Jiangxi University of Science and Technology

Chapter02 Getting Started in C Programming

Variables and Definitions

Fundamental Types_Integers:

THE C PROGRAMMING LANGUAGE

- C supports these integer types: char, short, int, long, long long (in C11) in a non-decreasing order of size.
- The actual size depends on the implementation.
- The integers (except char) are signed number (which can hold zero, positive and negative numbers).
- You could use the keyword unsigned [char|short|int|long|long long] to declare an unsigned integers (which can hold zero and positive numbers).
- There are a total 10 types of integers - signed|unsigned combined with char|short|int|long|long long.

Fundamental Types_ Characters:

THE C PROGRAMMING LANGUAGE

- Characters (e.g., 'a', 'Z', '0', '9') are encoded in ASCII into integers, and kept in type char.

For example,

- character '0' is 48 (decimal) or 30H (hexadecimal);
  character 'A' is 65 (decimal) or 41H (hexadecimal);
  character 'a' is 97 (decimal) or 61H (hexadecimal).
- Take note that the type char can be interpreted as character in ASCII code, or an 8-bit integer. Unlike int or long, which is signed, char could be signed or unsigned, depending on the implementation. You can use signed char or unsigned char to explicitly declare signed or unsigned char.

Fundamental Types_ Floating-point Numbers

THE C PROGRAMMING LANGUAGE

- There are 3 floating point types: float, double and long double, for single, double and long double precision floating point numbers.
- float and double are represented as specified by IEEE 754 standard.
- A float can represent a number between ±1.40239846×10^-45 and ±3.40282347×10^38, approximated.
- A double can represented a number between ±4.94065645841246544×10^-324 and ±1.79769313486231570×10^308, approximated.
- Take note that not all real numbers can be represented by float and double, because there are infinite real numbers. Most of the values are approximated.

The table below shows the *typical* size, minimum, maximum for the primitive types. Again, take note that the sizes are implementation dependent

THE C PROGRAMMING LANGUAGE

Category	Type	Description	Bytes (Typical)	Minimum (Typical)	Maximum (Typical)
Integers	int (or signed int)	Signed integer (of at least 16 bits)	4 (2)	-2147483648	2147483647
	unsigned int	Unsigned integer (of at least 16 bits)	4 (2)	0	4294967295
	char	Character (can be either signed or unsigned depends on implementation)	1		
	signed char	Character or signed tiny integer (guarantee to be signed)	1	-128	127
	unsigned char	Character or unsigned tiny integer (guarantee to be unsigned)	1	0	255
	short (or short int) (or signed short) (or signed short int)	Short signed integer (of at least 16 bits)	2	-32768	32767
	unsigned short (or unsigned shot int)	Unsigned short integer (of at least 16 bits)	2	0	65535
	long (or long int) (or signed long) (or signed long int)	Long signed integer (of at least 32 bits)	4 (8)	-2147483648	2147483647
	unsigned long (or unsigned long int)	Unsigned long integer (of at least 32 bits)	4 (8)	0	same as above
	long long (or long long int) (or signed long long) (or signed long long int)	Very long signed integer (of at least 64 bits)	8	-2^{63}	$2^{63}-1$
	unsigned long long (or unsigned long long int)	Unsigned very long integer (of at least 64 bits)	8	0	$2^{64}-1$

The table below shows the *typical* size, minimum, maximum for the primitive types. Again, take note that the sizes are implementation dependent

THE C PROGRAMMING LANGUAGE

Real Numbers	float	Floating-point number, ≈7 digits (IEEE 754 single-precision floating point format)	4	3.4e38	3.4e-38
	double	Double precision floating-point number, ≈15 digits (IEEE 754 double-precision floating point format)	8	1.7e308	1.7e-308
	long double	Long double precision floating-point number, ≈19 digits (IEEE 754 quadruple-precision floating point format)	12 (8)		
Wide Characters	wchar_t	Wide (double-byte) character	2 (4)		

2.4 Arithmetic Operations

THE C PROGRAMMING LANGUAGE

- Displaying Numerical Values

Table 2.8 Conversion Control Sequences

Sequence	Meaning
%d	Display an integer as a decimal (base 10) number
%c	Display a character
%f	Display the floating-point number as a decimal number with six digits after the decimal point (pad with zeros, if necessary)

Type Conversion Code

THE C PROGRAMMING LANGUAGE

Type	Type Conversion Code	Type & Format
Integers	%d (or %i)	(signed) int
	%u	unsigned int
	%o	int in octal
	%x, %X	int in hexadecimal (%X uses uppercase A-F)
	%hd, %hu	short, unsigned short
	%ld, %lu	long, unsigned long
	%lld, %llu	long long, unsigned long long
Floating-point	%f	float in fixed notation
	%e, %E	float in scientific notation
	%g, %G	float in fixed/scientific notation depending on its value
	%f, %lf (printf), %lf (scanf)	double: Use %f or %lf in printf(), but %lf in scanf().
	%Lf, %Le, %LE, %Lg, %LG	long double
Character	%c	char
String	%s	string

printf

THE C PROGRAMMING LANGUAGE

- **printf**
  - **precise output formatting**
    - Conversion specifications: flags, field widths, precisions, etc.
  - Can perform rounding, aligning columns, right/left justification, inserting literal characters, exponential format, hexadecimal format, and fixed width and precision
- Format

  printf(format-control-string**,** other-arguments **);**
  - format control string: describes output format
  - other-arguments: correspond to each conversion specification in format-control-string
    - each specification begins with a percent sign, ends with conversion specifier

2.4 Arithmetic Operations

THE C PROGRAMMING LANGUAGE

- Program 2.5

```
1.  #include <stdio.h>
2.  int main(){
3.          printf("%d + %d = %d\n", 15, 2, 15+2);
4.          printf("%d - %d = %d\n", 15, 2, 15-2);
5.          printf("%d * %d = %d\n", 15, 2, 15*2);
6.          printf("%d / %d = %d\n", 15, 2, 15/2);
7.          return 0;
8.  }
```

TASK


```
/* * Print Size of Fundamental Types (SizeofTypes.cpp). */
 #include <stdio.h>
int main() {
printf("sizeof(char) is %d bytes.\n", sizeof(char));
printf("sizeof(short) is %d bytes.\n", sizeof(short));
printf("sizeof(int) is %d bytes.\n", sizeof(int));
printf("sizeof(long) is %d bytes.\n", sizeof(long));
printf("sizeof(long long) is %d bytes.\n", sizeof(long long));
printf("sizeof(float) is %d bytes.\n", sizeof(float));
printf("sizeof(double) is %d bytes.\n", sizeof(double));
printf("sizeof(long double) is %d bytes.\n", sizeof(long double));
 return 0;
}
```

TASK

THE C PROGRAMMING LANGUAGE

```
int anInt = 12345;
float aFloat = 55.6677;
double aDouble = 11.2233;
char aChar = 'a';
char aStr[] = "Hello";
printf("The int is %d.\n", anInt); //The int is 12345.
printf("The float is %f.\n", aFloat); //The float is 55.667702.
printf("The double is %lf.\n", aDouble); //The double is 11.223300.
printf("The char is %c.\n", aChar); //The char is a.
printf("The string is %s.\n", aStr); //The string is Hello.
printf("The int (in hex) is %x.\n", anInt); //The int (in hex) is 3039.
printf("The double (in scientific) is %le.\n", aDouble); //The double (in scientific) is 1.122330e+01.

printf("The float (in scientific) is %E.\n", aFloat); //The float (in scientific) is 5.566770E+01.
```

TASK

THE C PROGRAMMING LANGUAGE

```
double value = 123.14159265; printf("value=%lf;\n", value); //value=123.141593;
printf("value=%6.2lf;\n", value); //value=123.14;
printf("value=%9.4lf;\n", value); //value= 123.1416;

printf("value=%3.2lf;\n", value); // Field-width too short. Ignored. //value=123.14;
```

Program to test


```
1  #include <stdio.h>
2  int main()
3   {
4  printf( "%d\n", 455 );
5  printf( "%i\n", 455 );  /* i same as d in printf */
6  printf( "%d\n", +455 );
7  printf( "%d\n", -455 );
8  printf( "%hd\n", 32000 );
9  printf( "%ld\n", 2000000000 );
10 printf( "%o\n", 455 );
11 printf( "%u\n", 455 );
12 printf( "%u\n", -455 );
13 printf( "%x\n", 455 );
14 printf( "%X\n", 455 );
15 return 0;
16 }
```

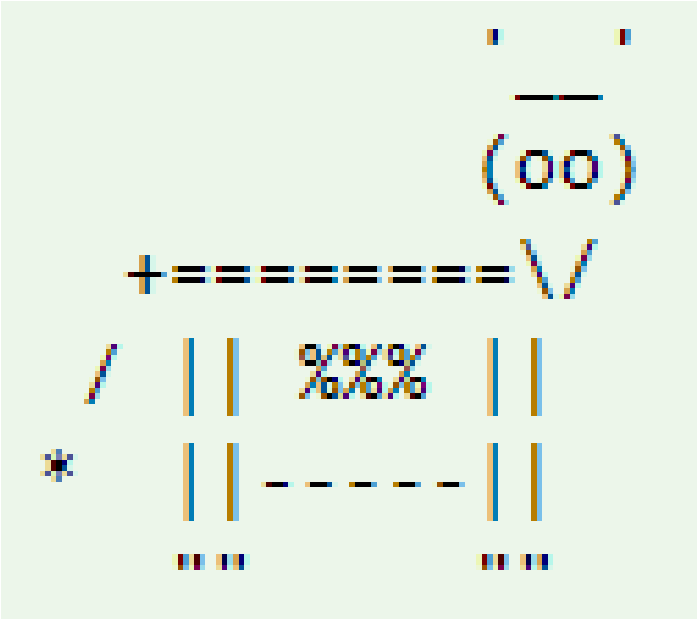
Decoration

THE C PROGRAMMING LANGUAGE

Escape Sequence	Description
\n	New-line (or Line-feed)
\r	Carriage-return
\t	Tab
\"	Double-quote (needed to include " in double-quoted string)
\'	Single-quote
\\	Back-slash (to resolve ambiguity)

➢ **TRY:** Write a program to print the following picture.

Take note that you need to use escape sequences to print special characters.



2.4 Arithmetic Operations

THE C PROGRAMMING LANGUAGE

➢ Expression Types 表达式类型

- **Expression:** any combination of operators and operands that can be evaluated to yield a value
- **Integer expression:** contains only integer operands; the result is an integer
- **Floating-point expression:** contains only floating-point operands; the result is a double-precision
- In a **mixed-mode expression** the data type of each operation is determined by the following rules:
  1. If both operands are integers, result is an integer
  2. If one operand is real, result is double-precision

2.4 Arithmetic Operations

THE C PROGRAMMING LANGUAGE

- Integer Division (整除)
- 15/2 = 7
  - Integers cannot contain a fractional part
  - Remainder (余数) is truncated
- % is the modulus or remainder operator (模运算符，取余运算符)
  - 9 % 4 is 1
  - 17 % 3 is 2
  - 14 % 2 is 0

2.4 Arithmetic Operations

THE C PROGRAMMING LANGUAGE

- Negation
  - A unary operator is one that operates on a single operand, e.g., negation (-)
  - The minus sign in front of a single numerical value negates (reverses the sign of) the number
  - **int a= -89;**

2.4 Arithmetic Operations

THE C PROGRAMMING LANGUAGE

➢ Summary of Arithmetic Operators

C supports the following arithmetic operators for numbers:

- short, int, long,
- long long, char (treated as 8-bit signed integer),
- unsigned short,
- unsigned int, unsigned long,
- unsigned long long, unsigned char,
- float, double and long double.

Table 2.9 Summary of Arithmetic Operators

Operation	Operator	Type	Operand	Result
Addition	+	Binary	Both are integers One operand is not an integer	Integer Double-precision
Subtraction	–	Binary	Both are integers One operand is not an integer	Integer Double-precision
Multiplication	*	Binary	Both are integers One operand is not an integer	Integer Double-precision
Division	/	Binary	Both are integers One operand is not an integer	Integer Double-precision
Modulus	%	Binary	Both are integers One operand is not an integer	Integer Double-precision
Negation	B	Unary	Integer or floating point	Same as operand

2.4 Arithmetic Operations

THE C PROGRAMMING LANGUAGE

- Operator Precedence and Associativity(运算符的优先级和结合性)

1. Expressions in **parentheses** are evaluated **first**, All **negations** are done **first**
2. **Multiplication**, **division**, and **modulus** operations are computed **next**; expressions containing more than one of these operators are evaluated **from left to right** as each operator is encountered
3. **Addition** and **subtraction** are computed **last**; expressions containing more than one addition or subtraction are evaluated **from left to right** as each operator is encountered

2.4 Arithmetic Operations

THE C PROGRAMMING LANGUAGE

- Operator Precedence and Associativity

- Example:
  - 8 + 5 * 7 % 2 * 4
  - = 8 + 35 % 2 * 4
  - = 8 + 1 * 4
  - = 8 + 4
  - = 12

Table 2.10 Operator Precedence and Associativity

Operator	Associativity
unary -	Right to left
* / %	Left to right
+ B	Left to right

2.5 Variables and Definitions

THE C PROGRAMMING LANGUAGE

- Variables (变量)
  - Variables are names given by programmers to computer storage
  - Variable name usually limited to 255 characters
  - Variable names are case sensitive
  - **int** a=45, b=12; **// Assignment statements**
  - int total=a+b;**// Assignment statements**

Watch 1

Name	Value	Type
a	45	int
b	12	int
&a	0x0022f984	int *
&b	0x0022f978	int *

Memory 1

Address: 0x0022F984

0x0022F978	+12 ....
0x0022F97C	-858993460
0x0022F980	-858993460
0x0022F984	+45 -...

2.5 Variables and Definitions

THE C PROGRAMMING LANGUAGE

- Definitions Statements (定义语句)
  - Definition statements define or tell the compiler how **much memory** is needed for data storage.
  - int total=100;
  - float firstnum=98;
  - double secnum=80.5;
  - char ch=65;

Name	Value	Type
total	100	int
firstnum	98.00	float
secnum	80.50	double
ch	65 'A'	char
sizeof(total)	4	unsigned int
sizeof(firstnum)	4	unsigned int
sizeof(secnum)	8	unsigned int
sizeof(ch)	1	unsigned int

2.5 Variables and Definitions

THE C PROGRAMMING LANGUAGE

- Initialization (初始化)
  - Declaration statements (声明语句) can be used to **store an initial value into declared variables**
  - **Literals** and **expressions** can all be used as initializers within a declaration statement.
  - float grade1=86.0;
  - float average=tatal/2.0;

2.5 Variables and Definitions

THE C PROGRAMMING LANGUAGE

- Program 2.7

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.          float grade1=86.0;
5.          float grade2=97.0
6.          float total=grade1+grade2;
7.          float average=tatal/2.0;
8.          printf("the average grade is %f\n",average);
9.          return 0;
10. }
```

2.6 Case Study

THE C PROGRAMMING LANGUAGE

- *Temperature Conversion*

  - **A friend of yours is going to Spain, where temperatures are reported using the Celsius temperature scale.**
  - **She has asked you to provide her with a list of temperatures in degrees Fahrenheit, and the equivalent temperature in degrees Celsius.**
  - **The formula relating the two temperatures is Celsius = 5/9(Fahrenheit − 32).**
  - **Initially, you are to write and test a program that correctly converts the Fahrenheit temperature of 75 degrees into its Celsius equivalent.**

2.6 Case Study

➢ Program 2.9 *Temperature Conversion*


```
1.  #include <stdio.h>
2.  int main(){
3.          float Celsius;
4.          float Fahrenheit=75.0;
5.          Celsius = 5.0/9.0*(Fahrenheit-32.0);
6.          printf("the Celsius equivalent of %5.2f degree Fahrenheit\n", Fahrenheit);
7.          printf("is %5.2f degree\n",Celsius);
8.          return 0;
9.  }
```

2.6 Case Study

THE C PROGRAMMING LANGUAGE

- Program 2.9 Temperature Conversion

aa.cpp

(Global Scope) main()

```
1  #include <stdio.h>
2  int main(){
3      float Celsius;
4      float Fahrenheit=75.0;
5      Celsius = 5.0/9.0*(Fahrenheit-32.0);
6      printf("the Celsius equivalent of %5.2f
   degree Fahrenheit\n", Fahrenheit);
7      printf("is %5.2f degree\n",Celsius);
8      return 0;
9  }
```

236 %

2.7 Summary

THE C PROGRAMMING LANGUAGE

- A C program consists of one or more functions
- A function is a C language description of an algorithm
- Many functions are supplied in a standard library of functions provided with each C compiler
- Simple C programs consist of the single function named `main()`
- An executable statement causes some specific action to be performed when the program is executed

2.7 Summary

THE C PROGRAMMING LANGUAGE

- All executable C statements must be terminated by a semicolon
- The `printf()` function displays text or numerical results
- The two basic numerical data types used almost exclusively in current C programs are integers and double-precision numbers
- An expression is a sequence of one or more operands separated by operators

2.7 Summary

- Expressions are evaluated according to the precedence and associativity of the operators used
- **printf()** can display all of C's data types
- Every variable in a C program must be
  - Declared with a data type
  - Used after it is declared
- Declaration statements inform the compiler of a function's valid variable names

Reference

- BOOK
- Some part of this PPT given by Prof 区 **Chengtian Ouyang)**
- with special thank
- https://www.codingunit.com/c-tutorial hello-world