Input Output Techniques

There are several functions that are used to take input and print output efficiently. Some of them are discussed here.

scanf:

It is the simplest of all. Most of the time, it is used to take inputs. Due to the problems of taking string as input, sometime gets() function is used instead of it.

Example 1:

*The important part of **scanf** function is its return value.*

```
a=scanf(“%d %d”,&b,&c);
```

a. The result will be 2 if two integer variables are given correctly as input.
b. The result will be 1 if first input is given correctly and second one is not given or given illegal input(such as string) or first one contains some illegal input. For example:

```
56 abc
53ab 34
45 ^Z
```

c. The result will be 0 if first input is given incorrectly.
d. The result will be -1 if end of file (EOF) is found without getting any input.

Example 2:

Suppose the problem states,

“There will be two integers in every line of the input and terminated by EOF (end of file).”

Then you should write

```
int a,b;
while(scanf(“%d %d”,&a,&b)==2) //two inputs are taken correctly
{
	// rest of the work
}
```

Example 3:

If the problem states,

“There will be three numbers (not necessarily integers) in every line of input and terminated by EOF (end of file).”

Then you should write

```
double a,b,c;
while(scanf(“%lf %lf %lf”,&a,&b,&c)==3) //three inputs are taken correctly
{
                  // rest of the work
}
```

Example 4:

If the problem states,

“There will be 3 integers and one name (not consisting of any space) and the input will be terminated if the given first integer is 0”

Then you should write

```
int a,b,c;
char name[100];
while(scanf(“%d %d %d %s”,&a,&b,&c,name)==4)
{
      if(a==0)
            break;
      // rest of the work
}
```

Example 5:

What would be the output of the following code?

```

char name[100];                 input is given as:
scanf(“%s”, name);              String input
printf(“%s”, name);
```

The output will be “String” but not “String input”. Because scanf() function takes string input without space. To take the whole line you can use gets() function or modified scanf() function. For example:

```
scanf(“%[^\n]”,name);
```

This will take the input correctly.

gets:

Suppose you have to take a line which contains a name, or names but can have spaces then you have to use gets.

Example 6:

Return value of gets() function:

a. gets() function returns the pointer of the argument on success.

b. On error, gets() function returns -1 or NULL. The error will occur because of end of file (EOF)
   For example:

```
char line[1001], *pt;
pt=gets(line);
printf("%s\n",pt);
```

For the correct input, pt points line[0] and prints the output that is given as input. For the incorrect input (such as EOF), pt will be -1 and prints "**(null)"** as output.

Example 7:

Suppose the problem states,

> "There will be one or more names in a single line and will be terminated by end of file"

Then you should write

```
char line[1001];
while(gets(line))
{
      // rest of the work
}
```

sscanf:

sscanf() function works as scanf() function. But instead of standard input, sscanf() function takes input from a string.

Example 8:

Suppose there are three integers in a string (array of characters), how will you take the three integers in separate variables?

See the example below

```
int a,b,c;
char line[1001];
//suppose line[] contains three integers
sscanf(line,"%d %d %d",&a,&b,&c);
```

sscanf is similar to scanf, except that you have to include the array name. Remember that scanf takes input from the standard input, but sscanf takes input from a string. And their behavior is same.

Example 9:

Suppose the problem states

“The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.
There can be one or more line(s) for a single input set, each line will contain two integers.”

Sample:

```
3

1 2
10 19
21 30
45 67

786 126
1099 122
11 12
11 989
17354 1221
112 12112

121 121
128 76
```

How to take input for this problem? Suppose a[100][2] is a two dimensional array, and you want to store all the integers in this array.

```
int TotalCase,caseno,a[101][2],n;
char line[1001];

scanf(“%d\n”,&TotalCase);
for(caseno=1;caseno<=TotalCase;caseno++)
{
      n=0;
      while(gets(line))
      {
            if(line[0]==’\0’) // Blank line
                  break;
            sscanf(line,“%d %d”,&a[n][0],&a[n][1]);
            n++;
      }
      // rest of the work
}
```

Example 10:

Suppose the problem states

"Each input set contains some names, they can be given in a single or multiple line(s), ans two consecutive sets will be separated by a single line."

Sample:

```
There are only three sets
But this one is quite tricky
And the first set ends here

Now the second case starts you can simply see that
there is a blank line between the cases

third case is same as second case

This is the fourth case and there is no blank line
after this case since it is the last case, so this case is
different
```

Can you take the input?

Hint: Suppose line[101][101] is a two dimensional character array. Store all the lines in this array. Remember that you don't have to store all the cases. Store one set, process it, then take the next one(forget the previous set). So, you can use line[][] for every set. Try it.

Combination of gets and scanf function

Example 11:

If the problem states,

"In each line there will be 2 integers and one name (consisting space) and the input will be terminated if the given first integer is 0. All variables are separated by exactly one space"

Sample Input:

```
20 30 Abu Ahmed Ferdaus
15 25 Chowdhury Farhan Ahmed
```

Then you should write

```
int a,b;
char name[100];
while(scanf("%d %d",&a,&b)==2)
{
      if(a==0)
            break;
      getchar();  //required to remove the space
      gets(name);
      // rest of the work
}
```

Example 12:

If the problem states,

“There will be 2 integers in a line and one name (consisting space) on the next line and the input will be terminated by end of file.

Sample Input:

```
20 30
Abu Ahmed Ferdaus
15 25
Chowdhury Farhan Ahmed
```

Then you should write

```
int a,b;
char name[100];
while(scanf(“%d %d”,&a,&b)==2)
{
      getchar();  //why is it required?? (Very important)
      gets(name);
      // rest of the work
}
```

Printf:

Most of the time, printf() function is used to print the output.

Example 13:

You should know the following formats of printf function:

```
int a=10, b=20;
double c=3.432343;
char v[100];
printf(“%6d %-6d\n”,a,b);
printf(“%4.3lf\n”,c);
printf(“%06d\n”,a,b);
sprintf(v, “%d %d”,a,b);
```

Example 14:

If the problem states,

“There will be a new line between two test cases”

Sample output:

```
ResultLine 1

ResultLine 2

ResultLine 3
```

“There will be a new line after each test case.”

```
ResultLine 1
ResultLine 2

ResultLine 3
```

freopen:

Suppose for the above case you have just written a code. But how to test your code? You can say, “I will run the code and type the input lines as described”. Yes your solution is correct but isnt it slower? And you can have errors.

What if you can save the input in a file, and run your code for that file. Then you only need to write the input once. Look at the code

```
int main()
{
      freopen(“Input.in”,”r”,stdin);

      // Your code for taking input
      return 0;
}
```

Then make a file named “Input.in” in the same folder (where your code is). And type the input in that file. Then if you run your code you will see that you’re code returns the result automatically (If you code is correct, too ☺). And if the file doesn’t exist you can see some abnormal behavior. So, be sure that the file exists.

Now what if your input file is in a different folder? Lets consider that the input file is in a folder named “input” which is in C drive and the file name is “abc.txt”. Then you should write

```
int main()
{
      freopen(“c:\\input\\abc.txt”,”r”,stdin);

      // Your code
      return 0;
}
```

Now you want to store your output in a file named “Output.out”.

```
int main()
{
      freopen(“Output.out”,”w”,stdout);

      // Your code for taking input
      return 0;
}
```

In the code above there is no input file, so you have to give the input by hand, but your output will be stored in a file named “Output.out” which will be modified or created (if there is no such file) in the same location.

Now suppose you want to take input from a file wich is located in C drive and the name is “jhg.txt”, and you want to store the output in a file named “uyt.txt”. And you are sure that “jhg,txt” exists. Then you can write

```
int main()
{
	freopen(“c:\\jhg.txt”,”r”,stdin);
	freopen(“d:\\uyt.txt”,”w”,stdout);

	// Your code
	return 0;
}
```

Now consider a task. Can you write a code, the output of your code will be the code itself?

Hint: can you open your code using freopen?

And remember that don’t submit a problem in uva which has freopen. Then you will get restricted function. So, before submitting make sure that you have deleted/disabled freopen.

Output:

To output something you can use printf.

See the code below.

```
int main()
{
	int aa,bb;
	double dd,ee;
	long double ll;
	char cc;
	char name[100];
	unsigned int uu;

	long long qq; // for uva
	__int64 qq; // for vc

	aa=2;
	bb=101;
	printf(“%d %d\n”,aa,bb);
	// for int the identifier is %d

	dd=1.06;
	ee=78;
	printf(“%lf %lf\n”,dd,ee);
	// for double the identifier is %lf
```

```
        cc='m';
        printf("%c\n",cc);
        // for char the identifier is %c

        name[0]='a';
        name[1]='b';
        name[2]='\0';
        printf("%s\n",name);
        // for string the identifier is %s

        ll=-89767;
        printf("%Lf\n",ll);
        // for long double the identifier is %Lf


        qq=7186212;
        printf("%lld\n",qq); // for uva
        // for long long the identifier is %lld // for uva
        qq=7186212;
        printf("%I64d\n",qq); // for vc
        // for __int64 the identifier is %I64d // for vc


        uu=18726;
        printf("%u\n",uu);
        // for unsigned int the identifier is %u

        // to print multiple thigs you can use

        printf("%d %lf %s %Lf\n",aa,dd,name,ll);

        return 0;
}
```

And remember that if you are using vc then use '__int64' for 64 bit integer. But to submit in uva you have to use 'long long'. And their identifiers are different. So, before submitting make sure that you have changed the name(__int64 to long long) and the identifier correctly. Or you can use predecessor. Otherwise you will get compilation error.

*Created by: **Jane Alam Jan (10th Batch)***