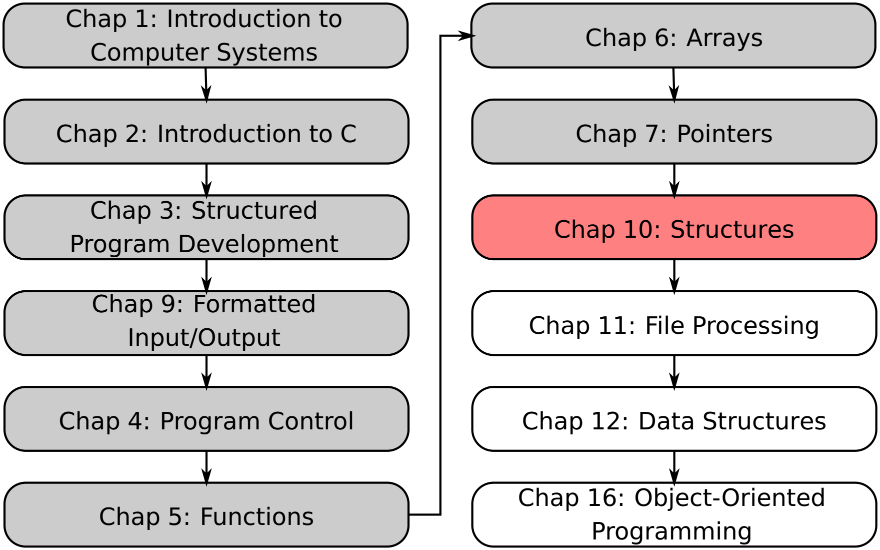
Computer Programming 143 – Lecture 24
C Structures II

Electrical and Electronic Engineering Department
University of Stellenbosch

Prof Johan du Preez
Mr Callen Fisher
Dr Willem Jordaan
Dr Hannes Pretorius
Mr Willem Smit

Copyright & Disclaimer

Module Overview


Chap 1: Introduction to Computer Systems
Chap 2: Introduction to C
Chap 3: Structured Program Development
Chap 9: Formatted Input/Output
Chap 4: Program Control
Chap 5: Functions
Chap 6: Arrays
Chap 7: Pointers
Chap 10: Structures
Chap 11: File Processing
Chap 12: Data Structures
Chap 16: Object-Oriented Programming

Lecture Overview

1. Arrays of Structures
2. `malloc()` and Structures
3. Example: Card Shuffling and Dealing Simulation (10.7)

Arrays of Structures I

```c
/* Example of creating and using an array of structures */
#include <stdio.h>

typedef struct {
    char *name;
    int atomicNumber;
    double atomicMass;
} Element; // defines structure type Element

int main( void )
{
    // array of 118 Element structures
    Element periodicTableOfElements[ 118 ];
    // pointer to type Element
    Element *elementPtr = periodicTableOfElements;

    // assign values to members of first array element using dot operator
    periodicTableOfElements[0].name = "Hydrogen";
```

Arrays of Structures II

```c
    periodicTableOfElements[0].atomicNumber = 1;
    periodicTableOfElements[0].atomicMass = 1.00794;

    // display values of members of first array element
    // using arrow operator
    printf( "Name: %s\nAtomic number: %d\nAtomic mass: %lf",
            elementPtr->name, elementPtr->atomicNumber,
            elementPtr->atomicMass );

    return 0; // indicates successful termination
} // end main
```

Output

```
Name: Hydrogen
      Atomic number: 1
      Atomic mass: 1.007940
```

malloc() and Structures I

Creating an array of structures during runtime

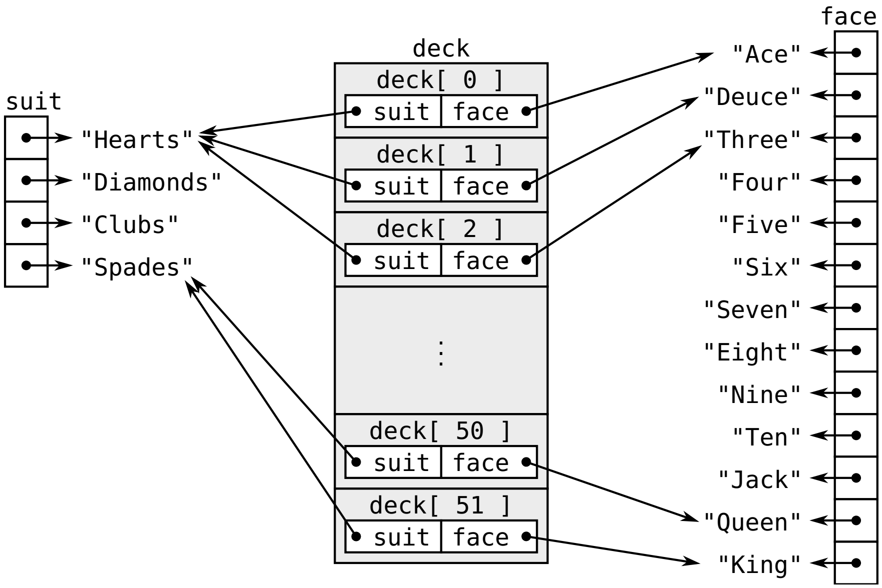
```
typedef struct {
    char *name;
    int atomicNumber;
    double atomicMass;
} Element;  // defines structure type Element

int numberOfElements = 118;
Element *periodicTableOfElements; // pointer to structure type Element

// reserves space for numberOfElements amount of structures of type
// Element
periodicTableOfElements = malloc( numberOfElements * sizeof( Element ));
periodicTableOfElements[0].name = "Hydrogen";

free( periodicTableOfElements ); // frees memory allocated previously
```

10.7 Example: Card Shuffling and Dealing

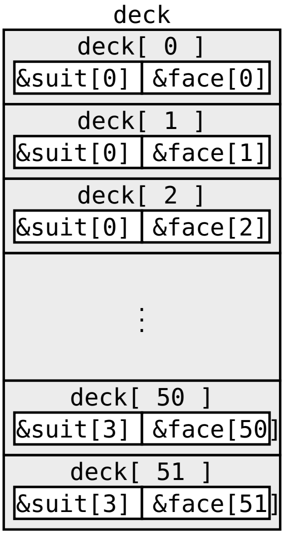
Problem statement

Design and implement an algorithm that shuffles and deals a 52-card deck

10.7 Example: Data structure



10.7 Example: Filled deck



Pseudocode

```
Create Card structure
Create array of 52 “Card"s
Initialise the constant array of suit names
Initialise the constant array of face names

For each Card in Array
    Initialise the face and the suit - in an ordered sequence

For each Card in Array
    Determine a random number between 0 and 51
    Swap the card at that random position with the current card

For each Card in Array
    Display the face and suit
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// card structure definition
struct card {
    const char *face; // define pointer face
    const char *suit; // define pointer suit
}; // end structure card

typedef struct card Card; // new type name for struct card

// prototypes
void fillDeck( Card * const wDeck, const char * wFace[],
    const char * wSuit[] );
void shuffle( Card * const wDeck );
void deal( const Card * const wDeck );
```

```c
int main( void )
{
    Card deck[ 52 ]; // define array of Cards

    // initialise array of pointers
    const char *face[] = { "Ace", "Deuce", "Three", "Four", "Five",
        "Six", "Seven", "Eight", "Nine", "Ten",
        "Jack", "Queen", "King" };

    // initialise array of pointers
    const char *suit[] = { "Hearts", "Diamonds", "Clubs", "Spades" };

    srand( time( NULL ) ); // randomise

    fillDeck( deck, face, suit ); // load the deck with Cards
    shuffle( deck ); // put Cards in random order
    deal( deck ); // deal all 52 Cards
    return 0; // indicates successful termination
} // end main
```

```c
// place strings into Cards structures
void fillDeck( Card * const wDeck, const char * wFace[],
    const char * wSuit[] )
{
    int i; // counter

    // loop through wDeck
    for ( i = 0; i <= 51; i++ ) {
        //set face field to point to wFace[] character string
        wDeck[ i ].face = wFace[ i % 13 ];
        //set suit field to point to wSuit[] character string
        wDeck[ i ].suit = wSuit[ i / 13 ];
    } // end for
} // end function fillDeck
```

```c
// shuffle cards
void shuffle( Card * const wDeck )
{
    int i; // counter
    int j; // variable to hold random value between 0 - 51
    Card temp; // define temporary structure for swapping Cards

    // loop through wDeck randomly swapping Cards
    for ( i = 0; i <= 51; i++ ) {
        j = rand() % 52;
        temp = wDeck[ i ];
        wDeck[ i ] = wDeck[ j ];
        wDeck[ j ] = temp;
    } // end for
} // end function shuffle
```

```c
// deal cards
void deal( const Card * const wDeck )
{
    int i; // counter

    // loop through wDeck
    for ( i = 0; i <= 51; i++ ) {
        printf( "%5s of %-8s", wDeck[ i ].face, wDeck[ i ].suit );
        if ((i + 1) % 4)
            printf(" ");
        else
            printf("\n");
    } // end for
} // end function deal
```

Perspective

Today

C Structures II

- Arrays of Structures
- `malloc()` and Structures
- Example: shuffling and dealing of cards

Next lecture

File Processing

- Sequential-access files

Homework

1. Study Section 10.7 in Deitel & Deitel
2. Do Self Review Exercises 10.1(a),(d),(f),(g),(k), and (l)