The C Programming Language – Part 2

(with material from Dr. Bin Ren, William & Mary Computer Science)

Overview

- Input/Output
- Structures and Arrays

Basic I/O

- **character-based**
  - putchar (c) – output
  - getchar () – input

- **formatted - standard I/O**
  - printf (stuff goes in here) - output
  - scanf (stuff goes in here) - input *** white space is important
  - format specifiers (% before specifier)

Basic I/O

```
#include <stdio.h>

int main (void)
{
  int i = 65;  /* what if 258 instead of 65? */
  char a;

  printf ("i = %d\n", i);

  printf (“i output with putchar = ");
  putchar (i);
  printf ("\n");

  a = (char) i;
  printf ("a = %c\n", a);

  printf ("i = %c\n", i);
  getchar ();

  return(0);
}
```

Basic I/O

```
#include <stdio.h>

int main()
{
    int x;

    scanf ("%d\n", &x);
    printf ("x = %d\n", x);
}
```

- **pointer needed for scanf variable**
  - must use a pointer to the variable to change the value of the variable

Conversion Characters

- % plus character
  - placeholder for variable replacement
  - determines interpretation of bits

Conversion Character	Displays Argument (Variable's Contents) As
%c	Single character
%d	Signed decimal integer (int)
%e	Signed floating-point value in E notation
%f	Signed floating-point value (float)
%g	Signed value in %e or %f format, whichever is shorter
%i	Signed decimal integer (int)
%o	Unsigned octal (base 8) integer (int)
%s	String of text
%u	Unsigned decimal integer (int)
%x	Unsigned hexadecimal (base 16) integer (int)
%%	(percent character)

Basic I/O

```
#include <stdio.h>

int main (void)
{
  int i = 65;

  printf ("i = %d\n", i);

  printf ("i output with putchar = ");
  putchar (i);
  printf ("\n");

 printf ("i = %i\n", i);

  i = getchar ();
  printf ("i = %c\n", i);
  printf ("i = %x\n", i);
  printf ("i = %d\n", i);

  return(0);
}
```

Basic I/O

```
#include <stdio.h>

int main ()
{
    int x;
    scanf ("%d", &x);    /* why need & ? */
    printf ("%d\n", x);

    float var;
    scanf ("%f", &var);    printf ("%f\n", var);
    scanf ("%d", &var);    printf ("%d\n", var);
    scanf ("%lf", &var);   printf ("%lf\n", var);

    int first, second;
    scanf ("%d %d", &first, &second);

    int i, j;
    scanf (" %d %*d %*d%*d %d ", &i, &j);
}
```

Conversion Characters for Output (printf)

Character	Argument type; Printed As
d,i	int; signed decimal notation.
o	int; unsigned octal notation (without a leading zero).
x,X	unsigned int; unsigned hexadecimal notation (without a leading 0xor 0X), using abcdef for 0x or ABCDEF for 0X.
u	int; unsigned decimal notation.
c	int; single character, after conversion to unsigned char
s	characters from the string are printed until a '\0' is reached or until the number of characters indicated by the precision have been printed.
f	double; decimal notation of the form [-]mmm.ddd, where the number of d's is given by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.
e,E	double; decimal notation of the form [-]m.dddddde+/-xx or [-]m.ddddddE+/-xx, where the number of d's is specified by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.
g,G	double; %eor %E is used if the exponent is less than -4 or greater than or equal to the precision; otherwise % fis used. Trailing zeros and a trailing decimal point are not printed.

Source: xkcd.com/571

Printing Decimal and Floating Point

- **integers: %nd**
  - n = width of the whole number portion for decimal integers
- **float: %m.nf**
  - m = total character width, including decimal point
  - n = precision width after decimal

%d	print as decimal integer
%6d	print as decimal integer, at least 6 characters wide
%f	print as floating point
%6f	print as floating point, at least 6 characters wide
%.2f	print as floating point, 2 characters after decimal point
%6.2f	print as floating point, at least 6 wide and 2 after decimal point

printf Examples

- **print two values separated by tab**

```
printf ("%d\t%d\n", fahr, celsius);
```

- **print the first number in a field three digits wide, and the second in a field six digits wide**

```
printf ("%3d %6d\n", fahr, celsius);
```

- **each % construction in the first argument of printf is paired with the corresponding second argument, third argument, etc.;**
  - must match up properly by number and type, or wrong values printed

```
printf ("\na=%f\nb=%f\nc=%f\nPI=%f", a, b, c, d);
c = a + b;   printf ("%d + %d = %d\n", a, b, c);
```

Escape Sequences

Escape code	Description
\n	newline
\r	carriage return
\t	tab
\v	vertical tab
\b	backspace
\f	form feed (page feed)
\a	alert (beep)
\'	single quote (')
\"	double quote (")
\?	question mark (?)
\\	backslash (\)

www.cpp.com

Formatted I/O

- **printf and scanf**
  - both formatted I/O
  - both use standard I/O location

- **printf**
  - converts values to character form according to format string
  - outputs to stdout

- **scanf**
  - converts characters according to the format string, followed by pointer arguments indicating where the resulting values are stored
  - inputs from stdin

scanf

- **requires two parameters**
  - format string argument with specifiers
  - set of variable pointers to store corresponding values
- **format string**
  - skips over all leading white space (spaces, tabs, newlines)
  - % and type indicator
    - in between: maximum field-width, type indicator modifier, or * (input suppression)
  - input stops at end of format string, type mismatch in reading
    - next call to scanf resume searching for input of correct type where previous scanf left off
- **return value**
  - # of values converted

scanf Conversion Strings

FORMAT	MEANING	VARIABLE TYPE
%d	read an integer value	int
%ld	read a long integer value	long
%f	read a real value	float
%lf	read a double precision real value	double
%c	read a character	char
%s	read a character string from the input	array of char

scanf Examples

```
int day, month, year;
scanf ("%d/%d/%d", &month, &day, &year);
Input: 01/29/64
month == 1  day == 29  year == 64
```

```
int anInt;
scanf ("%*s %i", &anInt);
Input:
Age: 29
anInt == 29
```

```
int anInt;
scanf ("%i%%", &anInt);
Input: 23%
anInt == 23
```

```
double d;
scanf ("%lf", &d);
Input: 3.14
d == 3.14
```

```
int anInt, anInt2;
scanf ("%2i", &anInt);
scanf (“%2i”, &anInt2);
Input:
2345
anInt == 23
anInt2 == 45
```

```
int anInt; long l;
scanf ("%d  %ld", &anInt, &l);
Input:
-23 200
anInt == -23
l == 200
```

```
string s;
scanf ("%9s", s);
Input:
VeryLongString
s == “VeryLongS”
```

- Note: pressing the Enter key means you have entered a character

More scanf Examples

Letter	Type of Matching Argument	Auto-skip; Leading White-Space	Example	Sample Matching Input
%	% (a literal, matched but not converted or assigned)	no	int anInt; scanf("%i%%", &anInt);	23%
d	int	yes	int anInt; long l; scanf("%d %ld", &anInt, &l);	-23 200
i	int	yes	int anInt; scanf("%i", &anInt);	0x23
o	unsigned int	yes	unsigned int aUInt; scanf("%o", &aUInt);	023
u	unsigned int	yes	unsigned int aUInt; scanf("%u", &aUInt);	23
x	unsigned int	yes	unsigned int aUInt; scanf("%d", &aUInt);	1A
a, e, f, g	float or double	yes	float f; double d; scanf("%f %lf", &f, &d);	1.2 3.4
c	char	no	char ch; scanf(" %c", &ch);	Q
s	array of char	yes	char s[30]; scanf("%29s", s);	hello
n	int	no	int x, cnt; scanf("X: %d%n", &x, &cnt);	X: 123 (*cnt==6*)

[	array of char	no	char s1[64], s2[64];
			scanf(" %[^\n]", s1);
			scanf("%[^\t] %[^\t]", s1, s2);
				Hello World
				field1 field2

scanf Examples

- to allow spaces to be input with the string

```
scanf (" %[^\n]s",a);
```

Increment Operators

- **prefix/postfix**

Example 1	Example 2

`x = 3;`
`y = ++x;`
`// x contains 4, y contains 4`
	`x = 3;`
	`y = x++;`
	`// x contains 4, y contains 3`

Ternary Operators

- **condition ? result1 : result2**

```
1 7==5 ? 4 : 3      // evaluates to 3, since 7 is not equal to 5.
2 7==5+2 ? 4 : 3    // evaluates to 4, since 7 is equal to 5+2.
3 5>3 ? a : b       // evaluates to the value of a, since 5 is greater than 3.
4 a>b ? a : b       // evaluates to whichever is greater, a or b.
```

Conditional Statement

- compound if

```
#include <stdio.h>

int main()
{
    int x;

    scanf ("%d\n", &x);

    if (x > 0)
      printf ("%d is positive\n", x);
    else if (x  < 0)
      printf ("%d is negative\n", x);
    else
      printf ("%d is zero\n", x);
}
```

Iteration

- while loop

```
#include <stdio.h>

int main ()
{
    int n = 10;

    while (n > 0) {
      printf ("%n, ", n);
      --n;
    }

    printf ("liftoff!\n");
}
```

```
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, liftoff!
```

Iteration

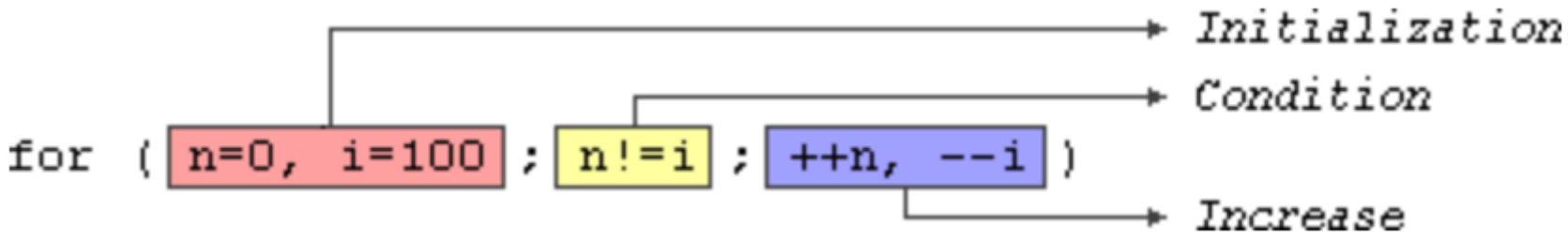
- for loop

```
#include <stdio.h>

int main ()
{
    int n;

    for (n = 10; n > 0; n--) {
      printf (“%n, ", n);
    }

    printf ("liftoff!\n");
}
```

```
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, liftoff!
```

Iteration

- for loop



break Statement

- break

```
#include <stdio.h>

int main ()
{
    int n;

    for (n = 10; n > 0; n--) {
      printf ("%n, ", n);

      if (n == 3) {
        printf ("Countdown aborted!\n");
        break;
      }
    }
}
```

```
10, 9, 8, 7, 6, 5, 4, 3, countdown aborted!
```

continue Statement

- continue

```
#include <stdio.h>

int main ()
{
    int n;

    for (n = 10; n > 0; n--) {
      if (n == 5) continue;

      printf ("%n, ", n);
    }

    printf (“liftoff!\n");
}
```

```
10, 9, 8, 7, 6, 4, 3, 2, 1, liftoff!
```

switch Statement

```
if (x == 1 || x == 2 || x == 3) {
   ("x is 1, 2, or 3\n");
}
else if (x == 4) {
   printf ("x is 4\n");
}
else {
   printf ("x is not 1, 2, 3, or 4\n");
}
```

```
switch (x) {
   case 1:
   case 2:
   case 3:  printf ("x is 1, 2, or 3\n");
            break;
   case 4: printf ("x is 4\n");
            break;
   default: printf ("x is not 1, 2, 3, or 4\n");
}
```