CSE240 – Introduction to Programming Languages (online)

Lecture 08:
Programming with C | typedef, enum, and struct type

Javier Gonzalez-Sanchez

javiergs@asu.edu
javiergs.engineering.asu.edu
Office Hours: By appointment

jgs

typedef and enum

typedef

- `typedef typename newname;`
- introduces a new name that becomes a synonym for the type given by the `typename` portion of the declaration.

```
typedef int booOoolean;
typedef char FlagType;

int main() {
  booOoolean x = 0;
  int counter
  FlagType y = 'A';
  // etc.
}
```

typedef

- **`typedef typename newname;`**
- introduces a new name that becomes a synonym for the type given by the **`typename`** portion of the declaration.

```
typedef int booOoolean;
typedef char FlagType;

int main() {
  booOoolean x = 0;
  int counter
  FlagType y = 'A';
  // etc.
}
```

```
#include <stdio.h>

typedef int cat;
typedef int dog;
typedef char letter;

int main()
{

    letter x = 'A';
    cat one = 1;
    cat two = 2;
    dog little = 3;
    dog big = 4;
    printf("Hello World");
    return 0;
}
```

typedef and enum

- **enum** allow us to define the allowed values for a new type

- The elements in an enumeration are **integer constants**, i.e., they are labels that represent an integer value

```
typedef enum {false, true} booOoolean;
typedef enum {Sun, Mon, Tue, Wed, Thu, Fri, Sat} days;

int main() {
  booOoolean a = false;
  int counter;
  days x = Mon, y = Fri;
  days today = x + y;
  printf("%d", today);
}
```

typedef and enum


```
#include <stdio.h>
typedef enum { red, amber, green} traffic_light;
typedef enum { No, Yes} logic;
void main( ) {
  traffic_light x = red;
  logic var = Yes;
  while (var == Yes) {
    switch (x) {
      case amber:
        x = red; printf("Red Light"); break;
      case red:
        x = green; printf("Green Light"); break;
      case green:
        x = amber; printf("Amber Light"); break;
    }
    var = No;
  }
}
```

struct

struct type

- A structure is is a composite data type declaration that defines a physically grouped list of variables to be placed under one name in a block of memory.

- It is created by the keyword **struct**.

- **Similar to a Java class; but DOES NOT allow methods**

struct type

```
struct type_name {
   type1    element1;
   type2    element2;
   . . .
   typen    elementn;
};

struct type_name a, b;
```

```
// Example
#include <stdio.h>

struct person {
    char name[30];
    int id;
};

void main( ) {
   struct person x, y;
   scanf("%s", x.name );
   scanf("%d", &x.id);
   printf("%s", x.name);
   printf("\n");
   printf("%d", x.id);
}
```

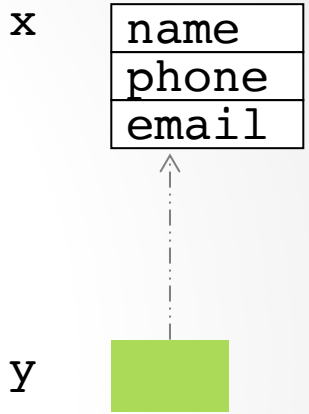
Arrays of struct

jgs

```
#include <stdio.h>

struct contact {
        char name[30];
        int  phone;
        char email[30];
};

struct contact contact_book[100]; // an array of structures

void main() {
 int index = 0;
  scanf("%d", &contact_book[index].phone);
  scanf("%s",  contact_book[index].name );
  scanf("%s",  contact_book[index].email);

  printf("\n %d", contact_book[index].phone);
  printf("\n %s", contact_book[index].name);
  printf("\n %s", contact_book[index].email);
}
```

Pointers to struct


```
struct contact {
    char name[30];
    int phone;
    char email[30];
};
main () {
  struct contact x;
  struct contact *y;

  scanf("%s", x.name); //dot with x
  scanf("%d", &x.phone);
  scanf("%s", x.email);
  printf ("%d \n", x.phone);
  y = &x;
  (*y).phone = 101010; //asterisk and dot with y
  y->phone = 404040; // arrow with y
  printf ("%d \n", y->phone);
}
```



CSE240 – Introduction to Programming Languages (online)

Javier Gonzalez-Sanchez
javiergs@asu.edu
Fall 2017