Project by: **Muhammad Anas Baig**

01-134152-037

BS(CS)-3A



[Document subtitle]

December 21, 2016

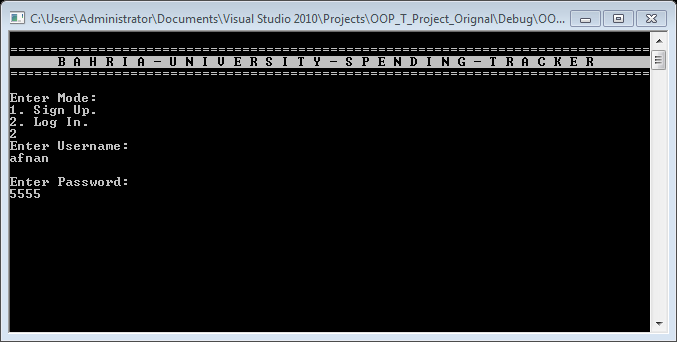
Title: **Spending(Budget) Tracking Application**

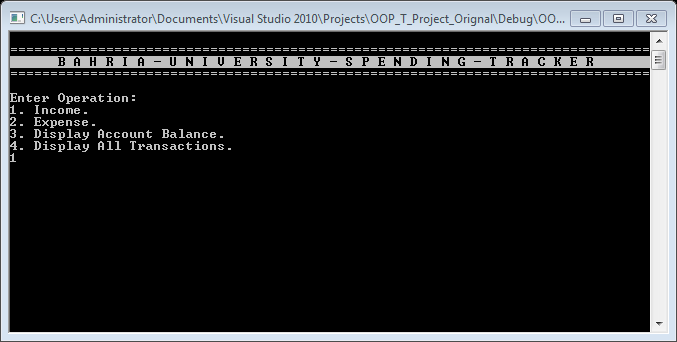
**Introduction:**

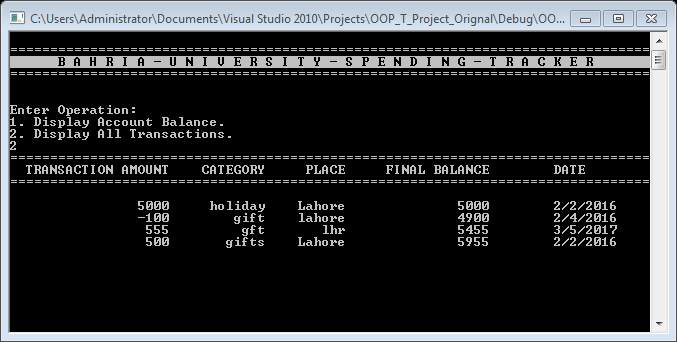
Spending Tracker is basically an application which keeps record of the daily **Income** and **Expenses** of the user. It helps the user to maintains his daily budget and saves money by keeping an eye on his spending.  
 **OOP Concepts Used:**

1. Inheritance.
2. Polymorphism.
3. Composition.
4. Operator Overloading.
5. Operator Overriding.
6. Files and Streams.
7. Pointers.
8. Classes.
9. Functions.  
   10. Strings  
     
   **Application Features:**

* Secure Sign Up and Log In.
* Separate file for each user which can’t be accessed without password.
* Add Incomes/Expenses with information such as date, place, and category.
* Displays remaining balance in Account.
* Displays Transactions List of the user
* Best and most important feature is that it can be used on a Public PC where each user has its own USERNAME and PASSWORD and has its own separate file which can’t be accessed without password.

**Output:**





**Application Code**

**main.cpp:**

#include "iomanip"

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include "new\_user.h"

#include "old\_user.h"

#include "transaction.h"

#include "income.h"

#include "expense.h"

#include "user.h"

#include "string"

#include "string.h"

#include "conio.h"

#include "fstream"

#include <iostream>

using namespace std;

int main()

{

system("CLS");

cout<<endl<<"================================================================================";

HANDLE m\_hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute ( m\_hConsole,

BACKGROUND\_RED |

BACKGROUND\_GREEN |

BACKGROUND\_BLUE );

std::cout <<" B A H R I A - U N I V E R S I T Y - S P E N D I N G - T R A C K E R ";

SetConsoleTextAttribute ( m\_hConsole,

FOREGROUND\_RED |

FOREGROUND\_GREEN |

FOREGROUND\_BLUE );

cout<<"================================================================================"<<endl;

int choice1, choice2, choice3, choice4; //variables to get user choices

user \*u; //pointer to user class

transaction \*t; //pointer to transaction class

cout<<"Enter Mode:"<<endl;

cout<<"1. Sign Up."<<endl;

cout<<"2. Log In."<<endl;

cin>>choice1; //gets mode input

if(choice1==1)

{

u = new new\_user; //dynamic memory allocation | polymorphism used

}

else

{

u = new old\_user; //dynamic memory allocation | polymorphism used

}

u->getData(); //gets transaction(income/expense) data from user

system("CLS");

cout<<endl<<"================================================================================";

// HANDLE m\_hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute ( m\_hConsole,

BACKGROUND\_RED |

BACKGROUND\_GREEN |

BACKGROUND\_BLUE );

std::cout <<" B A H R I A - U N I V E R S I T Y - S P E N D I N G - T R A C K E R ";

SetConsoleTextAttribute ( m\_hConsole,

FOREGROUND\_RED |

FOREGROUND\_GREEN |

FOREGROUND\_BLUE );

cout<<"================================================================================"<<endl;

cout<<endl;

system("CLS");

cout<<endl<<"================================================================================";

//HANDLE m\_hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute ( m\_hConsole,

BACKGROUND\_RED |

BACKGROUND\_GREEN |

BACKGROUND\_BLUE );

std::cout <<" B A H R I A - U N I V E R S I T Y - S P E N D I N G - T R A C K E R ";

SetConsoleTextAttribute ( m\_hConsole,

FOREGROUND\_RED |

FOREGROUND\_GREEN |

FOREGROUND\_BLUE );

cout<<"================================================================================"<<endl;

do

{

cout<<"Enter Operation:"<<endl;

cout<<"1. Income."<<endl;

cout<<"2. Expense."<<endl;

cout<<"3. Display Account Balance."<<endl;

cout<<"4. Display All Transactions."<<endl;

cin>>choice2; //gets operation choice

if(choice2==1)

{

t = new income; //dynamic memory allocation | polymorphism used

t->getAmount(u->getUsername(), u->getPassword()); //files of the user will be created on its username.txt(final balance) and password.txt(transactions details)

}

else if (choice2==2)

{

t = new expense; //dynamic memory allocation | polymorphism used

t->getAmount(u->getUsername(), u->getPassword()); //files of the user will be created on its username.txt(final balance) and password.txt(transactions details)

}

else if(choice2==3)

{

t= new income; //dynamic memory allocation | polymorphism used

string s = u->getUsername();

(\*t) + s; //operator overloaded which takes username as argument to open username.txt

}

else

{

t= new expense; //dynamic memory allocation | polymorphism used

t->displayTransaction(u->getPassword()); //files of the user will be created on its username.txt(final balance) and password.txt(transactions details)

}

/\*system("CLS");

cout<<endl<<"================================================================================";

//HANDLE m\_hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute ( m\_hConsole,

BACKGROUND\_RED |

BACKGROUND\_GREEN |

BACKGROUND\_BLUE );

std::cout <<" B A H R I A - U N I V E R S I T Y - S P E N D I N G - T R A C K E R ";

SetConsoleTextAttribute ( m\_hConsole,

FOREGROUND\_RED |

FOREGROUND\_GREEN |

FOREGROUND\_BLUE );

cout<<"================================================================================"<<endl;\*/

cout<<endl;

cout<<endl<<"Do you want to do operation again?"<<endl;

cout<<"1. YES."<<endl;

cout<<"2. NO."<<endl;

cin>>choice4;

}

while(choice4 == 1);

system("CLS");

cout<<endl<<"================================================================================";

SetConsoleTextAttribute ( m\_hConsole,

BACKGROUND\_RED |

BACKGROUND\_GREEN |

BACKGROUND\_BLUE );

std::cout <<" B A H R I A - U N I V E R S I T Y - S P E N D I N G - T R A C K E R ";

SetConsoleTextAttribute ( m\_hConsole,

FOREGROUND\_RED |

FOREGROUND\_GREEN |

FOREGROUND\_BLUE );

cout<<"================================================================================"<<endl;

cout<<endl<<"Enter Operation:"<<endl;

cout<<"1. Display Account Balance."<<endl;

cout<<"2. Display All Transactions."<<endl;

cin>>choice3;

if(choice3==1)

{

string s = u->getUsername();

(\*t) + s; //operator overloaded which takes username as argument to open username.txt

}

else

{

t= new expense;

t->displayTransaction(u->getPassword()); //functions gets password as argument which will open password.txt file for transaction details

}

getch();

}

**user.h Code:**

#pragma once

#include "string"

#include "string.h"

#include <iostream>

using namespace std;

class user

{

protected:

string username;

string password;

public:

user(void);

virtual void getData()=0; //pure virtual function to apply polymorphism and get Username and Password from User

string getUsername(); //get variable method

string getPassword(); //get variable method

};

**user.cpp:**

#include "user.h"

#include "string"

#include "string.h"

#include <iostream>

using namespace std;

user::user(void)

{

}

string user::getUsername() //get variable method

{

return(username);

}

string user::getPassword() //get variable method

{

return(password);

}

**new\_user.h Code:**

#pragma once

#include "user.h"

#include "string"

#include "string.h"

#include "fstream"

#include <iostream>

using namespace std;

class new\_user:public user

{

public:

new\_user(void); //constructor

void getData(); //get input of username and password from user

};

**new\_user.cpp Code:**

#include "new\_user.h"

#include "income.h"

#include "user.h"

#include "fstream"

#include "string"

#include "string.h"

#include <iostream>

using namespace std;

new\_user::new\_user(void) //constructor

{

}

void new\_user::getData() //get input from user

{

int checkpoint; //variable to store check value

string temp\_username; //stores username temporarily

string temp\_password; //stores password temporarily

do //asks user for choices and operaations as many times he want to do operations

{

checkpoint=0; //set to zero

cout<<"Enter Username:"<<endl;

cin>>temp\_username; //get username

cout<<endl<<"Enter Password:"<<endl;

cin>>temp\_password; //gets password

ifstream read\_file("credentials.txt", ios::binary); //file open read mode to check either entered username is already registered or not

for( ;!(read\_file.eof()); ) //checks for EOF, no intialization or increment statements necassary

{

read\_file.read(reinterpret\_cast<char \*>(this), sizeof(\*this)); //reads object from file

if(temp\_username == this->username) //if entered username is registered

{

cout<<"Username Already Found."<<endl<<endl;

checkpoint=1; //set to 1

}

}

read\_file.close(); //file close

}

while(checkpoint==1); //continues loop unless checkpoint=1 (i.e. username already registered)

if(checkpoint==0) //if duplicate usernaem not found

{

this->username = temp\_username; //stores username to object

this->password = temp\_password; //stores password to object

income i; //income class object

i.setZero(this->username, this->password); //this function sets all variables of income class to 0

ofstream write\_file("credentials.txt",ios::binary|ios::app); //file write mode open to save new user login information to file

write\_file.write(reinterpret\_cast<char \*>(this), sizeof(\*this)); //writes object to file

write\_file.close(); //file close

}

}

**old\_user.h Code:**

#pragma once

#include "user.h"

#include "fstream"

#include "string"

#include "string.h"

#include <iostream>

using namespace std;

class old\_user : public user //inheritance applied base class 'user'

{

public:

old\_user(void); //constructor

void getData(); //function to get input of USERNAME and PASSWORD from user

};

**old\_user.cpp:**

#include "old\_user.h"

#include "user.h"

#include "fstream"

#include "string"

#include "string.h"

#include <iostream>

using namespace std;

old\_user::old\_user(void) //constructor

{

}

void old\_user::getData() //variable to get input data from user

{

int checkpoint; //variable to store check value

string temp\_username; //temporary username variable

string temp\_password; //temporary password variable

do //loop to get input from user as many times he wants

{

checkpoint=0; //set to 0

cout<<"Enter Username:"<<endl;

cin>>temp\_username; //username input

cout<<endl<<"Enter Password:"<<endl;

cin>>temp\_password; //password input

ifstream read\_file("credentials.txt", ios::binary); //read mode file open which contains usernames and passwords of all users

for( ;!(read\_file.eof()); ) //no initialization or increment statements because there we only check loop with EOF

{

read\_file.read(reinterpret\_cast<char \*>(this), sizeof(\*this)); //reading credentials file

if((temp\_username==this->username) && (temp\_password==this->password)) //it checks either username and password are correct or not

{

checkpoint=1; //if username and password are entered correctly

}

}

if(checkpoint==0) //if username or password is incorrectly typed

{

cout<<"Incorrect USERNAME or PASSWORD!!! TRY AGAIN"<<endl<<endl;

}

read\_file.close(); //file close

}

while(checkpoint==0); //let it get input from user unless correct username and password is entered (i.e. checkpoint=1)

if(checkpoint==1) //condition for username and password are correctly typed

{

this->username = temp\_username; //stores username to object

this->password = temp\_password; //stores password to object

}

}

**date.h Code:**

#pragma once

#include<iostream>

using namespace std;

class Date

{

public:

int day, month, year;

public:

Date(void);

void getInput();

void displayDate();

};

**date.cpp Code:**

#include "Date.h"

#include<iostream>

using namespace std;

Date::Date(void)

{

}

void Date::getInput()

{

cout<<endl<<"Enter Date:-"<<endl;

cout<<"Day:"<<endl;

cin>>day;

cout<<"Month:"<<endl;

cin>>month;

cout<<"Year:"<<endl;

cin>>year;

}

void Date::displayDate()

{

cout<<day<<"/"<<month<<"/"<<year;

}

**transaction.h Code:**

#pragma once

#include "fstream"

#include "date.h"

#include "string"

#include "string.h"

#include <iostream>

using namespace std;

class transaction

{

protected:

int amount; //transaction amount

int balance; //final balance

string category; //type of transaction

string place; //place where transaction made

Date d; //to store date of transaction

public:

transaction(void); //constructor

virtual void getAmount(string, string)=0; //function to get input from user

virtual void displayTransaction(string)=0; //function to display all transactions

virtual void operator + (string)=0; //operator overloaded to display final balance in account

};

**transaction.cpp Code:**

#include "transaction.h"

#include "fstream"

#include "string"

#include "string.h"

#include <iostream>

using namespace std;

transaction::transaction(void) //constructor

{

**income.h Code:**

#pragma once

#include "transaction.h"

#include "date.h"

#include "fstream"

#include "string"

#include "string.h"

#include <iostream>

using namespace std;

class income:public transaction //inheritance applied base class 'transaction'

{

public:

income(void); //constructor

void setZero(string, string); //sets all variables to 0 of income class, also takes username and password as arguments to make file names

void getAmount(string, string); //gets user input

void displayTransaction(string); //displays all transactions

void operator + (string); //operator overloaded to display Account balance (argument is filename of the user which contains its all log)

};

**income.cpp Code:**

#include "iomanip"

#include "income.h"

#include "transaction.h"

#include "fstream"

#include "string"

#include "string.h"

#include <iostream>

using namespace std;

income::income(void) //constructor

{

}

void income::setZero(string username, string password ) //gets username and password to create 2 files, first one will get final balace details while second one will get all transaction details

{

this->amount=0; //set to zero

this->balance=0; //set to zero

ofstream username\_write(username.c\_str(), ios::binary); //file write mode open to store final balance

username\_write.write(reinterpret\_cast<char \*>(this), sizeof(\*this)); //writes object to file

username\_write.close(); //file close

ofstream password\_write(password.c\_str(), ios::binary); //file write mode open to store all transactions

password\_write.write(reinterpret\_cast<char \*>(this), sizeof(\*this)); //writes object to file

password\_write.close(); //file close

}

void income::getAmount(string username, string password)

{

int temp\_amount; //temporary amount storing variable

cout<<"Enter Income Amount:"<<endl;

cin>>temp\_amount; //gets input

ifstream username\_read(username.c\_str(), ios::binary); //file open read mode to get final balance

username\_read.read(reinterpret\_cast<char \*>(this), sizeof(\*this)); //reads object from file

balance = balance + temp\_amount; //adds amount to fianl balance

username\_read.close(); //file close

this->amount = temp\_amount; //sets amount

cout<<endl<<"Enter Transaction Category(e.g. gift, holiday, shopping etc.):"<<endl;

cin>>this->category; //transaction type

cout<<endl<<"Enter Transaction Place:"<<endl;

cin>>this->place; //transaction location

this->d.getInput(); //date input

ofstream username\_write(username.c\_str(), ios::binary); //file write mode open for final balance reading

username\_write.write(reinterpret\_cast<char \*>(this), sizeof(\*this)); //reads object form file

username\_write.close(); //fiel close

//-----------------------Above is work with username.txt file which only has 1 record for final balance------

//-----------------------Below is work with password.txt file which contains transactions list------

ofstream password\_write(password.c\_str(), ios::binary|ios::app); //file write mode open to read transactions details

password\_write.write(reinterpret\_cast<char \*>(this), sizeof(\*this)); //writes objec tto file

password\_write.close(); //file close

}

void income::displayTransaction(string password) //displays all transactions

{

ifstream file\_read(password.c\_str(), ios::binary); //file read mode open to read tranasaction details

cout<<"================================================================================";

cout<<setw(20)<<"TRANSACTION AMOUNT";

cout<<setw(12)<<"CATEGORY";

cout<<setw(10)<<"PLACE";

cout<<setw(18)<<"FINAL BALANCE";

cout<<setw(12)<<"DATE"<<endl;

cout<<"================================================================================"<<endl;

for(int i=0;!file\_read.eof();i++) //continues untill EOF reaches

{

file\_read.read(reinterpret\_cast<char \*>(this), sizeof(\*this)); //reads object from file

if(i==0) //skips garbage data

{

continue;

}

cout<<setw(20)<<amount<<setw(12)<<category<<setw(10)<<place<<setw(18)<<balance;

cout<<setw(9);

d.displayDate();

cout<<endl;

}

}

void income:: operator + (string username) //operator overloaded to display final balance

{

ifstream username\_read(username.c\_str(), ios::binary); //file read mode open to read final balance

username\_read.read(reinterpret\_cast<char \*>(this), sizeof(\*this)); //reads object form file

cout<<endl<<"Account Balance: "<<this->balance<<endl;

username\_read.close(); //file close

}

**expense.h Code:**

#pragma once

#include "transaction.h"

#include "date.h"

#include "fstream"

#include "string"

#include "string.h"

#include <iostream>

using namespace std;

class expense:public transaction

{

public:

expense(void); //constructor

void setZero(string, string); //set all variables to zero

void getAmount(string, string); //gets input from user

void displayTransaction(string); //displays all transactions

void operator + (string); //operator overloaded to display final balance of account

};

**expense.cpp Code:**

#include "expense.h"

#include "iomanip"

#include "transaction.h"

#include "fstream"

#include "string"

#include "string.h"

#include <iostream>

using namespace std;

expense::expense(void) //constructor

{

}

void expense::setZero(string username, string password ) //sets all values to zero

{

this->amount=0; //set to zero

this->balance=0; //set to zero

ofstream username\_write(username.c\_str(), ios::binary); //file write mode open to store final balance

username\_write.write(reinterpret\_cast<char \*>(this), sizeof(\*this)); //writes object to file

username\_write.close(); //file close

ofstream password\_write(password.c\_str(), ios::binary); //file write mode open to store transaction details to file

password\_write.write(reinterpret\_cast<char \*>(this), sizeof(\*this)); //writes object to file

password\_write.close(); //file close

}

void expense::getAmount(string username, string password) //gets input from user

{

int temp\_amount; //temporary amount storing variable

cout<<"Enter Expense Amount:"<<endl;

cin>>temp\_amount; //amount input

temp\_amount = (temp\_amount \* (-1)); //set debit amount to -ve so that when all transactions are displayed user could identify which one is income and which one is expense

ifstream username\_read(username.c\_str(), ios::binary); //file read mode open to store final balance

username\_read.read(reinterpret\_cast<char \*>(this), sizeof(\*this)); //reads object form file

balance = balance + temp\_amount; //adds amount to final balance

username\_read.close(); //file close

this->amount = temp\_amount; //amount set

cout<<endl<<"Enter Transaction Category(e.g. gift, holiday, shopping etc.):"<<endl;

cin>>this->category; //gets transaction type

cout<<endl<<"Enter Transaction Place:"<<endl;

cin>>this->place; //gets transaction location

this->d.getInput(); //gets date input

ofstream username\_write(username.c\_str(), ios::binary); //file wriite mode open tos tore final balance

username\_write.write(reinterpret\_cast<char \*>(this), sizeof(\*this)); //writes object to file

username\_write.close(); //file close

//-----------------------Above is work with username.txt file which only has 1 record for final balance------

//-----------------------Below is work with password.txt file which contains transactions list------

ofstream password\_write(password.c\_str(), ios::binary|ios::app); //file write mode open to store transaction details

password\_write.write(reinterpret\_cast<char \*>(this), sizeof(\*this)); //writes object to file

password\_write.close(); //file close

}

void expense::displayTransaction(string password) //displays all transactions

{

ifstream file\_read(password.c\_str(), ios::binary); //file read mode open to get transactions details

cout<<"================================================================================";

cout<<setw(20)<<"TRANSACTION AMOUNT";

cout<<setw(12)<<"CATEGORY";

cout<<setw(10)<<"PLACE";

cout<<setw(18)<<"FINAL BALANCE";

cout<<setw(12)<<"DATE"<<endl;

cout<<"================================================================================"<<endl;

for(int i=0;!file\_read.eof();i++) //read file untill EOF

{

file\_read.read(reinterpret\_cast<char \*>(this), sizeof(\*this)); //reads object form file

if(i==0) //skips garbage data

{

continue;

}

if(file\_read.eof()) //skips last iteration after EOF

{

break;

}

cout<<setw(20)<<amount<<setw(12)<<category<<setw(10)<<place<<setw(18)<<balance;

cout<<setw(9);

d.displayDate();

cout<<endl;

}

}

void expense:: operator + (string username) //diplays final balance

{

ifstream username\_read(username.c\_str(), ios::binary); //file read mode open to read final balance

username\_read.read(reinterpret\_cast<char \*>(this), sizeof(\*this)); //reads object from file

cout<<endl<<"Account Balance: "<<this->balance<<endl;

username\_read.close(); //file close

}