Brian Blalock

Project 3

Source code

```
#include <stdio.h>
#include <stdlib.h>


void add( float* A, float* B) ;
void subtract(float* A, float* B);
void multiply(float* A, float* B);
void power(float* A, float* B);
void factorial(float* A, float* B);

void getOperandA(float* A);
void getOperandB(float* B);
void getOperator(char* C);


int main(){

        printf("Brian Blalocks C Calculator\n"); //open prompt

        int option = -1;                                        // store menu option ( default -1 )

        float operandA = 0;                                     // store first operand( and result)

        float operandB = 0;                                             // store second operand
        char operator;                                  // store operator to choose what function to run

        char shouldExit = '\0';

        while (1) {
                do{
                printf("Select Option:\n1. Insert two operands (A and B) and an operator\n2. Use 
the previous result as operand A, insert operand (B) and an operator.\n");
                scanf( "%d", &option);
                } while( option != 1 && option != 2 );   // loop until valid option



                if( option == 1){                               // get operandA if option 1 is selected
                        getOperandA( &operandA);
        }
        getOperator(&operator);                     //get the operator
        getOperandB(&operandB);                     //get second operand


        switch(operator){                           //switch to determine function call

            case '+':
                add(&operandA, &operandB);                  // addition
                break;
            case '-':
                subtract(&operandA, &operandB);     //subtraction
                break;
            case 'x':
                multiply(&operandA, &operandB);     // multiplication
                break;

            case 'P':
                power(&operandA, &operandB);        // power
                break;
            case '!':
                // add A and B together
                factorial(&operandA, &operandB);                    //factorial
                break;

            default:

                break;

        }

        printf("Result = %f\n", operandA);      // print result , now stored in operandA



        printf( "Would you like to calculate something else? Y or N\n");
        scanf(" %c", &shouldExit);

        if(shouldExit == 'N'|| shouldExit == 'n'){
            break;
        }

    }
```

```
	printf("Goodbye");							// exit message

	return 0;
}

void getOperandA(float* A){						// gets first value

	printf("Insert operand A:\n");

	scanf("%f", A);

}
void getOperandB(float* B){						// gets second value

	printf("Insert operand B:\n");

	scanf("%f", B);

}
void getOperator(char* operator){					// gets the operator

	int needOperator = 1;				// stores if we have a valid operator or not
	do{
		printf( "Insert Operator(+,-,x,P,!):\n"); // prompt

		scanf(" %c", operator);				// gets char operator and store

		if( *operator == '+' || *operator == '-' || *operator == 'x' || *operator == 'P' ||
*operator == '!'){	//check if valid operator

			needOperator = 0;	// end the loop

		}
		else{	// if not valid

			printf("Operator not supported\n"); // display error, loop restarts
		}

	}while( needOperator);

}
```

```
void add(float* A, float* B){              //simple Addition

        *A = *A + *B;

}
void subtract(float* A, float* B){         // Simple subtraction

        *A = *A - *B;

}
void multiply( float* A, float* B){        // Simple multiplication

        *A = *A * *B;

}
void power(float* A, float* B){            // power function

        if( *A < 0 || *B < 0){                     // if negative display error prompt

                printf("Negative operands not supported for this operator.\n");
                *A = -1;
                return;

        }else if( *B == 0 ) {                      // if exponent is 0 , return 1
                *A = 1;
                return;
        }

        float initialVal = *A;                     // store initial value of A



        for(int i = 1; i < (int)*B ; i++){

                *A *= initialVal;                  // multiply A by the initial val of A, B times

        }

}

void factorial(float* A, float* B){                        // compute factorial

        if( *A < 0|| *B < 0){                      // if negative display error prompt
```

```
                printf("Negative operands not supported for this operator.\n");
                *A = -1;
                return;

        }



        int val = (int)*A + (int)*B;                                // initial value of A
        int result = val;                           // stores factorial
        for (int i = val-1; i >= 1; i--){     // loop down from value to 1

                result *= i;                                // do step of factorial;

        }

        *A = result;                                        // set A to result;

}
```

Print out from console:

```
$ ./project
Brian Blalocks C Calculator
Select Option:
1. Insert two operands (A and B) and an operator
2. Use the previous result as operand A, insert operand (B) and an operator.
1
Insert operand A:
2.35
Insert Operator(+,-,x,P,!):
+
Insert operand B:
3.65
Result = 6.000000
Would you like to calculate something else? Y or N
Y
Select Option:
1. Insert two operands (A and B) and an operator
2. Use the previous result as operand A, insert operand (B) and an operator.
2
Insert Operator(+,-,x,P,!):
-
```

```
Insert operand B:
3.567
Result = 2.433000
Would you like to calculate something else? Y or N
Y
Select Option:
1. Insert two operands (A and B) and an operator
2. Use the previous result as operand A, insert operand (B) and an operator.
2
Insert Operator(+,-,x,P,!):
x
Insert operand B:
2.3
Result = 5.595900
Would you like to calculate something else? Y or N
Y
Select Option:
1. Insert two operands (A and B) and an operator
2. Use the previous result as operand A, insert operand (B) and an operator.
1
Insert operand A:
2
Insert Operator(+,-,x,P,!):
P
Insert operand B:
2
Result = 4.000000
Would you like to calculate something else? Y or N
Y
Select Option:
1. Insert two operands (A and B) and an operator
2. Use the previous result as operand A, insert operand (B) and an operator.
2
Insert Operator(+,-,x,P,!):
!
Insert operand B:
0
Result = 24.000000
Would you like to calculate something else? Y or N
Y
Select Option:
1. Insert two operands (A and B) and an operator
2. Use the previous result as operand A, insert operand (B) and an operator.
```

```
1
Insert operand A:
3
Insert Operator(+,-,x,P,!):
[
Operator not supported
Insert Operator(+,-,x,P,!):
d
Operator not supported
Insert Operator(+,-,x,P,!):
+
Insert operand B:
4.5
Result = 7.500000
Would you like to calculate something else? Y or N
Y
Select Option:
1. Insert two operands (A and B) and an operator
2. Use the previous result as operand A, insert operand (B) and an operator.
1
Insert operand A:
-3
Insert Operator(+,-,x,P,!):
P
Insert operand B:
2
Negative operands not supported for this operator.
Result = -1.000000
Would you like to calculate something else? Y or N
Y
Select Option:
1. Insert two operands (A and B) and an operator
2. Use the previous result as operand A, insert operand (B) and an operator.
1
Insert operand A:
-3
Insert Operator(+,-,x,P,!):
!
Insert operand B:
-2
Negative operands not supported for this operator.
Result = -1.000000
Would you like to calculate something else? Y or N
```

```
Y
Select Option:
1. Insert two operands (A and B) and an operator
2. Use the previous result as operand A, insert operand (B) and an operator.
1
Insert operand A:
34.222
Insert Operator(+,-,x,P,!):
P
Insert operand B:
0
Result = 1.000000
Would you like to calculate something else? Y or N
N
Goodbye
```