Day18 A

Young W. Lim

2017-12-06 Wed

Outline

Based on

"C How to Program",
Paul Deitel and Harvey Deitel

Initializing Structures

- initilialized using initializer lists
- when fewer initializers, the members with no initializers
  are initialized to zero or NULL
- members of global structure variables
  are intialized to zero or NULL
  unless explicitly initialized
- structure variables may be initialized
  by a structure variable assignment of the same type
  by individual member assignments

Accessing Members of Structures

- the structure member operator (.)
    - accesses a structure member via the structure variable name
- the structure pointer operator (->)
    - accesses a structure member via a pointer to the structure variable

```
struct aaa {
  int a;
  char b;
};

struct aaa A;
struct aaa *p = &A;

(*p).a  // A.a
(*p).b  // A.b

p->a
p->b
```

typedef

- a mechanism for creating <u>synonyms</u> for previously defined types
  - to create shorter type names
  - to increase portability
- names for structure types are often defined with `typedef`
- often used to create synonyms for the basic data types

Using Structures with Functions

- structure variable may be passed to functions
  - by passing individual structure members
  - by passing entire structure (passing by value, default)
  - by passing a pointer to a structure variable (passing by reference)

Passing Arrays of Structures

- arrays of structures is passed to functions
    - by reference (default)
- pass the array name by reference

Passing Structures of Arrays

- structure of an array may be used
  to pass the array by value
  - because structures are passed by values
  - so its member arrays are passed also by values

Bit Fields

- can specify the number of bits
  where an `unsigned int` or `int` member
  of structure / union is stored
- better memory utlization
  by storing data in the minimum number of bits required

Bit Field Width

- `unsigned int` / `int` *member_name* : *integer_width*
- *width* ranges from 0 to the number of bits for `int`

```
#include <stdio.h>

struct aaa {
  unsigned char a:9;
  unsigned int b:7;
};

int main(void) {
  struct aaa A;

  printf("sizeof(A)= %ld \n", sizeof(A));
}

---
t.c:4:3: error: width of ‘a’ exceeds its type
   unsigned char a:9;
   ^
```

Unnamed Bit Fields

- unnamed bit field
    - for padding bits
- unnamed bit field with a zero width
    - for the alignment of the next bit field on a new storage unit boundary