Programming Paradigms

CT331 Week 3 Lecture 2

Finlay Smith

finlay.smith@universityofgalway.ie

User Defined Types: Enums

Enum

Enumeration:

enumeration

/ɪnjuːməˈreɪʃ(ə)n/ 🔊

noun

the action of mentioning a number of things one by one.
"the complete enumeration of all possible genetic states"

- *formal*
  the action of establishing the number of something.
  "detailed enumeration of the income of the household"

User Defined Enums

Use keyword enum:

enum flag {constant1, constant2, … constantN};

Eg:

enum State {Working = 1, Failed = 0};

enum week{Mon, Tue, Wed, Thur, Fri, Sat, Sun};

User Defined Enums

Constants are immutable

Constants must have a value

Enum creates constants with default values

User Defined Enums

```c
#include<stdio.h>

enum year{Jan, Feb, Mar, Apr, May, Jun, Jul,
          Aug, Sep, Oct, Nov, Dec};

int main()
{
    int i;
    for (i=Jan; i<=Dec; i++)
        printf("%d ", i);

    return 0;
}
```

User Defined Enums

Enum constants may have the same value:

```
enum State {Working = 1, Failed = 0, Freezed = 0};
```

User Defined Enums

Enum constants may have the same value:

```
enum State {Working = 1, Failed = 0, Freezed = 0};
```

Enum constants may have mixed default & assigned values:

```
enum day {sunday = 1, monday, tuesday = 5, wednesday,
thursday = 10, friday, saturday};
```

User Defined Enums

Enum constants must be unique (within their scope):

```
enum state  {working, failed};

enum result {failed, passed};



> compiler error
```

Using Enums

```
//create a variable check of type boolean
enum boolean { false, true };
enum boolean check;


//or


enum boolean
{
    false, true
} check;
```

Using Enums

```c
#include <stdio.h>

enum Suits {HEARTS, DIAMONDS, SPADES, CLUBS};
enum Suits current;

int main(int arg, char* argv[]){

  current = DIAMONDS;

  printf("current suit: %d\n", current);

  return 0;

}
```

Enum links

https://www.programiz.com/c-programming/c-enumeration

http://www.geeksforgeeks.org/enumeration-enum-c/