

## 1 objective

This report is based on real life banking system which allows user to attain all the banking features like deposit amount, withdraw amount, transfer amount etc.

## 2 statisitcal information

total number of line in c code= 166 lines  
total time required=18 hours  
total functions=9 functions

### 3 DEFINE FUNCTIONS-

#### 3.0.1 Deposit function:

deposit function shows how much amount you deposit to your bank account.

total amount is the summation of deposited amount and balance.

#### 3.0.2 Withdraw function:

withdraw functions means the amount that has been debited from your account.

total amount is the subtraction of amount present and withdraw money.

#### 3.0.3 Transfer function:

Transfer functions shows the amount the money that you transfer from one account to another.

#### 3.0.4 Checkdetails function:

this function shows all the details like total amount, total deposited amount,

total withdrawal amount ,transferred amount etc.

#### 3.0.5 Last function:

Exit functions shows that you clearly performed all the activities and gives the summary for your performed task.

**3.0.6 Create an account Function:**

This functions takes the details of a customer like their name,mobile number to create a new account in a bank.

**3.0.7 Account number function:**

This functions takes an account number from the user.

**3.0.8 Select bank function:**

This function help us to choose a bank in which you want to open an account.

**3.0.9 List function:**

List function calls the function one by one.you can call whatever function you wanna use.

**4 SOURCE CODE-****4.1 IN C LANGUAGE:**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int list();
void deposit();
void transfer();
void withdraw();
void checkdetail();
void last();
```

```
void create_an_account ();
void acc_num ();
void select_bank ();
int totalamount=1000,amount,amo,totaldeposit=0,totalwithdraw=0,tr;
int totaltr=0;
int acc;
char name[50];
int main()
{
    //clrscr();
    //printf("ENTER YOUR NAME:\n");
    //gets(a);
    //printf("ENTER THE ACCOUNT NUMBER:");
    //scanf("%d",&acc);
    //getchar();
    printf("select bank:\n");
    select_bank();
while(1)
{
    //clrscr();
    switch(list())
    {
        case 0:
            create_an_account();
            break;
        case 1:
            acc_num();
            break;
        case 2:
            deposit();
            totaldeposit+=amount;
            break;
        case 3:
            withdraw();
            if(amo<=totalamount)
            totalwithdraw+=amo;
            break;
        case 4:
            transfer();
            if(tr<=totalamount)
```

```
        totaltr+=tr;
        break;
    case 5:
        checkdetail();
        break;
    case 6:
        // clrscr();
        last();
        getch();
        exit(0);
    default:
        printf("\n invalid choice:");
    } //end of switch
    getch();
} //end of while
return 0;
}

int list()
{
    int ch;
    printf("\n0. create an account:");
    printf("\n1. account number:");
    printf("\n2. deposit amount:");
    printf("\n3. withdraw amount:");
    printf("\n4. transfer amount:");
    printf("\n5. check detail:");
    printf("\n6. exit ");
    printf("\n enter your choice:");
    scanf("%d",&ch);
    return(ch);
}

void deposit()
{
    printf("enter the amount you want to deposit:");
    scanf("%d",&amount);
    totalamount+=amount;
}

void withdraw(){
    printf("enter the amount you want to withdraw:");
```

```
scanf("%d",&amo);
if(amo<=totalamount)
totalamount-=amo;
else
printf("\n less amount withdraw is not possible");
}
void transfer()
{
printf("\n enter the amount you want to transfer:");
scanf("%d",&tr);
if(tr<=totalamount)
totalamount-=tr;
else
printf("\n less amount transfer is not possible");
}
void checkdetail()
{
printf("total amount:%d",totalamount);
printf("\n total deposited amount:%d",totaldeposit);
printf("\n total withdraw amount:%d",totalwithdraw);
printf("\n total transferred amount:%d",totaltr);
}
void last()
{
printf("\n***\n");
printf("\n your name=%s",name);
printf("\n account number=%d",acc);
printf("\ntotal amount:=%d",totalamount);
printf("\n total deposited amount:%d",totaldeposit);
printf("\n total withdraw amount:%d",totalwithdraw);
printf("\n total transferred amount:%d",totaltr);
printf("\n*THANKS*\n");
}
void create_an_account()
{
char name[50];
long long int mobile_num;
printf("enter your name:");
getchar();
gets(name);
```

```
        puts(name);
        printf("enter your phone number:");
        scanf("%lld",&mobile_num);
        printf("%lld\n",mobile_num);
        getchar();
    }
    void acc_num()
    {
        long long int acc;
        printf("ENTER THE ACCOUNT NUMBER:");
        scanf("%lld",&acc);
        printf("your account no. is:%lld\n",acc);
    }
    void select_bank(){
        printf("press 1 for HDFC bank\n");
        printf("press 2 for punjab national bank\n");
        printf("press 3 for axis bank\n");
        int bank;
        scanf("%d",&bank);
        switch(bank){
            case 1:
                printf("Welcome to HDFC bank\n");
                break;
            case 2:
                printf("Welcome to punjab national bank\n");
                break;
            case 3:
                printf("Welcome to axis bank\n");
                break;
            default:
                printf("\nno other bank is available");
        }
    }
```

## 5 Output in C language-

```
C:\Users\my\Documents\Untitled2.exe
select bank:
press 1 for HDFC bank
press 2 for punjab national bank
press 3 for axis bank
1
Welcome to HDFC bank
0. create an account:
1. account number:
2. deposit amount:
3. withdraw amount:
4. transfer amount:
5. check detail:
6. exit
enter your choice:0
enter your name:khushi jain
khushi jain
enter your phone number:9827113198
9827113198
0. create an account:
1. account number:
2. deposit amount:
3. withdraw amount:
4. transfer amount:
5. check detail:
6. exit
enter your choice:1
ENTER THE ACCOUNT NUMBER:098
your account no. is:98
0. create an account:
1. account number:
2. deposit amount:
3. withdraw amount:
4. transfer amount:
5. check detail:
6. exit
enter your choice:2
enter the amount you want to deposit:4000
0. create an account:
1. account number:
2. deposit amount:
3. withdraw amount:
4. transfer amount:
5. check detail:
6. exit
enter your choice:3
enter the amount you want to withdraw:56
0. create an account:
1. account number:
2. deposit amount:
3. withdraw amount:
4. transfer amount:
5. check detail:
6. exit
enter your choice:4
enter the amount you want to transfer:1900
0. create an account:
1. account number:
2. deposit amount:
3. withdraw amount:
4. transfer amount:
5. check detail:
6. exit
enter your choice:5
total amount:3844
total deposited amount:4000
total withdraw amount:56
total transferred amount:1900
0. create an account:
1. account number:
2. deposit amount:
3. withdraw amount:
4. transfer amount:
5. check detail:
6. exit
enter your choice:6
*****
your name:
account number:0
total amount:-3844
total deposited amount:4000
total withdraw amount:56
total transferred amount:1900
THANKS**
```

## 6 Profiling and Debugging



```

1 Flat profile:
2
3 Each sample counts as 0.01 seconds.
4 no time accumulated
5
6 % cumulative self self total
7 time seconds seconds calls ts/call ts/call name
8 0.00 0.00 0.00 7 0.00 0.00 list
9 0.00 0.00 0.00 1 0.00 0.00 acc_num
10 0.00 0.00 0.00 1 0.00 0.00 checkdetail
11 0.00 0.00 0.00 1 0.00 0.00 create_an_account
12 0.00 0.00 0.00 1 0.00 0.00 deposit
13 0.00 0.00 0.00 1 0.00 0.00 last
14 0.00 0.00 0.00 1 0.00 0.00 select_bank
15 0.00 0.00 0.00 1 0.00 0.00 transfer
16 0.00 0.00 0.00 1 0.00 0.00 withdraw
17
18 % the percentage of the total running time of the
19 time program used by this function.
20
21 cumulative a running sum of the number of seconds accounted
22 seconds for by this function and those listed above it.
23
24 self the number of seconds accounted for by this
25 seconds function alone. This is the major sort for this
26 listing.
27
28 calls the number of times this function was invoked, if
29 this function is profiled, else blank.
30
31 self the average number of milliseconds spent in this
32 ms/call function per call, if this function is profiled,
33 else blank.
34
35 total the average number of milliseconds spent in this
36 ms/call function and its descendants per call, if this
37 function is profiled, else blank.
38
39 name the name of the function. This is the minor sort
40 for this listing. The index shows the location of
41 the function in the gprof listing. If the index is
42 in parenthesis it shows where it would appear in
43 the gprof listing if it were to be printed.
44
45 Copyright (C) 2012-2017 Free Software Foundation, Inc.
46
47 Copying and distribution of this file, with or without modification,
48 are permitted in any medium without royalty provided the copyright
49 notice and this notice are preserved.
50
51 Call graph (explanation follows)
52
53
54 granularity: each sample hit covers 4 byte(s) no time propagated
55
56 index % time self children called name
57 0.00 0.00 0.00 7/7 main [00]
58 [2] 0.0 0.00 0.00 7 list [2]
59 -----
60 0.00 0.00 0.00 1/1 main [00]
61 [3] 0.0 0.00 0.00 1 acc_num [3]

```

```

62 -----
63 [4] 0.0 0.00 0.00 1/1 main [00]
64 [4] 0.0 0.00 0.00 1 checkdetail [4]
65 -----
66 [5] 0.0 0.00 0.00 1/1 main [00]
67 [5] 0.0 0.00 0.00 1 create_an_account [5]
68 -----
69 [6] 0.0 0.00 0.00 1/1 main [00]
70 [6] 0.0 0.00 0.00 1 deposit [6]
71 -----
72 [7] 0.0 0.00 0.00 1/1 main [00]
73 [7] 0.0 0.00 0.00 1 last [7]
74 -----
75 [8] 0.0 0.00 0.00 1/1 main [00]
76 [8] 0.0 0.00 0.00 1 select_bank [8]
77 -----
78 [9] 0.0 0.00 0.00 1/1 main [00]
79 [9] 0.0 0.00 0.00 1 transfer [9]
80 -----
81 [10] 0.0 0.00 0.00 1/1 main [00]
82 [10] 0.0 0.00 0.00 1 withdraw [10]
83 -----
84
85 This table describes the call tree of the program, and was sorted by
86 the total amount of time spent in each function and its children.
87
88 Each entry in this table consists of several lines. The line with the
89 index number at the left hand margin lists the current function.
90 The lines above it list the functions that called this function,
91 and the lines below it list the functions this one called.
92 This line lists:
93 index a unique number given to each element of the table.

```

The image displays three sequential screenshots of a Visual Studio Code editor window, showing a file named 'jain.txt'. The editor is open to a file with a recursive function profiler. The file contains comments explaining the profiler's output and a table of function calls.

**First Screenshot (Lines 93-117):**

```

93  Index A unique number given to each element of the table.
94  Index numbers are sorted numerically.
95  The index number is printed next to every function name so
96  it is easier to look up where the function is in the table.
97
98  % time This is the percentage of the 'total' time that was spent
99  in this function and its children. Note that due to
100 different viewpoints, functions excluded by options, etc,
101 these numbers will NOT add up to 100%.
102
103 self This is the total amount of time spent in this function.
104
105 children This is the total amount of time propagated into this
106 function by its children.
107
108 called This is the number of times the function was called.
109 If the function called itself recursively, the number
110 only includes non-recursive calls, and is followed by
111 a '+' and the number of recursive calls.
112
113 name The name of the current function. The index number is
114 printed after it. If the function is a member of a
115 cycle, the cycle number is printed between the
116 function's name and the index number.
117

```

**Second Screenshot (Lines 125-156):**

```

125 the function's children into this parent.
126
127 called This is the number of times this parent called the
128 function '/' the total number of times the function
129 was called. Recursive calls to the function are not
130 included in the number after the '/'.
131
132 name This is the name of the parent. The parent's index
133 number is printed after it. If the parent is a
134 member of a cycle, the cycle number is printed between
135 the name and the index number.
136
137 If the parents of the function cannot be determined, the word
138 '<spontaneous>' is printed in the 'name' field, and all the other
139 fields are blank.
140
141 For the function's children, the fields have the following meanings:
142
143 self This is the amount of time that was propagated directly
144 from the child into the function.
145
146 children This is the amount of time that was propagated from the
147 child's children to the function.
148
149 called This is the number of times the function called
150 this child '/' the total number of times the child
151 was called. Recursive calls by the child are not
152 listed in the number after the '/'.
153
154 name This is the name of the child. The child's index
155 number is printed after it. If the child is a
156 member of a cycle, the cycle number is printed

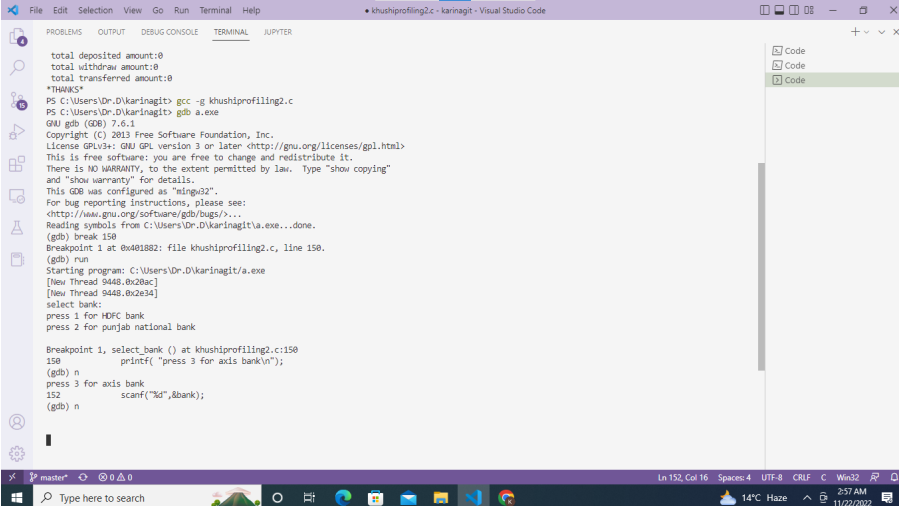
```

**Third Screenshot (Lines 159-178):**

```

159 was called. Recursive calls by the child are not
160 listed in the number after the '/'.
161
162 name This is the name of the child. The child's index
163 number is printed after it. If the child is a
164 member of a cycle, the cycle number is printed
165 between the name and the index number.
166
167 If there are any cycles (circles) in the call graph, there is an
168 entry for the cycle-as-a-whole. This entry shows who called the
169 cycle (as parents) and the members of the cycle (as children.)
170 The '+' recursive calls entry shows the number of function calls that
171 were internal to the cycle, and the calls entry for each member shows,
172 for that member, how many times it was called from other members of
173 the cycle.
174
175 Copyright (C) 2012-2017 Free Software Foundation, Inc.
176
177 Copying and distribution of this file, with or without modification,
178 are permitted in any medium without royalty provided the copyright
179 notice and this notice are preserved.
180
181 Index by function name
182
183 [3] acc_num [6] deposit [8] select_bank
184 [4] checkdetail [7] last [9] transfer
185 [5] create_an_account [2] list [10] withdraw
186

```



```
File Edit Selection View Go Run Terminal Help
• khushiprofiling2.c - karinagt - Visual Studio Code

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

total deposited amount:0
total withdraw amount:0
total transferred amount:0
*THANKS*
PS C:\Users\Dr.D\karinagt> gcc -g khushiprofiling2.c
PS C:\Users\Dr.D\karinagt> gbd a.exe
GNU gdb (GDB) 7.6.1
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv2+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "mingw32".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from C:\Users\Dr.D\karinagt\k.exe...done.
(gdb) break 150
Breakpoint 1 at 0x401882: file khushiprofiling2.c, line 150.
(gdb) run
Starting program: C:\Users\Dr.D\karinagt\k.exe
[New Thread 9448.0x20ac]
[New Thread 9448.0x2e34]
select bank:
press 1 for HDFC bank
press 2 for punjab national bank

Breakpoint 1, select_bank () at khushiprofiling2.c:150
150     printf("press 3 for axis bank\n");
(gdb) n
press 3 for axis bank
152     scanf("%d",&bank);
(gdb) n
```