เอกสารอ้างอิง

ภาษาไทย

ระพีพรรณ พิริยะกุล. โครงสร้างข้อมูลและเทคนิคการจัดไฟล์. กรุงเทพมหานคร: สำนักพิมพ์ยูไนเต็ดทีบุคส์, 2529.

สมบูรณ์ ชนกประสิทธิ์. การออกแบบและพัฒนาระบบเตรียมข้อมูลบนเครื่องไมโครคอมพิวเตอร์. วิทยานิพนธ์ปริญญามหาบัณฑิต จุฬาลงกรณ์มหาวิทยาลัย, 2532.

ภาษาอังกฤษ

Alen S. Fisher. Case Using Software Development Tools. Second Edition. John Wiley & Sons, inc., 1991.

Applied Information Systems, Inc. EasyEntry version 3.1 User's Guide. USA., 1984.

Donna Hussain, K.M. Hussain. Information Processing Systems for Management. Richard D. Irwin, Inc., USA., 1984.

General Electric Company. SOFTWARE ENGINEERING HANDBOOK. McGraw-Hill series in Software Engineering an Technology, 1986.

International Business Machines Corporation. 3742 Dual Data Station Operation's Guide. GA21-9136-3. Fifth Edition. IBM, 1977.

Kenmore S. Brathwaite. Applications Development Using CASE Tools. Academic Press, inc., 1990.

MicroPro International Corporation. DataStar Reference Manual Release 1.4. USA., 1982.

Nippon Electric Co., Ltd. System Description for N3600 Station Model 50 F3.

Ralph M. Stair, Jr. Principles of Data Processing Concepts, Application and Cases. Richard D. Irwin, Inc., USA., 1984.

Recognition Equipment Incorporated. Tartan Plus Data Entry. USA., 1986.

Recognition Equipment Incorporated. Tartan Plus PC Data Entry. USA., 1986.

Robert J. Condon. Data Processing Systems Analysis Design. Fourth Edition. Prentice-Hall Company, USA., 1985.

The Santa Cruz Operation, Inc. SCO UNIX System V/386 Development System C Language Reference. USA., 1989.

The Santa Cruz Operation, Inc. SCO UNIX System V/386 Development System C User's Guide. USA., 1989.

The Santa Cruz Operation, Inc. SCO UNIX System V/386 Development System Programmer's Guide. USA., 1989.

The Santa Cruz Operation, Inc. SCO UNIX System V/386 Development System Programmer's Reference. USA., 1989.

ภาคผนวก

ภาคผนวก ก

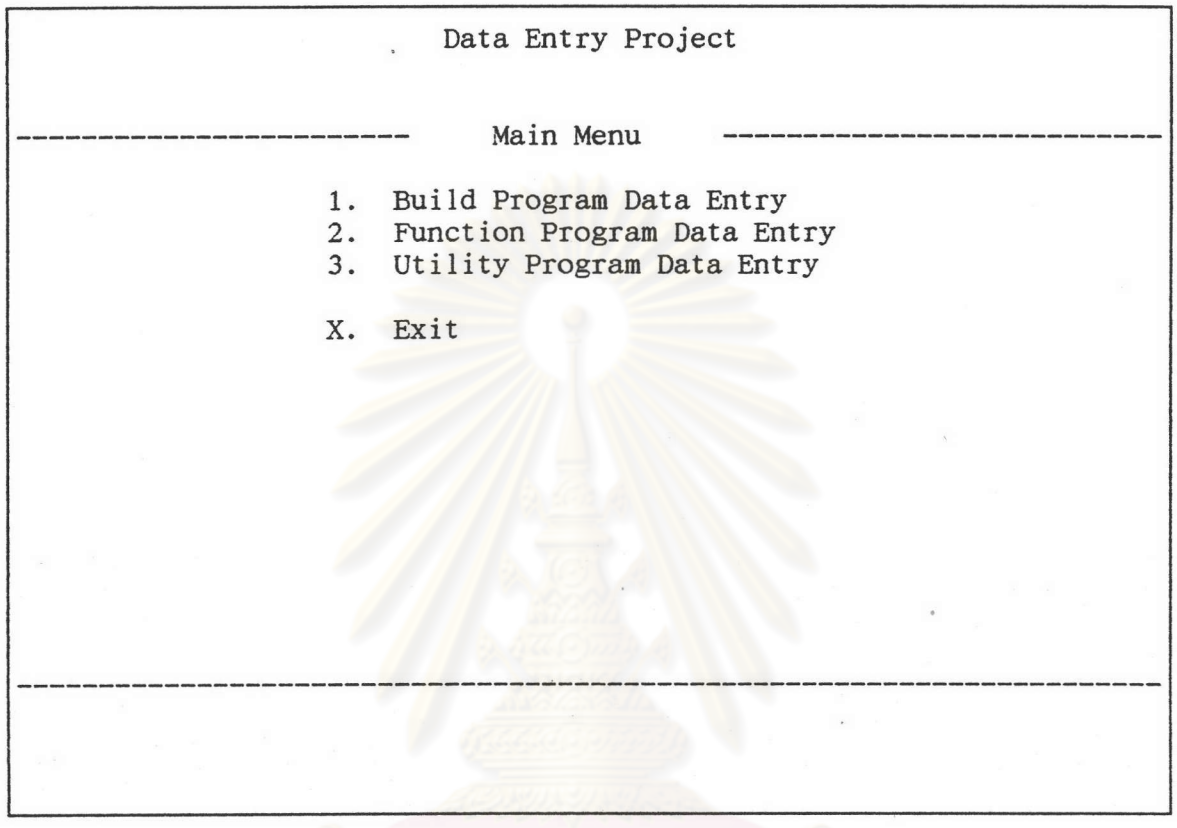
ตัวอย่างการใช้โปรแกรมสร้างโปรแกรมจัดเตรียมข้อมูล

ตัวอย่างการใช้งานในการสร้างโปรแกรมรับข้อมูลที่มีรูปแบบระเบียนข้อมูลประกอบด้วย เลขประจำตัว ชื่อ นามสกุล และเลขที่บัญชี โดยมีรูปแบบระเบียนดังนี้

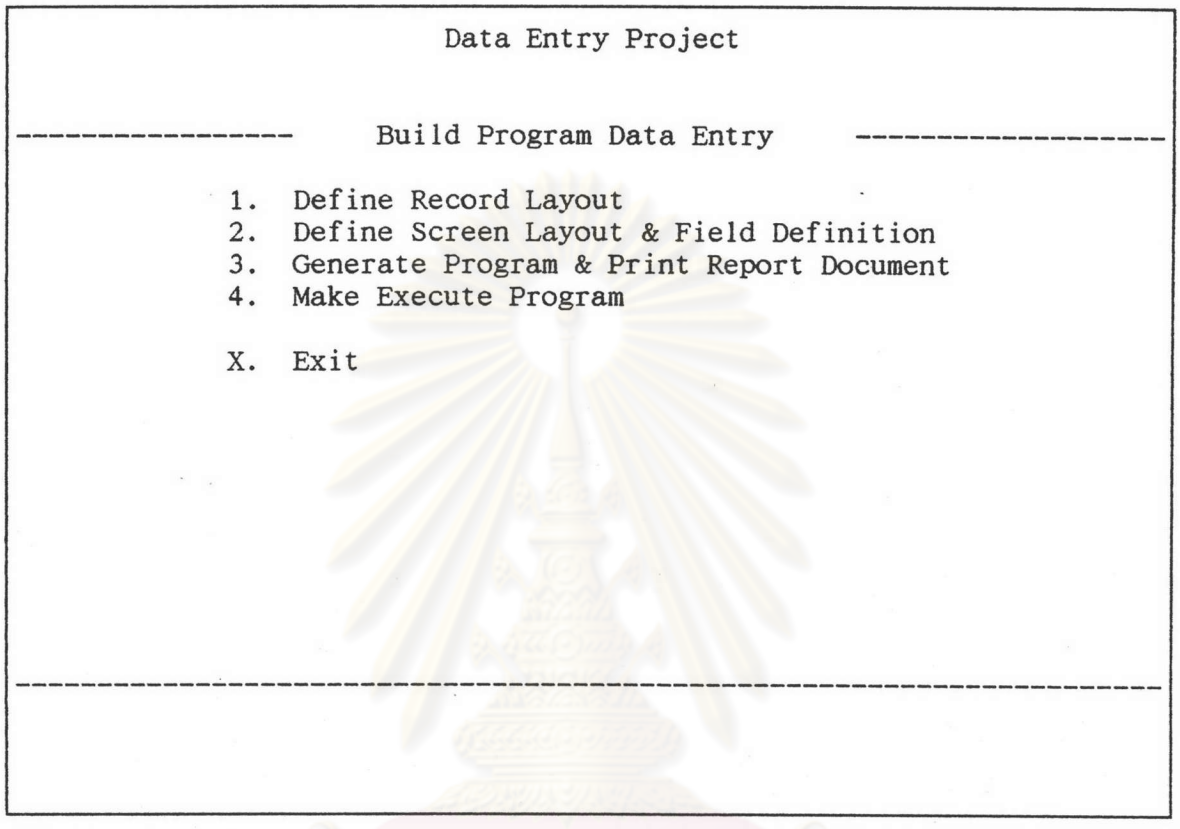
	ชนิด	ความยาว	จุดทศนิยม	ตำแหน่งเริ่มต้น	การเก็บ	เติมอักขระ
1. เลขประจำตัว	ตัวเลข	5	0	1	ชิดขวา	0
2. ชื่อ	ตัวอักษร	15		6	ชิดซ้าย	ช่องว่าง
3. นามสกุล	ตัวอักษร	20		21	ชิดซ้าย	ช่องว่าง
4. เลขที่บัญชี	ตัวเลข	10	0	41	ชิดขวา	0

ในการรับข้อมูลจะมีการกำหนดลำดับและคุณสมบัติของเขตข้อมูลดังนี้

1. เลขประจำตัว จะมีค่าเพิ่มขึ้นทีละหนึ่งโดยอัตโนมัติ ไม่มีการตรวจสอบค่าสูงสุดและต่ำสุด
2. เลขที่บัญชี จะมีการตรวจสอบค่าโดยใช้เลขโดดตัวสอบ วิธีโมดูลัส 10
3. ชื่อ จะมีการตรวจสอบไม่ให้ผ่านเขตข้อมูลนี้ได้ในกรณีที่ไม่มีข้อมูลในเขตข้อมูลนี้
4. นามสกุล จะมีการตรวจสอบไม่ให้ผ่านเขตข้อมูลนี้ได้ในกรณีที่ไม่มีข้อมูลในเขตข้อมูลนี้



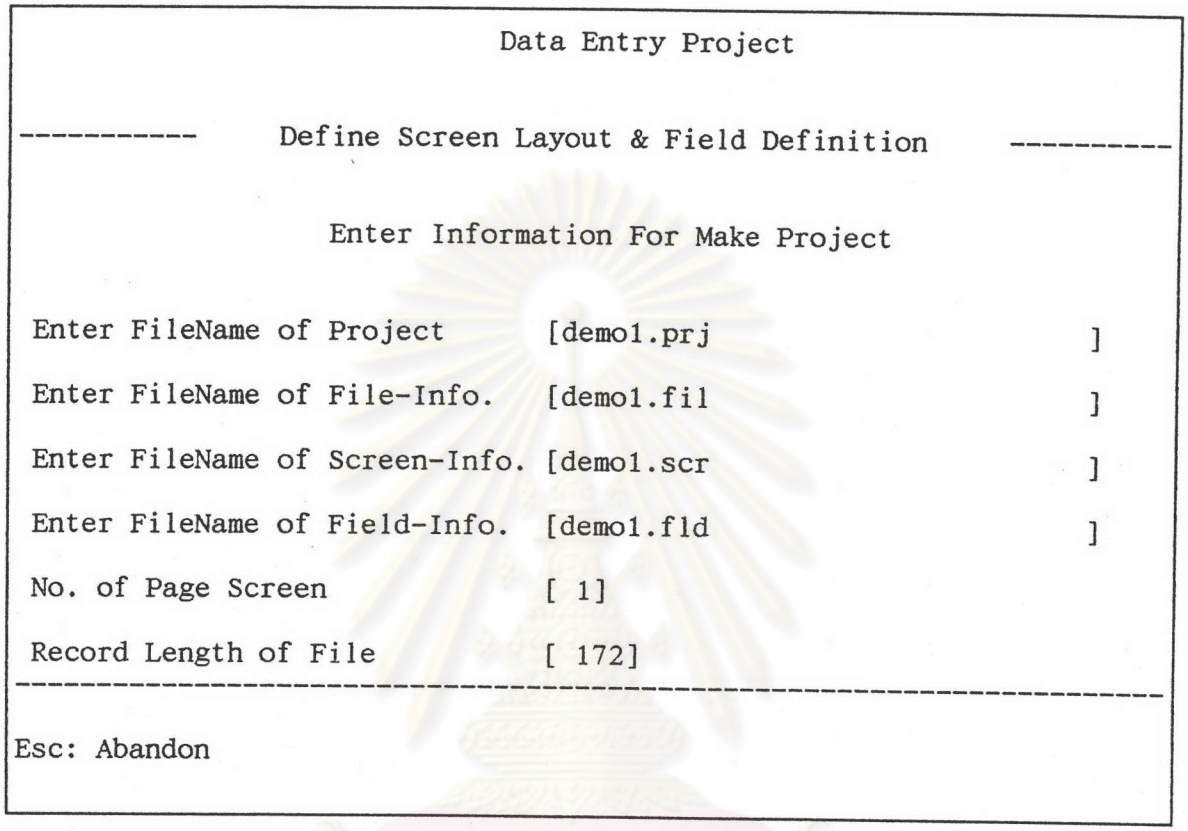
รูปที่ ก.1 แสดงหน้าจอรายการเมนูหลัก



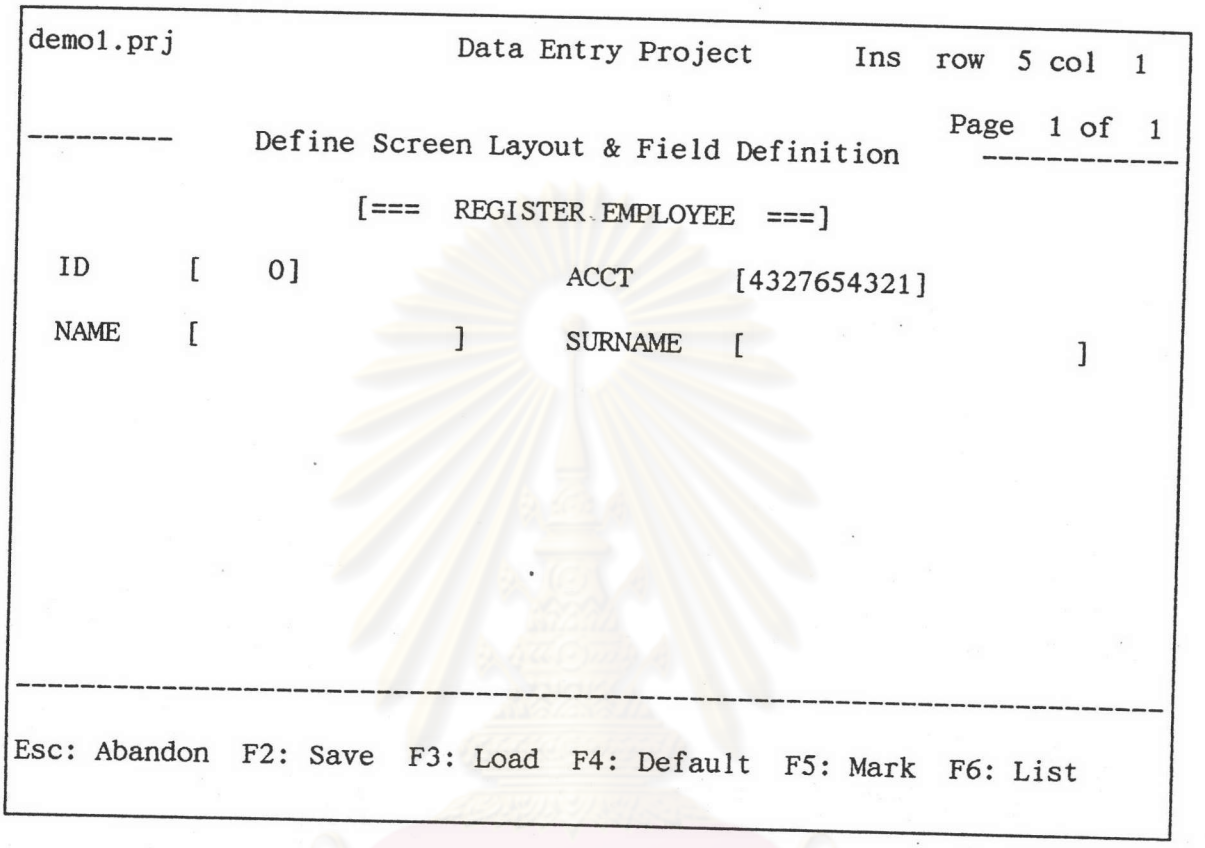
รูปที่ ก.2 แสดงหน้าจอรายการเมนูสำหรับสร้างโปรแกรมจัดเตรียมข้อมูล

```
demo1.fil                   Data Entry Project


--------------------    Define Record Layout    ---------------------

  No.        Field Name      Type  Length  Decimal  Start  Justify  Pad

 [  1] [ID               ] [N]  [  5]   [ 0]  [   1]   [R]    [Z]
 [  2] [NAME             ] [C]  [ 15]   [  ]  [   6]   [L]    [S]
 [  3] [SURNAME          ] [C]  [ 20]   [  ]  [  21]   [L]    [S]
 [  4] [ACCT             ] [N]  [ 10]   [ 0]  [  41]   [R]    [Z]
 [   ] [                 ] [ ]  [  0]   [  ]  [   0]   [ ]    [ ]
 [   ] [                 ] [ ]  [   ]   [  ]  [    ]   [ ]    [ ]
 [   ] [                 ] [ ]  [   ]   [  ]  [    ]   [ ]    [ ]
 [   ] [                 ] [ ]  [   ]   [  ]  [    ]   [ ]    [ ]
 [   ] [                 ] [ ]  [   ]   [  ]  [    ]   [ ]    [ ]
 [   ] [                 ] [ ]  [   ]   [  ]  [    ]   [ ]    [ ]
 [   ] [                 ] [ ]  [   ]   [  ]  [    ]   [ ]    [ ]
 [   ] [                 ] [ ]  [   ]   [  ]  [    ]   [ ]    [ ]
 [   ] [                 ] [ ]  [   ]   [  ]  [    ]   [ ]    [ ]
-----------------------------------------------------------------------

Esc: Abandon  F2: Save  F3: Load
```

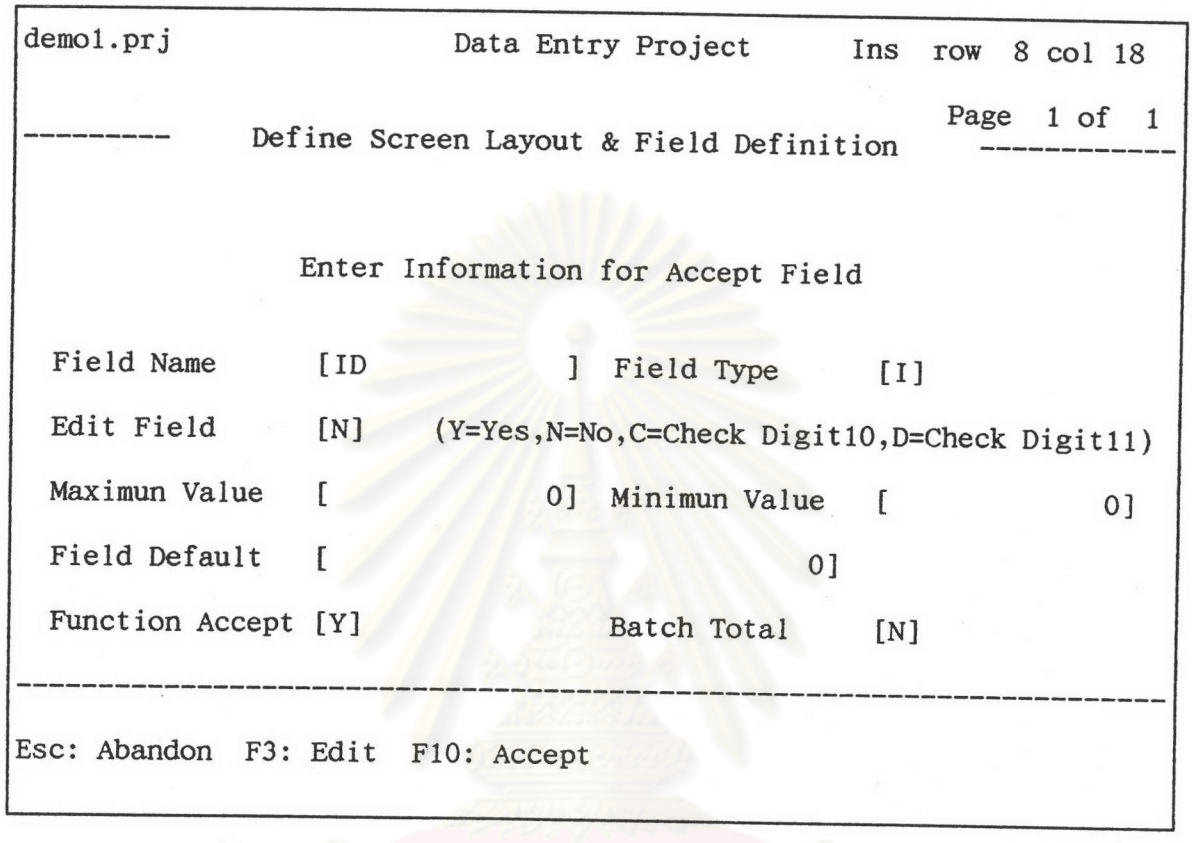
รูปที่ ก.3 แสดงหน้าจอสำหรับกำหนดรูปแบบระเบียนข้อมูล



รูปที่ ก.4 แสดงหน้าจอสำหรับรับข้อมูลที่ใช้เป็นข้อมูลในการสร้างโปรแกรม



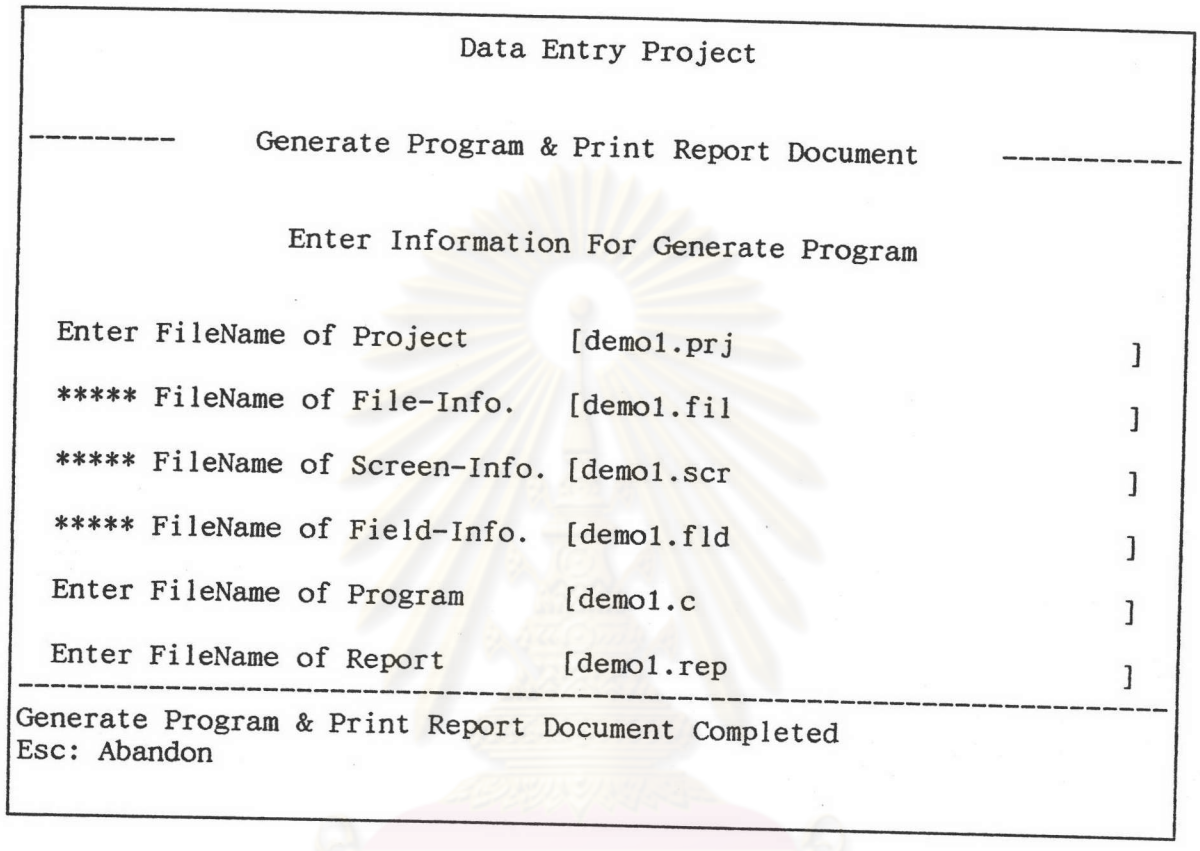
รูปที่ ก.5 แสดงหน้าจอสำหรับกำหนดฟอร์มสำหรับป้อนข้อมูล



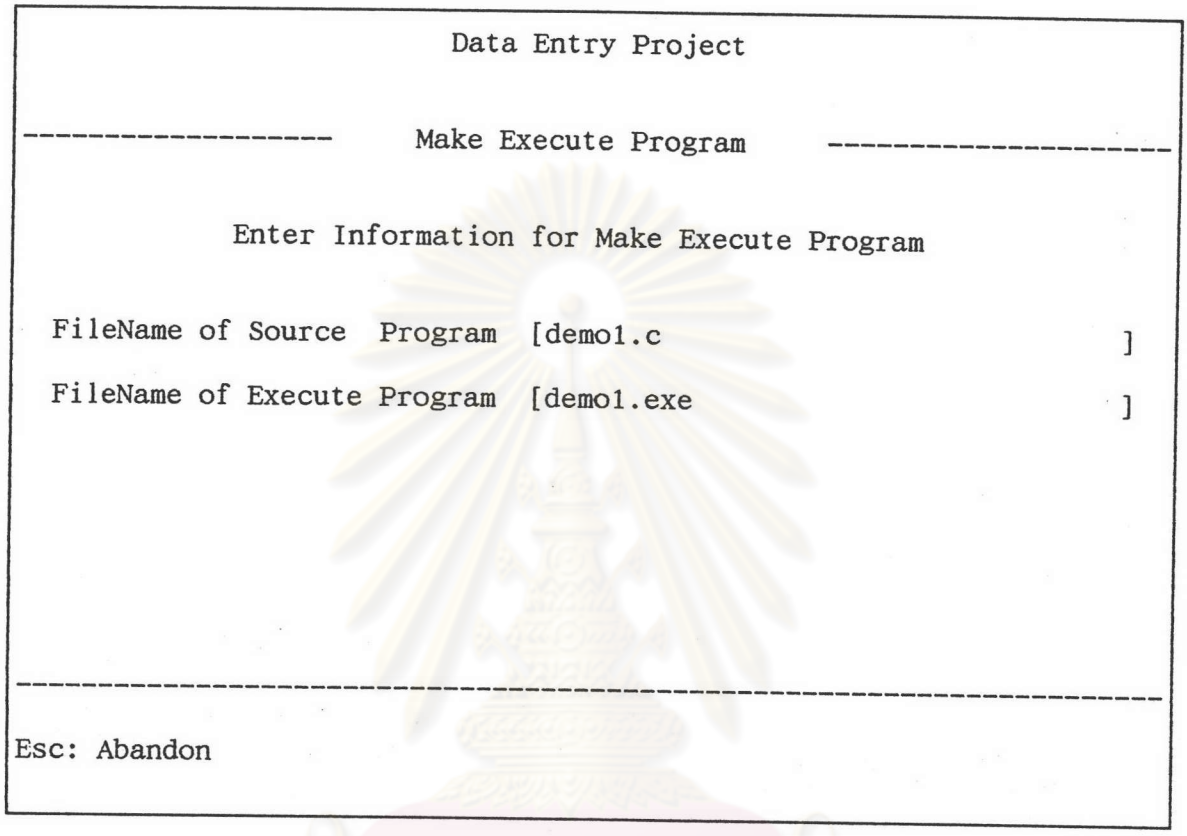
รูปที่ ก.6 แสดงหน้าจอสำหรับกำหนดคุณสมบัติของเขตข้อมูล

```
demo1.prj                    Data Entry Project        Ins  row  8 col 18

---------      Define Screen Layout & Field Definition      Page  1 of  1 ----------


Field Name     Type Len Dec Edit      Maximun      Minimun Function Batch

ID             I      5   0    N          0            0        Y       N
ACCT           N     10   0    C          0            0        Y       N
NAME           C     15   0    Y          0            0        Y       N
SURNAME        C     20   0    Y          0            0        Y       N




-------------------------------------------------------------------------------
Please any key to continue
Esc: Abandon  F2: Save  F3: Load  F4: Default  F5: Mark  F6: List
```

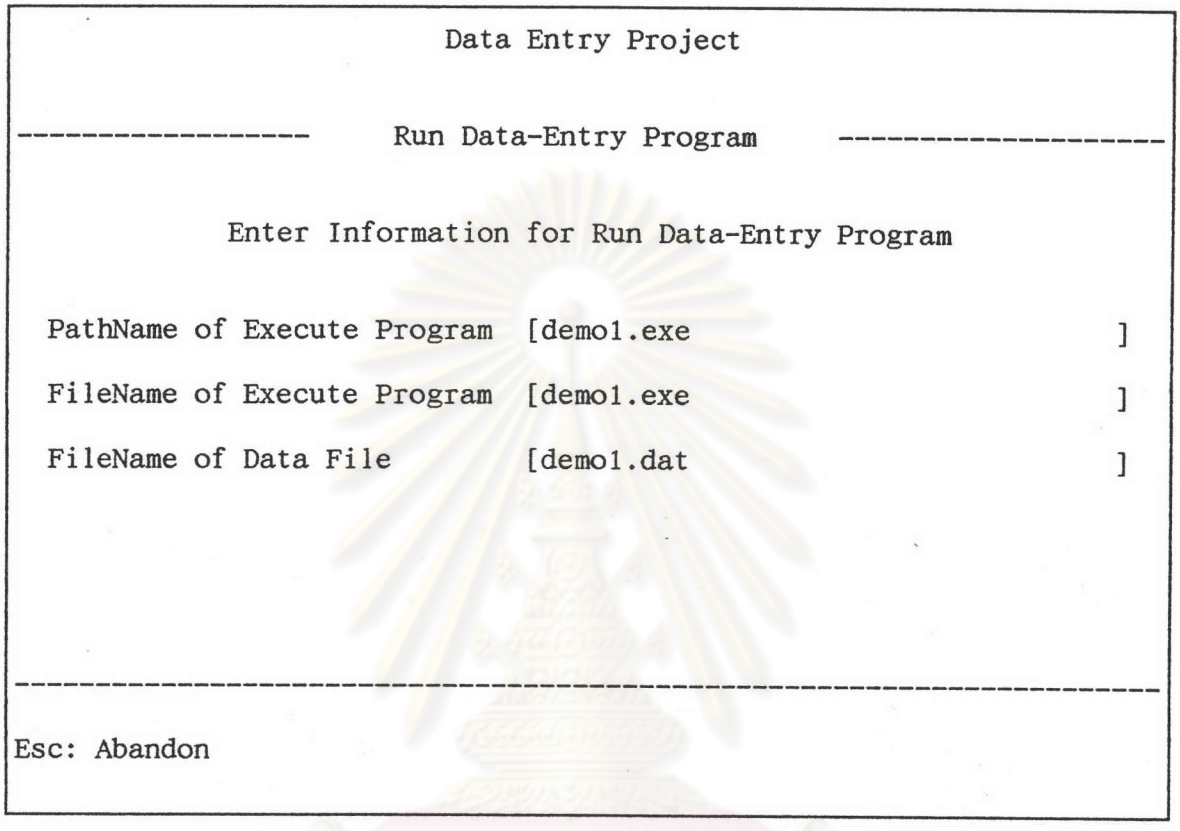
รูปที่ ก.7 แสดงหน้าจอสำหรับแสดงรายละเอียดของเขตข้อมูลที่กำหนด



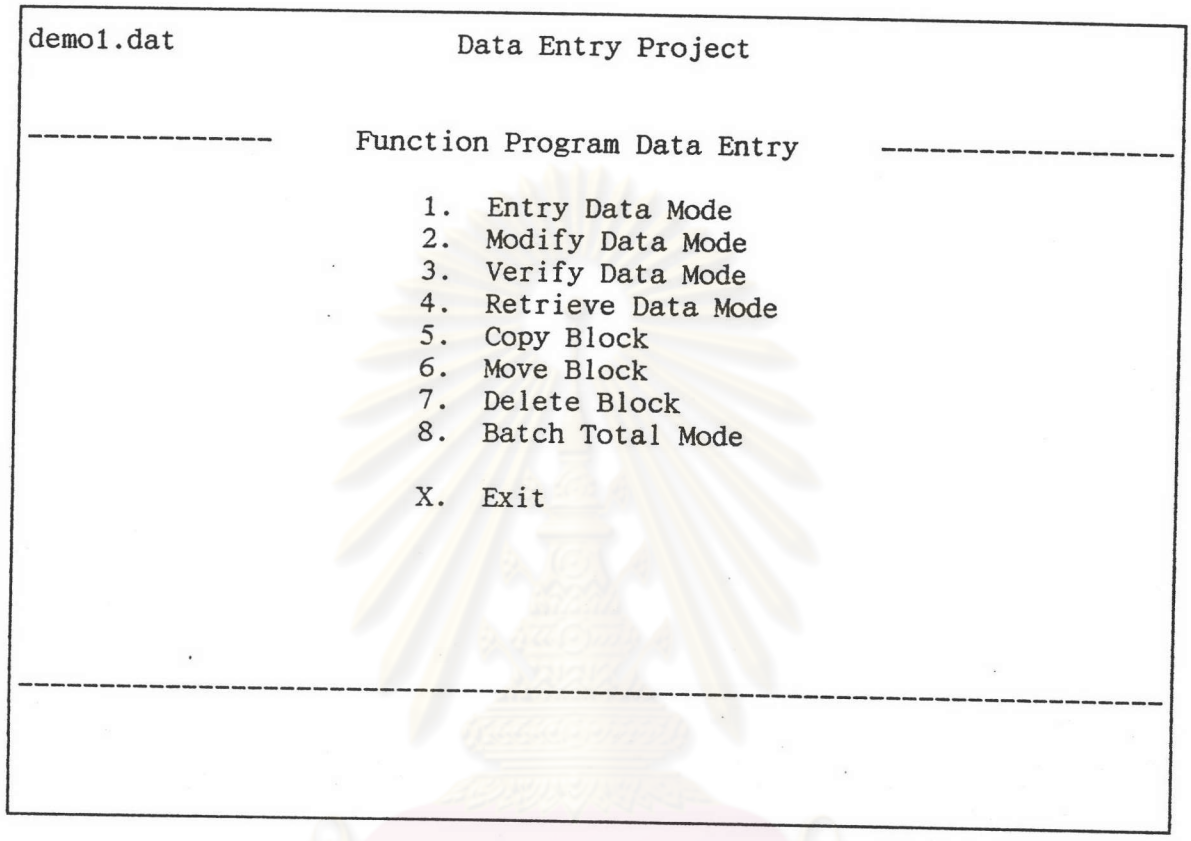
รูปที่ ก.8 แสดงหน้าจอสำหรับสร้างโปรแกรมและพิมพ์เอกสารประกอบ



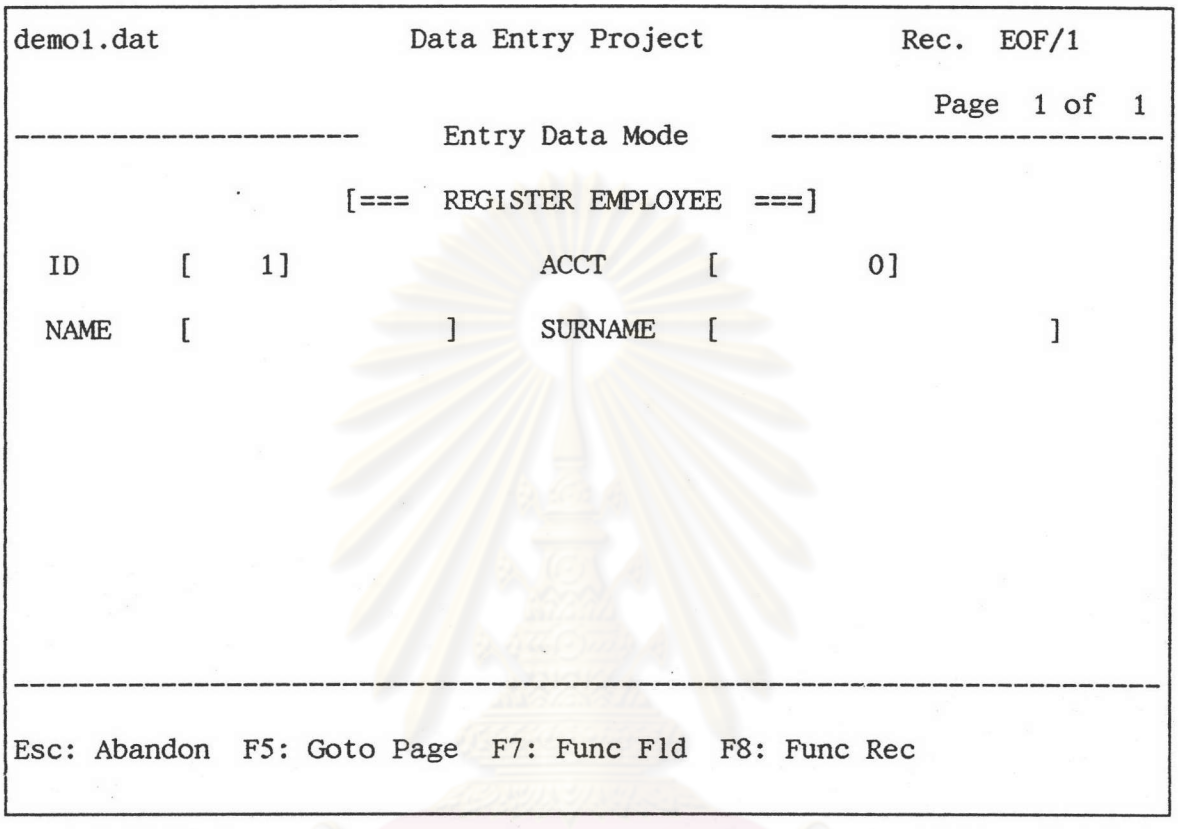
รูปที่ ก.9 แสดงหน้าจอสำหรับเปลี่ยนโปรแกรมที่สร้างได้ให้เป็นภาษาเครื่อง



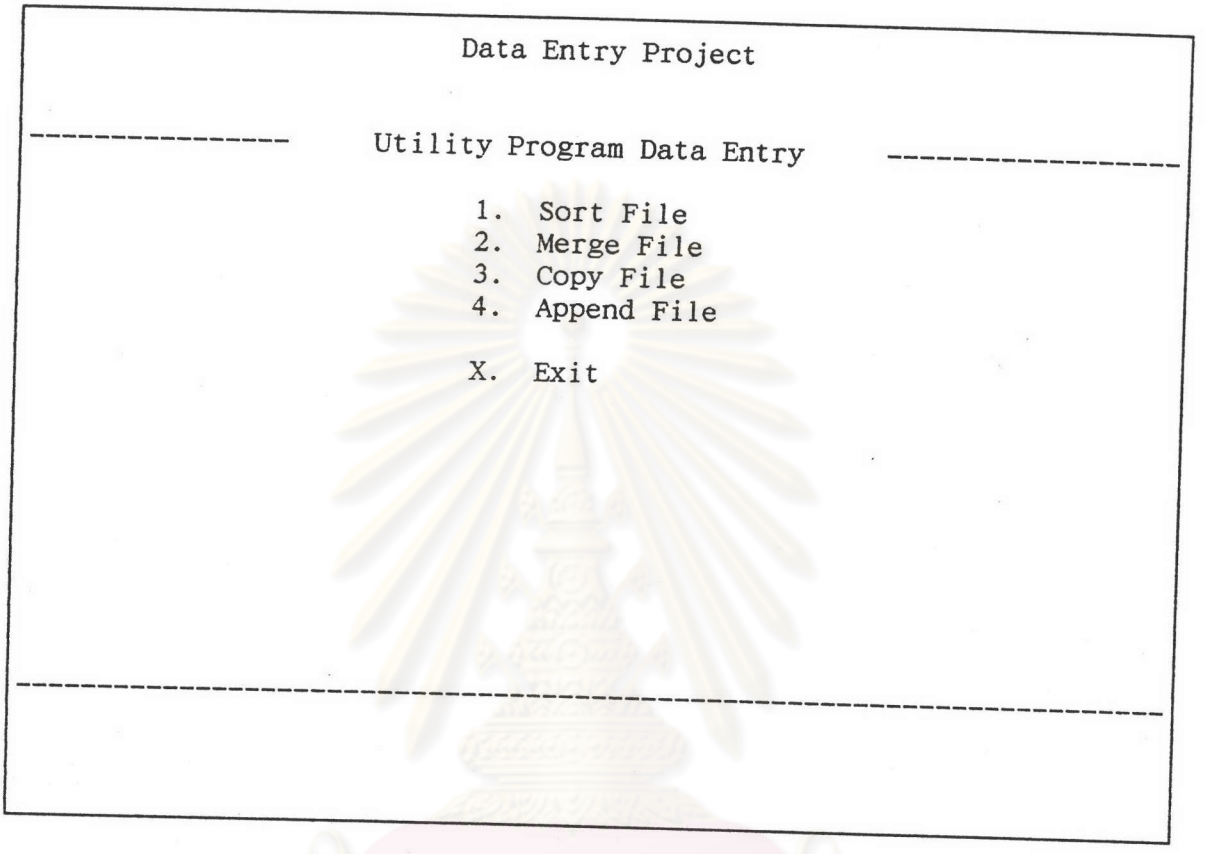
รูปที่ ก.10 แสดงหน้าจอสำหรับทำงานโปรแกรมจัดเตรียมข้อมูลที่ต้องการ



รูปที่ ก.11 แสดงหน้าจอรายการเมนูของโปรแกรมจัดเตรียมข้อมูล



รูปที่ ก.12 แสดงหน้าจอแบบฟอร์มสำหรับรับค่าข้อมูล



รูปที่ ก.13 แสดงหน้าจอรายการเมนูของโปรแกรมช่วยเสริมการทำงาน

ภาคผนวก ข

ตัวอย่างโปรแกรมที่ได้จากการสร้างโปรแกรมของระบบงาน

```
#include "userstru.h"

#define   ENTRY       0
#define   MODIFY      1
#define   VERIFY      2
#define   RETRIEVE    3
#define   nRecSize    51
#define   nRecSeek    52

/* ---------------------- */
/*  declare variable area  */
/* ---------------------- */
FILE *fd;
char wFNameDat[41] = "testgen.dat";
char w_Buf[nRecSize];
char *wFnMsg = "Esc: Abandon  F5: Goto Page  F7: Func Fld  F8: Func Rec";
long n_Cur,n_Last;
int n_Stat,n_Fld,n_MaxFld,n_Page,n_Func;

double nID;
double nACCT;
char wNAME[16];
char wSURNAME[21];

/* ------------------- */
/*  Main Line Routine  */
/* ------------------- */
main(argc,argv)
int  argc;
char *argv[];
{
  char *mtab[18];
  struct usr_menu m1;
  int n,choice;

  initscr();
  keypad(stdscr,TRUE);
  noecho();
  raw();
  mtab[0]     = "  1.  Entry Data Mode     ";
  mtab[1]     = "  2.  Modify Data Mode    ";
  mtab[2]     = "  3.  Verify Data Mode    ";
  mtab[3]     = "  4.  Retrieve Data Mode  ";
  mtab[4]     = "  5.  Copy Block          ";
  mtab[5]     = "  6.  Move Block          ";
  mtab[6]     = "  7.  Delete Block        ";
  mtab[7]     = "  8.  Batch Total Mode    ";
  mtab[8]     = "                          ";
  mtab[9]     = "  X.  Exit                ";
  m1.m_height = 10;
  m1.m_width  = strlen(mtab[0]);
  m1.m_title  = "Function Program Data Entry";
  m1.m_data   = &mtab[0];
  strcpy(wFNameDat,argv[1]);
```

```
  ScrTitle("Data Entry Project");
  SubTitle(m1.m_title);
  ShwFName(wFNameDat);
  choice      = 0;
  if ((n = RecSize(wFNameDat)) != 1)
  {
    if (n != nRecSize)
    {
      Msg("Length error");
      getch();
      choice  = 9;
    }
  }
  if (!FileExist(wFNameDat))
  {
    fd = fopen(wFNameDat,"w");
    fclose(fd);
  }
  while (choice < 9)
  {
    SubTitle(m1.m_title);
    ClrBox(5,1,21,80);
    ClrPosRec();
    switch((choice = usrmenu(&m1,choice)))
    {
      case 0 :
        EntryDta();
        break;
      case 1 :
        ModifyDta();
        break;
      case 2 :
        VerifyDta();
        break;
      case 3 :
        RetrieveDta();
        break;
      case 4 :
        CopyBlk();
        break;
      case 5 :
        MoveBlk();
        break;
      case 6 :
        DeleBlk();
        break;
      case 7 :
        BatchTot();
        break;
    }
    ClrFnMsg();
  }
  endwin();
}
```

```
/* -------------------- */
/*  Entry Data Routine  */
/* -------------------- */
EntryDta()
{
  SubTitle("Entry Data Mode");
  ClrBox(5,1,21,80);
  fd = fopen(wFNameDat,"r+");
  fseek(fd,0l,2);
  n_Last = ftell(fd) / (long) nRecSize;
  n_Cur  = n_Last + 1l;
  n_Fld  = 1;
  n_Page = 1;
  n_Func = ENTRY;
  ClrGetxx();
  while (n_Fld > 0)
    GetPage(n_Page);
  ClrPage();
  fclose(fd);
}

/* -------------------- */
/*  Modify Data Routine  */
/* -------------------- */
ModifyDta()
{
  SubTitle("Modify Data Mode");
  ClrBox(5,1,21,80);
  fd = fopen(wFNameDat,"r+");
  fseek(fd,0l,2);
  n_Last = ftell(fd) / (long) nRecSize;
  fseek(fd,0l,0);
  n_Cur  = 1l;
  n_Fld  = 1;
  n_Page = 1;
  n_Func = MODIFY;
  GetRecxx();
  while (n_Fld > 0)
    GetPage(n_Page);
  ClrPage();
  fclose(fd);
}

/* -------------------- */
/*  Verify Data Routine  */
/* -------------------- */
VerifyDta()
{
  SubTitle("Verify Data Mode");
  ClrBox(5,1,21,80);
  fd = fopen(wFNameDat,"r");
  fseek(fd,0l,2);
  n_Last = ftell(fd) / (long) nRecSize;
```

```
  fseek(fd,01,0);
  n_Cur  = 11;
  n_Fld  = 1;
  n_Page = 1;
  n_Func = VERIFY;
  GetRecxx();
  while (n_Fld > 0)
    GetPage(n_Page);
  ClrPage();
  fclose(fd);
}

/* ----------------------- */
/*  Retrieve Data Routine  */
/* ----------------------- */
RetrieveDta()
{
  SubTitle("Retrieve Data Mode");
  ClrBox(5,1,21,80);
  fd = fopen(wFNameDat,"r");
  fseek(fd,01,2);
  n_Last = ftell(fd) / (long) nRecSize;
  fseek(fd,01,0);
  n_Cur  = 11;
  n_Fld  = 1;
  n_Page = 1;
  n_Func = RETRIEVE;
  ClrGetxx();
  while (n_Fld > 0)
    GetPage(n_Page);
  ClrPage();
  fclose(fd);
}

/* -------------------- */
/*  Copy Block Routine  */
/* -------------------- */
CopyBlk()
{
  double nFrom;
  double nTo;
  double nAt;

  SubTitle("Copy Block");
  ClrBox(5,1,21,80);
  while (FldGetBlk(3,&nFrom,&nTo,&nAt) != IkEsc)
  {
    if (CopyBlock(wFNameDat,nRecSeek,(long) nFrom,(long) nTo,(long) nAt) == 0)
      Msg("*** Copy record completed. ***");
    else
      Msg("*** Cannot process this command ***");
  }
}
```

```
/* -------------------- */
/*  Move Block Routine  */
/* -------------------- */
MoveBlk()
{
  double nFrom;
  double nTo;
  double nAt;

  SubTitle("Move Block");
  ClrBox(5,1,21,80);
  while (FldGetBlk(3,&nFrom,&nTo,&nAt) != IkEsc)
  {
    if (MoveBlock(wFNameDat,nRecSeek,(long) nFrom,(long) nTo,(long) nAt) == 0)
      Msg("*** Move record completed. ***");
    else
      Msg("*** Cannot process this command ***");
  }
}

/* ---------------------- */
/*  Delete Block Routine  */
/* ---------------------- */
DeleBlk()
{
  double nFrom;
  double nTo;

  SubTitle("Delete Block");
  ClrBox(5,1,21,80);
  while (FldGetBlk(2,&nFrom,&nTo) != IkEsc)
  {
    if (DeleBlock(wFNameDat,nRecSeek,(long) nFrom,(long) nTo) == 0)
      Msg("*** Delete record completed. ***");
    else
      Msg("*** Cannot process this command ***");
  }
}

/* -------------------- */
/*  Batch Total Routine  */
/* -------------------- */
BatchTot()
{
  long n_Cnt;
  long i;

  SubTitle("Batch Total Mode");
  ClrBox(5,1,21,80);
  fd = fopen(wFNameDat,"r");
  fseek(fd,0l,2);
  n_Cnt = ftell(fd) / (long) nRecSize;
  fseek(fd,0l,0);
  for (i=0;i<n_Cnt;i++)
```

```
  {
    GetRecxx();
    fseek(fd,(long) nRecSeek,1);
  }
  fclose(fd);
  WrtXYAJSL(10,15,A_BOLD,'L',"Total Record of File          = ",35);
  WrtXYAS(10,58,A_BOLD,NumToStr((double) n_Cnt,7,0));
  getch();
}

/* ---------------------- */
/*  Function Key Routine  */
/* ---------------------- */
int FnKeyRtn(ch)
chtype ch;
{
  int i;
  long n1;

  switch(ch)
  {
    case IkF7 :   /* field function */
      ShwStat(FLD_FUNC);
      FnMsg("F5: Goto  F6: Dup");
      ch = getch();
      switch(ch)
      {
        case IkF6 :          /* dup field n */
          DupFldxx(n_Page,n_Fld);
          break;
        default :
          if ((i = FieldFunc(ch,n_Fld,n_MaxFld)) != 0)
            n_Fld = i;
          break;
      }
      ShwStat(n_Stat);
      FnMsg(wFnMsg);
      return(2);
    case IkF8 :   /* record function */
      ShwStat(REC_FUNC);
      FnMsg("F5: Goto  F6: Dup  Ins: Insert  Del: Delete");
      ch = getch();
      switch(ch)
      {
        case IkF6 :           /* dup record */
          fseek(fd,-(long) nRecSeek,1);
          GetRecxx();
          fseek(fd,(long) nRecSeek,1);
          break;
        case IkIns :          /* insert record */
          fclose(fd);
          CopyBlock(wFNameDat,nRecSeek,n_Cur,n_Cur,n_Cur);
          fd = fopen(wFNameDat,"r+");
          fseek(fd,(long) nRecSeek * (n_Cur - 11),0);
```

```
            n_Last++;
            ClrGetxx();
            break;
          case IkDel :          /* delete record */
            fclose(fd);
            DeleBlock(wFNameDat,nRecSeek,n_Cur,n_Cur);
            fd = fopen(wFNameDat,"r+");
            fseek(fd,(long) nRecSeek * (n_Cur - 11),0);
            n_Last--;
            GetRecxx();
            break;
          default :
            if ((nl = RecordFunc(ch,n_Cur,n_Last)) != 01)
            {
              fseek(fd,(long) nRecSeek * (nl - 11),0);
              GetRecxx();
            }
            break;
        }
        ShwStat(n_Stat);
        FnMsg(wFnMsg);
        GetPage(n_Page);
        return(1);
    }
    return(0);
}

/* -------------------- */
/*  Dup. Field Routine  */
/* -------------------- */
DupFldxx(nPage,nFld)
int nPage,nFld;
{
    char w_Buf[nRecSize];
    char w_Num[31];

    fseek(fd,-(long) nRecSeek,1);
    fread(w_Buf,nRecSize,1,fd);
    w_Buf[nRecSize-1] = '\0';
    switch(nPage)
    {
      case 1 :
        switch(nFld)
        {
          case 1 :
            strcpy(w_Num,Repl(' ',5));
            strncpy(w_Num,w_Buf+0,5);
            nID = StrToNum(w_Num,0);
            break;
          case 2 :
            strcpy(w_Num,Repl(' ',10));
            strncpy(w_Num,w_Buf+40,10);
            nACCT = StrToNum(w_Num,0);
            break;
```

```
        case 3 :
          strncpy(wNAME,w_Buf+5,15);
          break;
        case 4 :
          strncpy(wSURNAME,w_Buf+20,20);
          break;
      }
      break;
  }
}

/* ------------------------- */
/*  Validate Record Routine  */
/* ------------------------- */
int ValidRecxx()
{
  char wStr[31];

  strcpy(wStr,NumToText(nACCT,10,0,'0'));
  if (!CheckDigit10(wStr,"4327654321"))
  {
    Msg("check digit error");
    n_Page = 1;
    n_Fld  = 2;
    return(0);
  }
  if (isblank(wNAME))
  {
    Msg("cannot omitted this field");
    n_Page = 1;
    n_Fld  = 3;
    return(0);
  }
  if (isblank(wSURNAME))
  {
    Msg("cannot omitted this field");
    n_Page = 1;
    n_Fld  = 4;
    return(0);
  }
  return(1);
}

/* -------------------- */
/*  Put Record Routine  */
/* -------------------- */
PutRecxx()
{
  strncpy(w_Buf+0,NumToText(nID,5,0,'0'),5);
  strncpy(w_Buf+40,NumToText(nACCT,10,0,'0'),10);
  strncpy(w_Buf+5,wNAME,15);
  strncpy(w_Buf+20,wSURNAME,20);
  fprintf(fd,"%s\n",w_Buf);
  fseek(fd,01,1);
```

```
    if (n_Cur > n_Last)  n_Last++;
  }

  /* -------------------- */
  /*  Get Record Routine  */
  /* -------------------- */
  GetRecxx()
  {
    char w_Num[31];

    fread(w_Buf,nRecSize,1,fd);
    w_Buf[nRecSize-1] = '\0';
    fseek(fd,-(long) nRecSeek,1);
    strcpy(w_Num,Repl(' ',5));
    strncpy(w_Num,w_Buf+0,5);
    nID = StrToNum(w_Num,0);
    strcpy(w_Num,Repl(' ',10));
    strncpy(w_Num,w_Buf+40,10);
    nACCT = StrToNum(w_Num,0);
    strncpy(wNAME,w_Buf+5,15);
    strncpy(wSURNAME,w_Buf+20,20);
  }

  /* ------------------------ */
  /*  Page Get Field Routine  */
  /* ------------------------ */
  GetPage(n)
  int n;
  {
    switch(n)
    {
      case 1 :
        n_MaxFld = 4;
        FldGet01(n_Func);
        break;
      default :
        n_Fld  = 0;
        n_Page = 0;
        break;
    }
  }

  /* ------------------------ */
  /*  Display Screen Routine  */
  /* ------------------------ */
  MapGet01()
  {
    char *w_Line;

    ShwPage(1,1);
    ClrBox(5,1,21,80);
    w_Line = "[===  REGISTER EMPLOYEE  ===]";
    WrtXYAS(6,26,A_NORMAL,w_Line);
    w_Line = "ID            [    0]                    ACCT          [4327654321]";
```

```
  WrtXYAS(8,3,A_NORMAL,w_Line);
  w_Line = "NAME          [                ]          SURNAME       [
  WrtXYAS(10,3,A_NORMAL,w_Line);
}

/* ------------------------- */
/*  Clear Field Get Routine  */
/* ------------------------- */
ClrGetxx()
{
  strcpy(w_Buf,Repl(' ',nRecSize-1));
  nID = nID + 11;
  nACCT = (double) 0;
  strcpy(wNAME,Repl(' ',15));
  strcpy(wSURNAME,Repl(' ',20));
}

/* -------------------------- */
/*  Display Field Get Routine  */
/* -------------------------- */
ShwGet01()
{
  n_Cur = ftell(fd) / (long) nRecSize + 11;
  ShwPosRec(n_Cur,n_Last);
  WrtXYAS(8,17,A_BOLD,NumToStr(nID,5,0));
  WrtXYAS(8,55,A_BOLD,NumToStr(nACCT,10,0));
  WrtXYAS(10,17,A_BOLD,wNAME);
  WrtXYAS(10,55,A_BOLD,wSURNAME);
}

/* ------------------- */
/*  Get Field Routine  */
/* ------------------- */
FldGet01(nFunc)
int nFunc;
{
  int  i;

  char wStr[31];

  MapGet01();
  ShwGet01();
  FnMsg(wFnMsg);
  while (n_Fld > 0 && n_Fld < n_MaxFld+1)
  {

    if (n_Fld == 1)
    {
      TLastkey = TGetNum(8,17,A_REVERSE,A_BOLD,&nID,5,0);
      ClrMsg();
      switch(TLastkey)
      {
        case IkEsc :
          n_Fld = 0;
```

```
        break;
      case IkUp :
      case IkLeft :
        n_Fld = 1;
        break;
      default :
        if ((i = PageFunc(TLastkey,n_Page,1)) != 0)
        {
          n_Fld  = 1;
          n_Page = i;
          return;
        }
        if (FnKeyRtn(TLastkey) > 0)
          break;
        switch(nFunc)
        {
          case VERIFY :
            if (strncmp(w_Buf+0,NumToText(nID,5,0,'0'),5) != 0)
            {
              Msg("incorrect data");
              break;
            }
            n_Fld = 2;
            break;
          case RETRIEVE :
            if (TLastkey == IkDown)
            {
              n_Fld = 2;
              break;
            }
            fseek(fd,(long) nRecSeek,1);
            FindText(fd,nRecSize,1,5,NumToText(nID,5,0,'0'));
            fseek(fd,-(long) nRecSeek,1);
            GetRecxx();
            GetPage(n_Page);
            break;
          default :
            n_Fld = 2;
            break;
        }
    }
}

if (n_Fld == 2)
{
  TLastkey = TGetNum(8,55,A_REVERSE,A_BOLD,&nACCT,10,0);
  ClrMsg();
  switch(TLastkey)
  {
    case IkEsc :
      n_Fld = 0;
      break;
    case IkUp :
    case IkLeft :
```

```
        n_Fld = 1;
        break;
      default :
        if ((i = PageFunc(TLastkey,n_Page,1)) != 0)
        {
          n_Fld  = 1;
          n_Page = i;
          return;
        }
        if (FnKeyRtn(TLastkey) > 0)
          break;
        switch(nFunc)
        {
          case VERIFY :
            if (strncmp(w_Buf+40,NumToText(nACCT,10,0,'0'),10) != 0)
            {
              Msg("incorrect data");
              break;
            }
            n_Fld = 3;
            break;
          case RETRIEVE :
            if (TLastkey == IkDown)
            {
              n_Fld = 3;
              break;
            }
            fseek(fd,(long) nRecSeek,1);
            FindText(fd,nRecSize,41,10,NumToText(nACCT,10,0,'0'));
            fseek(fd,-(long) nRecSeek,1);
            GetRecxx();
            GetPage(n_Page);
            break;
          default :
            if (TLastkey < '0' || TLastkey > '9')
            {
              Msg("invalid data");
              break;
            }
            strcpy(wStr,NumToText(nACCT,10,0,'0'));
            if (!CheckDigit10(wStr,"4327654321"))
            {
              Msg("check digit error");
              break;
            }
            n_Fld = 3;
            break;
        }
    }
  }

  if (n_Fld == 3)
  {
    TLastkey = TGetStr(10,17,A_REVERSE,A_BOLD,wNAME);
```

```
    ClrMsg();
    switch(TLastkey)
    {
      case IkEsc :
        n_Fld = 0;
        break;
      case IkUp :
      case IkLeft :
        n_Fld = 2;
        break;
      default :
        if ((i = PageFunc(TLastkey,n_Page,1)) != 0)
        {
          n_Fld  = 1;
          n_Page = i;
          return;
        }
        if (FnKeyRtn(TLastkey) > 0)
          break;
        switch(nFunc)
        {
          case VERIFY :
            if (strncmp(w_Buf+5,wNAME,15) != 0)
            {
              Msg("incorrect data");
              break;
            }
            n_Fld = 4;
            break;
          case RETRIEVE :
            if (TLastkey == IkDown)
            {
              n_Fld = 4;
              break;
            }
            fseek(fd,(long) nRecSeek,1);
            FindText(fd,nRecSize,6,15,wNAME);
            fseek(fd,-(long) nRecSeek,1);
            GetRecxx();
            GetPage(n_Page);
            break;
          default :
            if (isblank(wNAME))
            {
              Msg("cannot omitted this field");
              break;
            }
            n_Fld = 4;
            break;
        }
    }
}
```

```
if (n_Fld == 4)
{
  TLastkey = TGetStr(10,55,A_REVERSE,A_BOLD,wSURNAME);
  ClrMsg();
  switch(TLastkey)
  {
    case IkEsc :
      n_Fld = 0;
      break;
    case IkUp :
    case IkLeft :
      n_Fld = 3;
      break;
    default :
      if ((i = PageFunc(TLastkey,n_Page,1)) != 0)
      {
        n_Fld  = 1;
        n_Page = i;
        return;
      }
      if (FnKeyRtn(TLastkey) > 0)
        break;
      switch(nFunc)
      {
        case VERIFY :
          if (strncmp(w_Buf+20,wSURNAME,20) != 0)
          {
            Msg("incorrect data");
            break;
          }
          if (n_Cur < n_Last)
          {
            fseek(fd,(long) nRecSeek,1);
            GetRecxx();
          }
          n_Fld  = 1;
          n_Page = 1;
          GetPage(n_Page);
          break;
        case RETRIEVE :
          if (TLastkey == IkDown)
          {
            break;
          }
          fseek(fd,(long) nRecSeek,1);
          FindText(fd,nRecSize,21,20,wSURNAME);
          fseek(fd,-(long) nRecSeek,1);
          GetRecxx();
          GetPage(n_Page);
          break;
        default :
          if (isblank(wSURNAME))
          {
            Msg("cannot omitted this field");
            break;
```

```
                }
                if (!ValidRecxx())
                {
                  GetPage(n_Page);
                  break;
                }
                PutRecxx();
                Msg("already saved data");
                if (n_Cur >= n_Last)
                {
                  if (nFunc == MODIFY)
                  {
                    fseek(fd,-(long) nRecSeek,1);
                    GetRecxx();
                  }
                  else
                    ClrGetxx();
                }
                else
                  GetRecxx();
                n_Fld  = 1;
                n_Page = 1;
                GetPage(n_Page);
                break;
            }
        }
    }

  }
}
```

ภาคผนวก ค

การเปรียบเทียบคุณสมบัติของระบบเตรียมข้อมูล

ตารางที่ ค.1 แสดงการเปรียบเทียบคุณสมบัติของระบบเตรียมข้อมูลที่สร้างได้ กับ ระบบเตรียมข้อมูลอื่นๆ

Feature	IBM 3742	N6300 Model 50 F3	Data Point	Tartan Plus	Data Star	Easy Entry	Prg. Gen.
Max.record length (chars)	128	128	253	1024	255	no info.	1024
Form Preparation							
- screen painting	-	X	X	X	X	X	X
- form definition language	-	-	X	X	-	X	X
Field Definition							
Field type							
- character field	X	X	X	X	X	X	X
- numeric field	X	X	X	X	X	X	X
- date field	-	-	X	-	-	X	X
Required entry field							
- at least 1 char.	-	-	X	-	-	-	X
- entire field	X	X	X	X	X	X	X
Calculated field							
- numeric expression	-	-	X	-	X	-	-
- string expression	-	-	-	-	X	-	-
Right/Left justification	X	X	X	X	X	X	X
Pad character	X	X	X	X	X	X	X
Float character	-	-	-	-	X	-	-
Verify field							
- sight verification	-	-	-	-	X	-	-
- retype verification	X	X	X	X	X	X	X
- list verification (table lookup)	-	-	X	X	X	X	-
Check digit field							
- divisible by 11 with no remainder	-	-	-	-	X	-	-
- modulus 10	X	X	-	-	-	-	X
- modulus 11	X	X	-	-	-	-	X
Range check field	-	-	X	X	X	X	X
Automatic blank-filled field	X	X	-	-	-	X	X
Automatic entered field							
- reading values from seperate file	-	-	-	-	X	-	-
- constant value	X	X	X	X	X	X	X
Display attribute							
- display only	X	X	X	X	X	X	X
- non display	-	X	-	-	-	-	-

ตารางที่ ค.1 แสดงการเปรียบเทียบคุณสมบัติของระบบเตรียมข้อมูลที่สร้างได้ กับ ระบบเตรียมข้อมูลอื่นๆ (ต่อ)

Feature	IBM 3742	N6300 Model 50 F3	Data Point	Tartan Plus	Data Star	Easy Entry	Prg. Gen.
Batch total field	X	X	X	X	-	X	X
Sequence checking field							
- ascendency check	-	-	-	-	X	-	-
- descandency check	-	-	-	-	-	-	-
Operation Mode							
Form print-out mode	-	X	-	X	X	X	X
Entry mode							
- add new record	X	X	X	X	X	X	X
- copy from previous record	X	X	X	X	X	X	X
Data retrieval mode							
- search by key	-	-	-	X	X	X	-
- search in sequential order	X	X	X	X	X	X	X
- search in index-file order	-	-	-	X	X	X	-
- search by sector #	X	X	-	-	-	-	-
- backward/forward retrieval	X	X	X	X	X	X	X
Data modification mode							
- update record	X	X	X	X	X	X	X
- insert record	X	X	X	X	X	X	X
- delete record	X	X	X	X	X	X	X
Verify mode							
- batch verification from working-file	-	-	-	-	X	-	-
- individual record verification	X	X	X	X	X	X	X
Batch total mode	X	X	X	X	-	X	X
Production statistics mode							
- record counter	X	X	X	X	-	X	X
- keystroke counter	X	X	X	X	-	X	-
- keystroke per hour	-	-	-	X	-	X	-
- verify correction keystroke counter	X	X	X	X	-	X	-
- elasped time in batch	-	-	-	X	-	X	-
Print data mode	-	-	X	-	X	X	-

หมายเหตุ X หมายถึง มีคุณสมบัติ - หมายถึง ไม่มีคุณสมบัติ

ประวัติผู้เขียน

นายพิสุทธิ์ วัฒนกุล เกิดวันที่ 27 พฤษภาคม 2509 สำเร็จการศึกษาวิทยาศาสตร์บัณฑิต (สาขาคณิตศาสตร์) จาก คณะวิทยาศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ เมื่อปีการศึกษา 2531 เข้าศึกษาระดับปริญญามหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. 2533