3.0. To Implement Hashing Technique

# include <iostream> # include <conio.h> using namespace std;

class hash

{

private: public:

long int arr[20];

void direct(void); void sub(void);

void modulodiv(void); void modulodivc(void); void display(void);

void digitextraction(void); void digitextractionc(void); void midsqr(void);

void midsqrc(void);

hash()

{

int i; for(i=0;i<20;i++)

{

};

void hash::direct()

{

arr[i] = 0;

}

}

int address,key;

cout<<"Enter the value of key : "; cin>>key; if((key>=0)&&(key<20))

{

}

else

{ address = key; arr[address] = key;

}

}

cout<<"Invalid Input";

void hash::sub()

{

int address,key;

cout<<"Enter the value of key: "; cin>>key; if((key>=0)&&(key<20))

{

}

else

{

}

address = (100-key)%20; arr[address] = key;

cout<<"Invalid Input";

}

void hash::modulodiv()

{

int address,key;

cout<<"Enter the value of key: "; cin>>key; if((key>=0)&&(key<100))

{

}

else

{

}

}

address = key%20; arr[address] = key;

cout<<"Invalid Input";

void hash::modulodivc()

{

int address,key;

cout<<"Enter the value of key"; cin>>key; if((key>=0)&&(key<100))

{

address = key%20; if(arr[address] == 0)

{

}

else

{

}

else

{

}

arr[address]=key;

address = address+1; arr[address]=key;

cout<<"\nInvalid Input";

}

}

void hash::digitextraction()

{

int address,key;

cout<<"Enter the value of key : "; cin>>key; if((key>=10000)&&(key<99999))

{

}

else

{

}

}

address = key%100; address = key%20; arr[address] = key;

cout<<"Invalid Input";

void hash::digitextractionc()

{

int address,key;

cout<<"Enter the value of key"; cin>>key; if((key>=10000)&&(key<99999))

{

address = key%20; if(arr[address] == 0)

{

}

else

{

}

else

{

}

arr[address]=key;

address = address+1; arr[address]=key;

cout<<"\nInvalid Input";

}

}

void hash::midsqr()

{

long int key;

int temp,square,address; cout<<"Enter the value of key :"; cin>>key; if((key>=0)&&(key<10000))

{

}

else

{

}

}

temp = key/100; square = temp \* temp;

address = ((square/10)%10); arr[address] = key;

cout<<"Invalid Output";

void hash::midsqrc()

{

long int key;

int temp,square,address; cout<<"Enter the value of key :"; cin>>key; if((key>=0)&&(key<99999))

{

temp = key/100; square = temp \* temp;

address = ((square/10)%10); if(arr[address] == 0)

{

}

else

{

}

arr[address] = key;

address = address+1; arr[address] = key;

}

else

{

}

}

cout<<"Invalid Output";

void hash::display()

{

int i; for(i=0;i<20;i++)

{

}

int main()

{

}

int opt; hash h;

cout<<"\nkey at "<<i<<" : "<<arr[i];

char ch = 'y'; while(ch == 'y')

{

cout<<"1.Direct:"; cout<<"\n2.Subtraction:"; cout<<"\n3.Modulo Division"; cout<<"\n4.Modulo Division Collission"; cout<<"\n5.Digit Extraction"; cout<<"\n6.Digit Extraction Collission"; cout<<"\n7.Mid Square"; cout<<"\n8.Mid Square Collission"; cout<<"\n9.Display";

cout<<"\n\nEnter an option:"; cin>>opt;

switch(opt)

{

case 1:

case 2:

case 3:

case 4:

case 5:

case 6:

h.direct();

cout<<"\nDo you want to continue\n\n"; break;

h.sub();

cout<<"\nDo you want to continue\n\n"; break;

h.modulodiv();

cout<<"\nDo you want to continue\n\n"; break;

h.modulodivc();

cout<<"\nDo you want to continue\n\n"; break;

h.digitextraction();

cout<<"\n\nDo you want to continue\n\n"; break;

h.digitextractionc();

case 7:

case 8:

case 9:

}

cout<<"\n\nDo you want to continue\n\n"; break;

h.midsqr();

cout<<"\n\nDo you want to continue\n\n"; break;

h.midsqrc();

cout<<"\n\nDo you want to continue\n\n"; break;

h.display();

cout<<"\n\nDo you want to continue\n\n"; break;

ch=getch();

}

}