;

Break out of the switch;

In case 2:

Output weight standard deviation;

Break out of the switch;

In case 3:

Output working years standard deviation;

Break out of the switch;

In case 4:

Output salary standard deviation;

Break out of the switch;

In case 5:

Output allowance standard deviation;

Break out of the switch;

Prompt user if wanted to continue;

Input Y or N for repetition with validation;

While uppercase input is Y;

Else if choice == 4.

Do,

Declare int id and bool f = 0;

Output deletion using employee id;

Input id;

Call delete function with input id and f;

If f is true,

Output profile deleted;

Else,

Output profile not found;

Prompt user if wanted to continue;

Input Y or N for repetition with validation;

While uppercase input is Y;

Else if choice == 5,

Do,

Declare and assign char n, o, p of array size 30, r of array size 10 with empty sets, string q with empty set, int c=1 and int e, h and w = 0;

Output menu for input;

While c is not 0,

Input validation loop for values from 1 to 9;

Switch cases for input c,

In case 1:

Prompt input name keywords and get line input for n up to 30 characters;

Break out of the switch;

In case 2:

Prompt input country and get line input for o up to 30 characters;

Break out of the switch;

In case 3:

Prompt input designation keywords and get line input for p up to 30 characters;

Break out of the switch;

In case 4:

Prompt input gender and get line input for q;

Break out of the switch;

In case 5:

Prompt input education level and input for r;

Break out of the switch;

In case 6:

Prompt input employee id and input for e;

Break out of the switch;

In case 7:

Prompt input height and input for h;

Break out of the switch;

In case 8:

Prompt input weight and get line input for n up to 30 characters;

Break out of the switch;

In case 9:

Assign 0 to c;

Break out of the switch;

Loop for y smaller than total profiles x with initial y = 0 and y plus 1 to itself;

Declare and assign 0 to s;

Call advanced search function with input words and figures of array y, n, o, p, q, r, e, h, w, and s;

If s is 1,

Output employee name, country with width 20 at the left, designation, gender and education level;

Output employee id, birthdate, height, weight, working years, salary and allowances;

Prompt user if wanted to continue;

Input Y or N for repetition with validation;

While uppercase input is Y;

Else if choice == 6,

Do,

Declare int att and double average;

Output the menu of attributes for mean;

Input validation loop for values from 1 to 5;

Call average function with input att and average;

Switch cases for input att,

In case 1:

Output the average height;

Break out of the switch;

In case 2:

Output the average weight;

Break out of the switch;

In case 3:

Output the average working years;

Break out of the switch;

In case 4:

Output the average salary;

Break out of the switch;

In case 5:

Output the average allowance;

Break out of the switch;

Prompt user if wanted to continue;

Input Y or N for repetition with validation;

While uppercase input is Y;

Else if choice == 7,

Output thanks for using;

Assign false to run;

Close inwords and infigures;

Return 0;

Pause the system;

Void add function with parameters profiles\_words addw, profiles\_figures addf and bool& dupe

Open and input the text files profiles\_figures.txt with ifstream object infigures;

Declare a struct named figures with array size 2400 using profiles\_figures;

Declare and assign integer x = 0;

Loop while infigures doesn’t reach end of file,

Input employee id, date of birth, height, weight, working years, salary and allowances for figure struct with infigures;

Add 1 to x;

For int y = 0; y smaller than x; y adds 1 to itself,

If employee id of infigures same as employee id of input;

Assign dupe to 1;

Break;

If dupe equals to 0,

Declare output for the text files profiles\_words.txt and profiles\_figures.txt with ofstream object outwords and outfigures respectively by appending;

Output name, country, designation, gender and education level for words struct with outwords;

Output employee id, date of birth, height, weight, working years, salary and allowances for figure struct with outfigures;

Close infigures;

Void select function with parameters profiles\_words swords, profiles\_figures sfigures, char\* letter, int a, int I and bool& s

Switch cases for input a,

For case 1:

If the input is a part of the variable name of swords,

Assign 1 to s;

For case 2:

If the input is a part of the variable country of swords,

Assign 1 to s;

For case 3:

If the input is a part of the variable designation of swords,

Assign 1 to s;

For case 4:

If the input is == gender variable of swords,

Assign s = 1;

For case 5:

If the input is a part of the variable level of swords,

Assign 1 to s;

For case 6:

If the input is == employee variables of sfigures;

Assign s = 1;

For case 7:

If the input is == height variables of sfigures;

Assign s = 1;

For case 8:

If the input is == weight variables of sfigures;

Assign s = 1;

Void standard deviation function with parameters int att and double& standard

Declare and assign 0 to x;

Open and input the text files profiles\_figures.txt with ifstream infigures;

Declare input the text files profiles\_figures.txt with ifstream object infigures;

Declare double total of each attributes and square total with initiation to 0;

Declare a struct named figures with array size 2400 using profiles\_figures;

Loop while infigures doesn’t reach end of file,

Input for every variable in struct figures respectively from infigures;

Each attribute will be summed to the total by adding the attributes to the total;

Each attribute will be squared and summed to the total by adding the attributes to the total;

1 will added to x;

Switch for the attributes

Each case with standard deviation calculation of square root of absolute values of attribute total divide by x minus the square of total attribute divide by x;

Void delete function with parameter int id and bool& f

Open and input the text files profiles\_words.txt and profiles\_figures.txt with ifstream object inwords and infigures respectively;

Declare structs named words and figures with array size 2400 using profiles\_words and profiles\_figures repectively;

Declare and assign 0 to x and a;

Loop while infigures doesn’t reach end of file,

Input employee id, date of birth, height, weight, working years, salary and allowances for figure struct with infigures;

Input employee id, date of birth, height, weight, working years, salary and allowances for figure struct with infigures;

Add 1 to x;

Open and output text files profiles\_words.txt and profiles\_figures.txt with ofstream object outwords and outfigures respectively;

Loop for y smaller than total profiles x with initial y = 0 and y plus 1 to itself;

If id == employee variable of figure,

Assign 1 to f;

Skip;

If a is true,

Add a line to outwords and outfigures;

Assign 1 to a;

Output name, country, designation, gender and education level for words struct with outwords;

Output employee id, date of birth, height, weight, working years, salary and allowances for figure struct with outfigures;

Close inwords, infigures, outwords and outfigures;

Void advanced search function with parameters profiles\_words awords, profiles\_figures afigures, char\* n, char\* o, char\* p, string q, char\* r, int e, int h, int w and int& s

Declare and assign 0 to bool bn, bo, bp, bq, br, be, bh, bw;

If n is an empty set or part of variable name of awords,

Assign 1 to bn;

If o is an empty set or part of variable country of awords,

Assign 1 to bo;

If p is an empty set or part of variable designation of awords,

Assign 1 to bp;

If q is empty or equals to variable gender of awords,

Assign 1 to bq;

If r is an empty set or part of variable level of awords,

Assign 1 to br;

If e is equals to 0 or equals to variable employee of afigures,

Assign 1 to be;

If h is equals to 0 or equals to variable employee of afigures,

Assign 1 to bh;

If w is equals to 0 or equals to variable employee of afigures,

Assign 1 to bw;

If bool of all the attributes are true,

Assign s = 1;

Void average function with parameters int att, double& average

Declare and assign x = 0;

Input the text files profiles\_figures.txt with ifstream object infigures;

Declare double total of each attributes with initiation to 0;

Declare a struct named figures with array size 2400 using profiles\_figures;

Loop while infigures doesn’t reach end of file,

Input for every variable in struct figures respectively from infigures;

Each attribute will be summed to the total by adding the attributes to the total;

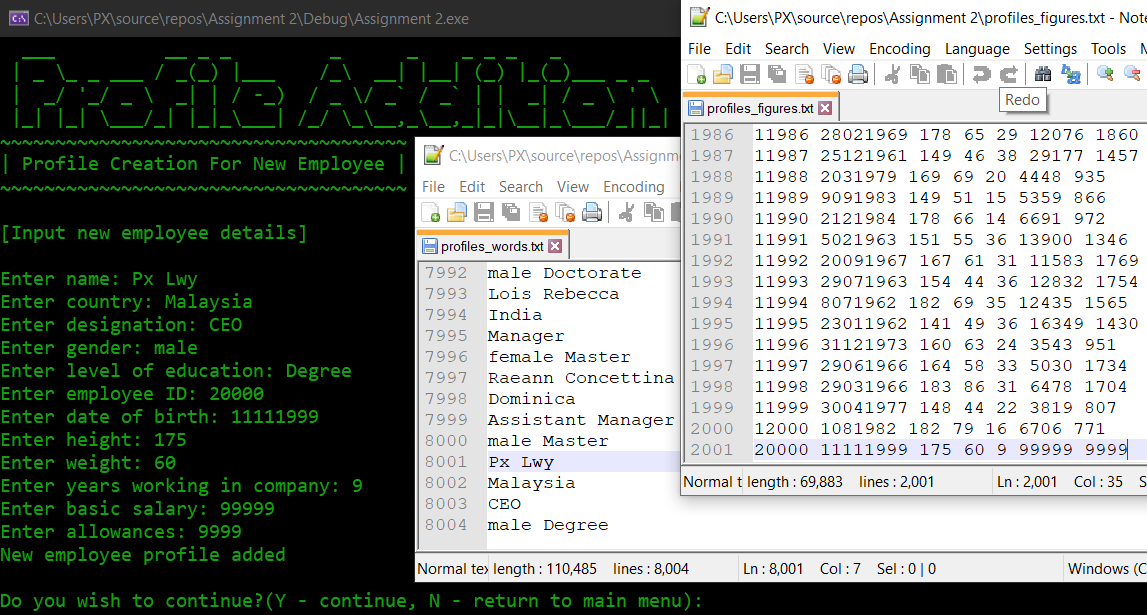
1 will added to x;

Switch for the attributes,

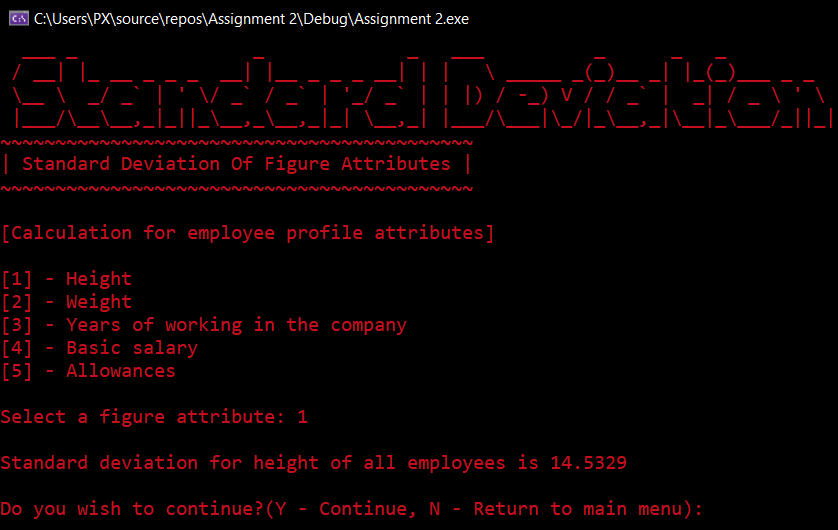
Each case with mean calculation of attribute total divide by x;

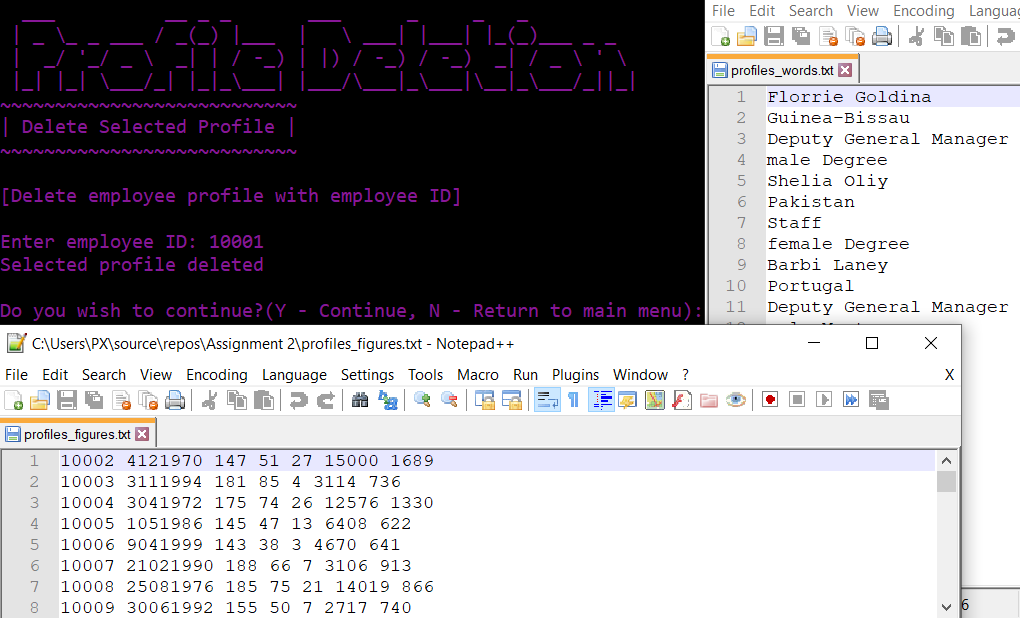
Close infigures;

**4. Test cases**











**5. Source code**

#include <iostream>

#include <string>

#include <fstream>

#include <cmath>

#include <iomanip>

using namespace std;

struct profiles\_figures {

double employee;

double date;

double height;

double weight;

double years;

double basic;

double allowances;

};

struct profiles\_words {

char name[30];

char country[40];

char designation[30];

string gender;

char level[10];

};

void add(profiles\_words, profiles\_figures, bool&);

void select(profiles\_words, profiles\_figures, char\*, int, int, bool&);

void sd(int, double&);

void del(int, bool&);

void as(profiles\_words, profiles\_figures, char\*, char\*, char\*, string, char\*, int, int, int, int&);

void aver(int, double&);

int main() {

bool run = true;

while (run) { // for looping back before quiting program

ifstream inwords("profiles\_words.txt");

ifstream infigures("profiles\_figures.txt");

if (infigures.fail() || inwords.fail()) { // if no files detected

cout << "Cannot open profiles" << endl;

break;

}

else {

profiles\_words words[2400];

profiles\_figures figures[2400];

int x = 0, choice;

char again;

while (!inwords.eof()) { // read and put every attribute word(s)/figure(s) into array

inwords.getline(words[x].name, 30);

inwords.getline(words[x].country, 40);

inwords.getline(words[x].designation, 30);

inwords >> words[x].gender >> words[x].level;

inwords.ignore();

infigures >> figures[x].employee >> figures[x].date >> figures[x].height >> figures[x].weight >> figures[x].years >> figures[x].basic >> figures[x].allowances;

infigures.ignore();

x++;

}

system("cls");

system("Color 1");

cout << R"( \_\_\_\_ \_\_ \_\_\_ \_\_\_\_ \_\_

/ \_\_/\_ \_ \_\_\_ / /\_\_ \_\_ \_\_\_\_\_ \_\_\_ \_\_\_ / \_ \\_\_\_\_\_\_\_ / \_(\_) /\_\_

/ \_// ' \/ \_ \/ / \_ \/ // / -\_) -\_|\_-< / \_\_\_/ \_\_/ \_ \/ \_/ / / -\_)

/\_\_\_/\_/\_/\_/ .\_\_/\_/\\_\_\_/\\_, /\\_\_/\\_\_/\_\_\_/ /\_/ /\_/ \\_\_\_/\_//\_/\_/\\_\_/

/\_/ /\_\_\_/

\_ \_\_ \_ \_\_ \_

/ |/ /\_\_ \_\_ \_\_(\_)\_\_ \_\_\_\_ \_/ /\_(\_)\_\_ \_\_\_

/ / \_ `/ |/ / / \_ `/ \_ `/ \_\_/ / \_ \/ \_ \

/\_/|\_/\\_,\_/|\_\_\_/\_/\\_, /\\_,\_/\\_\_/\_/\\_\_\_/\_//\_/

/\_\_\_/)" << endl;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n| Welcome to the program /\\\ Total employees: " << x << " |\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~";

cout << "\n\n[What would you like to do]\n\n[1] - Profile Creation\n[2] - Profile Search\n[3] - Standard Deviation of Figure Attributes\n[4] - Profile Deletion\n[5] - Advanced Profile Search\n[6] - Average of Figure Attributes\n[7] - Exit Program\n\n";

while (true) { // input validation

cout << "Please select an option: ";

if (cin >> choice && choice >= 1 && choice <= 7) {

break;

}

else {

cout << "Invalid input!" << endl;

}

cin.clear();

cin.ignore(1000, '\n');

}

if (choice == 1) { // addition module

system("Color 2");

do {

bool dupe = 0;

profiles\_words addw;

profiles\_figures addf;

system("cls");

cout << R"( \_\_\_ \_\_ \_ \_ \_ \_ \_ \_ \_ \_

| \_ \\_ \_ \_\_\_ / \_(\_) |\_\_\_ /\_\ \_\_| |\_\_| (\_) |\_(\_)\_\_\_ \_ \_

| \_/ '\_/ \_ \ \_| | / -\_) / \_ \/ \_` / \_` | | \_| / \_ \ ' \

|\_| |\_| \\_\_\_/\_| |\_|\_\\_\_\_| /\_/ \\_\\_\_,\_\\_\_,\_|\_|\\_\_|\_\\_\_\_/\_||\_|)" << endl;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n| Profile Creation For New Employee |\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~";

cout << "\n\n[Input new employee details]\n\n"; //get every value in both struct and pass to add function

cout << "Enter name: ";

cin.ignore();

cin.getline(addw.name, 30);

cout << "Enter country: ";

cin.getline(addw.country, 40);

cout << "Enter designation: ";

cin.getline(addw.designation, 30);

cout << "Enter gender: ";

getline(cin, addw.gender);

cout << "Enter level of education: ";

cin.getline(addw.level, 10);

cout << "Enter employee ID: ";

cin >> addf.employee;

cout << "Enter date of birth: ";

cin >> addf.date;

cout << "Enter height: ";

cin >> addf.height;

cout << "Enter weight: ";

cin >> addf.weight;

cout << "Enter years working in company: ";

cin >> addf.years;

cout << "Enter basic salary: ";

cin >> addf.basic;

cout << "Enter allowances: ";

cin >> addf.allowances;

add(addw, addf, dupe);

if (dupe) { // check duplicate of employee id

cout << "Duplicate of employee id detected\n\n";

}

else {

cout << "New employee profile added\n\n";

}

cout << "Do you wish to continue?(Y - continue, N - return to main menu): ";

cin >> again;

while (toupper(again) != 'Y' && toupper(again) != 'N') { //input validation

cout << "Please enter Y or N: ";

cin >> again;

}

} while (toupper(again) == 'Y');

}

else if (choice == 2) { // search module

system("Color 3");

do {

char n[30];

int a, i = 0;

system("cls");

cout << R"( \_\_\_ \_ \_\_\_ \_

| \_\_|\_ \_\_ \_ \_\_| |\_\_\_ \_ \_ \_\_\_ \_\_\_ / \_\_| \_\_\_ \_\_ \_ \_ \_ \_\_| |\_

| \_|| ' \| '\_ \ / \_ \ || / -\_) -\_) \\_\_ \/ -\_) \_` | '\_/ \_| ' \

|\_\_\_|\_|\_|\_| .\_\_/\_\\_\_\_/\\_, \\_\_\_\\_\_\_| |\_\_\_/\\_\_\_\\_\_,\_|\_| \\_\_|\_||\_|

|\_| |\_\_/)" << endl;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~\n| Employee Profile Search |\n~~~~~~~~~~~~~~~~~~~~~~~~~~~\n\n";

cout << "[Search with the following attributes]\n\n";

cout << "[1] - Name\t\t[2] - Country\t\t[3] - Designation\t[4] - Gender\n[5] - Education Level\t[6] - Employee ID\t[7] - Height\t\t[8] - Weight\n\n";

while (true) { // input validation

cout << "Please select an attribute: ";

if (cin >> a && a >= 1 && a <= 8)

break;

else

cout << "Invalid input!" << endl;

cin.clear();

cin.ignore(1000, '\n');

}

cin.ignore();

if (a == 1 || a == 2 || a == 3 || a == 4 || a == 5) {

cout << "Enter search keyword(s): ";

cin.getline(n, 30);

cout << endl << "Search results:" << endl << endl;

}

else if (a == 6 || a == 7 || a == 8) {

cout << "Enter search figure(s): ";

cin >> i;

cout << endl << "Search results:" << endl << endl;

}

for (int y = 0; y < x; y++) { // pass every value from array to select function for comparison

bool s = 0;

select(words[y], figures[y], n, a, i, s);

if (s == 1) {

cout << "Name: " << words[y].name << " Country: " << setw(20) << left << words[y].country << " Designation: " << words[y].designation << "\nGender: "

<< words[y].gender << " Education level: " << words[y].level << endl;

cout << setprecision(8) << "ID: " << figures[y].employee << " Birthdate: " << figures[y].date << " Height: " << figures[y].height << "cm Weight: " << figures[y].weight

<< "kg\nWorking years: " << figures[y].years << " Basic salary: RM" << figures[y].basic << " Allowances: RM" << figures[y].allowances << endl << endl;

}

}

cout << "Do you wish to continue?(Y - Continue, N - Return to main menu): ";

cin >> again;

while (toupper(again) != 'Y' && toupper(again) != 'N') {

cout << "Please enter Y or N: ";

cin >> again;

}

} while (toupper(again) == 'Y');

}

else if (choice == 3) { // standard deviation module

system("Color 4");

do {

int att;

double standard;

system("cls");

cout << R"( \_\_\_ \_ \_ \_ \_\_\_ \_ \_ \_

/ \_\_| |\_ \_\_ \_ \_ \_ \_\_| |\_\_ \_ \_ \_ \_\_| | | \ \_\_\_\_\_ \_(\_)\_\_ \_| |\_(\_)\_\_\_ \_ \_

\\_\_ \ \_/ \_` | ' \/ \_` / \_` | '\_/ \_` | | |) / -\_) V / / \_` | \_| / \_ \ ' \

|\_\_\_/\\_\_\\_\_,\_|\_||\_\\_\_,\_\\_\_,\_|\_| \\_\_,\_| |\_\_\_/\\_\_\_|\\_/|\_\\_\_,\_|\\_\_|\_\\_\_\_/\_||\_|)" << endl;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n| Standard Deviation Of Figure Attributes |\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";

cout << "\n[Calculation for employee profile attributes]\n\n";

cout << "[1] - Height\n[2] - Weight\n[3] - Years of working in the company\n[4] - Basic salary\n[5] - Allowances\n\n";

while (true) { // input validation

cout << "Select a figure attribute: ";

if (cin >> att && att >= 1 && att <= 5)

break;

else

cout << "Invalid input!" << endl;

cin.clear();

cin.ignore(1000, '\n');

}

cout << endl;

sd(att, standard); // pass in attribute selection and return sd value

switch (att) {

case 1:

cout << "Standard deviation for height of all employees is " << standard;

break;

case 2:

cout << "Standard deviation for weight of all employees is " << standard;

break;

case 3:

cout << "Standard deviation for years of working in the company for all employees is " << standard;

break;

case 4:

cout << "Standard deviation for basic salary of all employess is " << standard;

break;

case 5:

cout << "Standard deviation for allowance of all employess is " << standard;

break;

}

cout << endl << endl;

cout << "Do you wish to continue?(Y - Continue, N - Return to main menu): ";

cin >> again;

while (toupper(again) != 'Y' && toupper(again) != 'N') {

cout << "Please enter Y or N: ";

cin >> again;

}

} while (toupper(again) == 'Y');

}

else if (choice == 4) { // deletion module

system("Color 5");

do {

int id;

bool f = 0;

system("cls");

cout << R"( \_\_\_ \_\_ \_ \_ \_\_\_ \_ \_ \_

| \_ \\_ \_ \_\_\_ / \_(\_) |\_\_\_ | \ \_\_\_| |\_\_\_| |\_(\_)\_\_\_ \_ \_

| \_/ '\_/ \_ \ \_| | / -\_) | |) / -\_) / -\_) \_| / \_ \ ' \

|\_| |\_| \\_\_\_/\_| |\_|\_\\_\_\_| |\_\_\_/\\_\_\_|\_\\_\_\_|\\_\_|\_\\_\_\_/\_||\_|)" << endl;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~\n| Delete Selected Profile |\n~~~~~~~~~~~~~~~~~~~~~~~~~~~\n\n[Delete employee profile with employee ID]\n\nEnter employee ID: ";

cin >> id;

del(id, f);

if (f) {

cout << "Selected profile deleted\n\n";

}

else { // if no match for employee id

cout << "No profiles are deleted\n\n";

}

cout << "Do you wish to continue?(Y - Continue, N - Return to main menu): ";

cin >> again;

while (toupper(again) != 'Y' && toupper(again) != 'N') {

cout << "Please enter Y or N: ";

cin >> again;

}

} while (toupper(again) == 'Y');

}

else if (choice == 5) { // advance search module

system("Color 6");

do {

char n[30] = { '\0' }, o[30] = { '\0' }, p[30] = { '\0' }, r[10] = { '\0' };

string q = "\0";

int c = 1, e = 0, h = 0, w = 0;

system("cls");

cout << R"( \_ \_ \_ \_\_\_ \_

/\_\ \_\_| |\_ \_\_\_\_ \_ \_ \_ \_\_ \_\_\_ \_\_| | / \_\_| \_\_\_ \_\_ \_ \_ \_ \_\_| |\_

/ \_ \/ \_` \ V / \_` | ' \/ \_/ -\_) \_` | \\_\_ \/ -\_) \_` | '\_/ \_| ' \

/\_/ \\_\\_\_,\_|\\_/\\_\_,\_|\_||\_\\_\_\\_\_\_\\_\_,\_| |\_\_\_/\\_\_\_\\_\_,\_|\_| \\_\_|\_||\_|)" << endl;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n| Search Multiple Attributes |\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n\n[Please select search criteria up to 8 attributes] (input '9' to start searching)\n\n";

cout << "[1] - Name\t\t[2] - Country\t\t[3] - Designation\t[4] - Gender\n[5] - Education Level\t[6] - Employee ID\t[7] - Height\t\t[8] - Weight\n[9] - Search\n\n";

while (c != 0) { // option to input search keyword or values for other attributes before search

while (true) { // input validation

cout << "Insert search keyword(s)/figure(s) for: ";

if (cin >> c && c >= 1 && c <= 9)

break;

else

cout << "Invalid input!" << endl;

cin.clear();

cin.ignore(1000, '\n');

}

cin.ignore();

switch (c) {

case 1:

cout << "Enter name keyword(s): ";

cin.getline(n, 30);

break;

case 2:

cout << "Enter country keyword(s): ";

cin.getline(o, 30);

break;

case 3:

cout << "Enter designation keyword(s): ";

cin.getline(p, 30);

break;

case 4:

cout << "Enter gender keyword(s): ";

getline(cin, q);

break;

case 5:

cout << "Enter level of education keyword(s): ";

cin >> r;

break;

case 6:

cout << "Enter employee ID figure(s): ";

cin >> e;

break;

case 7:

cout << "Enter height figures(s): ";

cin >> h;

break;

case 8:

cout << "Enter weight figures(s): ";

cin >> w;

break;

case 9:

c = 0;

break;

}

}

cout << "\nAdvanced search results:\n\n";

for (int y = 0; y < x; y++) {

int s = 0;

as(words[y], figures[y], n, o, p, q, r, e, h, w, s);

if (s == 1) {

cout << "Name: " << words[y].name << " Country: " << setw(20) << left << words[y].country << " Designation: " << words[y].designation << "\nGender: "

<< words[y].gender << " Education level: " << words[y].level << endl;

cout << setprecision(8) << "ID: " << figures[y].employee << " Birthdate: " << figures[y].date << " Height: " << figures[y].height << "cm Weight: " << figures[y].weight

<< "kg\nWorking years: " << figures[y].years << " Basic salary: RM" << figures[y].basic << " Allowances: RM" << figures[y].allowances << endl << endl;

}

}

cout << "Do you wish to continue?(Y - continue, N - return to main menu): ";

cin >> again;

while (toupper(again) != 'Y' && toupper(again) != 'N')

{

cout << "Please enter Y or N: ";

cin >> again;

}

} while (toupper(again) == 'Y');

}

else if (choice == 6) { // average module

system("Color 6");

do {

int att;

double average;

system("cls");

cout << R"( \_ \_\_ \_\_ \_

/\_\\_\_ \_\_\_\_\_ \_ \_ \_\_ \_ \_\_ \_ \_\_\_ \ \ / /\_ \_| |\_ \_ \_\_\_

/ \_ \ V / -\_) '\_/ \_` / \_` / -\_) \ V / \_` | | || / -\_)

/\_/ \\_\\_/\\_\_\_|\_| \\_\_,\_\\_\_, \\_\_\_| \\_/\\_\_,\_|\_|\\_,\_\\_\_\_|

|\_\_\_/)" << endl;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n| Average Value Of Figure Attributes |\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n\n";

cout << "[Calculation for employee profile attributes]\n\n";

cout << "[1] - Height\n[2] - Weight\n[3] - Years of working in the company\n[4] - Basic salary\n[5] - Allowances\n\n";

while (true) { // input validation

cout << "Select a figure attribute: ";

if (cin >> att && att >= 1 && att <= 5)

break;

else

cout << "Invalid input!" << endl;

cin.clear();

cin.ignore(1000, '\n');

}

aver(att, average);

cout << endl;

switch (att) {

case 1:

cout << "Average value for height of all employees is " << average;

break;

case 2:

cout << "Average value for weight of all employees is " << average;

break;

case 3:

cout << "Average value for years of working in the company for all employees is " << average;

break;

case 4:

cout << "Average value for basic salary of all employess is " << average;

break;

case 5:

cout << "Average value for allowance of all employess is " << average;

break;

}

cout << "\n\nDo you wish to continue?(Y - continue, N - return to main menu): ";

cin >> again;

while (toupper(again) != 'Y' && toupper(again) != 'N') {

cout << "Please enter Y or N: ";

cin >> again;

}

} while (toupper(again) == 'Y');

}

else if (choice == 7) { // exit program

system("Color 9");

system("cls");

cout << R"( \_\_\_\_\_ \_ \_ \_\_ \_\_

|\_ \_| |\_ \_\_ \_ \_ \_ | |\_\_ \ \ / /\_\_ \_ \_

| | | ' \/ \_` | ' \| / / \ V / \_ \ || |

|\_| |\_||\_\\_\_,\_|\_||\_|\_\\_\ |\_|\\_\_\_/\\_,\_|

\_\_\_ \_ \_ \_

| \_\_|\_\_ \_ \_ | | | |\_\_(\_)\_ \_ \_\_ \_

| \_/ \_ \ '\_| | |\_| (\_-< | ' \/ \_` |

|\_|\\_\_\_/\_| \\_\_\_//\_\_/\_|\_||\_\\_\_, |

|\_\_\_/

\_\_\_\_\_ \_ \_\_\_ \_ \_

|\_ \_| |\_ \_\_\_ / \_\_|\_ \_ \_\_| |\_ \_\_\_ \_ \_\_ | |

| | | ' \/ -\_) \\_\_ \ || (\_-< \_/ -\_) ' \|\_|

|\_| |\_||\_\\_\_\_| |\_\_\_/\\_, /\_\_/\\_\_\\_\_\_|\_|\_|\_(\_)

|\_\_/)" << "\n \_\_\_\_\_\_ \n / \\\n| ^ ^ |\n| \\/ |\n \\\_\_\_\_\_\_/\n\nThe program will exit now.\n";

run = false;

}

}

inwords.close();

infigures.close();

}

return 0;

system("pause");

}

void add(profiles\_words addw, profiles\_figures addf, bool& dupe) { // add function

ifstream infigures("profiles\_figures.txt");

profiles\_figures figures[2400];

int x = 0;

while (!infigures.eof()) {

infigures >> figures[x].employee >> figures[x].date >> figures[x].height >> figures[x].weight >> figures[x].years >> figures[x].basic >> figures[x].allowances;

infigures.ignore();

x++;

}

for (int y = 0; y < x; y++) { // find duplicate of employee id input

if (figures[y].employee == addf.employee) {

dupe = 1;

break;

}

}

if (dupe == 0) {

ofstream outwords("profiles\_words.txt", ios::app); // add to bottom list of file

ofstream outfigures("profiles\_figures.txt", ios::app);

outwords << endl << addw.name << endl << addw.country << endl << addw.designation << endl << addw.gender << " " << addw.level; // add new employee details to bottom of file

outfigures << setprecision(8) << endl << addf.employee << " " << addf.date << " " << addf.height << " " << addf.weight << " " << addf.years << " " << addf.basic << " " << addf.allowances;

}

infigures.close();

}

void select(profiles\_words swords, profiles\_figures sfigures, char\* letter, int a, int i, bool& s) { // select function

switch (a) {

case 1:

if (strstr(swords.name, letter)) { // compare and if letter is part of words attributes then output non zero

s = 1;

}

break;

case 2:

if (strstr(swords.country, letter)) {

s = 1;

}

break;

case 3:

if (strstr(swords.designation, letter)) {

s = 1;

}

break;

case 4:

if (swords.gender == letter) { // compare and if letter is equals to gender attribute

s = 1;

}

break;

case 5:

if (strstr(swords.level, letter)) {

s = 1;

}

break;

case 6:

if (sfigures.employee == i) { // compare and if letters is equals to figures attributes

s = 1;

}

break;

case 7:

if (sfigures.height == i) {

s = 1;

}

break;

case 8:

if (sfigures.weight == i) {

s = 1;

}

break;

}

}

void sd(int att, double& standard) { // standard deviation function

int x = 0;

ifstream infigures("profiles\_figures.txt");

double totalh = 0, totalw = 0, totaly = 0, totalb = 0, totala = 0;

double totalh2 = 0, totalw2 = 0, totaly2 = 0, totalb2 = 0, totala2 = 0;

profiles\_figures figures[2400];

while (!infigures.eof()) { // get every attribute figures from file

infigures >> figures[x].employee >> figures[x].date >> figures[x].height >> figures[x].weight >> figures[x].years >> figures[x].basic >> figures[x].allowances;

totalh += figures[x].height; // total up each figures

totalw += figures[x].weight;

totaly += figures[x].years;

totalb += figures[x].basic;

totala += figures[x].allowances;

totalh2 += pow(figures[x].height, 2); // square and total up each figures

totalw2 += pow(figures[x].weight, 2);

totaly2 += pow(figures[x].years, 2);

totalb2 += pow(figures[x].basic, 2);

totala2 += pow(figures[x].allowances, 2);

x++;

}

switch (att) {

case 1:

standard = sqrt(abs(totalh2 / x - pow(totalh / x, 2))); // calculate sd for each figure values

break;

case 2:

standard = sqrt(abs(totalw2 / x - pow(totalw / x, 2)));

break;

case 3:

standard = sqrt(abs(totaly2 / x - pow(totaly / x, 2)));

break;

case 4:

standard = sqrt(abs(totalb2 / x - pow(totalb / x, 2)));

break;

case 5:

standard = sqrt(abs(totala2 / x - pow(totala / x, 2)));

break;

}

infigures.close();

}

void del(int id, bool& f) { // delete function

ifstream inwords("profiles\_words.txt");

ifstream infigures("profiles\_figures.txt");

profiles\_words words[2400];

profiles\_figures figures[2400];

int x = 0, a = 0;

while (!inwords.eof()) { // get each attribute word(s)/figure(s) into array

inwords.getline(words[x].name, 30);

inwords.getline(words[x].country, 40);

inwords.getline(words[x].designation, 30);

inwords >> words[x].gender >> words[x].level;

inwords.ignore();

infigures >> figures[x].employee >> figures[x].date >> figures[x].height >> figures[x].weight >> figures[x].years >> figures[x].basic >> figures[x].allowances;

x++;

}

ofstream outwords("profiles\_words.txt");

ofstream outfigures("profiles\_figures.txt");

for (int y = 0; y < x; y++) { // loop for checking and output

if (id == figures[y].employee) { // skip output if employee id same as input

f = 1; // detects if employees deleted

continue;

}

if (a) { // do not endl at the top of text file

outwords << endl;

outfigures << endl;

}

a = 1;

outwords << words[y].name << endl << words[y].country << endl << words[y].designation << endl << words[y].gender << " " << words[y].level;

outfigures << setprecision(8) << figures[y].employee << " " << figures[y].date << " " << figures[y].height << " " << figures[y].weight

<< " " << figures[y].years << " " << figures[y].basic << " " << figures[y].allowances;

}

inwords.close();

infigures.close();

outwords.close();

outfigures.close();

}

void as(profiles\_words awords, profiles\_figures afigures, char\* n, char\* o, char\* p, string q, char\* r, int e, int h, int w, int& s) { // advanced search function

bool bn = 0, bo = 0, bp = 0, bq = 0, br = 0, be = 0, bh = 0, bw = 0;

if (n[0] == '\0' || strstr(awords.name, n)) { // set bool to true if attributes matches input or no input (default)

bn = 1;

}

if (o[0] == '\0' || strstr(awords.country, o)) {

bo = 1;

}

if (p[0] == '\0' || strstr(awords.designation, p)) {

bp = 1;

}

if (q == "\0" || awords.gender == q) {

bq = 1;

}

if (r[0] == '\0' || strstr(awords.level, r)) {

br = 1;

}

if (e == 0 || afigures.employee == e) {

be = 1;

}

if (h == 0 || afigures.height == h) {

bh = 1;

}

if (w == 0 || afigures.weight == w) {

bw = 1;

}

if (bn && bo && bp && bq && br && be && bh && bw) { //if conditions met then return s to output profiles

s = 1;

}

}

void aver(int att, double& average) { // average function

int x = 0;

ifstream infigures("profiles\_figures.txt");

double totalh = 0, totalw = 0, totaly = 0, totalb = 0, totala = 0;

profiles\_figures figures[2400];

while (!infigures.eof()) { // get all figure attributes from file to array

infigures >> figures[x].employee >> figures[x].date >> figures[x].height >> figures[x].weight >> figures[x].years >> figures[x].basic >> figures[x].allowances;

totalh += figures[x].height; // calculate total for all figure attributes

totalw += figures[x].weight;

totaly += figures[x].years;

totalb += figures[x].basic;

totala += figures[x].allowances;

x++;

}

switch (att) {

case 1:

average = totalh / x; // calculate mean for all figure attributes

break;

case 2:

average = totalw / x;

break;

case 3:

average = totaly / x;

break;

case 4:

average = totalb / x;

break;

case 5:

average = totala / x;

break;

}

infigures.close();

}