# PDS Lab Section 15

Helper slides on using characters

## Using characters

- Use the char data type to declare variables
- Use %c to read one character in scanf and %c to print one character in printf
- Char variables can be used in exactly the same way as int and float
  - Store character values, compare, add/subtract/…
- Character constants
  - Contains a single character enclosed within a pair of single quote marks
  - Examples :: ‘2’, ‘+’, ‘Z’

```
int main()
{
        char c1, c2;
        scanf(“%c%c”, &c1, &c2);
        printf(“the characters read are %c, %c\n”, c1, c2);
        if (c1 != c2)
                printf(“%c is not equal to %c\n”, c1, c2);
}
```

```
int main()
{
        char c1;
        scanf(“%c”, &c1);
        if (c1 == ‘+’ || c1 == ‘-’)
                printf(“%c is one of plus and minus\n”, c1);
}
```

## VERY IMPORTANT

- Remember that everything that you enter from the keyboard is a character, like when you give a space or hit Enter (the character ‘\n’)
- A scanf will read any character
- So to read more than one character in your program, enter all characters with no space or Enter key press, press the Enter key at the end

## Example

- Suppose in your program you have

  scanf(“%c%c”, &c1, &c2);

  scanf(“%c”, &c3);

  - So you want to enter three characters
- Suppose you want to enter the characters x, y, z
  - Then, while giving the input, type xyz and then only press Enter
    - This puts c1 = ‘x’, c2 = ‘y’, c3 = ‘z’   CORRECT
  - If you type x, then press Enter, then type y, press Enter, type z press Enter (as you are doing now with int and float), you get
    - c1 = ‘x’, c2 = ‘\n’ (the character corresponding to key press Enter), c3 = ‘y’   **WRONG**
  - If you type space instead of Enter between x and y and z, similarly the character for space will be stored   **WRONG**
- Just type a small program to read and print 2 characters and make these mistakes and see what happens **(TRY IT BEFORE NEXT CLASS)**

## More on the char type

- Is actually stored as an integer internally
- Each character has an integer code (between 32 and 127) associated with it (ASCII code value)
- Internally, storing a character means storing its integer code
  - Printing the character with %d in printf will print its ascii value
- All operators that are allowed on int are allowed on char
  - 32 + ‘a’ will evaluate to 32 + 97 (the integer ascii code of the character ‘a’) = 129
  - Same for other operators
- Can switch on chars constants in switch, as they are integer constants

## Example

```
int a;
a = ‘c’ * 3 + 5;
printf(“%d”, a);
```

**Will print 302 (99*3 + 5)**

**(ASCII code of ‘c’ = 99)**

```
char c = ‘A’;
printf(“%c = %d”, c, c);
```

**Will print A = 65**

**(ASCII code of ‘A’ = 65)**

**Assigning char to int is fine. But other way round is dangerous, as size of int is larger (usually 4 bytes, char is 1 byte)**

## More on ASCII Code

- The code of a character is represented by an 8-bit unit. Since an 8-bit unit can hold a total of $2^8$=256 values and the computer character set is much smaller than that, some values of this 8-bit unit do not correspond to visible characters
- You do NOT need tor remember any asci code
- But we can do interesting thing with char type knowing some properties of Ascii codes

- Ascii codes of the characters ‘a’ to ‘z’ are contiguous (no gaps)
  - 97 to 122
- Ascii codes of the characters ‘A’ to ‘Z’ are contiguous (no gaps)
  - 65 to 90
- Ascii codes of the characters ‘0’ to ‘9’ are contiguous (no gaps)
  - 60 to 71

## Example: checking if a character is a lowercase alphabet

```
int main()
{
        char c1;
        scanf("%c", &c1);
        /* the asci code of c1 must lie between the
           asci cods of 'a' and 'z' */
        if (c1 >= 'a' && c1<= 'z')
            printf("%c is a lowercase alphabet\n", c1);
        else printf("%c is not a lowercase alphabet\n", c1);
}
```

## Example: converting a character from lowercase to uppercase

```
int main()
{
        char c1;
        scanf("%c", &c1);
        /* convert to uppercase if lowercase, else leave as it is */
        if (c1 >= 'a' && c1<= 'z')
        /* since asci codes of uppercase letters are contiguous, the
           uppercase version of c1 will be as far away from the asci code
           of 'A' as it is from the asci code of 'a' */
        c1 = 'A' + (c1 – 'a');
        printf(("The letter is %c\n", c1);
}
```

## Simple practice problems

- In a for loop, print out all uppercase alphabets and their asci codes (initialize a char variable ch with ‘A’, then print ch, ch+1, ch+2 (upto??) as both %c and %d
- Read in a character and check if it is an alphanumeric character (alphabet or digits only)
- Read in two characters and if both of them are digits between 0 and 9, then compute their sums in decimal (not the sum of their asci code values), store in an int type variable and print it. So if you give the characters ‘3’ and ‘4’, it should print 7.
- Keep reading and printing characters in a loop, until the character ‘1’ is entered. When ‘1’ is entered print a message and stop.