Day16.C

C Formatted IO

Young W. Lim

December 9, 2017

0.1 scanf()

```
:::::::::::::::
h1.c
:::::::::::::::
#include <stdio.h>

#define CASE 3 // 1, 2, 3, 4

#ifndef CASE
#define 1
#endif

int main(void) {

   int i,r;
   char c;


#if CASE == 1

   puts("======(a)=======================================");

   puts("Type 123(LF)--------%d---------");
   r=scanf("%d", &i);
   printf("i=%d r=%d item(s) \n\n", i, r);  // 1 ... 123

   puts("Type 123A(LF)-------%d-----%c--");
   r=scanf("%d", &i);
   printf("i=%d r=%d item(s) \n\n", i, r);  // 2 ... 123
   r=scanf("%c", &c);
   printf("c=%c r=%d item(s) \n\n", c, r);  // 3 ... A

#elif CASE == 2

   puts("======(b)=======================================");

   puts("Type 123 A(LF)------%d %c------");
   r=scanf("%d %c", &i, &c);
   printf("i=%d c=%c r=%d item(s) \n\n", i, c, r); // 1 ... 123 A

   puts("Type 123A(LF)-------%d%c------");
   r=scanf("%d%c", &i, &c);
   printf("i=%d c=%c r=%d item(s) \n\n", i, c, r); // 2 ... 123 A

   puts("Type 123A(LF)-------%d %c-----");
   r=scanf("%d %c", &i, &c);
   printf("i=%d c=%c r=%d item(s) \n\n", i, c, r); // 3 ... 123 A

   puts("Type 123   A(LF)----%d %c------");
```

```
   r=scanf("%d %c", &i, &c);
   printf("i=%d c=%c r=%d item(s) \n\n", i, c, r); // 4 ... 123 A

   puts("Type 123(LF)--------%d%c-------");
   r=scanf("%d%c", &i, &c);
   printf("i=%d c=%c r=%d item(s) \n\n", i, c, r); // 5 ... 123 (LF)


#elif CASE == 3

   puts("======(c)=======================================");

   puts("Type 123(LF)(LF)1(LF)----%d\\n-------");
   r=scanf("%d ", &i);
   printf("i=%d r=%d item(s) \n\n", i, r);  // 1 ... 123

   puts("Type 123(LF)1(LF)--------%d\\n-------");
   r=scanf("%d ", &i);
   printf("i=%d r=%d item(s) \n\n", i, r);  // 2 ... 1

   puts("Type 123 a b c (LF)-------%d\\n-------");
   r=scanf("%d ", &i);
   printf("i=%d r=%d item(s) \n\n", i, r);  // 3 ... 123

   r=scanf("%d", &i);
   printf("i=%d r=%d item(s) \n\n", i, r);  // 4 ... 1

   r=scanf("%d", &i);
   printf("i=%d r=%d item(s) \n\n", i, r);  // 5 ... 123

   r=scanf(" %c", &c);
   printf("c=%c r=%d item(s) \n\n", c, r);  // 6 ... a

   r=scanf(" %c", &c);
   printf("c=%c r=%d item(s) \n\n", c, r);  // 7 ... b

   r=scanf(" %c", &c);
   printf("c=%c r=%d item(s) \n\n", c, r);  // 8 ... c

#elif CASE == 4

   puts("======(d)=======================================");

   puts("Type 123(LF)(LF)1(LF)----%d\\n-------");
   r=scanf("%d\n", &i);
   printf("i=%d r=%d item(s) \n\n", i, r);   // 1 ... 123

   puts("Type 123(LF)1(LF)--------%d\\n-------");
   r=scanf("%d\n", &i);
   printf("i=%d r=%d item(s) \n\n", i, r);   // 2 ... 1
```

```
   puts("Type 123 a b c (LF)-------%d\\n-------");
   r=scanf("%d\n", &i);
   printf("i=%d r=%d item(s) \n\n", i, r);   // 3 ... 123

   r=scanf("%d\n", &i);
   printf("i=%d r=%d item(s) \n\n", i, r);   // 4 ... 1

   r=scanf("%d\n", &i);
   printf("i=%d r=%d item(s) \n\n", i, r);   // 5 ... 123

#else
   puts("CASE = 1, 2, 3, or 4 ");
#endif

}


:::::::::::::::
h1.out
:::::::::::::::
```

```
======(a)=====================================
Type 123(LF)--------%d---------
123
i=123 r=1 item(s)

Type 123A(LF)-------%d-----%c--
123A
i=123 r=1 item(s)

c=A r=1 item(s)
```

```
======(b)=====================================
Type 123 A(LF)------%d %c------
123 A
i=123 c=A r=2 item(s)

Type 123A(LF)-------%d%c------
123A
i=123 c=A r=2 item(s)

Type 123A(LF)-------%d %c-----
123A
i=123 c=A r=2 item(s)

Type 123   A(LF)----%d %c------
123   A
i=123 c=A r=2 item(s)

Type 123(LF)--------%d%c-------
123
i=123 c=
 r=2 item(s)
```

```
======(c)=====================================
Type 123(LF)(LF)1(LF)----%d\n-------
123

1
i=123 r=1 item(s)

Type 123(LF)1(LF)--------%d\n-------
123
i=1 r=1 item(s)

Type 123 a b c (LF)-------%d\n-------
1
i=123 r=1 item(s)

i=1 r=1 item(s)

123 a b c
i=123 r=1 item(s)

c=a r=1 item(s)

c=b r=1 item(s)

c=c r=1 item(s)
```

```
======(d)=====================================
Type 123(LF)(LF)1(LF)----%d\n-------
123

1
i=123 r=1 item(s)

Type 123(LF)1(LF)--------%d\n-------
123
d=1 r=1 item(s)

Type 123 a b c (LF)-------%d\n-------
1
i=123 r=1 item(s)

123 a b c
i=1 r=1 item(s)

i=123 r=1 item(s)
```

case (a)

1.
   - `r=scanf("%d", &i);`
   - input stream : 123 ↵
   - `%d` matches `123`
   - `printf("i=%d r=%d item(s) \n\n", i, r);`
   - output stream : i=123 r=1 item(s)↵↵

2.
   - `r=scanf("%d", &i);`
   - input stream : 123 A ↵
   - `%d` matches `123`
   - `printf("i=%d r=%d item(s) \n\n", i, r);`
   - output stream : i=123 r=1 item(s) ↵↵

   - `r=scanf("%c", &c);`
   - input stream updated : A ↵
   - `%c` matches `A`
   - `printf("c=%c r=%d item(s) \n\n", c, r);`
   - output stream : c=A r=1 item(s) ↵↵

case (b)

1.
   - `r=scanf("%d %c", &i, &c));`
   - input stream : 123 ␣ A ↵
   - %d matches 123
   - " " matches " "
   - %c matches A
   - `printf("i=%d c=%c r=%d item(s) \n\n", i, c, r);`
   - output stream : i=123 c=A r=2 item(s)↵↵

2.
   - `r=scanf("%d%c", &i, &c));`
   - input stream : 123 A ↵
   - %d matches 123
   - %c matches A
   - `printf("i=%d c=%c r=%d item(s) \n\n", i, c, r);`
   - output stream : i=123 c=A r=2 item(s)↵↵

3.
   - `r=scanf("%d %c", &i, &c));`
   - input stream : 123 A ↵
   - %d matches 123
   - " " matches nothing
   - %c matches A
   - `printf("i=%d c=%c r=%d item(s) \n\n", i, c, r);`
   - output stream : i=123 c=A r=2 item(s)↵↵

4.
   - `r=scanf("%d %c", &i, &c));`
   - input stream : 123 ␣␣ A ↵
   - %d matches 123
   - " " matches " "
   - %c matches A
   - `printf("i=%d c=%c r=%d item(s) \n\n", i, c, r);`
   - output stream : i=123 c=A r=2 item(s)↵↵

5.
   - `r=scanf("%d%c", &i, &c));`
   - input stream : 123 ↵
   - %d matches 123
   - %c matches ↵
   - `printf("i=%d c=%c r=%d item(s) \n\n", i, c, r);`
   - output stream : i=123 c=↵ r=2 item(s) ↵↵

case (c)

1.
   - `r=scanf("%d ", &i);`
   - input stream : 123 ↵ ↵ 1 ↵
   - `%d` matches 123
   - `" "` matches ↵ ↵
   - `printf("i=%d r=%d item(s) \n\n", i, r);`
   - output stream : i=123 r=1 item(s)↵↵

2.
   - `r=scanf("%d ", &i);`
   - input stream updated : 1 ↵ , 123 ↵
   - `%d` matches 1
   - `" "` matches ↵
   - `printf("i=%d r=%d item(s) \n\n", i, r);`
   - output stream : i=1 r=1 item(s)↵↵

3.
   - `r=scanf("%d ", &i);`
   - input stream updated: 123 ↵, 1 ↵
   - `%d` matches 123
   - `" "` matches ↵
   - `printf("i=%d r=%d item(s) \n\n", i, r);`
   - output stream : i=123 r=1 item(s)↵↵

4.
   - `r=scanf("%d", &i);` `"%d"`
   - input stream updagted: 1 ↵ (without additional input)
   - `%d` matches 1
   - `printf("i=%d r=%d item(s) \n\n", i, r);`
   - output stream : i=1 r=1 item(s)↵↵

5.
   - `r=scanf("%d", &i);`
   - input stream : 123 ␣ a ␣ b ␣ c ␣ ↵
   - `%d` matches 123
   - `printf("i=%d r=%d item(s) \n\n", i, r);`
   - output stream : i=123 r=1 item(s) ↵↵

6.
- `r=scanf(" %c", &c);`
- input stream updated: ␣ a ␣ b ␣ c ␣ ↵
- `" "` matches `" "`
- `%c` matches a
- `printf("c=%c r=%d item(s) \n\n", c, r);`
- output stream : c=a r=1 item(s) ↵↵

7.
- `r=scanf(" %c", &c);`
- input stream updated: ␣ b ␣ c ␣ ↵
- `" "` matches `" "`
- `%c` matches b
- `printf("c=%c r=%d item(s) \n\n", c, r);`
- output stream : c=b r=1 item(s) ↵↵

8.
- `r=scanf(" %c", &c);`
- input stream updated: ␣ c ␣ ↵
- `" "` matches `" "`
- `%c` matches c
- `printf("c=%c r=%d item(s) \n\n", c, r);`
- output stream : c=c r=1 item(s) ↵↵

case (d)

1.
   - `r=scanf("%d\n", &i);`
   - input stream : 123 ↵ ↵ 1 ↵
   - `%d` matches 123
   - `\n` matches ↵ ↵
   - `printf("i=%d r=%d item(s) \n\n", i, r);`
   - output stream : i=123 r=1 item(s)↵↵

2.
   - `r=scanf("%d\n", &i);`
   - input stream updated : 1 ↵ , 123 ↵
   - `%d` matches 1
   - `\n` matches ↵
   - `printf("i=%d r=%d item(s) \n\n", i, r);`
   - output stream : i=1 r=1 item(s)↵↵

3.
   - `r=scanf("%d\n", &i);`
   - input stream updated: 123 ↵, 1 ↵
   - `%d` matches 123
   - `\n` matches ↵
   - `printf("i=%d r=%d item(s) \n\n", i, r);`
   - output stream : i=123 r=1 item(s)↵↵

4.
   - `r=scanf("%d\n", &i);` `"%d\n"`
   - input stream updagted: 1 ↵ , 123 ␣ a ␣ b ␣ c ␣ ↵
   - `%d` matches 1
   - `\n` matches ↵
   - `printf("i=%d r=%d item(s) \n\n", i, r);`
   - output stream : i=1 r=1 item(s)↵↵

5.
   - `r=scanf("%d\n", &i);`
   - input stream updated : 123 ␣ a ␣ b ␣ c ␣ ↵
   - `%d` matches 123
   - `\n` matches ↵
   - `printf("i=%d r=%d item(s) \n\n", i, r);`
   - output stream : i=123 r=1 item(s) ↵↵

0.2 printf()

```
::::::::::::::
h2.c
::::::::::::::
#include <stdio.h>
#include <math.h>

//==========(0)==============================
#ifndef PI
#define PI 3.14159265358979323846
#endif

#define PIL 3.14159265358979323846L

// most compiler in <math.h>
// #define M_PI 3.14159265358979323846
//                1234567890123456789O

int main(void) {

  printf("--------------------------\n");
  printf("      1234567890123456789O\n");
  printf("      ...................\n");
  printf("PI= 3.14159265358979323846\n");
  printf("--------------------------\n");


//==========(a)==============================
  printf("PI= %f   \t\t\t\t%%f\n", PI);
  printf("PI= %e   \t\t\t%%e\n", PI);
  printf("PI= %g   \t\t\t\t%%g\n\n", PI);

//==========(b)==============================
  printf("PI= %.20f \t\t%%.20f\n", PI);
  printf("PI= %.20e \t\t%%.20e\n", PI);
  printf("PI= %.20g \t\t%%.20g\n\n", PI);

//==========(c)==============================
  printf("PI= %.20f \t\t%%.20lf\n", PI);
  printf("PI= %.20e \t\t%%.20le\n", PI);
  printf("PI= %.20g \t\t%%.20lg\n\n", PI);

//==========(d)==============================
  printf("100PI= %f   \t\t\t%%f\n", PI*100);
  printf("100PI= %e   \t\t\t%%e\n", PI*100);
  printf("100PI= %g   \t\t\t%%g\n\n", PI*100);

//==========(e)==============================
  printf("100PI= %.20f \t%%.20f\n", PI*100);
```

```
  printf("100PI= %.20e \t%%.20e\n", PI*100);
  printf("100PI= %.20g \t\t%%.20g\n\n", PI*100);

//==========(f)==============================
  printf("100PI= %.20f \t%%.20f\n", PI*100);
  printf("100PI= %.20e \t%%.20e\n", PI*100);
  printf("100PI= %.20g \t\t%%.20g\n\n", PI*100);

//==========( )==============================
  printf("PI= %.22Lf \t\t%%.22Lf\n", PIL);
  printf("PI= %.22Le \t%%.22Le\n", PIL);
  printf("PI= %.22Lg \t\t%%.23Lg\n", PIL);


}
```

```
::::::::::::::
h2.out
::::::::::::::
--------------------------
      12345678901234567890
      ....................
PI= 3.14159265358979323846
--------------------------
PI= 3.141593                              %f
PI= 3.141593e+00                          %e
PI= 3.14159                               %g

PI= 3.14159265358979311600                %.20f
PI= 3.141592653589793116000e+00           %.20e
PI= 3.14159265358979311600                %.20g

PI= 3.14159265358979311600                %.20lf
PI= 3.141592653589793116000e+00           %.20le
PI= 3.14159265358979311600                %.20lg

100PI= 314.159265                         %f
100PI= 3.141593e+02                       %e
100PI= 314.159                            %g

100PI= 314.15926535897932581065           %.20f
100PI= 3.141592653589793258110e+02        %.20e
100PI= 314.15926535897932581              %.20g

100PI= 314.15926535897932581065           %.20f
100PI= 3.141592653589793258110e+02        %.20e
100PI= 314.15926535897932581              %.20g

PI= 3.1415926535897932385128              %.22Lf
PI= 3.14159265358979323851280e+00         %.22Le
PI= 3.14159265358979323851300             %.23Lg
```

macro

- if `MACRO` is defined do the controlled text

```
#ifdef MACRO
controlled text
#endif /* MACRO */
```

- if `MACRO` is not defined do the controlled text

```
#ifndef MACRO
controlled text
#endif /* MACRO */
```

format conversion specifier

- `%e` exponential form ($m \cdot 10^e$)
  - 6 default digits after the decimal point
  - 3.1 4 1 5 9 3 e+00
- `%f` normal (fixed point) form
  - 6 default digits after the decimal point
  - 3.1 4 1 5 9 3
- `%g` either exponential or normal form
  - 6 default digits excluding decimal point
  - 3.1 4 1 5 9

double precision

- ----------
- 12345678901234567890
- ....
- PI= 3.14159265358979323846
- PI= 3.14159265358979311600 %.20f
- PI= 3.14159265358979311600e+00 %.20e
- PI= 3.14159265358979116 %.20g
- PI= 3.14159265358979311600 %.20lf
- PI= 3.14159265358979311600e+00 %.20le
- PI= 3.14159265358979116 %.20lg

- the followings are from

  `https://en.wikipedia.org/wiki/Double-precision_floating-point_format`

- Sign bit: 1 bit
- Exponent: 11 bits
- Significand precision: 53 bits (52 explicitly stored)
- The exponent field can be interpreted as either an 11-bit signed integer from 1024 to 1023 (2's complement) or an 11-bit unsigned integer from 0 to 2047, which is the accepted biased form in the IEEE 754 binary64 definition. If the unsigned integer format is used, the exponent value used in the arithmetic is the exponent shifted by a bias for the IEEE 754 binary64 case, an exponent value of 1023 represents the actual zero (i.e. for 2^{e-1023} to be one, e must be 1023). Exponents range from 1022 to +1023 because exponents of 1023 (all 0s) and +1024 (all 1s) are reserved for special numbers.
- The 53-bit significand precision gives from 15 to 17 significant decimal digits precision ($2^{-53} \approx 1.11 \times 10^{-16}$).

- `%f` or `%lf` for `double` type numbers
- `%Lf` for `long double` type numbers