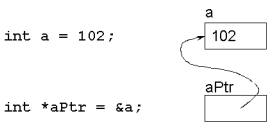
Pointers

- A pointer is a type of variable that refers to a value. (It contains an address in memory)
- It is a kind of indirect addressing



Pointer Variable Definition in C++

- Pointers are typed, just as other variables
- '*' prefix is used to indicate a pointer variable

```
int *intPtr;
double *doublePtr;
```


Address Operator in C++

- Address of a variable in memory is obtained by '&' operator
- '*' is used to indicate a pointer variable

```
int i;
int *intPtr = &i;
double x;
double *doublePtr = &x;
```


Examples

```
int i = 5;
int *iPtr = &i;
std::cout << i << std::endl;    // 5
std::cout << &i << std::endl;   // 43b69f73
std::cout << iPtr << std::endl; // 43b69f73
```


Dereferencing operator in C++

- How to obtain the value in the location a pointer is referring to?
- '*' operator is used for indirect access to the value
- It is also called dereferencing operator or indirection operator

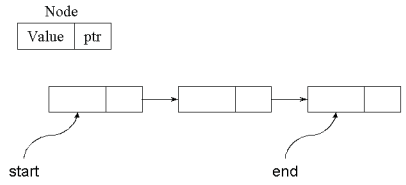
```
int i = 5;
int *iPtr = &i;
cout << *iPtr << endl;      // 5
```


Exercise

- Try the small example pieces of code for pointers (last three slides)

Linked List Structure

- It is a list that can only be accessed sequentially



Node Class

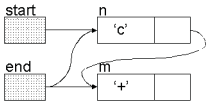
- Nodes are used to as cells to keep the data

```
class Node
{
private:
   char val;
   Node *ptr;
public:
   Node(char);
   Node *getNext();
   void setNext(Node);
   char getValue();
   void setValue(char);
};
```


Using Node Class

- Static Memory Allocation

```
Node *start;
Node *end;
Node n = Node('c');
start = &n;
end = &n;
Node m = Node('+');
end->setNext(m);
end = &m;
```



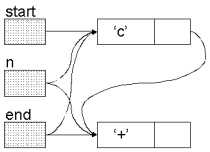
Memory Allocation

- Dynamic versus Static memory allocation
- Amount of *static* memory required by the program is known at the compilation time
- Memory requested and allocated at the run time is known as *dynamic* memory

Using Node Class

- Dynamic Memory Allocation

```
Node *start;
Node *end;
Node *n = new Node('c');
start = n;
end = n;
n = new Node('+');
end->setNext(n);
end = n;
```



Dynamic Memory Allocation

```
Node *n = new Node('c');
Node *start = n;
Node *end = n;

n = new Node('+');
end->setNext(n);
end = n;

n = new Node('+');
end->setNext(n);
end = n;

cout << start->getValue();
cout << start->getNext()->getValue();
cout << start->getNext()->getNext()->getValue() << endl;
```