CS 32 Spring 2021

Week 3 Discussion

TA: Manoj Reddy

LA: Katherine Zhang

Pointers

- Every variable has an address in memory
- Can get the address of a variable using the & operator
- Pointer is nothing but a variable to that stores a memory address instead of a regular value
  - Memory address can be another variable's address
- Pointer variable occupies 4/8 bytes of memory depending on 32/64-bit architecture
  - int x = 20;
  - int *ptr = & x;
  - *ptr = 22;

To print the value of the pointer variable myPtr , which statement would you use? **D**

```
A. cout << &myPtr;          D. cout << myPtr;
B. cout << *myPtr;          E. cout << myPtr&;
C. cout << myPtr*;
```

Which of the following operations is NOT allowed for a pointer?

A. adding an integer to a pointer
B. dereferencing a pointer
C. multiplying a pointer by an integer
D. using a pointer variable in a relational expression
E. getting the address of a pointer

C

Which of the following values can NOT be assigned to a pointer to an integer variable.

A. The address of a float variable
B. 0 (zero)
C. NULL
D. The address of an integer variable

A

Given this function prototype: void square (int *);

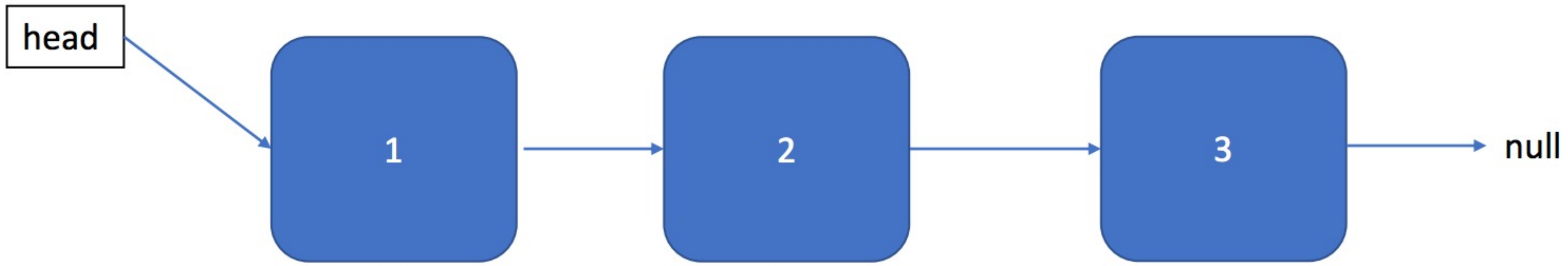
Which of the function calls below would be correct to call function square with integer variable y defined in the calling function.

A. square[y];
B. square (*y);
C. square (&y);
D. square (y);

C

Linked List

- Data structures is a core component of CS32
- Each data structure has pros and cons
- Tradeoff (Insertion, Deletion, Lookup …)
- Linked List are widely used in various applications
- Important to understand memory structure



Basic Outline

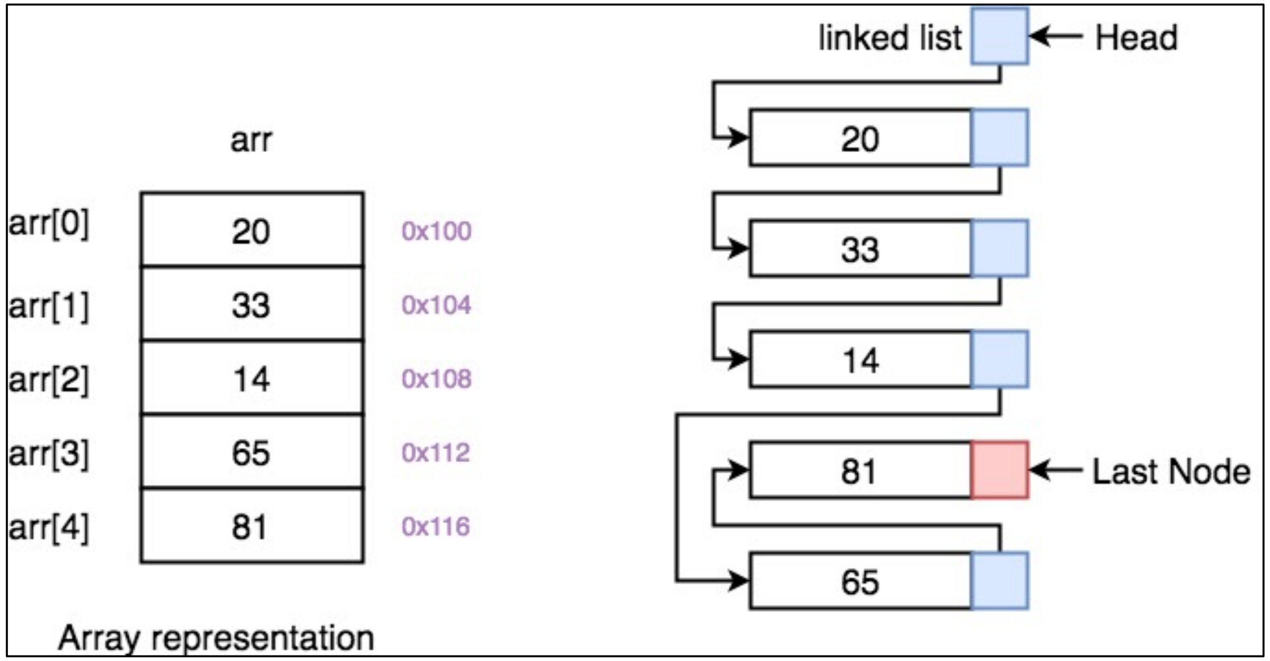
```
class LinkedList
{
public:
        LinkedList(){…}
        void addToFront(int v) { … }
        void addToRear(int v) { … }
        void deleteItem(int v) { … }
        bool findItem(int v) { … }
        void printItems()
        ~LinkedList() { … }

private:
       Node *head;
};
```

```
struct Node
{
   int value;
   Node *next;
};
```

```
int main(){
   LinkedList l1;
   l1.addToFront(20);
   l1.addToRear(22)
   l1.printItems();
}
```

Tradeoffs

- Comparing arrays vs linked list, which is faster?
  - Accessing 999th item
  - Inserting a new item at front
  - Removing an item in the middle
- Which is easier to program?
  - Which data structure will take less time to program and debug?



LA Worksheet