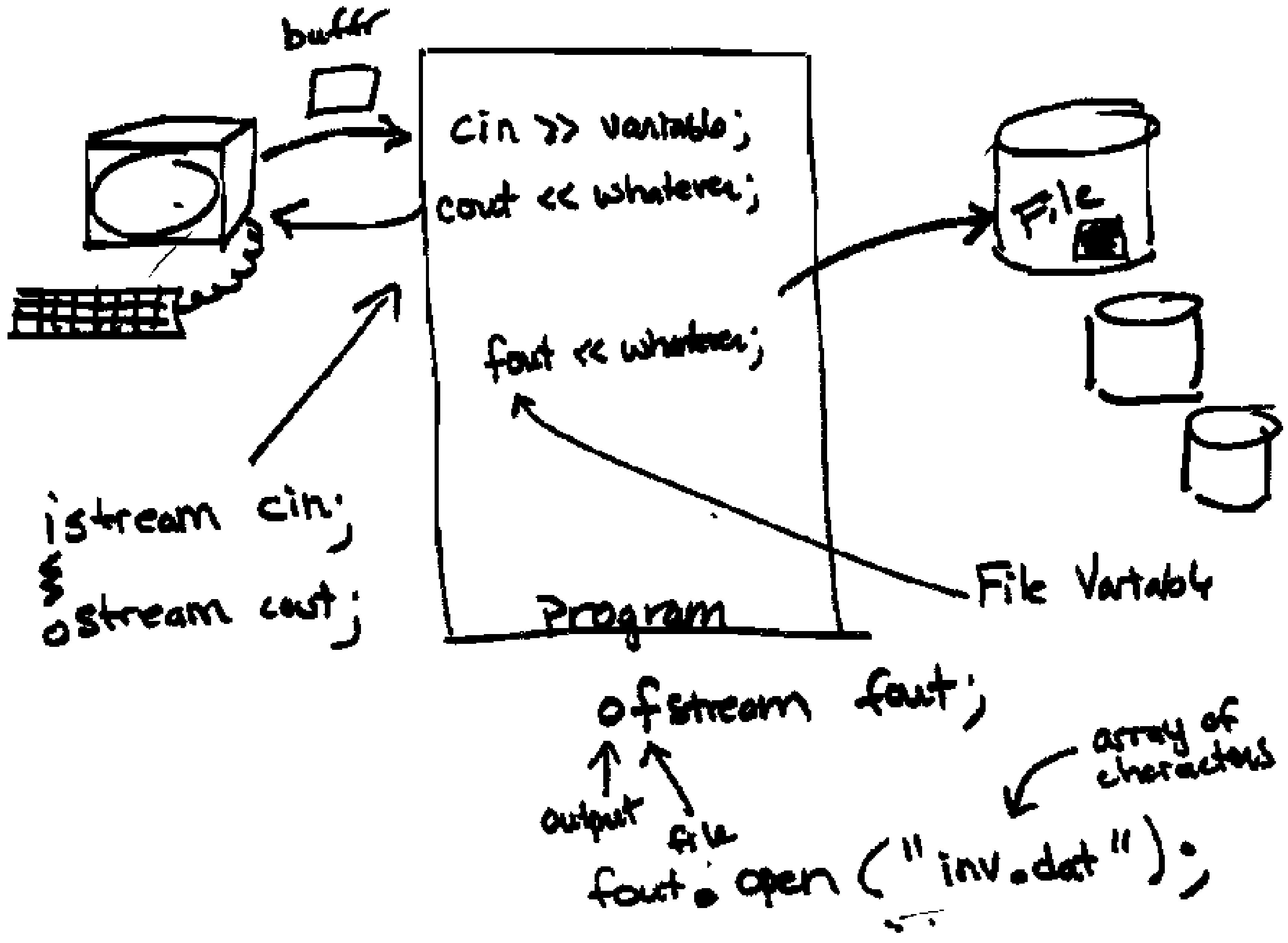
External File I/O



To wipe out file.... to write to a file: #include <fstream>

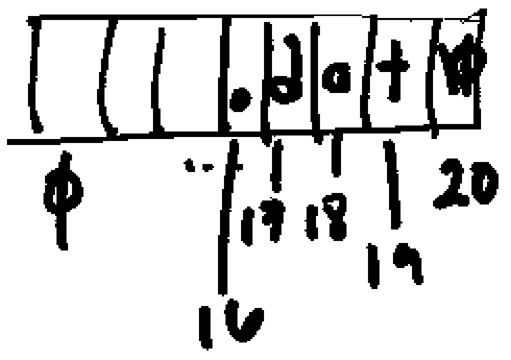
1.
```
ofstream fout;
fout.open( "inv.dat");
```

2.
```
char filename [21];
cin >> filename; cin.ignore();
strcat(filename, ".dat");
fout.open ( filename);
```

cstring library

3.

Safer



```
char filename [21];   int length;
                        constant
   :
cin.width(21);
cin >> filename;  //Max 20 character
cin.ignore(100, '\n');

length = strlen(filename);
if (length <= 16)
      strcat(filename, ".dat");
```

4.

```
cin.get(filename, 21, '\n');
cin.ignore(100, '\n');
```

Allows filenames to have blanks (spaces)

To Keep the File in tact: 3.

```
fout. open( filename, ios::app);

// fout is True (non zero) if
      open was successful.
//  fout is False (zero) otherwise
  if ( fout) //was open successful
  {
        :
        :
        : fout << variable << '\n';
     fout.close(); fout.clear();
  }
```

//write a function to output
// information (character) to a file :

```
void ~~[illegible]~~ output( ofstream & fileout, char array
                                                        )
  {     fileout ~~&~~ << array << '\n';
  }
```

call

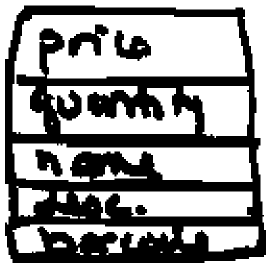
```
→ output(fout, "data");
→ output(fout2, "data");
```

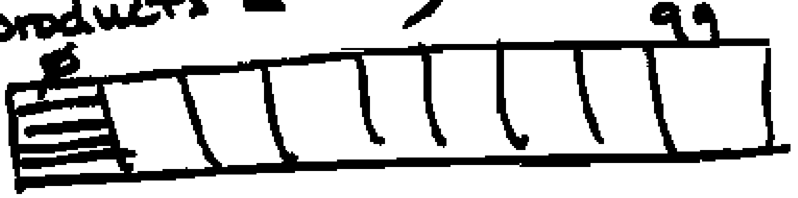
Better

Before MAIN

```
#include ....
        const int SIZE_NAME = 21;
        const int SIZE_DESC = 131;
        const int SIZE_BAR = 13;
        struct Inventory
        {
              float price;
              int   quantity;
              char  name [SIZE_NAME];
              char  description [SIZE_DESC];
              char  barcode [SIZE_BAR];
        };
```

```
int main()
{
      Inventory product;          data type
                                  variable
      Inventory all_products [100];
      int num_prod=0;
```

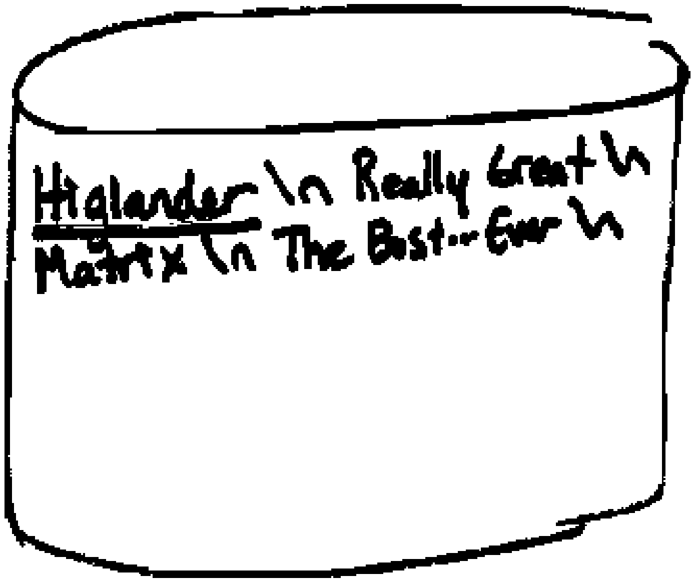




```
void  save_inventory (Inventory & product,
                      char filename [])
      [illegible]ve
{
      ofstream fout;
      fout.open (filename , ios::app);
      if ( fout)
      {
          fout << product.price << ':'
               << produd.quantity << ':'
               << product.name << '\n'
               << product.description << '\n'
               << product.barcode << '\n';
      [illegible] fout.close();
      }
}
```

Input from a File

```
ifstream   fin;
fin.open ("inv.dat");  // fin.open(filename);
                                    ↑
                                 array of
                                 characters
if (fin)  // most likely there is
          // a file to read from
{
    fin.get(movie, 81, '\n');
              ↑      ↑
           array  sizeofarray
    while (fin && !fin.eof())
    {
        fin.ignore();
        fin.get(comments, 131, '\n');
        fin.ignore();
```



```
        cout << movie <<'\t' << comments <<'\n';
        fin.get (movie, 81,'\n');
    }
    fin.close();
    fin.clear();
```

fin >> variable;

<u>fin.eof()</u>

True - if the <u>previous</u> input
operation failed

False - otherwise

```
void  get_inventory (Inventory all[], int & num,
                                        char filename[])
{
      ifstream  filein;
      filein.open(filename);
      if (filein)
      {
            filein >> all[num].price;
            while (filein && !filein.eof())
            {
                  filein.ignore();  // ':'
                  filein >> all[num].quantity;
                  filein.ignore();  // ':'
                  filein.get(all[num].name,
                                  SIZE_NAME, '\n');
                  filein.ignore();  // '\n's
                  filein.get(all[num].description,
                                  SIZE_DESC, '\n');
                  filein.ignore();
                  filein.get(all[num].barcode,
                                  SIZE_BAR, '\n');
```

```
                    ++num;
                    filein >>all[num].price;
                }
                filein.close();
            }
        }
```