

Projects

APPENDIX

A



A.1 MINOR PROJECT: CREDIT CalC

Learning Objectives

After going through this project work, you will be able to

- learn how real-world applications are developed using C++ programming, and
- develop menu-driven applications.

A.1.1 Credit CalC

Credit CalC is a command-driven banking application that helps determine the credit worthiness of a loan applicant. The application takes into account various factors that could affect the loan repayment capability of the applicant in future. Some of key factors taken into consideration are:

- Applicant's age
- Monthly salary
- Current liabilities, such as existing loan EMLs
- Bank transaction details
- Accommodation status, i.e., owned house or rented
- Spouse's employment status
- Whether the parents are dependent or not
- Company tier

After analyzing the above factors, the **Credit CalC** application calculates a credit score and rates the applicant as high risk, average risk or low risk customer. Accordingly, a decision is taken to grant or deny credit or loan to the applicant.

Table A.1 lists the key coding elements of the Credit CalC application:

Table A.1. Key coding elements

Element	Description
applicant	This class is used to store and process loan applicant's details.
get_data()	This applicant class function is used to receive and store loan applicant's details.

(Contd.)

Table A.1 (Contd.)

calc_liability()	This applicant class function is used to assess the loan applicant's current liabilities and calculate a credit score.
display_score()	This applicant class function is used to display the credit score and rate the applicant as high risk, average risk or low risk customer.

Table A.2 depicts the logic used by the **Credit CalC** application for calculating the credit score:

Table A.2. Application logic

Score	Logic
Initial stage	<ul style="list-style-type: none"> At the start of the application, the score is initialized to zero
Applicant's age	<ul style="list-style-type: none"> If the age of the applicant lies between 22 and 30, the score is incremented by 2 If the age lies between 30 and 35, the score is incremented by 1 If the age is above 35, the score is decremented by 1
Applicant's liabilities	<ul style="list-style-type: none"> If the sum of home loan and personal loan EMIs is less than one-fourth of the monthly salary, then the score is incremented by 5 If the sum is above 1/4th of the monthly salary but less than its 1/3rd, then the score is incremented by 3 If the sum is above 1/3rd of the monthly salary but less than its half, then the score is incremented by 1 If the sum is greater than half of the monthly salary, then the score is decremented by 1
Number of cheque bounce	<ul style="list-style-type: none"> If the number of cheque bounces in the last six months is greater than or equal to 2, then the score is decremented by 2 If the number of cheque bounces is equal to 1, then the score is decremented by 1 If there are no cheque bounces in the last six months, then the score is incremented by 1
Home loan versus personal loan EMIs	<ul style="list-style-type: none"> If the total of home loan EMIs is greater than or equal to the total of personal loan EMIs, then the score is incremented by 1; else the score is decremented by 1
Own or rented house?	<ul style="list-style-type: none"> If the applicant lives in owned house then the score is incremented by 1; else if the applicant lives in a rented accommodation then the score is decremented by 1
Spouse working?	<ul style="list-style-type: none"> If the applicant's spouse is working then the score is incremented by 1; else the score is decremented by 1
Dependent parents?	<ul style="list-style-type: none"> If the applicant has dependent parents then the score is decremented by 1; else the score is incremented by 1
Company tier	<ul style="list-style-type: none"> If the company with which the applicant is employed is a tier 1 company then the score is incremented by 3 If it is a tier 2 company then the score is incremented by 2; else, the score is incremented by 1

A.1.2 Application Code

The code for **Credit Calc** application is given below. It contains comments at appropriate places to help understand the coding elements better.

CREDIT CALC APPLICATION

```

/*Header Files*/
#include<iostream>
#include<conio.h>
#include<string>

#include<stdlib.h>
#include "windows.h"
using namespace std;

/*Function definition for setting cursor position in the console
window*/
void gotoxy(int x, int y)
{
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}

/*Class definition*/
class applicant
{
    /*Defining data members for storing applicant details*/
    string name;
    int age;
    string address;
    int m_sal;
    int h_loan;
    int p_loan;
    int no_chq_bounce;
    char own_house;
    char spouse_working;
    char dependent_parents;
    int company_tier;

public:
    /*Class funtion declarations*/
    void get_data();
    int calc_liability();
    void display_score();
};

```

(Contd.)

```

/*Function for receiving and storing loan applicant's details*/
void applicant::get_data()
{
    system("cls");
    cout<<"Name of applicant: ";
    getline(cin,name);
    cout<<"Age: ";
    cin>>age;

    cin.clear ();
    cin.ignore (1000, '\n');

    cout<<"Address: ";
    getline(cin,address);

    cin.clear ();
    cin.ignore (1000, '\n');

    cout<<"Monthly Salary: ";
    cin>>m_sal;
    cout<<"Total Home Loan EMI: ";
    cin>>h_loan;
    cout<<"Total Personal Loan EMI: ";
    cin>>p_loan;
    cout<<"Number of cheque bounces in last six months: ";
    cin>>no_chq_bounce;
    cout<<"Own house? (y or n): ";
    cin>>own_house;
    cout<<"Spouse working? (y or n): ";
    cin>>spouse_working;
    cout<<"Dependent parents? (y or n): ";
    cin>>dependent_parents;
    cout<<"Company Tier? (1, 2 or 3): ";
    cin>>company_tier;
} //End of get_data()

/*Function for calculating credit score*/
int applicant:: calc_liability()
{
    int score=0; /*Initializing the score variable for storing credit
score*/
    int liability;

    if(age>22&&age<=30)
        score=score+2;
    else
        if(age>30&&age<=35)
            score=score+1;
        else
            score=score-1;
}

```

(Contd.)

```

liability=h_loan+p_loan;
if(liability <= m_sal/4)
    score=score+5;
if(liability > m_sal/4 && liability <= m_sal/3)
    score=score+3;
if(liability > m_sal/3 && liability <= m_sal/2)
    score=score+1;
if(liability > m_sal/2)
    score=score-1;
if(no_chq_bounce>1)
    score=score-2;
if(no_chq_bounce==1)
    score=score-1;
if(no_chq_bounce==0)
    score=score+1;
if(p_loan > h_loan)
    score=score-1;
else
    score=score+1;
if(own_house=='y')
    score=score+1;
if(spouse_working=='y')
    score=score+1;
if(dependent_parents=='y')
    ;
else
    score=score+1;
if(company_tier==1)
    score=score+3;
if(company_tier==2)
    score=score+2;
else
    score=score+1;
return(score);
} //End of calc_liability()

/*Function for displaying credit score and applicant rating*/
void applicant:: display_score()
{
    int sc=calc_liability(); /*Calling calc_liability() function to
calculate applicant's credit score*/
    system("cls");

    /*Displaying the results*/
    if(sc>9)
        cout<<"The applicant "<<name<<" is at low risk.\nCredit score =
"<<sc<<"\nCredit can be given.";
    else
    {
        if(sc>5&&sc<=9)

```

(Contd.)

```

        cout<<"The applicant "<<name<<" is at average risk.\nCredit score
= "<<sc<<"\nCredit can be given with due precaution.";
        else
            cout<<"The applicant "<<name<<" is at high risk.\nCredit score =
"<<sc<<"\nCredit can not be given.";
    }
    getch();
} //End of display_score()

/*Main Function*/
void main()
{
    char flag='f';
    applicant a;
    char ch;
    while(1)
    {
        system("cls");
        gotoxy(31,8);
        cout<<"Credit Calc";
        gotoxy(31,9);
        cout<<"_____";
        gotoxy(22,12);
        cout<<"1 - Enter loan applicant's details";
        gotoxy(22,13);
        cout<<"2 - Display credit score";
        gotoxy(22,14);
        cout<<"3 - Exit";
        gotoxy(22,16);
        cout<<"Select an option by typing the numeric code: ";
        ch=getch();
        switch(ch)
        {
            case('1'):
            {
                a.get_data();
                flag='t';
                break;
            }

            case('2'):
            {
                if(flag=='f')
                {
                    gotoxy(22,18);
                    cout<<"Loan applicant's details not yet entered! Press any key
to continue..";

```

(Contd.)

```

        getch();
    }
    else
        a.display_score();
    break;
}
    case('3'):
    {
        exit(1);
        break;
    }

default:
{
    gotoxy(22,18);
    cout<<"Invalid Choice! Press any key to continue..";
    getch();
}
} //End of switch-case block
} //End of while block
} //End of main()

```

A.1.3 Application Output

The following is a series of screenshots depicting how the **Credit CalC** application functions.

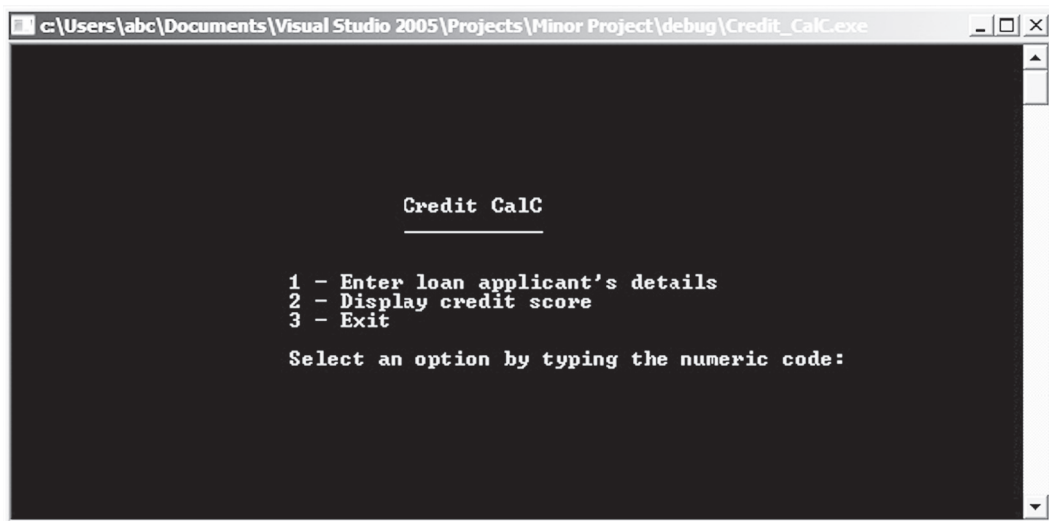
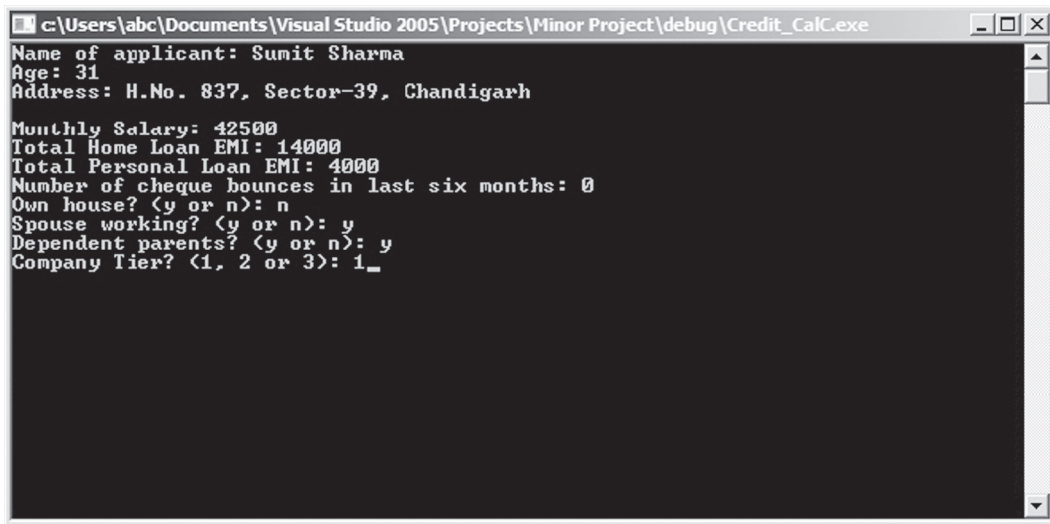


Fig. A.1 Main screen

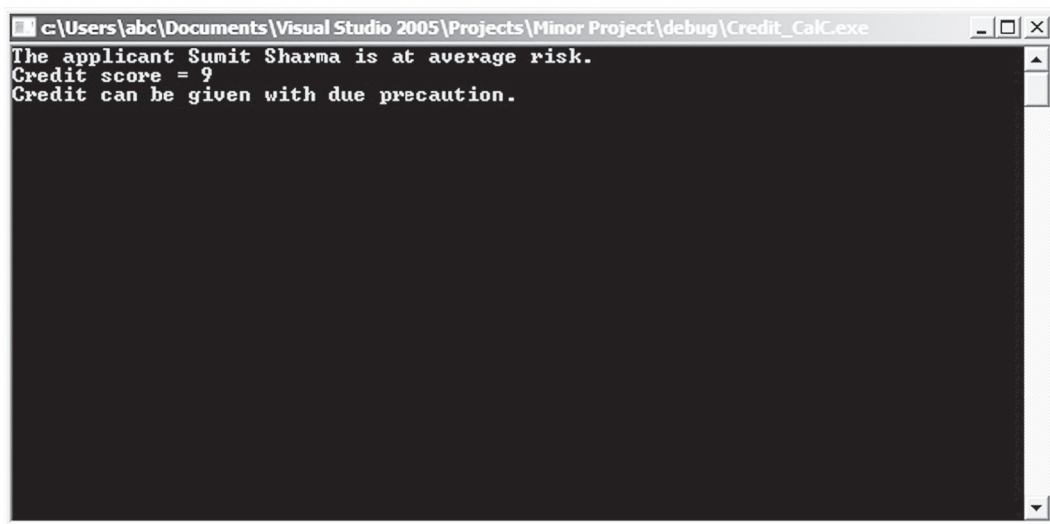


```

c:\Users\abc\Documents\Visual Studio 2005\Projects\Minor Project\debug\Credit_CalC.exe
Name of applicant: Sumit Sharma
Age: 31
Address: H.No. 837, Sector-39, Chandigarh
Monthly Salary: 42500
Total Home Loan EMI: 14000
Total Personal Loan EMI: 4000
Number of cheque bounces in last six months: 0
Own house? (y or n): n
Spouse working? (y or n): y
Dependent parents? (y or n): y
Company Tier? (1, 2 or 3): 1_

```

Fig. A.2 Entering applicant's details



```

c:\Users\abc\Documents\Visual Studio 2005\Projects\Minor Project\debug\Credit_CalC.exe
The applicant Sumit Sharma is at average risk.
Credit score = 9
Credit can be given with due precaution.

```

Fig. A.3 Displaying credit score



A.2 MAJOR PROJECT: PR2PO

Learning Objectives

After going through this project work, you will be able to

- learn how real-world applications are developed using C++ programming,
- develop menu-driven applications, and
- learn how file handling operations are performed in C++.

Background

It is a common practice in manufacturing companies to create purchase requisitions for the goods or items required. A **purchase requisition** typically contains the item code of the good required, its quantity and the date by when it is required. Based on the submitted purchase requisition, the Buyer or the Purchasing Officer then generates a purchase order. A **purchase order** or PO is a legal document used by organizations for placing orders with their suppliers for procurement of goods.

A.2.1 PR2PO

PR2PO is a command-driven procurement application that helps the users to

- create a purchase requisition
- view requisition details
- convert purchase requisition into a purchase order

Table A.3 lists the procurement-related assumptions for the **PR2PO** application. It details the various item codes, price and supplier details.

Table A.3. *PR2PO data set*

Item Code	Price	Supplier
1	865	SKS Inc
2	2.25	SKS Inc
3	44	SKS Inc
4	10	SKS Inc
5	20	SKS Inc
6	7710	SKS Inc
7	33.33	SKS Inc
8	48.97	SKS Inc
9	8	SKS Inc
10	0.5	SKS Inc
11	0.01	KC Spare Parts Ltd.
12	88	KC Spare Parts Ltd.
13	999	KC Spare Parts Ltd.
14	25000	KC Spare Parts Ltd.
15	87	KC Spare Parts Ltd.
16	4	KC Spare Parts Ltd.
17	22	KC Spare Parts Ltd.
18	200	KC Spare Parts Ltd.

(Contd.)

Table A.3. (Contd.)

19	20	KC Spare Parts Ltd.
20	150	KC Spare Parts Ltd.
>20	If the item code is greater than 20 then the price field of the purchase order is kept empty so that it can be manually filled by the Buyer.	If the item code is greater than 20 then the purchase order keeps the Supplier field as 'Other' and adds a blank space for the Buyer to enter the supplier name manually.

In addition to the item code, price and supplier details, the **PR2PO** application also adds standard terms and conditions to the purchase order. The text for these terms and conditions is taken from the terms.txt file, which is stored in the application root folder. Here are the contents of the terms.txt file:

```
//Contents of terms.txt file
```

```
Standard Terms and Conditions
```

1. PO should be acknowledged within 7 days of date of issue.
2. PO will be null and void if goods are not delivered by delivery date.
3. Defective goods will be returned back to the Supplier.
4. Invoice can only be raised after 20 days of date of goods receipt.
5. Damages during transit (if any) will be borne by the Supplier.

Table A.4 lists the key coding elements of the **PR2PO** application.

Table A.4. Key coding elements

Element	Description
requisition	This class is used to store the purchase requisition details that are later used for generating the purchase order.
get_data()	This requisition class function is used to receive and store requisition details. Every time a new requisition is created, the get_data() function assigns it a unique identifier (requisition number).
display()	This requisition class function is used to display the purchase requisition.
po	This class is used to represent a purchase order. Every time a new purchase order is created, the class constructor assigns it a unique identifier (purchase order number).
generate()	This friend function of requisition and po classes is used to generate a purchase order from a purchase requisition. The purchase order is stored in the <i>Current_PO.txt</i> file in the application root folder.

A.2.2 Application Code

The code for **PR2PO** application is given below. It contains comments at appropriate places to help understand the coding elements better.

PR2PO APPLICATION

```

#include<iostream>
#include<conio.h>
#include<string>
#include<stdlib.h>
#include<fstream>
#include<ctime>
#include "windows.h"

using namespace std;

/*Function definition for setting cursor position in the console
window*/
void gotoxy(int x, int y)
{
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}

//Price list of Items
//Each array index value is used as an identifier for item codes
float price_list[20]={865,2.25,44,10,20,7710,33.33,48.97,8,0.5,0.01,
88,999,25000,87,4,22,200,20,150};

class po; //Forward declaration of po class (needed while using
friend functions)

/*Class definition*/
class requisition
{
    /*Defining data members for storing requisition details*/
    static int req_no;
    int item_code;
    int qty;
    string delivery_date;

public:
    /*Class funtion declarations*/
    void get_data();
    void display();
    friend void generate(requisition, po); //friend function
    declaration
};

/*Function for receiving requisition details*/

```

(Contd.)

```

void requisition:: get_data()
{
    system("cls");
    req_no++;
    cout<<"Product Item Code: ";
    cin>>item_code;
    cout<<"Quantity: ";
    cin>>qty;
    cout<<"Delivery Date: ";
    cin.clear ();
    cin.ignore (1000, '\n');
    getline(cin,delivery_date);
}

//Defining static data member
int requisition:: req_no;

//Function for displaying requisition details
void requisition:: display()
{
    system("cls");
    cout<<"Requisition Number: "<<req_no;
    cout<<"\nProduct Item Code: "<<item_code;
    cout<<"\nQuantity: "<<qty;
    cout<<"\nDelivery Date: "<<delivery_date;
    getch();
}

class po
{
    static int po_no;
public:
    //Constructor is used for incrementing the static variable po_no
    po ()
    {
        po_no++;
    }

    friend void generate(requisition, po); //friend function
    declaration
};

//Defining static data member
int po:: po_no;

//Defining friend function generate()
void generate(requisition r, po p)
{
    ofstream fout;
    //Retreiving system date and time

```

(Contd.)

(Contd.)

```
char flag='f';
char ch;
//Displaying a menu to accept user commands
while(1)
{
    system("cls");
    gotoxy(35,8);
    cout<<" PR2PO";
    gotoxy(35,9);
    cout<<"_____";
    gotoxy(22,12);
    cout<<"1 - Create New Requisition";
    gotoxy(22,13);
    cout<<"2 - Display Requisition";
    gotoxy(22,14);
    cout<<"3 - Generate PO";
    gotoxy(22,15);
    cout<<"4 - Exit";
    gotoxy(22,18);
    cout<<"Select an option by typing the numeric code: ";
    ch=getch();
    switch(ch)
    {
        case('1'):
        {
            r.get_data();
            flag='t';
            break;
        }

        case('2'):
        {
            if(flag=='f')
            {
                gotoxy(22,20);
                cout<<"Requisition details not yet entered! Press any key to
                    continue..";
                getch();
            }
            else
            {
                r.display();
                break;
            }
        }

        case('3'):
        {
            po p;
            generate(r,p);
        }
    }
}
```

(Contd.)

```

gotoxy(22,20);
cout<<"PR has been converted into a PO. You can view the
Current_PO.txt file to view the generated PO and take its print
out";
getch();
break;
}

case('4'):
{
    exit(1);
    break;
}
default:
{
    gotoxy(22,20);
    cout<<"Invalid Choice! Press any key to continue..";
    getch();
}
} //End of switch-case block
} //End of while block
} //End of main()

```

A.2.3 Application Output

Following figures are a series of screenshots depicting how the **PR2PO** application functions.



Fig. A.4 Main screen

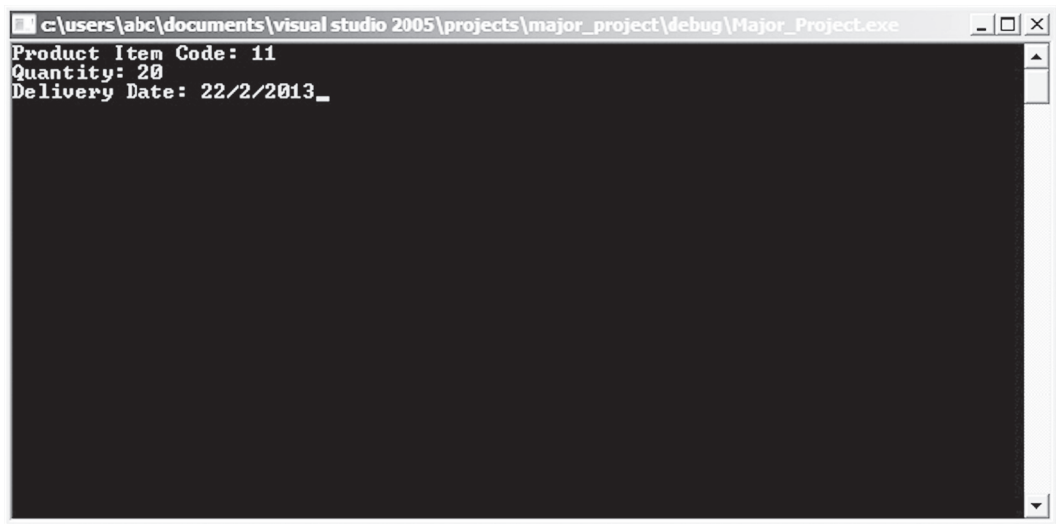


Fig. A.5 *Entering requisition details*

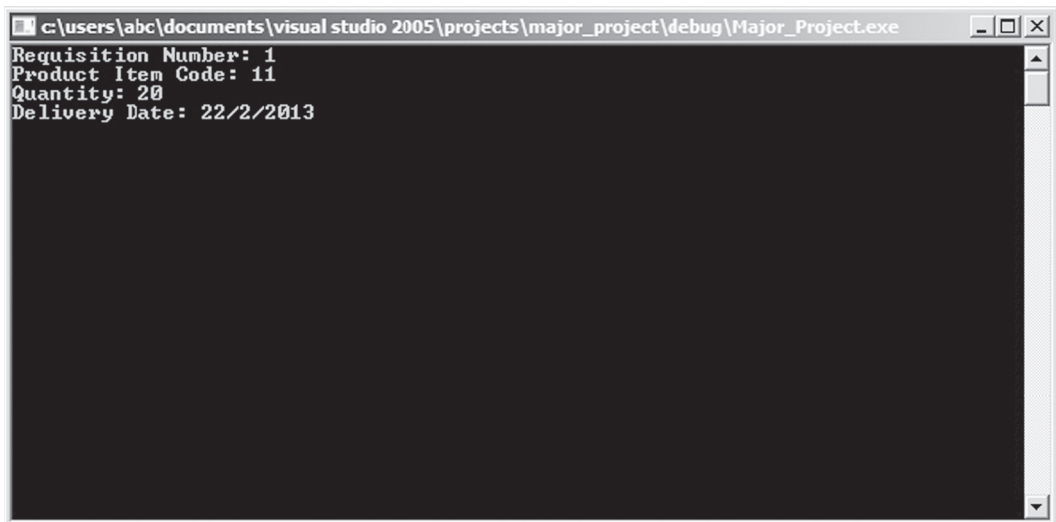


Fig. A.6 *Viewing requisition details*

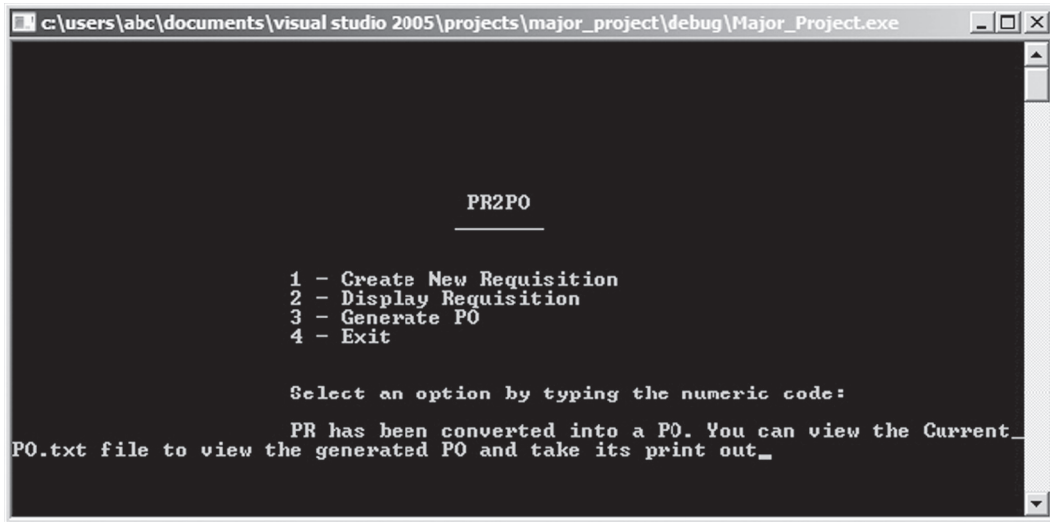


Fig. A.7 Generating purchase order

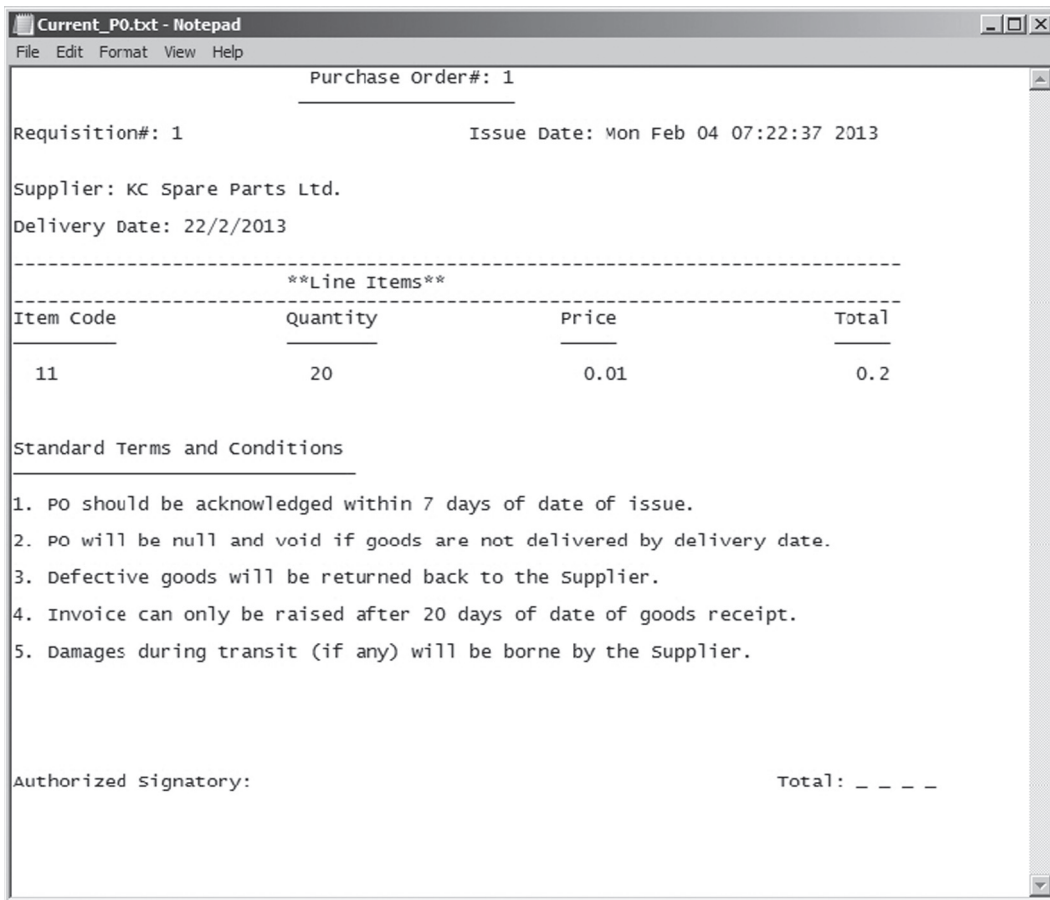


Fig. A.8 Viewing purchase order