CSE 220 – C Programming

Expressions and Data Types

Basic Types

- Integer types
- Floating types
- Character types
- Type conversion
- Bool

Integer types

- Whole numbers
- Signed/Unsigned
  - Signed: most significant bit denotes the sign:
    - 1 if –
    - 0 if +
  - By default, integers are signed
- Length (machine dependent):
  - int: 16/32bits
  - long int, long: 32/64bits
  - short int, short: 16bits

long int ≥ int ≥ short int

- sizeof operator: number of bytes:
  - sizeof(char): 1   sizeof(int): 4   sizeof(x): 4

Exercise

Which of the below types is bigger (or equal) in size to ***int***?

- `short`
- `int`
- `long`
- None of the above

Integer types

31 ***30*** ***0***

- 111111111 00101000 00111000 00000110
  - signed int x: - or + ($2^{30} + 2^{29} +$ ….)
  - unsigned int x: 2^{31} + …
- Integer overflow:
  - 0111111111111111 + 0000000000000001
    - Assume the above are signed ints
  - Result does not fit in 16 bits
  - If signed: behavior undefined
  - If unsigned: correct answer modulo 2^n
    - n is the number of bits

Integer Constants

- C allows constants to be written in:
  - Decimal: base 10
    - Digits between 0 and 9, must not begin with 0
    - 34 199
  - Octal: base 8
    - Digits between 0 and 7, must begin with 0
    - 034 07777
  - Hexadecimal: base 16
    - Digits between 0 and 9, letters between a and f (case doesn't matter), must begin with 0x
    - 0xFA 0X2fCB 0xfddd

Floating types

- float: single precision (32bits, 6 digits)
- double: double precision (64 bits, 15 digits)
- long double: extended precision

$$\boxed{1}\ \boxed{0\ \text{-----}\ 0}\quad\boxed{1\ \text{-------------}\ 1}$$

- Float Bits:
  - Sign (1)
  - Exponent (8)
  - Fraction (23)

Floating Constants

- Contain a decimal point and/or exponent:
  - 36.0 36. .36e2 36E0 360e-1
- By default: stored as double
- To force as float:
  - 360.0f
- To force as double:
  - 360.0l
    - Note the last letter is an l (like in lama)

Character types

- char: single character
- ASCII code:
  - 7bit code, 128 characters
  - A is 1000001 (=65)
  - B is 1000010 (=66)
  - See http://www.asciitable.com/
- Treated like integers
  - char c = 65; char c = 'A';
  - c += 1 => c becomes 'B'
  - c += 'a' – 'A' => 'b'
  - char d = 32 char d = ' '
  - 'a'*'z'/'X'

int	char	int	char
48	0	90	Z
49	1	…	
…		97	a
57	9	98	b
…		…	
65	A	122	z
66	B		
…			

Character types

How to turn an uppercase character into lowercase?

```
char bigLetter = ‘Q’;

char smallLetter1 = bigLetter + 32;

char smallLetter2 = bigLetter + (‘q’
– ‘Q’);

char smallLetter3 = bigLetter + (‘a’
– ‘A’);


char choice;

scanf(“%c”, &choice);

If (choice >= ‘A’ && choice <= ‘Z’)

      choice = choice + (‘a’ – ‘A’);
```

int	char	int	char
48	0	90	Z
49	1	…	
…		97	a
57	9	98	b
…		…	
65	A	122	z
66	B		
…			

Exercise

Write a program that reads a character from the user and output all characters starting from that character to Z.

Example: if user enters P, the program prints: P Q R S T U V W X Y Z

```
int main(void) {
   char startingLetter, tmpLetter;
   scanf(“%c”, &startingLetter);
   for (tmpLetter = startingLetter; tmpLetter <=
‘Z’; tmpLetter++)
      printf(“%c  ”, tmpLetter);

   return 0;
}
```

Reading and Writing

- scanf with %c to read a single character
- Does not skip spaces by default
  - scanf(“%c”, &mychar);
    - If the first character of input is whitespace, that is what is put in the char.
  - scanf(“ %c”, &mychar); //space in format string
    - This code ignores leading whitespace and stores the first letter/digit/punctuation that is sees
- printf(“%c”, mychar);
    - You print with the %c specifier.

Example

```
#include <stdio.h>
int main(void) {
        int age;
        char favorite_letter;

        printf("What is your age?\n");
        scanf("%d", &age);
        printf("Your age is %d.\n", age);

        printf("What is your favorite letter?\n");
        scanf(" %c", &favorite_letter); // Note the space
        printf("Your favorite letter is '%c'.", favorite_letter);
        return 0;
}
```

getchar and putchar

- ch = getchar();
- putchar(ch);
- Faster than scanf and printf
- But less useful.

Exercise

What is the output of the following program if the user enters:

a newline b newline c newline

```
char ch;
for (int idx=0; idx<3; idx++) {
   printf("Enter a single character >> ");
   ch = getchar();
   putchar(ch);
}
```

#Iterations: 3
The 3 characters read: a newline b

```
Enter a single character >> a
aEnter a single character >>
Enter a single character >> b
b
```

getchar and putchar are redundant to scanf

- For the simplicity of the class, I will not ask you to use getchar or putchar in any assignment, exam or lab exercise.

Type Conversion

- Implicit conversion:
  - When operands have different types
  - A.k.a, when right side of assignment does not match left side
- Main rule:
  - Convert to narrowest type that fits (promotion)

    myFloat + myInt: safer to convert int to float
  - float => double => long double
  - int => unsigned int => long int => unsigned long int

Type Conversion

- Explicit conversion: Casting
  - int j = (int) f;
  - float fraction = myFloat – (int) myFloat;
  - float result = ((float) x) / y;
  - float result = 1.0f/2;
- Casts treated as unary operators, have high precedence
  - float result = (float) x / y;

bool Type

- An optional type you can add to your program is `bool`
- `bool` is either `1` or `0` and can be initialized with `true` or `false`.
- Examples:
  - `bool b = 1;`
  - `bool is_cool = true;`
  - `is_cool = 0;`
  - `b = false;`
  - `bool favorite_class = class_num == 220;`
- You need to include `stdbool.h` at the top of your file to use `bool`.
  - See example next slide.

Type Definition

```
#include <stdio.h>
#include <stdbool.h>
int main(void) {
  int answer;
  bool match;
  scanf(“%d”, & answer);
  match = answer == 5;
  printf("%d", match);
  return 0;
}
```

Basic Types

- Integer types
- Floating types
- Character types
- Type conversion
- Bool