Session 4: Data Structures

Data Structures and Algorithm 1 - Lab

Yahya Tawil
15 Oct 2021

Last Session

- Review the concept of Class in C++.
- We built the linked list data structure using classes.
- The complete code is available here: https://github.com/yahyatawil/HKU-21-22-Data-Structures-Algorithm/blob/main/slides/lab3/LinkedList.cpp

More on Linked List

- Add/delete nodes from the linked list.
- Add destructor to the Linked List class and node class.
- How to `Reset` the linked list.
- How to count linked list length

Linked List in C

```c
#include <stdio.h>
struct Node {
    int data;
    struct Node* next;} ;
typedef struct Node Node_t;

void print_linkedlist(Node_t * node)
{
    if(node->next == NULL)
    {
      printf("x");
      return;
    }
    else
      printf("%d->",node->data);
     print_linkedlist(node->next);
}
```

```c
int main(){
  Node_t head;
  head.data = 10;

  Node_t node2;
  head.next= &node2;
  node2.data = 20;

  Node_t node3;
  node2.next = &node3;
  node3.data = 30;

  Node_t tail;
  node3.next= &tail;
  tail.next = NULL;
  tail.data = 40;
  print_linkedlist(&head);return
0;}
```

Linked List in C

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;} ;
typedef struct Node Node_t;
void reset(Node_t * node)
{
    Node_t * next = node->next;
    if(node->next == NULL)
    {node->next=NULL;
      free(node);
     printf("List is empty \r\n");
      return;}
    else
    {node->next=NULL;
    free(node);     }
    reset(next);}
```

```
  Node_t *head = malloc(sizeof(Node_t));
  head->data = 10;
  Node_t *node2 =
malloc(sizeof(Node_t));;
  head->next= node2;
  node2->data = 20;
  Node_t* node3 =
malloc(sizeof(Node_t));;
  node2->next = node3;
  node3->data = 30;
  Node_t *tail = malloc(sizeof(Node_t));;
  node3->next= tail;
  tail->next = NULL;
  tail->data = 40;
  print_linkedlist(head);
  reset(head);
  print_linkedlist(head);
```

C++ Basics (Class Destructor)

- when object reaches end of lifetime, typically some cleanup required before object passes out of existence.
- destructors often serve to release resources associated with object.
- destructor for class `T` always has name `T::~T`

```
class Widget {
public:
    Widget(int bufferSize) { // Constructor.
        // allocate some memory for buffer
        bufferPtr_ = new char[bufferSize];
    }
    ~Widget() { // Destructor.
        // free memory previously allocated
        delete [] bufferPtr_;
    }
    // copy constructor, assignment operator, ...
private:
    char* bufferPtr_; // pointer to start of buffer
};
```

More on Linked List

- Is calling the last destructor for linked list is enough ?

No

```
~Linkedlist()
{
    cout<<"delete list"<<endl;
}
```

```
Node N;
cout<<"size of node:"<<sizeof(N)<<endl;

Linkedlist* List = new Linkedlist();
List->printList();
List->insertNode(10);
List->printList();
List->insertNode(20);
List->printList();
List->insertNode(30);
List->printList();
delete List;

cout<<"old node:"<<dec<<oldNode->data<<endl;
```

More on Linked List

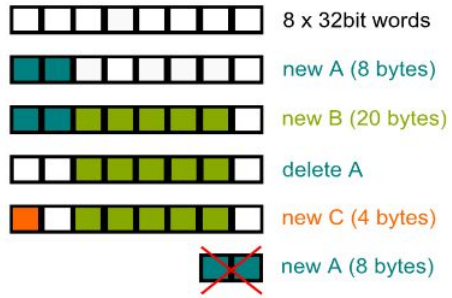
- Is calling this destructor for linked list is enough ? **Yes**

```
~Linkedlist()
{
    // Traverse till end of list
    Node* temp = head;
    Node* temp1 ;
    while (temp->next != NULL) {
        // Update temp
        temp1 = temp->next;
        delete temp;
        temp = temp1;
    }

    cout<<"delete list"<<endl;
}
```

C++ Basics (New and Delete)

- `new` to allocate memory at run time (dynamic allocation) and `delete` the deallocate the allocated memory.
- Form of allocation: Single or array.
- `Malloc/free` in c.
- Heap memory(from data memory) is used for allocation.
- Pro: Can be used when the variable size is unknown.
- Con: Can leads to fragmentation.
- Make sure to deallocate the space at the end.

```
using namespace std;
int main(){
    cout<<"Enter the array size:";
    int size;
    cin>>size;
    int * array = new int[size];

    cout<<"array
allocated:"<<hex<<&array[0]<<&array
[size-1];
    return 0;
}
```



Assignment 2: Exercise 2

- Extend your code in exercise 1 and make the linked list capable of having different data type in each node (char, float or int). Hint: you need to make use of method/constructor overloading.