C Programming for MSc

Lecture 4:
Keyboard buffer and Functions

Prof. Stephen Smith

E-mail: stephen.smith@york.ac.uk

Based on lecture notes by Dr Julian Miller

Contents

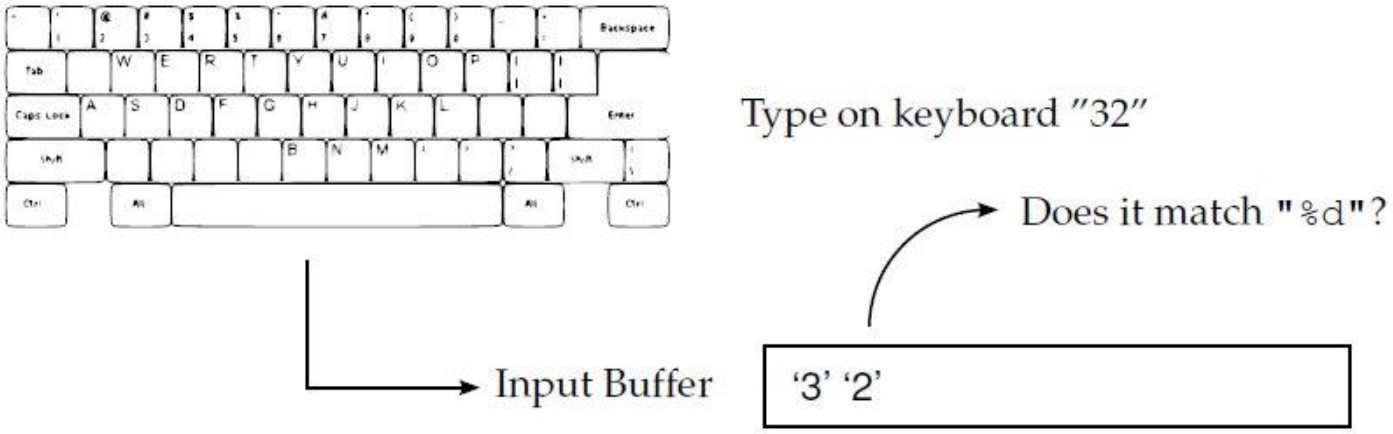
- **`scanf()`** and the input buffer
- Flushing the input buffer
- Functions
- Function prototypes
- Library functions

`scanf()` and the keyboard buffer (1)

- `scanf()` gets its data from the keyboard buffer

```
int number;

printf("Please enter an integer number: ");
scanf("%d", &number);
```



`scanf()` and the keyboard buffer (2)

- `scanf()` also returns a number equal to the number of placeholders matched
- Example

```
int num_placeholders_matched;
int number;

printf("Please enter an integer number: ");
num_placeholders_matched = scanf("%d", &number);
```

fflush(stdin)

- When the characters in the keyboard buffer *do not match* the placeholders, **scanf()** leaves them in the buffer
  - This will cause a second **scanf()** call to not work properly. To avoid this use **fflush(stdin)**

```
int num_placeholders_matched, number;

printf("Please enter an integer number: ");
num_placeholders_matched = scanf("%d", &number);

if (num_placeholders_matched != 1)
{
  fflush(stdin);
}
```

Functions

- Functions are parts of a program that can be run by other parts of a program
- You have used predefined functions (e.g. **`printf(), scanf()`**)
- Functions look like this

  `type function_name(<parameter list>)`
- Where ‘type’ means a C data type, e.g.
  - **`int, char, double, unsigned, void`**

https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html

User defined functions

```
#include <stdio.h>

void display_welcome(void)
{
    printf("Welcome\n");
}

int square(int number)
{
    return number*number;
}

int main(void)
{
    int value;

    display_welcome();

    printf("Enter an integer: ");
    scanf("%d, &value");

    printf("The square of %d is %d\n2",value,square(value));

    return 0;
}
```

Function prototypes

- These are declarations of functions
- When they are used the order of functions is unimportant, provided it occurs after the protype
- They allow the compiler to check the function call against its definition to see if there are any differences in the arguments and the return type
- e.g.

```
void display_welcome(void);

int square(int number);
```

Designing Boolean functions

- Often in program complex decisions need to me made more than once. Writing small functions that test conditions and return 1 or 0 can be very useful
- Example 1: To check if the projectile is above ground

```
int projectile_above_ground(int pos_y, int y_ground
{
   return (pos_y < y_ground);
}
```

- Example 2: Suppose a random box is drawn on the screen and the projectile is not allowed to penetrate it

```
int projectile_collides_with_box(int xprojectile, int yprojectile,
                                 int xbox, int ybox,
                                 int boxwidth, int boxheight)
{
    int xinvalid, yinvalid;

    xinvalid = (xprojectile >= xbox) && (xprojectile <= xbox+boxwidth);
    yinvalid = (yprojectile >= ybox) && (yprojectile <= ybox+boxheight);
}
```

Mathematical functions: math.h

- math.h is one of many library functions available for C
- In math.h there are many mathematical functions that you can use:

  `sqrt(),pow(),sin(),cos(),asin(),...`

  – they all return type **`double`** and take type **`double`** as arguments

  – however you can still use them with **`int`** arguments as they get converted to type **`double`**.

Mathematical functions: `pow()`

```
double pow(double x, double y);
```

The **`pow()`** function raises the first argument to the power of the second argument and returns the result. For example:

```
result = pow(2, 3);
```

would assign the value 8 to the variable **`result`**. **`pow()`** is defined in math.h

```
#include <math.h>
```

Summary

- Considered features of the keyboard buffer
- Introduced functions