

```

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

void main()
{
    FILE *infile,*outfile;
    int Num,i;
    int NumList[20];
    int Total;
    int Count;

    infile = fopen("numlist.txt","r");
    if (infile == NULL)
    {
        fprintf(stderr,"Can't open numlist.txt\n");
        exit(100);
    }
    outfile = fopen("newnum.txt","w");
    if (outfile == NULL)
    {
        fprintf(stderr,"Can't open newnum.txt\n");
        fclose(infile);
        exit(100);
    }
    fscanf(infile,"%d",&Num);
    for (Count = 0 ; !feof(infile) ; Count++)
    {
        if (Count >= 20)
        {
            fprintf(stderr,"Too many numbers, Stopping at 20\n");
            break;
        }
        NumList[Count] = Num;
        fscanf(infile,"%d",&Num);
    }
    fclose(infile);
    for (i=0 ; i<Count ; i++)
    {
        NumList[i] *= NumList[i];
    }
    for (i=0,Total=0 ; i<Count ; i++)
    {
        fprintf(outfile,"%d -- %d\n",i,NumList[i]);
        Total += NumList[i];
    }
    fprintf(outfile,"Total -- %d\n",Total);
    fclose(outfile);
}

```

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

void main()
{
    int NumTab[10][10];
    int i,j;

    for (i=0 ; i<10 ; i++)
    {
        NumTab[i][0] = i+1;
        for (j=1 ; j<10 ; j++)
        {
            NumTab[i][j] = NumTab[i][j-1] + NumTab[i][0];
        }
    }
    for (i=0 ; i<10 ; i++)
    {
        for (j=0 ; j<10 ; j++)
        {
            fprintf(stdout,"%3.3d ",NumTab[i][j]);
        }
        putc('\n',stdout);
    }
    putc('\n',stdout);
    putc('\n',stdout);
    for (i=0 ; i<10 ; i++)
    {
        for (j=0 ; j<10 ; j++)
        {
            fprintf(stdout,"%-3d ",NumTab[i][j]);
        }
        putc('\n',stdout);
    }
}
```

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

void main()
{
    int NumTab[20];
    int i;
    int *TabPtr;

    for (i=0,TabPtr=NumTab ; i<20 ; i++,TabPtr++)
    {
        *TabPtr = i*i;
    }
    for (i=0,TabPtr=NumTab ; i<20 ; i++,TabPtr++)
    {
        fprintf(stdout,"%d -- %d\n",i,*TabPtr);
    }
}
```

```

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

void main()
{
    int NumTab[10][10];
    int i,j;
    int (*pi)[10],*pj,*Zptr,*ppj;

    for (i=0,pi=NumTab ; i<10 ; i++,pi++)
    {
        Zptr = *pi;
        *Zptr = i+1;
        for (j=1,ppj=Zptr,pj=Zptr+1 ; j<10 ; j++,pj++,ppj++)
        {
            *pj = *ppj + *Zptr;
        }
    }
    for (i=0,pi=NumTab ; i<10 ; i++,pi++)
    {
        for (j=0,pj=*pi ; j<10 ; j++,pj++)
        {
            fprintf(stdout,"%3.3d ",*pj);
        }
        putc('\n',stdout);
    }
    putc('\n',stdout);
    putc('\n',stdout);
    for (i=0,pi=NumTab ; i<10 ; i++,pi++)
    {
        for (j=0,pj=*pi ; j<10 ; j++,pj++)
        {
            fprintf(stdout,"%3d ",*pj);
        }
        putc('\n',stdout);
    }
}

```

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

void main()
{
    char Tc,Str[30],*Sptr;
    int i;

    strcpy(Str,"Some Stuff");
    for (i=0 ; Str[i] ; i++)
    {
        Tc = Str[i];
        printf("%c\n",Tc);
    }
    for (Sptr = Str ; *Sptr ; Sptr++)
    {
        Tc = *Sptr;
        printf("%c\n",Tc);
    }
}
```

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

void main()
{
    int Numbers[20],i,count;

    scanf("%d",Numbers);
    count = 0;
    while (!feof(stdin))
    {
        count++;
        scanf("%d",&(Numbers[count]));
    }
    for (i=0 ; i<count ; i++)
    {
        if (Numbers[i] == 1)
        {
            continue;
        }
        if (Numbers[i] < 3)
        {
            break;
        }
    }
    printf("i = %d\n",i);
}
```

```

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

void main()
{
    // A struct is a single data type
    // containing other data items.
    // Each type of structure requires a name.
    // This name appears only in the declarations.
    struct MyStruct
    {
        int MyInteger;
        long MyLong;
        short MyShort;
        char MyCharacter;
        char MyString[20];
    }
    // All Data from the keyword "struct"
    // through the closing brace is the
    // type name. Just like char and int.
    // The type name is followed by variable
    // declarations: (just as for int and char)
    NewStruct;

    // To access a struct, write the variable
    // name of the struct, followed by a period,
    // followed by the name of the element.

    NewStruct.MyInteger = 10;
    NewStruct.MyLong = 1234567;
    NewStruct.MyCharacter = 'c';
    strcpy(NewStruct.MyString, "abc");
    printf("Enter a number: ");

    // & is required
    scanf("%d",&NewStruct.MyShort);

    printf("MyInteger = %d\n",NewStruct.MyInteger);
    printf("MyLong = %ld\n",NewStruct.MyLong);
    printf("MyShort = %d\n",NewStruct.MyShort);
    printf("MyCharacter = %c\n",NewStruct.MyCharacter);
    printf("MyString = %s\n",NewStruct.MyString);
}

```

```

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

void main()
{
    int i,count;
    struct TwoNum
    {
        int Num1;
        long Num2;
    }
    Temp;

    // Once a struct has been declared, it becomes a type.
    // The type can be used in subsequent declarations
    // In subsequent declarations, the declaration of
    // the elements should be omitted.
    struct TwoNum TNArray[20];

    printf("Enter a list of numbers two per line\n");
    // &'s are required
    scanf("%d %d",&Temp.Num1,&Temp.Num2);
    count = 0;
    while (count < 20 && !feof(stdin))
    {
        // You can assign one struct to another
        // as long as they are of the same type
        TNArray[count] = Temp;
        count++;
        scanf("%d %d",&Temp.Num1,&Temp.Num2);
    }
    for (i=0 ; i<count ; i++)
    {
        // for an array of structures, the index comes first, then the
        // sub-element name.
        printf("item %d, num1 %d, num2 %d\n",i,TNArray[i].Num1,TNArray[i].Num2);
    }
}

```

```

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

void main()
{
    int i,count;
    struct TwoNum
    {
        int Num1;
        long Num2;
    }
    Temp;

    // You can declare arrays of structures
    // and pointers to structures
    struct TwoNum TNArray[20],*SPtr;

    printf("Enter a list of numbers two per line\n");
    scanf("%d %d",&Temp.Num1,&Temp.Num2);
    count = 0;
    while (count < 20 && !feof(stdin))
    {
        TNArray[count] = Temp;
        count++;
        scanf("%d %d",&Temp.Num1,&Temp.Num2);
    }

    // We will use pointers to access the array elements
    // We assign the array name to the pointer (arrays and
    // pointers are still the same)
    // We increment the pointer with ++
    // With pointers, Increment means NEXT ELEMENT
    for (i=0,SPtr=TNArray ; i<count ; i++,SPtr++)
    {
        // When you have a pointer to a struct you use the
        // arrow operator -> to access the elements of the struct.
        // SPtr->Num1 is equivalent to (*SPtr).Num1
        // *SPtr.Num1 IS NOT THE SAME AS (*SPtr).Num1
        printf("item %d, num1 %d, num2 %d\n",i,SPtr->Num1,SPtr->Num2);
    }
}

```

```

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

void main()
{
    // A structure definition can contain
    // another structure definition
    struct Line
    {
        struct Point
        {
            int x,y;
        }
        Beg,End;
    }
    MyLine;

    // The nested definition can be "pulled out"
    // and used separately
    struct Point MyPoint;

    printf("Enter the coordinates of point 1\n");
    scanf("%d %d",&MyPoint.x,&MyPoint.y);
    if (feof(stdin))
    {
        exit(100);
    }
    MyLine.Beg = MyPoint;
    printf("Enter the coordinates of point 2\n");
    scanf("%d %d",&MyPoint.x,&MyPoint.y);
    if (feof(stdin))
    {
        exit(100);
    }
    MyLine.End = MyPoint;
    MyPoint.x = (MyLine.Beg.x+MyLine.End.x)/2;
    MyPoint.y = (MyLine.Beg.y+MyLine.End.y)/2;
    printf("The midpoint is (%d,%d)\n",MyPoint.x,MyPoint.y);
}

```

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

void main()
{
    int i;

    // A structure can contain a pointer to
    // another structure of the same type.
    struct NameItem
    {
        struct NameItem *Next;
        char Name[20];
    }
    NameList[30];

    // You can turn the array into a linked list
    // using the following code.

    for (i=0; i<29 ; i++)
    {
        // Adding an integer to a pointer advances the
        // pointer that many items
        NameList[i].Next = NameList + i + 1;
    }
    NameList[29].Next = NULL;
}
```

```

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

void main()
{
    // If you have many structures of the same type,
    // You can simplify declarations with a typedef.

    // you declare a new type the same way as you
    // declare a variable, you just start the whole
    // thing with the keyword "typedef"
    typedef struct Point
    {
        int x,y;
    }
    POINT, *PPOINT;

    // POINT and PPOINT are NOT variables, they are new types.

    // The new type can be used in other typedefs
    // NOTE: Messing up a typedef usually generates thousands
    // of error messages
    typedef struct Line
    {
        POINT Beg,End;
    }
    LINE, *PLINE;

    // The following declares a point and a line.
    POINT MyPoint;
    LINE MyLine;

    // The following declares a pointer to a line
    PLINE LinePtr;

    // The following declares a pointer to a point
    PPOINT PointPtr;

    printf("Enter the coordinates of point 1\n");
    scanf("%d %d",&MyPoint.x,&MyPoint.y);
    if (feof(stdin))
    {
        exit(100);
    }
    MyLine.Beg = MyPoint;
    printf("Enter the coordinates of point 2\n");
    scanf("%d %d",&MyPoint.x,&MyPoint.y);
    if (feof(stdin))
    {
        exit(100);
    }
    MyLine.End = MyPoint;
    MyPoint.x = (MyLine.Beg.x+MyLine.End.x)/2;
    MyPoint.y = (MyLine.Beg.y+MyLine.End.y)/2;
    printf("The midpoint is (%d,%d)\n",MyPoint.x,MyPoint.y);
    // A couple of useless assignments:
    // To assign the address of a point to a pointer to a point,
    // use the & operator.
    PointPtr = &MyPoint;
    LinePtr = &MyLine;
}

```

```

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

void main()
{
    int count;

    // Processing a linked list
    struct NameItem
    {
        struct NameItem *Next;
        char Name[20];
    }
    NameList[30], *ListPtr;
    char TempStr[20];

    printf("Enter some words, one word per line\n");
    scanf("%s",TempStr);
    for (count=0; !feof(stdin) && count<30 ;)
    {
        // if this is not the start of the array make the
        // previous element point to this element
        if (count > 0)
        {
            NameList[count-1].Next = NameList + count;
        }
        // flag this item as the end of the list
        NameList[count].Next = NULL;
        strcpy(NameList[count].Name,TempStr);
        count++;
        scanf("%s",TempStr);
    }

    // process the linked list
    // This is a standard list-processing loop
    for (ListPtr = NameList ; ListPtr ; ListPtr = ListPtr->Next)
    {
        printf("%s\n",ListPtr->Name);
    }
}

```