Chapter 11

Data Structures

OBJECTIVES

After reading this chapter, the reader should be able to:

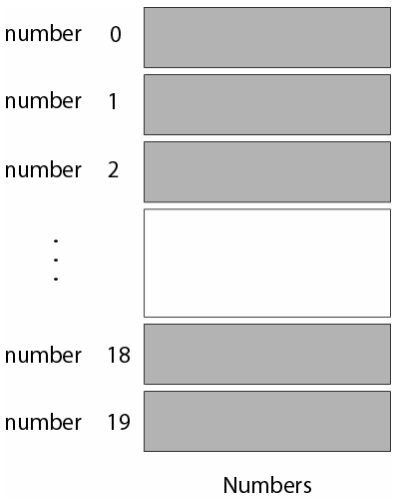
- Understand arrays and their usefulness.
- Understand records and the difference between an array and a record.
- Understand the concept of a linked list and the difference between an array and a linked list.
- Understand when to use an array and when to use a linked-list.

11.1 ARRAYS

Brooks/Cole Thomson Learning

Figure 11-1

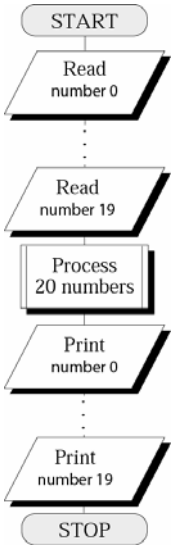
Twenty individual variables
Declare 20 variables!



Brooks/Cole Thomson Learning

Figure 11-2

Processing individual variables



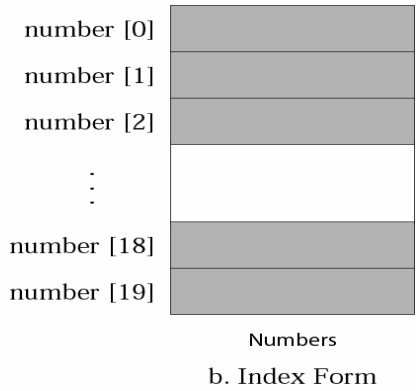
Brooks/Cole
Thomson Learning

Figure 11-3

Arrays with subscripts and indexes

Declare 20 variables→ Declare a array with 20 elements

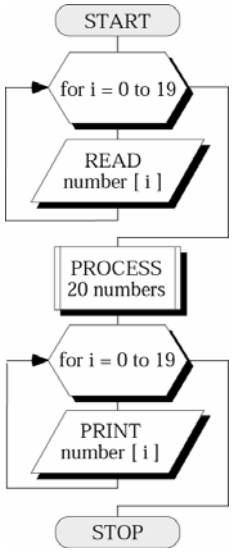
```
int number0, number1, …
number0=0;
number1=1;
…
change to

int number[20];
for (i=0;i=19; i++)
{
  number[i]=i;
}
```



Brooks/Cole
Thomson Learning

Figure 11-4

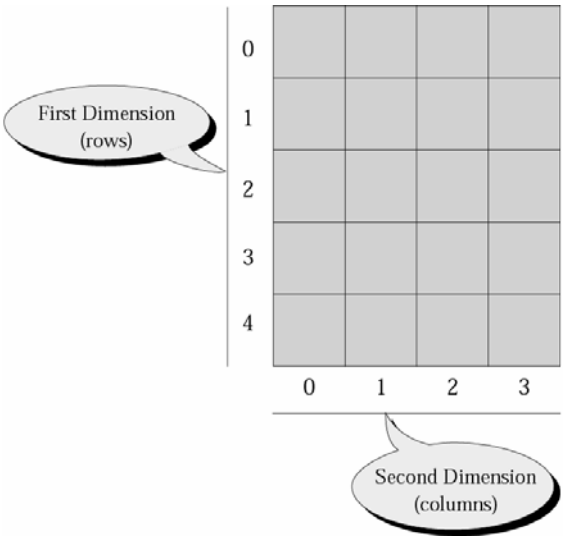
Processing an array



Brooks/Cole
Thomson Learning

Figure 11-7- Part I

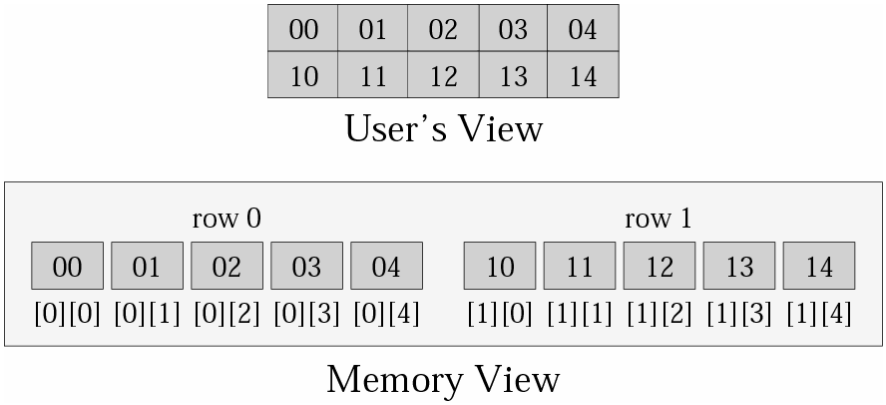
Two-dimensional array



Brooks/Cole
Thomson Learning

Figure 11-8

Memory layout

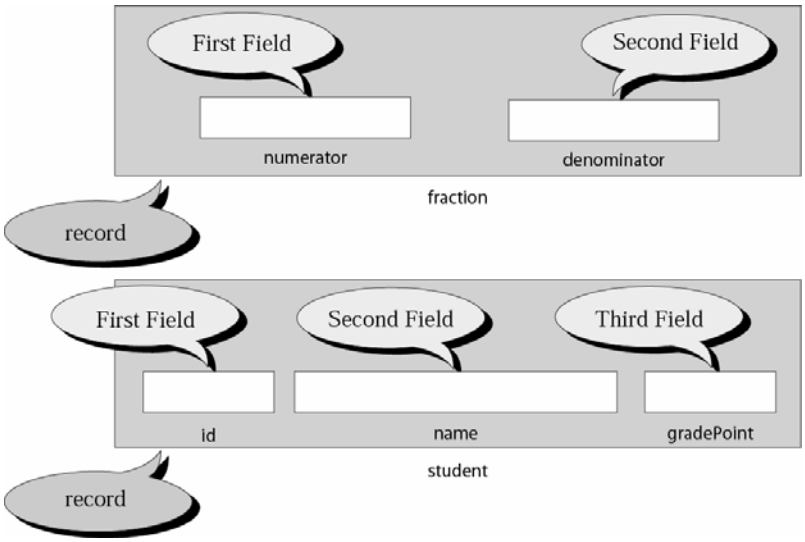


Row-major data allocation scheme.

11.2 RECORDS

Figure 11-9

Records



Brooks/Cole Thomson Learning

Note:

The elements in a record can be of the same or different types. But all elements in the record must be related.

Brooks/Cole Thomson Learning

Record in C and C++

```
struct fraction
{
  int numerator,  denominator;
};
fraction V;
V.numerator=1;
V.denominator=2;
```


Figure 11-10

Linked lists

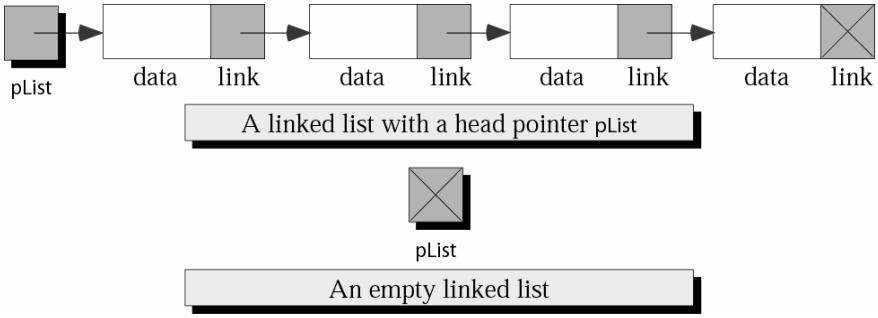


Figure 11-11

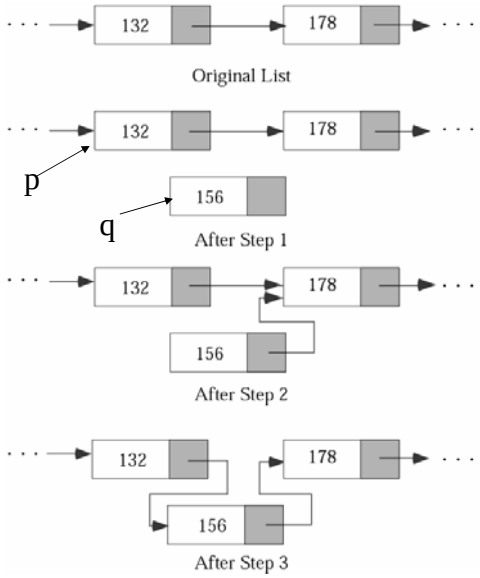
Node


```
struct Node
{
  int Data;
  Node* Ptr;
}
Node*  Ptr;
```

Figure 11-12

Inserting a node

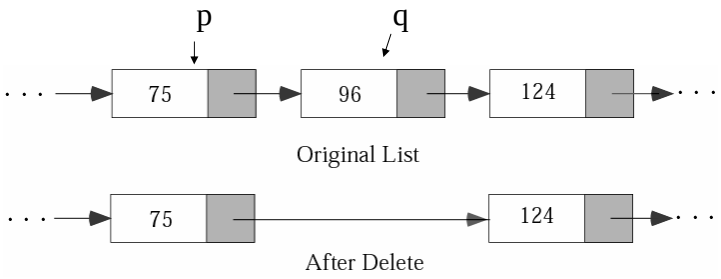
q.Ptr=p.Ptr;
p.Ptr=q;



Brooks/Cole Thomson Learning

Figure 11-13

Deleting a node

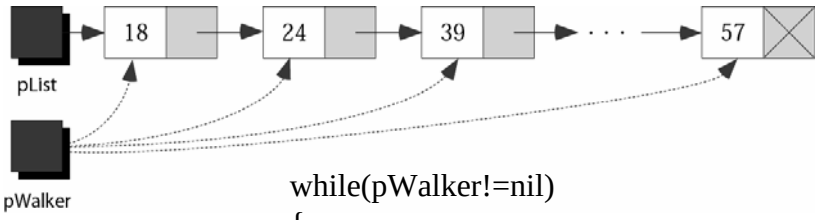


Node* q=p.Ptr;
p.Ptr=q.Ptr;
delete q;

Brooks/Cole Thomson Learning

Figure 11-14

Traversing a list




```
while(pWalker!=nil)
{
  pWalker=pWalker.Ptr;
  //Do something meaningful
}
```

Brooks/Cole
Thomson Learning