MA122 - Computer Programming and Apllications

Indian Institute of Space Science and Technology

February 17, 2017

Lecture 14

1 Structures

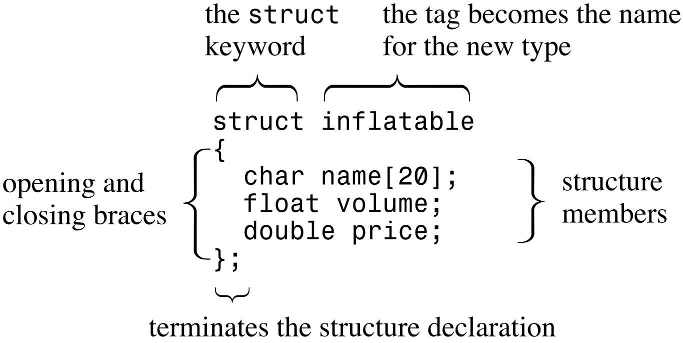
2 Address

3 pointer

Declaration

```c
struct inflatable // structure declaration
{
  char name[20];
  float volume;
  double price;
};
```

Declaration



the struct keyword

the tag becomes the name for the new type

```
struct inflatable
{
    char name[20];
    float volume;
    double price;
};
```

opening and closing braces

structure members

terminates the structure declaration

Figure 4.6 Parts of a structure description.

Program

```
#include <iostream>
struct inflatable // structure declaration
{
  char name[20];
  float volume;
  double price;
};
int main()
{
  using namespace std;
  inflatable guest =
  {
    "Glorious Gloria", // name value
    1.88,             // volume value
    29.99             // price value
  }; // guest is a structure variable of type
      inflatable
  // It’s initialized to the indicated values
```

Program

```
  inflatable pal =
  {
    "Audacious Arthur",
    3.12,
    32.99
  }; // pal is a second variable of type inflatable

  cout << "Expand your guest list with " << guest.name;
  cout << " and " << pal.name << "!\n";
  //pal.name is the name member of the pal variable
  cout << "You can have both for $";
  cout << guest.price + pal.price << "!\n";
  return 0;
}
```

Declaration

external declaration — can be used in all functions in file

local declaration — can be used only in this function

type `parts` variable

type `perks` variable

type `parts` variable

can't declare a type `perks` variable here

```
#include <iostream>
using namespace std;
struct parts
{
  unsigned long part_number;
  float part_cost;
};
void mail();
int main()
{
  struct perks
  {
    int key_number;
    char car[12];
  };
  parts chicken;
  perks mr_blug;
  ...
  ...
}
void mail()
{
  parts studebaker;
  ...
  ...
}
```

Program

```
#include <iostream>
struct inflatable
{
  char name[20];
  float volume;
  double price;
};
int main()
{
  using namespace std;
  inflatable bouquet =
  {
    "sunflowers",
    0.20,
    12.49
  };
```

Program

```
  inflatable choice;


  cout << "bouquet: " << bouquet.name << " for $";
  cout << bouquet.price << endl;

  choice = bouquet; // assign one structure to another

  cout << "choice: " << choice.name << " for $";
  cout << choice.price << endl;

  return 0;
}
```

book 1

```
#include <iostream>
#include <cstring>

using namespace std;

struct Books {
    char  title[50];
    char  author[50];
    char  subject[100];
    int   book_id;
};
```

book 1

```cpp
int main( ) {
    struct Books Book1;      // Declare Book1 of type Book
    struct Books Book2;      // Declare Book2 of type Book

    // book 1 specification
    strcpy( Book1.title, "Learn C++ Programming");
    strcpy( Book1.author, "Chand Miyan");
    strcpy( Book1.subject, "C++ Programming");
    Book1.book_id = 6495407;

    // book 2 specification
    strcpy( Book2.title, "Telecom Billing");
    strcpy( Book2.author, "Yakit Singha");
    strcpy( Book2.subject, "Telecom");
    Book2.book_id = 6495700;
```

book 2

```

    // Print Book1 info
    cout << "Book 1 title : " << Book1.title <<endl;
    cout << "Book 1 author : " << Book1.author <<endl;
    cout << "Book 1 subject : " << Book1.subject <<
        endl;
    cout << "Book 1 id : " << Book1.book_id <<endl;


    // Print Book2 info
    cout << "Book 2 title : " << Book2.title <<endl;
    cout << "Book 2 author : " << Book2.author <<endl;
    cout << "Book 2 subject : " << Book2.subject <<
        endl;
    cout << "Book 2 id : " << Book2.book_id <<endl;

    return 0;
}
```

Structures as Function Arguments

```cpp
#include <iostream>
#include <cstring>

using namespace std;
void printBook( struct Books book );

struct Books {
    char  title[50];
    char  author[50];
    char  subject[100];
    int   book_id;
};

int main( ) {
    struct Books Book1;      // Declare Book1 of type
        Book
    struct Books Book2;      // Declare Book2 of type
        Book
```

Structures as Function Arguments

```

    // book 1 specification
    strcpy( Book1.title, "Learn C++ Programming");
    strcpy( Book1.author, "Chand Miyan");
    strcpy( Book1.subject, "C++ Programming");
    Book1.book_id = 6495407;

    // book 2 specification
    strcpy( Book2.title, "Telecom Billing");
    strcpy( Book2.author, "Yakit Singha");
    strcpy( Book2.subject, "Telecom");
    Book2.book_id = 6495700;
```

Structures as Function Arguments

```

    // Print Book1 info
    printBook( Book1 );

    // Print Book2 info
    printBook( Book2 );

    return 0;
}

void printBook( struct Books book ) {
    cout << "Book title : " << book.title <<endl;
    cout << "Book author : " << book.author <<endl;
    cout << "Book subject : " << book.subject <<endl;
    cout << "Book id : " << book.book_id <<endl;
}
```

Lecture 14

1 Structures

2 Address

3 pointer

How to find the address of a variable

```
#include <iostream>
int main()
{
  using namespace std;


  int donuts = 6;
  double cups = 4.5;

  cout << "donuts value = " << donuts;
  cout << " and donuts address = " << &donuts << endl;


  cout << "cups value = " << cups;
  cout << " and cups address = " << &cups << endl;

  return 0;
}
```

Lecture 14

dereferencing

```cpp
#include <iostream>
int main()
{
  using namespace std;

  int updates = 6;      // declare a variable

    int * p_updates;     // declare pointer to an int

  p_updates = &updates; // assign address of int to pointer
  // express values two ways

  cout << "Values: updates = " << updates;

  cout << ", *p_updates = " << *p_updates << endl;
```

program

```
  // express address two ways

  cout << "Addresses: &updates = " << &updates;

  cout << ", p_updates = " << p_updates << endl;

  // use pointer to change value

  *p_updates = *p_updates + 1;

  cout << "Now updates = " <<updates << endl;

  return 0;
}
```

```
int jumbo = 23;
int * pe = &jumbo;
```

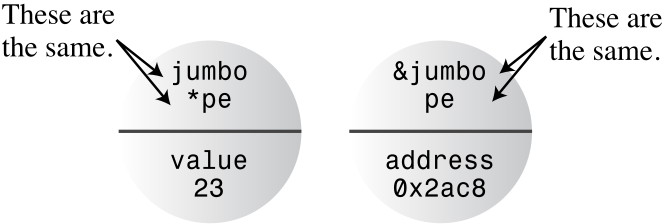
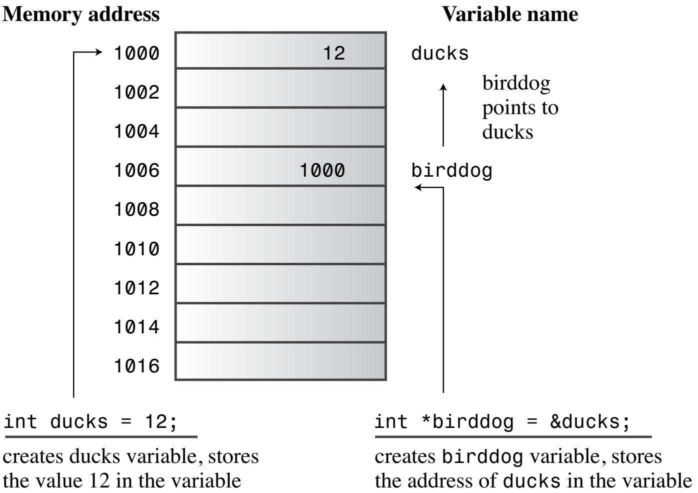


Figure 4.8 Two sides of a coin.



```
int ducks = 12;
```

creates ducks variable, stores the value 12 in the variable

```
int *birddog = &ducks;
```

creates `birddog` variable, stores the address of `ducks` in the variable

Figure 4.9 Pointers store addresses.

Initialize

```
#include <iostream>
int main()
{
  using namespace std;

  int higgens = 5;
  int * pt = &higgens;

  cout << "Value of higgens = " << higgens
  << "; Address of higgens = " << &higgens << endl;

  cout << "Value of *pt = " << *pt
  << "; Value of pt = " << pt << endl;

  return 0;
}
```