**Title:**

Budget Synthesizer

**Class:**

CSC – 17A

**Section:**

47975

**Due Date:**

December 9, 2012

**Introduction:**

**Title:** Budget Synthesizer

I made a program that calculates your net income, by taking in your salary and calculating your pay rate and hours taking into account your overtime, and double time. This program also adds other income, subtracts other expenses and it gives back your net income or budget.

**Total Lines:** 642

**Actual Coded Lines:** 398

**Empty Lines:** 80

**Number of variables:** 20

**Comment Lines:** 144

**Variable: Location:**

**short:**

**hours =** regular hours **struct Pay**

**othours =** overtime hours **Paycheck::getPaywOT(),**

**Paycheck::getPaywDT()**

**dthours =** double time hours **Paycheck::getPaywDT()**

**weeks =** number of weeks earning this specific pay **default\_program()**

**int:**

**choice menu()**

**const short:**

**NUM\_EXPENSES:** Number of expenses **default\_program()**

**STRING\_SIZE:** string length **Global**

**LETTER:** string length **Main()**

**COL:** cost entries **Global**

**SIZE:** name length **Global**

**float:**

**rate =** pay obtained after working one regular hour **struct Pay**

**regpay =** pay obtained from regular hours **Paycheck::getPaywDT()Paycheck::getPaywDT()**

**otpay=** pay obtained from overtime hours **Paycheck::getPaywDT()Paycheck::getPaywDT()**

**otrate=** pay earned every overtime hours **Paycheck::getPaywDT()Paycheck::getPaywDT()**

**dtpay=** pay obtained from double time hours

**Paycheck::getPaywDT(),**

**Paycheck::getPaywDT()**

**dtrate=** pay earned every double time hours

**Paycheck::getPaywDT(),**

**Paycheck::getPaywDT()**

**pay =** pay obtained after one week  **struct Pay**

**total =** total pay earned **struct Pay**

**temp=** the input other income / expenses **default\_program()**

**expenses=** the total of expenses **struct Pay**

**income=** total of extra income **struct Pay**

**netpay =** money left over after expenses have been subtracted **struct Pay**

**cost[NUM\_EXPENSES][ COL]:** Entries of cost of expenses **default\_program()**

**char:**

**p:** pointer to name array **default\_program()**

**name[LETTER]:** input name **default\_program()**

**Expenses[NUM\_EXPENSES][ STRING\_SIZE]:** Names of expenses **default\_program()**

**choice:** run program or exit **Main()**

**Function location**

**Passing a value default\_program()**

**Pass by reference default\_program()**

**Passing an array default\_program()**

**Passing a 2D array default\_program()**

**Void menu() Main()**

**Void default\_program menu()**

**Returning pointers default\_program()**

**Pointers location**

**p:** pointer to name array **default\_program()**

**Arrays location**

**name[Letters] Main()**

**Expenses[NUM\_EXPENSES][STRING\_SIZE]default\_program**

**cost[NUM\_EXPENSES][ COL] default\_program**

**Classes:**

**Loan Loan.h**

**Paycheck Paycheck.h**

**Pseudo Code:**

declare headers

const short STRING\_SIZE=16;

const short COL=2;

const short SIZE=24;

structure Pay

char name[SIZE];

int hours;

float rate,

expense,

income,

pay,

total;

\*enterName (parameters:char\*,short l);

menu();

default\_function();

totalpay(parameters:float &refPay);

net(parameters:float t,float e);

startingValues(parameters:char array[][STRING\_SIZE],float cost[][COL],short ne);

int main()

Declare constant int size

CALL enterName (name,LETTERS);

Declare and Initialize

Do

Enter choice

choice = tolower(choice);

switch(choice)

IF choice = y call menu()

IF choice = n display message

default = Display message

while choice = y

RETURN EXIT\_SUCCESS;

menu()

for(;;)

cout << "What would you like to do?";

cout << endl;

DISPLAY menu

ENTER choice

switch (choice)

IF choice = 1 call default\_function();

break;

default: break;

IF choice=2 return;

default\_function()

declare and initialize

declare the amount of customers to run this program for

for( i=0; while i<amount of customers; increment +1)

enter the customer’s name

enter hours worked

enter hourly rate

format output

IF 0>=hours <= 40

calculate pay

display pay

total = totalpay(pay);

display total

ELSEIF (hours <= 60)

calculate overtime hours

calculate overtime pay rate

calculate overtime pay

calculate regular pay

calculate pay

display pay

total = totalpay(pay);

display total

ELSEIF 60> hours <= 168

calculate double time hours

calculate regular hour

calculate overtime hours

calculate overtime pay rate

calculate double time pay rate

calculate regular pay rate

calculate overtime pay

calculate double time pay

calculate regular pay

calculate pay

display pay

Call totalpay(argument:pay)

RETURN total

display total

ELSE

Display message

ENTER income earned from SSI / Pension

WHILE SSI / Pension <0

ENTER income earned from SSI / Pension

ENDWHILE

Calculate other income

ENTER income earned from stocks

WHILE stocks<0

ENTER income earned from stocks

ENDWHILE

Calculate other income

Display other income

Calculate total income

Display total income

Declare constant array size

Declare and Initialize an array to hold names of expenses

Declare floating 2d array to hold the predicted and true cost of expenses

CALL startingValues(Expenses,cost,NUM\_EXPENSES)

DISPLAY message

DISPLAY table

FOR i=0 to NUM\_EXPENSES

DISPLAY the name of the expense

ENTER cost of the expense

CALCULATE total expenses

NEXT i

DISPLAY total expenses

CALL net( totalIncome, expenses )

RETURN netpay

DISPLAY this month’s net income

Reintialize netpay and totalIncome

DISPLAY message

DISPLAY table

FOR i=0 to NUM\_EXPENSES

Calculate next month’s budget for i expenses

DISPLAY the name of the expense and cost of the expense

NEXT i

CALL net(totalIncome,expenses)

RETURN netpay

DISPLAY next months predicted income;

LoanCalculator()

DECLARE Variables

ENTER Values

CALCULATES monthly payments

CALCULATES total pay

CALCULATES total Interest

DISPLAY monthly payments

DISPLAY total pay

DISPLAY total Interest

SavingInterestCal()

DECLARE Variables

ENTER Values

CALCULATES rate

CALCULATES New Balance

DISPLAY Old Balance

DISPLAY Years

DISPLAY New Balance

total Interest \*enterName (char\* array,short l)

ENTER name

startingValues( array[][STRING\_SIZE], cost[][COL], ne )

DISPLAY message

DISPLAY table

FOR i=0 to ne

DISPLAY the name of the expense

ENTER the predicted cost of the expense

NEXT i

totalpay(float &ref)

Declare and Initialize

ENTER how many weeks you earned this pay

FOR week = 1 to weeks

ADD ref to total

NEXT week

RETURN total

net(t, e)

Declare n

Calculate n

RETURN n