Filing With Objects

Formatted File I/O

- In formatted I/O, numbers are stored on disk as a series of characters (just as we have been doing until now).
- Thus 6.02, rather than being stored as a 4-byte type float or an 8-byte type double, is stored as the characters ‘6’, ‘.’, ‘0’, and ‘2’.
- This can be inefficient for numbers with many digits, but it’s appropriate in many situations and easy to implement.
- Characters and strings are stored more or less normally.

Binary I/O

- In binary I/O numbers are stored as they are in the computer’s RAM memory, rather than as strings of characters.
- In binary I/O an int is stored in 4 bytes, whereas its text version might be “12345”, requiring 5 bytes.
- Similarly, a float is always stored in 4 bytes, while its formatted version might be “6.02314e13”, requiring 10 bytes

Binary I/O

- You can write a few numbers to disk using formatted I/O, but if you're storing a large amount of numerical data it's more efficient to use binary I/O, in which numbers are stored as they are in the computer's RAM memory, rather than as strings of characters.
- In binary I/O an int is stored in 4 bytes, whereas its text version might be "12345", requiring 5 bytes. Similarly, a float is always stored in 4 bytes, while its formatted version might be "6.02314e13", requiring 10 bytes.

Binary I/O

- We use two new functions:
  - write(), a member of ofstream; and
  - read(), a member of ifstream.
- These functions think about data in terms of bytes.
- They don't care how the data is formatted, they simply transfer a buffer full of bytes from and to a disk file.
- The parameters to write() and read() are the address of the data buffer and its length.
- The address must be cast, using reinterpret_cast, to type char*, and the length is the length in bytes (characters), not the number of data items in the buffer.

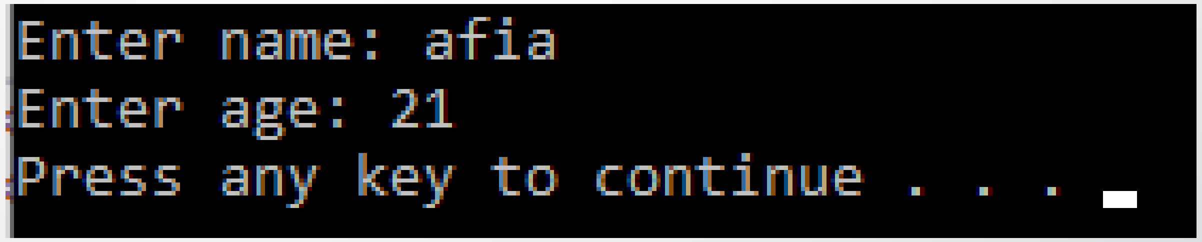
```
WriteFileObject.write( reinterpret_cast<char*>(&object), MAX*sizeof(object) );
ReadFileObject.read( reinterpret_cast<char*>(&object), MAX*sizeof(object) );
```

Writing an Object to Disk

- #include <fstream> //for file functions
- #include <iostream>
- #include <string>
- using namespace std;
- class person //class of persons
- {
- protected:
- char name[80]; //person's name
- short age; //person's age
- public:
- void getData() //get person's data
- {
- cout << "Enter name: "; cin >> name;
- cout << "Enter age: "; cin >> age;
- }
- };

Writing an Object to Disk

- int main()
- {
- person pers; //create a person
- pers.getData(); //get data for person
- //create ofstream object
- ofstream outfile("PERSON.DAT", ios::binary);
- //write to it
- outfile.write(reinterpret_cast<char*>(&pers), sizeof(pers));
- outfile. close();
- system("pause");
- return o;
- }



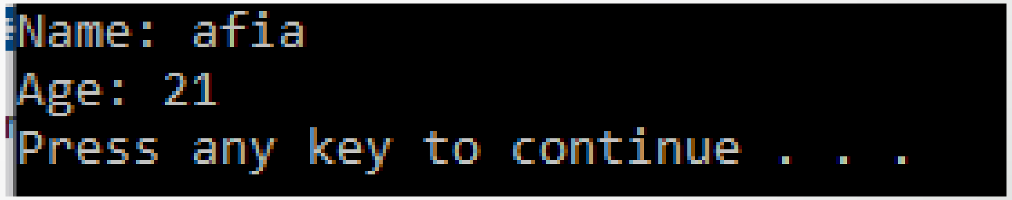
```
Enter name: afia
Enter age: 21
Press any key to continue . . .
```

Reading an Object from Disk

- #include <fstream> //for file streams
- #include <iostream>
- using namespace std;
- class person //class of persons
- {
- protected:
- char name[80]; //person's name
- short age; //person's age
- public:
- void showData() //display person's data
- {
- cout << "Name: " << name << endl;
- cout << "Age: " << age << endl;
- }
- };

Reading an Object from Disk

- int main()
- {
- person pers; //create person variable
- ifstream infile("PERSON.DAT", ios::binary); //create stream
- //read stream
- infile.read(reinterpret_cast<char*>(&pers), sizeof(pers));
- pers.showData(); //display person
- system("pause");
- return 0;
- }


```
Name: afia
Age: 21
Press any key to continue . . .
```

I/O with Multiple Objects

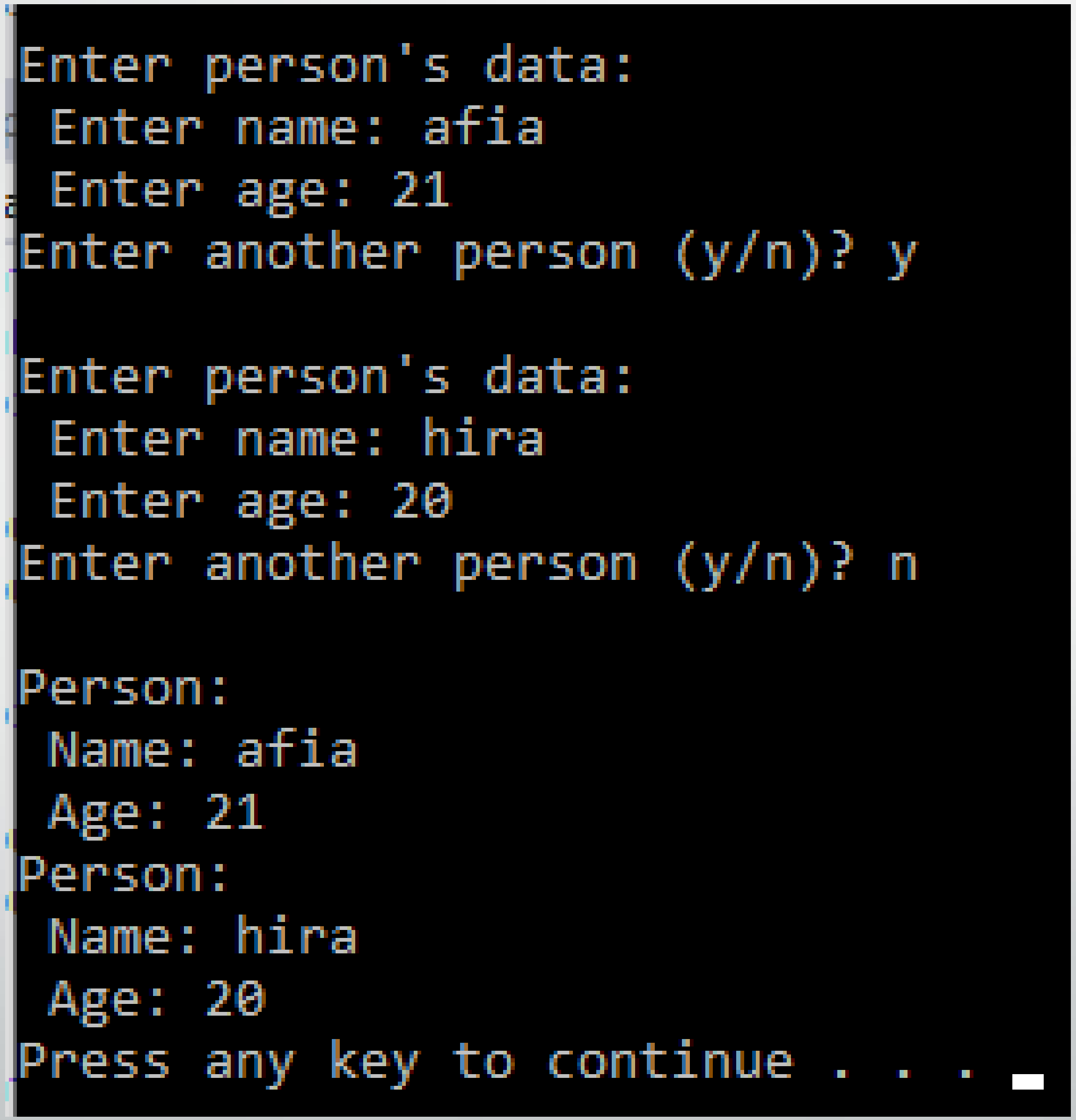
- #include <fstream> //for file streams
- #include <iostream>
- using namespace std;
- class person //class of persons
- {
- protected:
- char name[80]; //person's name
- int age; //person's age
- public:
- void getData() //get person's data
- {
- cout << "\n Enter name: "; cin >> name;
- cout << " Enter age: "; cin >> age;
- }
- void showData() //display person's data
- {
- cout << "\n Name: " << name;
- cout << "\n Age: " << age;
- }
- };

I/O with Multiple Objects

- int main()
- {
- char ch;
- person pers; //create person object
- fstream file; //create input/output file
- //open for append
- file.open("GROUP.DAT", ios::app | ios::out |
- ios::in | ios::binary);
- do //data from user to file
- {
- cout << "\nEnter person’s data:";
- pers.getData(); //get one person’s data
- //write to file
- file.write(reinterpret_cast<char*>(&pers), sizeof(pers));
- cout << "Enter another person (y/n)? ";
- cin >> ch;
- } while(ch=='y'); //quit on ‘n’
- file.close();//close file

I/O with Multiple Objects

- file.open("GROUP.DAT", ios::in | ios::binary);
- //read first person
- file.read(reinterpret_cast<char*>(&pers), sizeof(pers));
- while(file.read(reinterpret_cast<char*>(&pers), sizeof(pers))
- {
- pers.showData(); //read another person
- }
- system("pause");
- return 0;
- }



```
Enter person's data:
 Enter name: afia
 Enter age: 21
Enter another person (y/n)? y

Enter person's data:
 Enter name: hira
 Enter age: 20
Enter another person (y/n)? n

Person:
 Name: afia
 Age: 21
Person:
 Name: hira
 Age: 20
Press any key to continue . . .
```