Department of Computer Science

University of Engineering and Technology, Lahore

Structures and Linked Lists

Suppose that you want to process the following data related to a single laptop

1. Manufacturer name
2. Model
3. Processor
4. RAM in GBs
5. Hard Disk size
6. Generation
7. Price

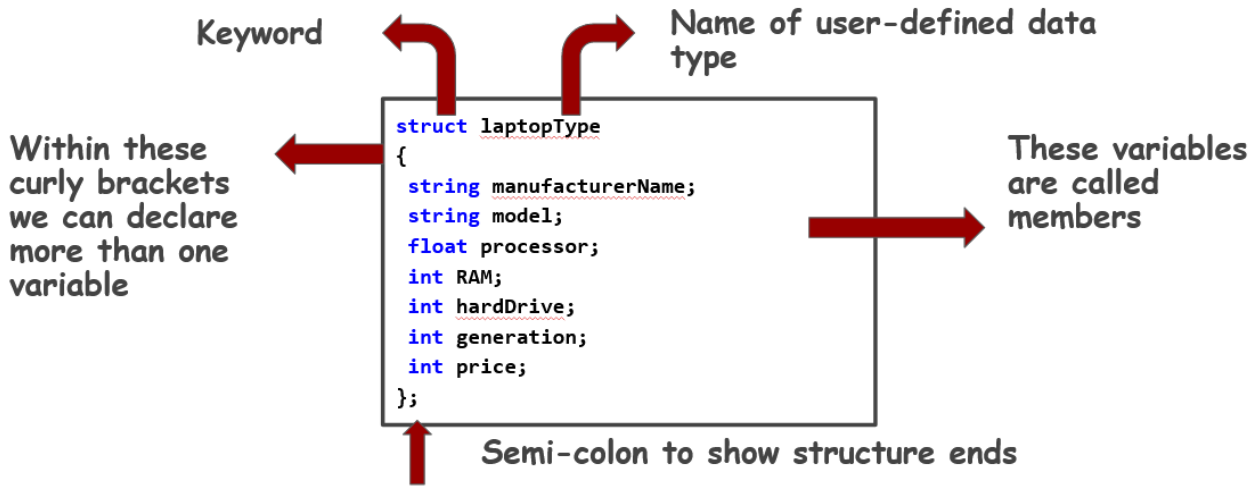
All this information is related to a single laptop, but the data types of these are different.
We can store such information in Parallel arrays. But This is not an efficient solution, because the information is scattered in different arrays, which is only linked through indexes.

C++ provides us an opportunity to define our own data type according to the requirement. This Data-type is called Structure.

Structure:
A collection of a fixed number of components in which the components are accessed by a name. The components may be of different types.

Example 1:

Define a `struct laptopType` to store the following data about a laptop: Manufacturer (`string`), model type (`string`), processor in gigahertz (`float`), ram (`int`) in GB, hard drive size (`int`) in GB, generation (`int`), and the price (`double`)



Example 2:
Assume the definition of laptopType. Declare a laptopType variable and write statements to store the following information:
Manufacturer—Computer Corporation,
model—HP,
processor speed—2.3,
RAM—12 GB,
hard drive size—500 GB,
Generation—5,
and the price—70000.

Solution

```
#include<iostream>
using namespace std;

struct laptopType
{
 string manufacturerName;
 string model;
 float processor;
 int RAM;
 int hardDrive;
 int generation;
 int price;
};
```

```
main()
{
    laptopType myLaptop;
    myLaptop.manufacturerName = "Computer Corporation";
    myLaptop.model = "HP";
    myLaptop.processor = 2.3;
    myLaptop.RAM = 12;
    myLaptop.hardDrive = 500;
    myLaptop.generation = 5;
    myLaptop.price = 70000;
}
```

Example 3:
Assume the definition of Example 1, which defines the struct laptopType. Write a function that prompts the user to input data about a laptop.

Solution

```
#include<iostream>
using namespace std;

struct laptopType
{
 string manufacturerName;
 string model;
 float processor;
 int RAM;
 int hardDrive;
 int generation;
 int price;
};

void takeInput()
{
    laptopType myLaptop;
    cout << "Enter Manufacturer Name: ";
    cin >> myLaptop.manufacturerName;
    cout << "Enter Model: ";
```

```
    cin >> myLaptop.model;
    cout << "Enter Processor Speed: ";
    cin >> myLaptop.processor;
    cout << "Enter RAM: ";
    cin >> myLaptop.RAM;
    cout << "Enter Hard Drive Size: ";
    cin >> myLaptop.hardDrive;
    cout << "Enter Generation: ";
    cin >> myLaptop.generation;
    cout << "Enter Price: ";
    cin >> myLaptop.price;
}
main()
{
    takeInput();
}
```

Example 4:
Now, Write a function that prompts the user to input data about a laptop and then return that structure laptopType to the main function.

Solution

```
#include<iostream>
using namespace std;

struct laptopType
{
 string manufacturerName;
 string model;
 float processor;
 int RAM;
 int hardDrive;
 int generation;
 int price;
};

laptopType takeInput()
{
```

```
    laptopType myLaptop;
    cout << "Enter Manufacturer Name: ";
    cin >> myLaptop.manufacturerName;
    cout << "Enter Model: ";
    cin >> myLaptop.model;
    cout << "Enter Processor Speed: ";
    cin >> myLaptop.processor;
    cout << "Enter RAM: ";
    cin >> myLaptop.RAM;
    cout << "Enter Hard Drive Size: ";
    cin >> myLaptop.hardDrive;
    cout << "Enter Generation: ";
    cin >> myLaptop.generation;
    cout << "Enter Price: ";
    cin >> myLaptop.price;
    return myLaptop;
}
main()
{
    laptopType l1 = takeInput();
}
```

Example 4:
Now, Write a function that outputs the data of a laptopType structure on the console.

Solution

```
#include<iostream>
using namespace std;

struct laptopType
{
 string manufacturerName;
 string model;
 float processor;
 int RAM;
 int hardDrive;
 int generation;
 int price;
```

```
};
laptopType takeInput()
{
    laptopType myLaptop;
    cout << "Enter Manufacturer Name: ";
    cin >> myLaptop.manufacturerName;
    cout << "Enter Model: ";
    cin >> myLaptop.model;
    cout << "Enter Processor Speed: ";
    cin >> myLaptop.processor;
    cout << "Enter RAM: ";
    cin >> myLaptop.RAM;
    cout << "Enter Hard Drive Size: ";
    cin >> myLaptop.hardDrive;
    cout << "Enter Generation: ";
    cin >> myLaptop.generation;
    cout << "Enter Price: ";
    cin >> myLaptop.price;
    return myLaptop;
}

void displayOutput(laptopType l)
{
    cout << "Manufacturer Name: " << l.manufacturerName << endl;
    cout << "Model: " << l.model << endl;
    cout << "Processor: " << l.processor << endl;
    cout << "RAM: " << l.RAM << endl;
    cout << "HardDrive Size: " << l.hardDrive << endl;
    cout << "Generation: " << l.generation << endl;
    cout << "Price: " << l.price;
}
main()
{
    laptopType l1 = takeInput();
    displayOutput(l1);
}
```

Output

```
Enter Manufacturer Name: Comp_corporation
Enter Model: HP
Enter Processor Speed: 2.3
Enter RAM: 12
Enter Hard Drive Size: 500
Enter Generation: 5
Enter Price: 70000
Manufacturer Name: Comp_corporation
Model: HP
Processor: 2.3
RAM: 12
HardDrive Size: 500
Generation: 5
Price: 70000
```

Example 5:
Now make an array of laptopType that can store 5 laptops data.

Solution

```
#include<iostream>
using namespace std;

struct laptopType
{
 string manufacturerName;
 string model;
 float processor;
 int RAM;
 int hardDrive;
 int generation;
 int price;
};
laptopType takeInput()
{
    laptopType myLaptop;
    cout << "Enter Manufacturer Name: ";
    cin >> myLaptop.manufacturerName;
    cout << "Enter Model: ";
    cin >> myLaptop.model;
```

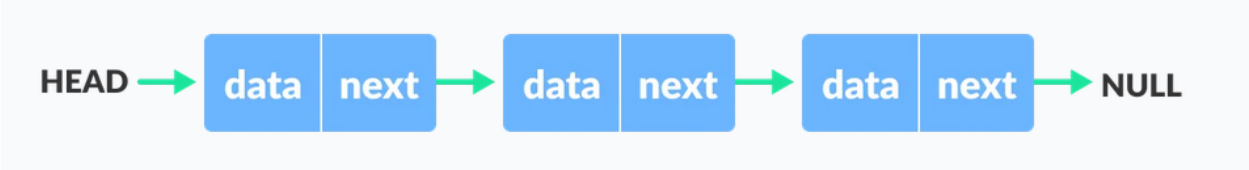
```
    cout << "Enter Processor Speed: ";
    cin >> myLaptop.processor;
    cout << "Enter RAM: ";
    cin >> myLaptop.RAM;
    cout << "Enter Hard Drive Size: ";
    cin >> myLaptop.hardDrive;
    cout << "Enter Generation: ";
    cin >> myLaptop.generation;
    cout << "Enter Price: ";
    cin >> myLaptop.price;
    return myLaptop;
}

void displayOutput(laptopType l)
{
    cout << "Manufacturer Name: " << l.manufacturerName << endl;
    cout << "Model: " << l.model << endl;
    cout << "Processor: " << l.processor << endl;
    cout << "RAM: " << l.RAM << endl;
    cout << "HardDrive Size: " << l.hardDrive << endl;
    cout << "Generation: " << l.generation << endl;
    cout << "Price: " << l.price << endl;
}
main()
{
    laptopType l[5];
    for(int x=0; x<5; x++)
    {
        l[x] = takeInput();
    }
    for (int i = 0; i < 5; i++)
    {
        displayOutput(l[i]);
    }
}
```

Previously, we have used an array of structure to store the information of 5 Laptops.
In this we had specify the size of the array before compilation i.e. `laptopType l[5];`

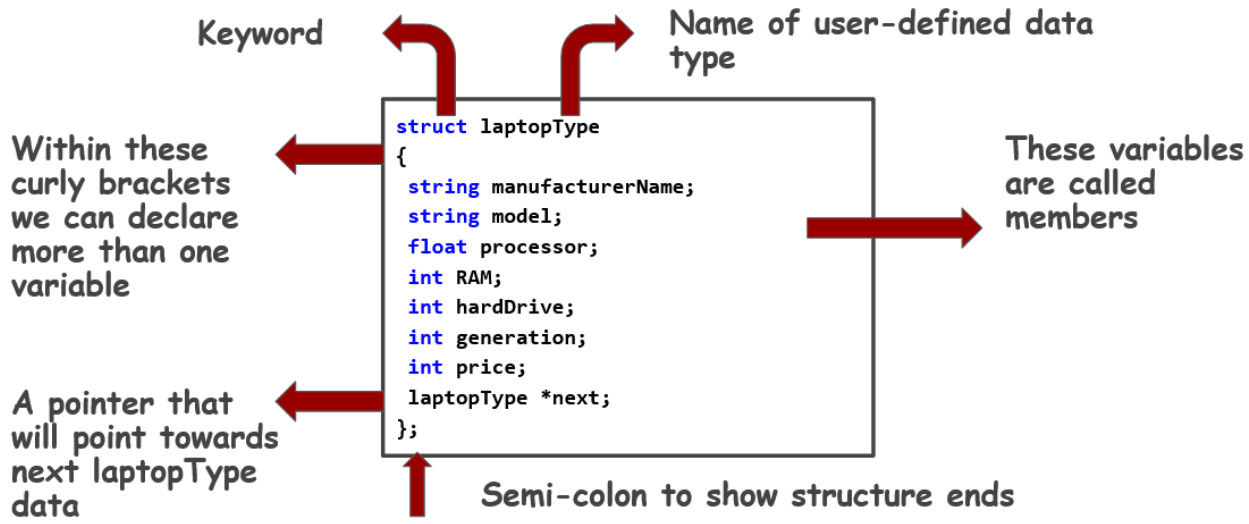
There is a way, in which we do not have to specify the size at compilation time, so, we could add the records dynamically as we need at the execution time.
This is done through link lists.

Linked List: A linked list is a linear data structure that includes a series of connected nodes. Here, each node stores the data and the address of the next node. For example,



You have to start somewhere, so we give the address of the first node a special name called HEAD. Also, the last node in the linked list can be identified because its next portion points to NULL.

Example 5:
Now let's add a pointer as a member in the laptopType structure.



Example 6:
First of all, make a pointer that will point towards the start of the Linked List.
Initially, there is no laptop record therefore, the starting pointer (let's call it head) will point towards NULL.

Solution
`#include<iostream>` `using namespace std;`

```
struct laptopType
{
 string manufacturerName;
 string model;
 float processor;
 int RAM;
 int hardDrive;
 int generation;
 int price;
 laptopType *next;
};

laptopType *head;
head = NULL;
```

The Dot(.) operator is used to normally access members of a structure.
The Arrow(->) operator exists to access the members of the structure using pointers.

Example 7:
Write a function that will take input of the data related to the laptop and then store it into the laptopType structure dynamically using new keyword.

Solution

```
#include <iostream>
using namespace std;

struct laptopType
{
    string manufacturerName;
    string model;
    float processor;
    int RAM;
    int hardDrive;
    int generation;
    int price;
```

```
    laptopType *next;
};

laptopType *head = NULL;

void takeInput();
laptopType *getLastRecord(laptopType *temp);
void addRecord(string manu, string mod, float pro, int ram, int hard,
int gene, int pri);

void takeInput()
{
    string manu;
    string model;
    float pro;
    int ram, hard, gene, price;
    cout << "Enter Manufacturer Name: ";
    cin >> manu;
    cout << "Enter Model: ";
    cin >> model;
    cout << "Enter Processor Speed: ";
    cin >> pro;
    cout << "Enter RAM: ";
    cin >> ram;
    cout << "Enter Hard Drive Size: ";
    cin >> hard;
    cout << "Enter Generation: ";
    cin >> gene;
    cout << "Enter Price: ";
    cin >> price;
    addRecord(manu, model, pro, ram, hard, gene, price);
}

laptopType *getLastRecord(laptopType *temp)
{
    while (temp->next != NULL)
    {
        temp = temp->next;
    }
```

```
    return temp;
}

void addRecord(string manu, string mod, float pro, int ram, int hard,
int gene, int pri)
{
    laptopType *record = new laptopType;
    record->manufacturerName = manu;
    record->model = mod;
    record->processor = pro;
    record->RAM = ram;
    record->hardDrive = hard;
    record->generation = gene;
    record->price = pri;
    record->next = NULL;
    if (head == NULL)
    {
        head = record;
    }
    else
    {
        laptopType *temp = getLastRecord(head);
        temp->next = record;
    }
}

main()
{
    takeInput();
    takeInput();
}
```

Example 8:
Write a function that will display all the data related to the laptop which is stored dynamically using a linked list.

Solution

```
#include <iostream>
```

```
using namespace std;

struct laptopType
{
    string manufacturerName;
    string model;
    float processor;
    int RAM;
    int hardDrive;
    int generation;
    int price;
    laptopType *next;
};

laptopType *head = NULL;

void takeInput();
laptopType *getLastRecord(laptopType *temp);
void addRecord(string manu, string mod, float pro, int ram, int hard,
int gene, int pri);
void printSingleRecord(laptopType *l);
void displayAllRecords();

void takeInput()
{
    string manu;
    string model;
    float pro;
    int ram, hard, gene, price;
    cout << "Enter Manufacturer Name: ";
    cin >> manu;
    cout << "Enter Model: ";
    cin >> model;
    cout << "Enter Processor Speed: ";
    cin >> pro;
    cout << "Enter RAM: ";
    cin >> ram;
    cout << "Enter Hard Drive Size: ";
    cin >> hard;
```

```
    cout << "Enter Generation: ";
    cin >> gene;
    cout << "Enter Price: ";
    cin >> price;
    addRecord(manu, model, pro, ram, hard, gene, price);
}

laptopType *getLastRecord(laptopType *temp)
{
    while (temp->next != NULL)
    {
        temp = temp->next;
    }
    return temp;
}

void addRecord(string manu, string mod, float pro, int ram, int hard,
int gene, int pri)
{
    laptopType *record = new laptopType;
    record->manufacturerName = manu;
    record->model = mod;
    record->processor = pro;
    record->RAM = ram;
    record->hardDrive = hard;
    record->generation = gene;
    record->price = pri;
    record->next = NULL;
    if (head == NULL)
    {
        head = record;
    }
    else
    {
        laptopType *temp = getLastRecord(head);
        temp->next = record;
    }
}
```

```
void printSingleRecord(laptopType *l)
{
    cout << "Manufacturer Name: " << l->manufacturerName << endl;
    cout << "Model: " << l->model << endl;
    cout << "Processor: " << l->processor << endl;
    cout << "RAM: " << l->RAM << endl;
    cout << "HardDrive Size: " << l->hardDrive << endl;
    cout << "Generation: " << l->generation << endl;
    cout << "Price: " << l->price << endl;
}

void displayAllRecords()
{
    laptopType *temp = head;
    while (temp != NULL)
    {
        printSingleRecord(temp);
        temp = temp->next;
    }
}
main()
{
    takeInput();
    takeInput();
    displayAllRecords();
}
```

Example 8:
Let's write a function that can search a specific type of model in the data stored in the linked list.

Solution

```
void searchRecord(string m)
{
    laptopType *temp = head;
    bool isFound = false;
    while (temp != NULL)
    {
```

```
        if(temp->model == m)
        {
            cout << "Model found" << endl;
            isFound = true;
            break;
        }
        temp = temp->next;
    }
    if(isFound == false)
    {
        cout << "Model not found" << endl;
    }
}
```

Challenge: (Video Profile Activity)

Make 2 structures

```
struct student
{
    string name;
    student *next;
    marks *m_next;
};
```

First structure contains the student name and 2 pointers. 1 points towards the next **student** structure and other points towards the **marks** structure.

```
struct marks
{
    string subject;
    float data;
    marks *next;
};
```

Second structure contains the subject name and its marks and a pointer that points towards the next marks structure.

Your goal is to grow the student structure in both directions. In the downward direction the students increase and in the horizontal direction the new marks of the students are added.

If the User gives the new student then you have to add at the end of the link list in the vertical direction. If the user gives the name of the student who is already present in the link list then you have to add the marks in the other link list in the horizontal direction.

Hint: use the **gotoxy()** Function to display the contents on the specific location.

Sample Output: