Type and Constant

Computer Science Department

Eastern Washington University

Yun Tian (Tony) Ph.D.

Recall

- Basic Syntax
- Data types
- Declaration and Definition
- C variables

Topic for today

- Constants of many types
- Boolean in C

Constant Concept

- The constants refer to fixed values that the program may not alter during its execution.
- These fixed values are also called literals.
- Constants can be of any of the basic data types,
  - like an integer constant, a floating constant, a character constant, or a string literal.

Constant Concept

- The constants are treated just like regular variables except that their values cannot be modified after their definition.

Integer Constant

- An integer literal can be a decimal, octal, or hexadecimal constant.
  - A prefix specifies the base or radix: 0x or 0X for hexadecimal, 0 for octal, and nothing for decimal.
- An integer literal can also have a suffix that is a combination of U and L, for unsigned and long, respectively.
- The suffix can be uppercase or lowercase and can be in any order.

Integer Constant

- Examples

```
212      /* Legal */
215u     /* Legal */
0xFeeL   /* Legal */
078      /* Illegal: 8 is not an octal digit */
032UU    /* Illegal: cannot repeat a suffix */
```

Integer Constant

- Examples

```
85       /* decimal */
0213     /* octal */
0x4b     /* hexadecimal */
30       /* int */
30u      /* unsigned int */
30l      /* long */
30ul     /* unsigned long */
```

EASTERN WASHINGTON UNIVERSITY start something **big**

Floating-point constant

- You can represent floating point literals either in decimal form or exponential form(using **e** or **E**).
- Examples

```
3.14159     /* Legal */
314159E-5L /* Legal */
510E   /* Illegal: incomplete exponent */
210f   /* Illegal: no decimal or exponent */
.e55   /* Illegal: missing integer or fraction */
```

Character constants

- Character literals are enclosed in single quotes,
  - e.g., 'x' and can be stored in a simple variable of char type.

    char yesorno = ‘y’;
- A character literal can be a plain character (e.g., 'x') or an escape sequence (e.g., '\t').

Character constants

- There are certain characters in C when they are preceded by a backslash,
  - They will have special meaning
  - and they are used to represent like newline (\n) or tab (\t).
- Here, you have a list of some of such escape sequence codes,

Character Constants

Escape sequence	Meaning
\\	\ character
\'	' character
\"	" character
\?	? character
\a	Alert or bell
\b	Backspace
\f	Form feed
\n	Newline
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\ooo	Octal number of one to three digits
\xhh . . .	Hexadecimal number of one or more digits

Character Constants

```
#include <stdio.h>
int main()
{
	printf("Hello\tWorld\n\n");
	return 0;
}
```

String Constants

- String literals or constants are enclosed in double quotes "“.
- "hello, dear“
- "Hello\tWorld\n\n“

Define Constants

- There are two simple ways in C to define constants:
  - Using **#define** preprocessor.
  - Using **const** keyword.

Define Constants

```
#include <stdio.h>
#define LENGTH 10   //we call this Macro
#define WIDTH 5
#define NEWLINE '\n‘
int main()
{
   int area;
   area = LENGTH * WIDTH;
   printf("value of area : %d", area);
   printf("%c", NEWLINE);
   return 0;
}
```

A *macro* is a fragment of code which has been given a name. Whenever the name is used, it is replaced by the contents of the macro.

Define Constants

```
#include <stdio.h>
 int main()
 {
       const int LENGTH = 10;   // use const prefix to declare constants
       const int WIDTH = 5;
       const char NEWLINE = '\n';
       int area; area = LENGTH * WIDTH;
       printf("value of area : %d", area);
       printf("%c", NEWLINE);
       return 0;
}
```

Define Constants

```
#include <stdio.h>
 int main()
 {
      const int LENGTH = 10;   // use const prefix to declare constants
      const int WIDTH = 5;
      const char NEWLINE = '\n';
      WIDTH = 10;              // what if we change the constant?    Compile-error
      int area; area = LENGTH * WIDTH;
      printf("value of area : %d", area);
      printf("%c", NEWLINE);
      return 0;
}
```

More Data Types

- Size of basic types depends on specific platforms.
- But the following guidelines are true,
  - sizeof(short) < sizeof(int)
  - sizeof(char) < sizeof(short)
  - sizeof(long) > size(int)

More Data Types

- Type Conversion
- If type cast is not used, values are converted to the largest type.
  - E.g. all floating types are considered larger than all integer types.
  - Demo of autoCast.c

Demo of C Basic Type

```
#include <stdio.h>

int main()
{
    double d = 12.4f;
    int i = 12;
    float f = 3.2f;
    double d2 = 2.f;
    float d3 = 210e3;

    printf("size of result = %d",sizeof(d / i) );

    printf("size of int is %d",sizeof(i) );

    printf("size of double is %d",sizeof(d) );
    printf("result = %.2f",d / i );
}
```

More Data Types

- No true boolean type in C
  - boolean are integral with 0(zero) representing false,
  - any other values(non-zero) representing true.
- if(x = 0) always evaluates false
  - It assigns 0 to x, then the whole expression(x=0) get value of 0.→ false.
  - Be very careful when use this way.

Summary

- Summary
  - Constant of many types
  - Boolean in C
- Tomorrow:
  - Many operator and their precedence order in C