



DASF004: Basic and Practice in Programming

❖ Lab 12: Structure



In this lab ...

- ❖ Introduction to Structure and Enumeration
- ❖ What you need to submit in this lab (Lab #12):
 - » Lab Exercise #12 by Wednesday 11:59 pm
 - » Assignment #12 by Tuesday 11:59 pm



You can create new type using `struct` and `enum`

- Using a keyword `enum`, a set of integer constants can be represented by a set of identifiers.
- The values in an `enum` start with 0, unless specified otherwise, and are incremented by 1.

```
enum days {Mon, Tue, Wed, Thu, Fri, Sat, Sun};
```

This creates a new data type, `days`, in which the identifiers are set automatically to the integers 0 to 6.



You can create new type using struct and enum

- You can then declare a variable using the new type `days`, and perform other logical calculations:

```
days Today = Wed;
if (Today == Sun)
    printf("Happy Holiday!");
else if (Today <= Fri)
    printf("Working day...");
```



You can create new type using `struct` and `enum`

- `struct` is to create a new type consists of a collection of types:

```
struct StuInfo {  
    char Name[10];  
    int IDNo;  
};
```

- After you define the new type using `struct`, you can declare the new variable with the type:

```
StuInfo Student1;
```



You can create new type using struct and enum

- The . operator:

```
struct StuInfo {  
    char Name[10];  
    int IDNo;  
};
```

```
StuInfo StudentTest;  
strcpy(StudentTest.Name, "John");  
StudentTest.IDNo = 1357;
```

Lab Exercise 12:

- You are provided with a data file in ASCII format.
- The data file contains 80 students information in a class.
 - Name, ID Number, and 2 exam score.
 - Examine this file first
- Write a program to
 - import all these 80 students information into a struct defined as follow:

```
struct StuInfo {  
    char Name[10];  
    int IDNo;  
    int Exam[2];  
};
```

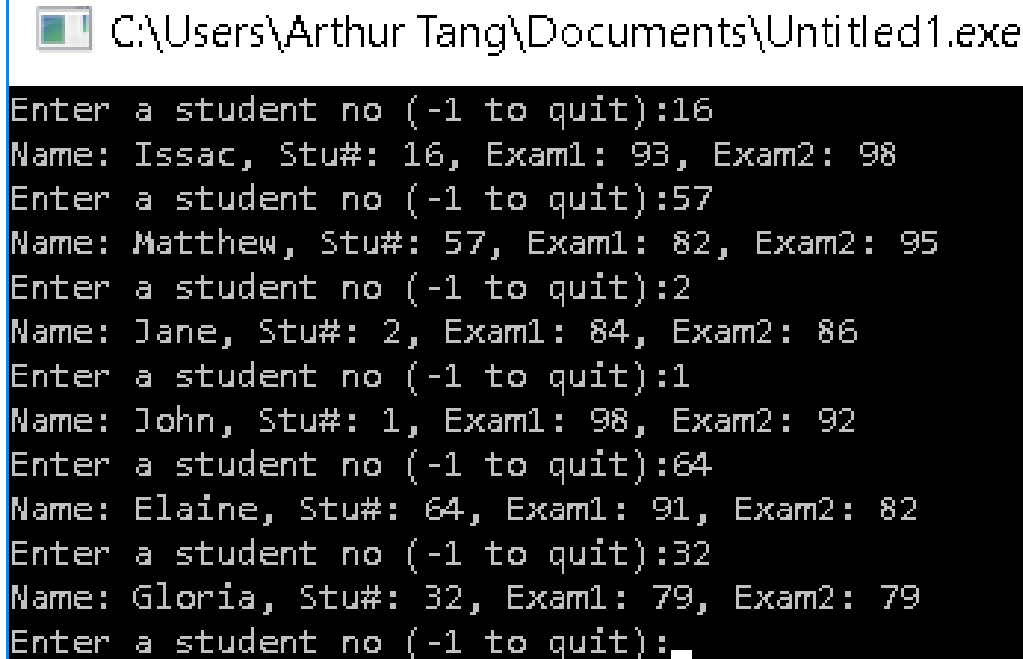
- Store information of these 80 students using ONE array (not 80 different variables).
- Your program should be able to import file containing any number of students (i.e. it should work with other input files with different number of students)
- Your program will then ask the user for the student number, and display all the information of that student
- Your program will be running continuously, until user enter -1 as the student number to quit the program.

Data file:

- Available on iCampus (“lab12_datafile.txt”)
- Each row in the file correspond to student information for one student

Name	Student No.	Exam1 Score	Exam2 Score
John	1	98	92
Jane	2	84	86
Mary	3	76	87
Joe	4	92	95

If your code is correct, you should generate the following sample output:



```
C:\Users\Arthur Tang\Documents\Untitled1.exe
Enter a student no (-1 to quit):16
Name: Issac, Stu#: 16, Exam1: 93, Exam2: 98
Enter a student no (-1 to quit):57
Name: Matthew, Stu#: 57, Exam1: 82, Exam2: 95
Enter a student no (-1 to quit):2
Name: Jane, Stu#: 2, Exam1: 84, Exam2: 86
Enter a student no (-1 to quit):1
Name: John, Stu#: 1, Exam1: 98, Exam2: 92
Enter a student no (-1 to quit):64
Name: Elaine, Stu#: 64, Exam1: 91, Exam2: 82
Enter a student no (-1 to quit):32
Name: Gloria, Stu#: 32, Exam1: 79, Exam2: 79
Enter a student no (-1 to quit):_
```

- You can verify the results with the data file

Question

Your student info is stored in an array with 80 items of StuInfo

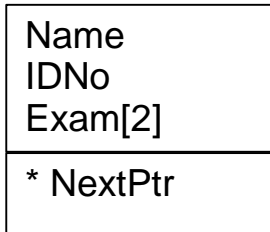
- `StuInfo GEDB029[80];`

What if you want to add one more student?

- You can't ...
- You have to create another array and copy the entire array to the new array
- `StuInfo NewGEDB029[81];`
- **Copy everything from GEDB029 to NewGEDB029**
- **Add information of the new student to NewGEDB029**

Lets Modify our struct slightly...

```
struct StuInfo
{ char Name[10];
  int IDNo;
  int Exam[2];
  StudInfo * NextPtr;
};
```

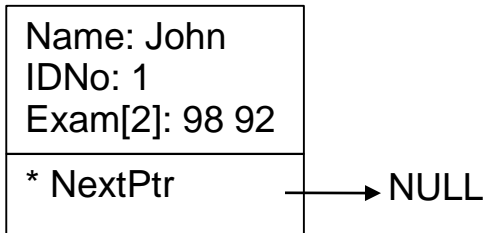


Adding a node

```
struct StuInfo
{ char Name[10];
  int IDNo;
  int Exam[2];
  StuInfo * NextPtr;
};

int main (void)
{ StuInfo Node1;
  strcpy(Node1.Name, "John");
  Node1.IDNo = 1;
  Node1.Exam[0]=98; Node1.Exam[1]=92;
  Node1.NextPtr = NULL;
}
```

Node1



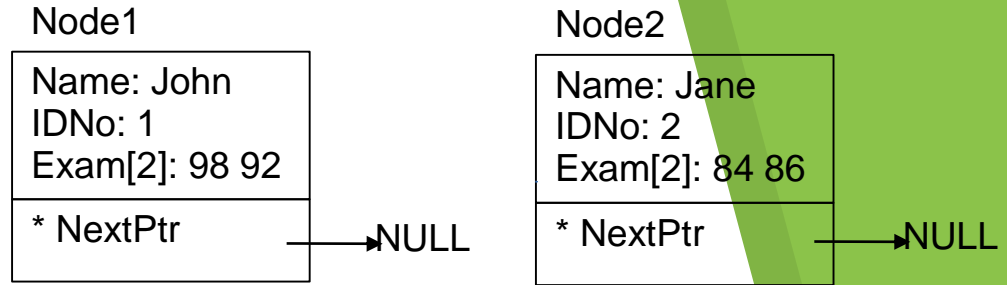
Adding another node

```
struct StuInfo
{
    char Name[10];
    int IDNo;
    int Exam[2];
    StuInfo * NextPtr;
};

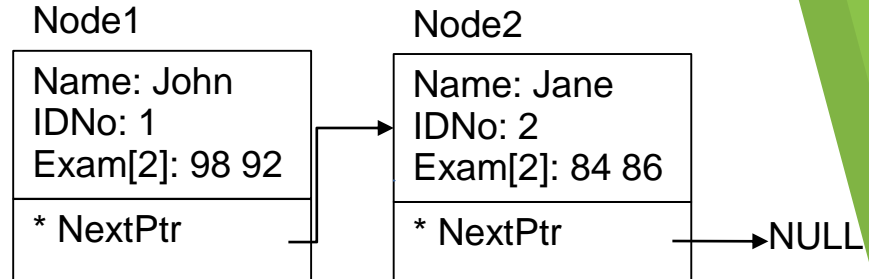
int main (void)
{
    StuInfo Node1;
    strcpy(Node1.Name, "John");
    Node1.IDNo = 1;
    Node1.Exam[0]=98; Node1.Exam[1]=92;
    Node1.NextPtr = NULL;

    StuInfo Node2;
    strcpy(Node2.Name, "Jane");
    Node2.IDNo = 2;
    Node2.Exam[0]=84; Node2.Exam[1]=86;
    Node2.NextPtr = NULL;
    Node1.NextPtr = & Node2;
}
```

1. Create Node2 First



2. Modify NextPtr of Node1 pointing to Node2



Forming a Linked List with 5 nodes

```
struct StuInfo
{ char Name[10];
  int IDNo;
  int Exam[2];
  StuInfo * NextPtr;
};
```

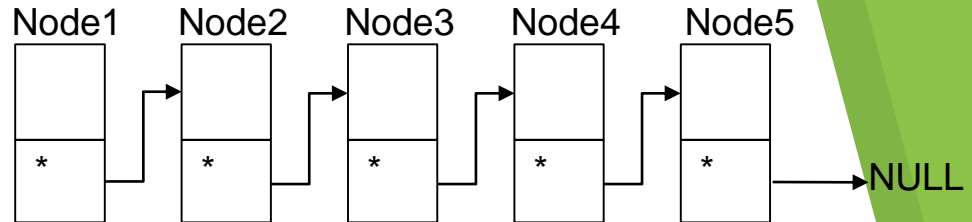
```
int main (void)
{ StuInfo Node1, Node2, Node3, Node4, Node5;
  strcpy(Node1.Name, "John");
  Node1.IDNo = 1;
  Node1.Exam[0]=98; Node1.Exam[1]=92;
  Node1.NextPtr = NULL;
```

```
  strcpy(Node2.Name, "Jane");
  Node2.IDNo = 2;
  Node2.Exam[0]=84; Node2.Exam[1]=86;
  Node2.NextPtr = NULL;
  Node1.NextPtr = & Node2;
```

```
  strcpy(Node3.Name, "Mary");
  Node3.IDNo = 3;
  Node3.Exam[0]=76; Node3.Exam[1]=87;
  Node3.NextPtr = NULL;
  Node2.NextPtr = & Node3;
```

```
  strcpy(Node4.Name, "Joe");
  Node4.IDNo = 4;
  Node4.Exam[0]=92; Node4.Exam[1]=95;
  Node4.NextPtr = NULL;
  Node3.NextPtr = & Node4;
```

```
  strcpy(Node5.Name, "Peter");
  Node5.IDNo = 5;
  Node5.Exam[0]=98; Node5.Exam[1]=36;
  Node5.NextPtr = NULL;
  Node4.NextPtr = & Node5;
}
```



Now you have a Linked List with 5 nodes ...

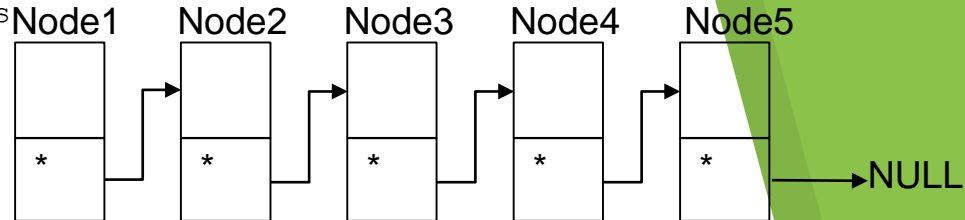
How do you access information on each node?

e.g. Printing the name of each node

```
struct StuInfo
{ char Name[10];
  int IDNo;
  int Exam[2];
  StuInfo * NextPtr;
};

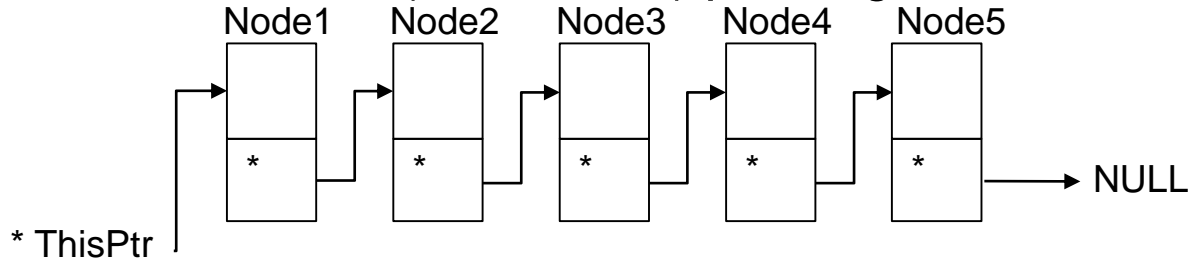
void PrintAllName(StuInfo Node)
{ // How do you implement a function to pass through
  // each node and then print the name of each node???
}

int main (void)
{ StuInfo Node1, Node2, Node3, Node4, Node5;
  ... ..
  ... .. // Establish the linked list with 5 nodes
  ... ..
  PrintAllName(Node1);
}
```



Printing a member of each node in a Linked List

1. Create a Node Pointer (`*ThisPtr`) pointing to the first node

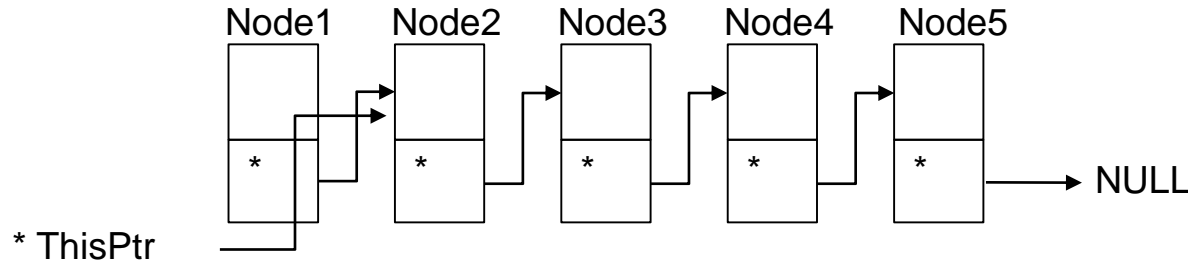


2. Access the member using the the `*ThisPtr` Pointer

```
» printf("Name: %s\n", ThisPtr->Name);
```

3. Modify `ThisPtr` pointing to the next node

```
» ThisPtr = ThisPtr->NextPtr;
```



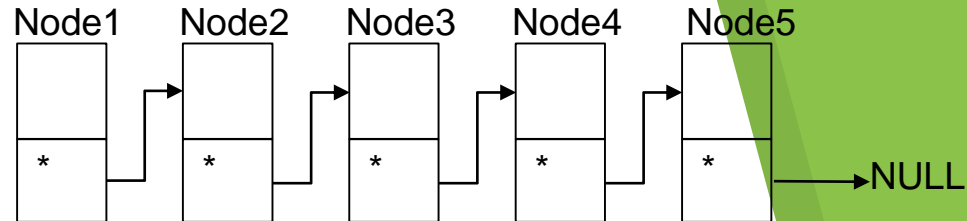
4. Repeat 2&3 until `ThisPtr` is pointing to NULL

How do you access information on each node?

```
struct StuInfo
{ char Name[10];
  int IDNo;
  int Exam[2];
  StuInfo * NextPtr;
};
```

```
void PrintAllName(StuInfo Node)
{ StuInfo* ThisPtr = &Node1; // A pointer ThisPtr pointing to the first node
  while(ThisPtr != NULL)      // Repeat while ThisPtr is not point to NULL
  { printf("Name: %s\n",ThisPtr->Name); // Access information in the current node
    ThisPtr = ThisPtr->NextPtr;        // Change ThisPtr pointing to the next node
  }
}
```

```
int main (void)
{ StuInfo Node1, Node2, Node3, Node4, Node5;
  ... ..
  ... .. // Establish the linked list with 5 nodes
  ... ..
  PrintAllName(Node1);
}
```



Lab Assignment #12:

- Your program will read the file from Lab Exercise #12 as input, and then store the data as a linked list using the data from the input file.
- Your program should import files containing any number of students (i.e. it should work with other input files with different number of students)
- Implement 3 functions:
 - `void PrintAllName(StuInfo Node)`
 - This function will take the first node of the linked list as parameter, and print out the name of all students in the node
 - `float AverageScore(StuInfo Node)`
 - This function will take the first node of the linked list as parameter, and then return the average score of both exams of all students in the linked list as return value (i.e. average score of 80 students is the average value of 160 scores, since each students has 2 scores)
 - `StuInfo * BestStudent(StuInfo Node)`
 - This function will take the first node of the linked list as parameter, and then return a pointer pointing to the best student (i.e. highest average score)

Lab Assignment #12:

- Your program will call these 3 functions, and then:
 - Display the name of all students
 - Display the average score of all students
 - Display the name, student ID and the two scores of the best student
- Submit your source code on iCampus before Tuesday 11:59 pm