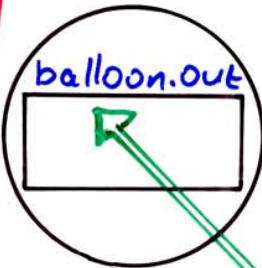
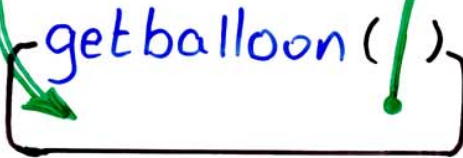
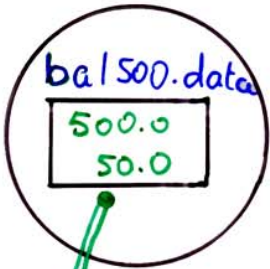
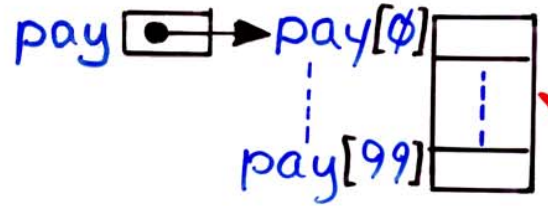
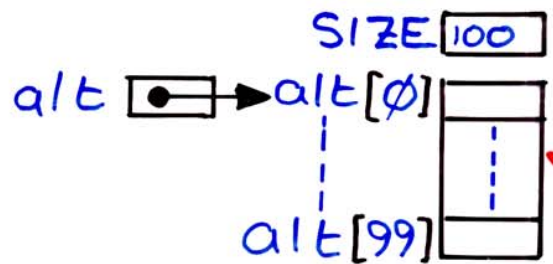
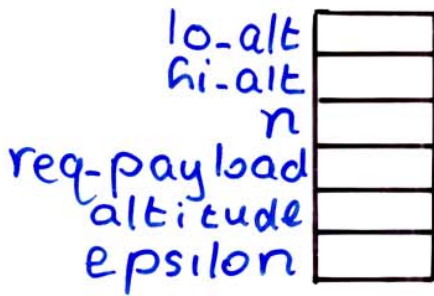


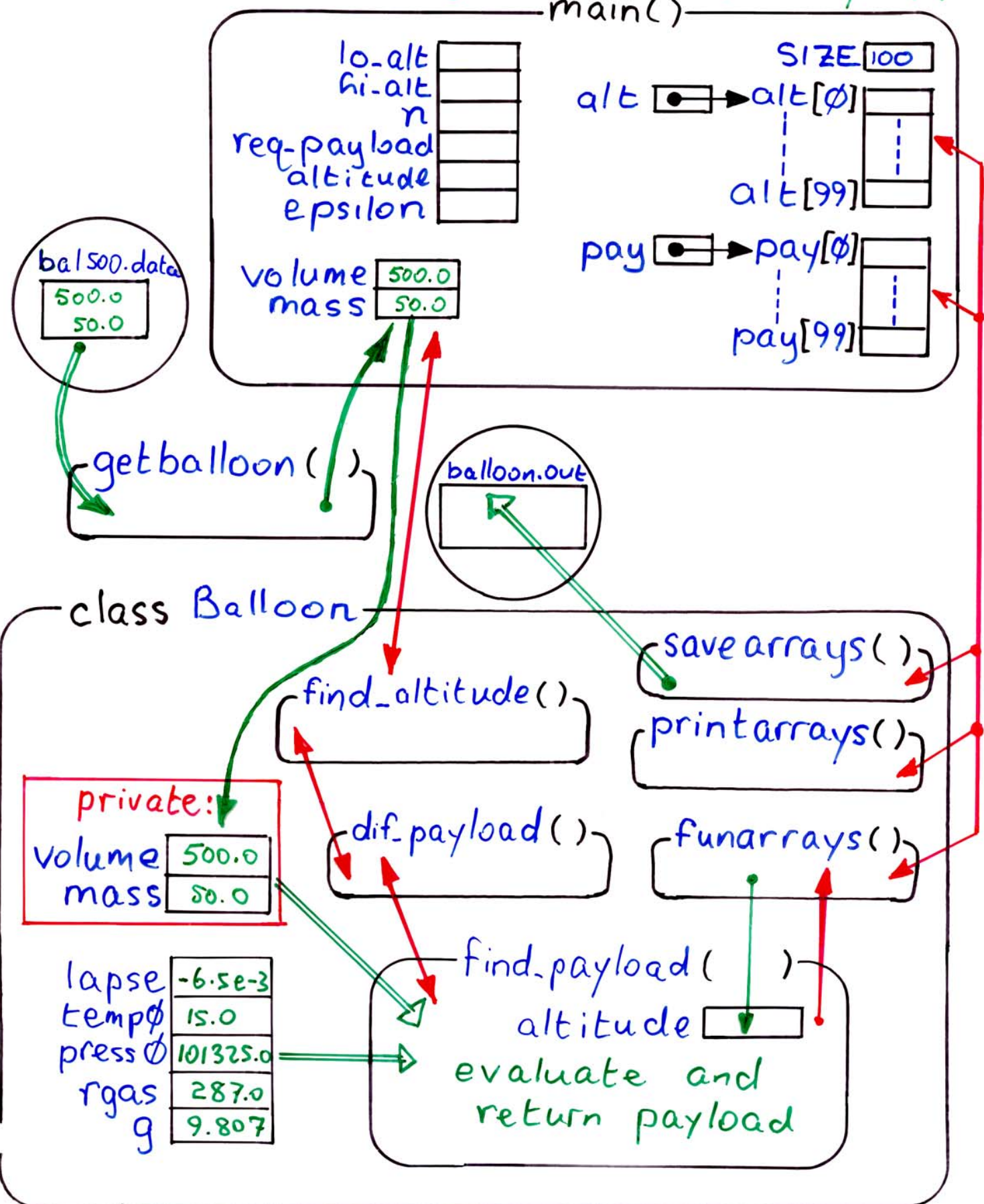
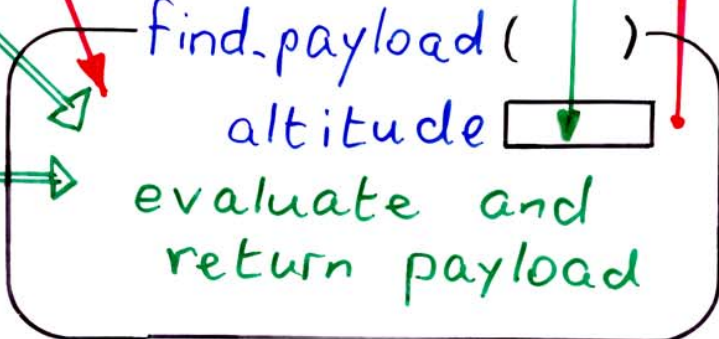
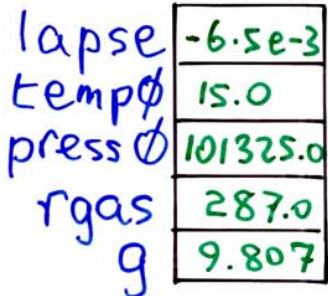
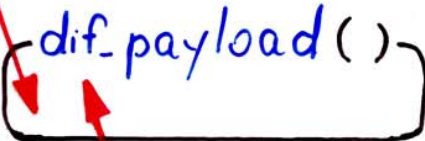
balarray.h

balarray.cpp

main()



class Balloon



```
class Balloon { // copied from balfile.h
    void funarrays (float lo_alt, float hi_alt,
                  float alt[], float pay[], int n);
    void printarrays (float alt[], float pay[], int n);
    void savearrays (float alt[], float pay[], int n);
};
void getballoon (float & volume, float & mass);
int const SIZE = 100;
```

```
int main (void) // copied from balfile.cpp
{
    float alt[SIZE], pay[SIZE];
    float lo_alt, hi_alt, dalt;
    int n;
    cout << "enter low & high bounds of altitude\n";
    cin >> lo_alt, hi_alt;
    cout << "values are: " << lo_alt << hi_alt << endl;
    cout << "enter number of points\n";
    cin >> n;
```

```
myballoon.funarrays (lo_alt, hi_alt, alt, pay, n);
```

```
myballoon.printarrays (alt, pay, n);
myballoon.savearrays (alt, pay, n);
```

```
return 0;
```

```
}
```

```
void Balloon::funarrays(float lo_alt, float hi_alt,
                       float alt[], float pay[], int n)
{
    float dalt = (hi_alt - lo_alt) / (n - 1);
    for (int i = 0; i < n; i++) {
        alt[i] = lo_alt + i * dalt;
        pay[i] = find_payload(alt[i]);
    }
}
```

```
void Balloon::printarrays(float alt[],
                          float pay[], int n)
{
    cout << "altitude(m) payload(kg) \n";
    for (int i = 0; i < n; i++) {
        cout << alt[i] << pay[i] << endl;
    }
}
```

```
void Balloon::savearrays(float alt[],
                          float pay[], int n)
{
    char const tab = '\t';
    ofstream outfile("balloon.out");
    assert(outfile);
    for (int i = 0; i < n; i++)
        outfile << alt[i] << tab << pay[i] << endl;
    cout << n << " rows data to file \n";
    outfile.close();
}
```