Jiangxi University of Science and Technology

Ch06 Modularity Using Functions: Part I

Lecture0604 Standard Library Functions

6.4 Standard Library Functions

➢Standard Library Functions

- —The **standard library** consists of 15 header files
- —Before using these functions, you must know
  - The **name** of each available function
  - The **arguments** required by each function
  - The **data type** of the result (if any) returned by each function
  - A **description** of what each function does
- —How to include the library containing the desired function
  - #include <header-file-name>

C Standard Library

- The **C standard library** (aka **libc**) is a standardized collection of **header** files and **library** routines, which are used to implement common operations, such as input/output and string handling etc.
- C does not have built in keywords for these tasks, so nearly all C programs rely on the standard library.

Header files in C language

No.	Header Name	Job	Example
1	stdio.h	Input/ Output function	**scanf, printf**
2	conio.h	Console input/ output	**Scanf, printf, getch**
3	assert.h	Diagnostics function	**assert**
4	ctype.h	Character handling function	**isdigit(), islower(), toupper()**
5	float.h	Making more portable program	**FLT_MIN_EXP, FLT_MAX_EXP**
6	math.h	Mathematical function	**pow(), sqrt()**
7	setjmp.h	Nonlocal jump function	**setjmp(), longjmp()**
8	signal.h	Signal handling function	**sig_atomic_t , SIG_DFL**
9	stdarg.h	Variable argument list function	**va_list , va_start()**

Header files in C language

No	Header Name	Job	Example
10	stdlib.h	General utility function	**int abs() ,** **double atof()**
11	string.h	String function	**int memcmp() ,** **char *strcpy()**
12	time.h	Date and time function	**time_t mktime() ,** **time_t time()**
13	complex.h	A set of function for manipulating complex numbers	**cabs(), cpow()**
14	stdalign.h	For querying and specifying the alignment of objects	**_Alignas;** **_Alignof;**
15	errno.h	For tasting error code	**EDOM Domain Error()**
16	stdatomic.h	For atomic operation on data shared between threads	**atomic_flag() ,** **atomic_init()**
17	stdnoreturn.h	For specifying non- returning function	**noreturn void fatal()**

Header files in C language

THE C PROGRAMMING LANGUAGE

No.	Header Name	Job	Example
18	uchar.h	Types and functions for manipulating Unicode character	**NULL . char16_t**
19	fenv.h	A set of functions for controlling floating-point environment	**int feclearexcept(), int fetestexcept()**
20	wchar.h	Defines wide string handling functions	**int fwprintf() , int fwscanf()**
21	tgmath.h	Type- generic mathematical function	**copysign() , exp2()**
22	stdarg.h	Accessing a varying number of arguments passed to function	**va_start() , va_arg()**
23	stdbool.h	Define a boolean data type	**_Bool , true , false**
24	locale.h	Localization function	**LC_ALL, LC_TIME**

<stdio.h>

THE C PROGRAMMING LANGUAGE

input/output functionaliteis

Header Name	Job	Example

#include<stdio.h>
(Standard input-output header)
	Used to perform input and output operations in C like scanf() and printf().	#include <stdio.h>
		int main()
		{
		printf("C Programming");
		return 0;
		}

<string.h>

THE C PROGRAMMING LANGUAGE

string manipulation and memory handling

Header Name	Job	Example

#include<string.h>
(String header)
	Perform string manipulation operations like strlen and strcpy.	#include <stdio.h>
		#include <string.h>
		
		int main()
		{
		char str[]=“jxust";
		int length;
		
		//string length
		length=strlen(str);
		printf("String Length: %d\n",length);
		
		return 0;
		}

<conio.h>

THE C PROGRAMMING LANGUAGE

Header Name	Job	Example

#include<conio.h> (Console input-output header)	Perform console input and console output operations like clrscr() to clear the screen and getch() to get the character from the keyboard.	#include<conio.h>
		int main()
		{
		int iLoop;
		clrscr();
		for(iLoop=0; iLoop <= 15; iLoop++)
		{
		textcolor(iLoop);
		textbackground(15-iLoop);
		cprintf(“jxust");
		cprintf("\r\n");
		}
		getch();
		return 0;
		}

<stdlib.h>

THE C PROGRAMMING LANGUAGE

conversion, pseudo-random numbers, memory allocation, process control, environment, signalling, searching, and sorting.

Header Name	Job	Example

#include<stdlib.h> (Standard library header)	Perform standard utility functions like dynamic memory allocation, using functions such as malloc() and calloc().	#include <stdio.h>
		#include <stdlib.h>
		
		int main()
		{
		printf("Hello world!\n");
		return 0;
		}

<math.h>

common math functions

THE C PROGRAMMING LANGUAGE

Header Name	Job	Example

#include<math.h> (Math header)	Perform mathematical operations like sqrt() and pow(). To obtain the square root and the power of a number respectively.	#include <stdio.h>
		#include <math.h>
		
		int main()
		{
		int number, a;
		printf("Please enter a number from keyboard to calculate the absolute value\n");
		scanf("%d", &number);
		a = abs(number);
		printf("Calculated absolute value is : %d\n", a);
		
		return 0;
		}

<ctype.h>

to classify characters by their types or to convert between upper and lower case

THE C PROGRAMMING LANGUAGE

Header Name	Job	Example

#include<ctype.h> (Character type header)	Perform character type functions like isaplha() and isdigit(). To find whether the given character is an alphabet or a digit respectively.	#include<stdio.h>
		#include<ctype.h>
		int main()
		{
		char ch;
		printf("Enter a character: ");
		scanf("%c",&ch);
		if(isalnum(ch))
		printf("%c is an alphanumeric character.\n",ch);
		else
		printf("%c is not an alphanumeric character.\n",ch);
		
		return 0;
		}

<time.h>

THE C PROGRAMMING LANGUAGE

time/date formats and manipulation

Header Name	Job	Example

#include<time.h> (Time header)	Perform functions related to date and time like setdate() and getdate(). To modify the system date and get the CPU time respectively.	#include <stdio.h>
		#include <time.h>
		int main()
		{
		struct tm* ptr;
		time_t lt;
		lt = time(NULL);
		ptr = localtime(<);
		printf("%s", asctime(ptr));
		return 0;
		}

<assert.h>

THE C PROGRAMMING LANGUAGE

Contains the assert macro, helpful in detecting logical errors and other types of bug in debugging versions

Header Name	Job	Example

#include<assert.h>
(Assertion header)
	It is used in program assertion functions like assert(). To get an integer data type in C as a parameter which prints stderr only if the parameter passed is 0.	#include <stdio.h>
		#include <assert.h>
		int main()
		{
		int x = 7;
		x = 9;
		assert(x==7);
		return 0;
		}

<locale.h>

to set and select locale

THE C PROGRAMMING LANGUAGE

Header Name	Job	Example

#include<locale.h>
(Localization header)
	Perform localization functions like setlocale() and localeconv().
	To set locale and get locale conventions respectively.
		#include <stdio.h>
		#include <locale.h>
		int main()
		{
		struct lconv *a;
		setlocale (LC_ALL, "");
		a = localeconv();
		printf("Symbol to separate the values in thousand: %s\n\n", a->thousands_sep);
		printf("Symbol to define the values in currency: %s\n\n", a->currency_symbol);
		return 0;
		}

<signal.h>

various exceptional conditions

Header Name	Job	Example

#include<signal.h>
(Signal header)
	Perform signal handling functions like signal() and raise().
	To install signal handler and to raise the signal in the program respectively
		#include <signal.h>
		#include <stdio.h>
		#include <stdlib.h>
		static void catch_function(int signal) {
		puts("Interactive attention signal caught.");
		}
		int main(void) {
		if (signal(SIGINT, catch_function) == SIG_ERR) {
		fputs("An error occurred while setting a signal handler.\n", stderr);
		return EXIT_FAILURE;
		}
		puts("Raising the interactive attention signal.");
		if (raise(SIGINT) != 0) {
		fputs("Error raising the signal.\n", stderr);
		return EXIT_FAILURE;
		}
		puts("Exiting.");
		return 0;
		}

<stdarg.h>

THE C PROGRAMMING LANGUAGE

to allows functions to accept an variable number of arguments

Header Name	Job	Example

#include<stdarg.h>
(Standard argument header)
	Perform standard argument functions like va_start and va_arg().
	To indicate start of the variable-length argument list and to fetch the arguments from the variable-length argument list in the program respectively.
		#include <stdio.h>
		#include <stdarg.h>
		void printargs(int arg1, ...)
		{
		va_list ap;
		int i;
		va_start(ap, arg1);
		for (i = arg1; i >= 0; i = va_arg(ap, int))
		printf("%d ", i);
		va_end(ap);
		putchar('\n');
		}
		int main(void)
		{
		printargs(5, 2, 14, 84, 97, 15, 24, 48, -1);
		printargs(84, 51, -1);
		printargs(-1);
		printargs(1, -1);
		return 0;
		}

6.4 Standard Library Functions

➢*Mathematical Library Function (math.h)*

Table 6.1 Mathematical Library Functions (require the `math.h` header file)

Prototype	Description
`double fabs(double)`	Returns the absolute value of its double-precision argument. (Note, the function `int abs(int)` is defined in the `stdlib.h` header file.)
`double ceil(double)`	Returns a floating-point value that is the smallest integer that is greater than or equal to its argument value.
`double floor(double)`	Returns a floating-point value that is the largest integer that is less than or equal to its argument value.
`double fmod(double, double)`	Returns the remainder of its first argument divided by its second argument.
`double exp(double)`	Returns e raised to its double-precision argument.
`double log(double)`	Returns the natural logarithm (base e) of its argument.
`double log10(double)`	Returns the common logarithm (base 10) of its argument.

6.4 Standard Library Functions

➢*Mathematical Library Function (math.h)*

Table 6.1 Mathematical Library Functions (require the `math.h` header file)

Prototype	Description
`double fabs(double)`	Returns the absolute value of its double-precision argument. (Note, the function `int abs(int)` is defined in the `stdlib.h` header file.)
`double ceil(double)`	Returns a floating-point value that is the smallest integer that is greater than or equal to its argument value.
`double floor(double)`	Returns a floating-point value that is the largest integer that is less than or equal to its argument value.
`double fmod(double, double)`	Returns the remainder of its first argument divided by its second argument.
`double exp(double)`	Returns e raised to its double-precision argument.
`double log(double)`	Returns the natural logarithm (base e) of its argument.
`double log10(double)`	Returns the common logarithm (base 10) of its argument.

6.4 Standard Library Functions

➢*Mathematical Library Function*

Table 6.1 Mathematical Library Functions (require the `math.h` header file) (continued)

Prototype	Description
`double sqrt(double)`	Returns the square root of its argument.
`double pow(double, double)`	Returns its first argument raised to the power of its second argument.
`double sin(double)`	Returns the sine of its argument. The argument must be in radians.
`double cos(double)`	Returns the cosine of its argument. The argument must be in radians.
`double tan(double)`	Returns the tangent of its argument. The argument must be in radians.
`double asin(double)`	Returns the angle (in radians) whose sine is the argument.
`double acos(double)`	Returns the angle (in radians) whose cosine is the argument.
`double atan(double)`	Returns the angle (in radians) whose tangent is the argument.
`double atan2(double, double)`	Returns the angle (in radians) whose tangent is the first argument divided by the second argument.
`double sinh(double)`	Returns the hyperbolic sine of its argument.
`double cosh(double)`	Returns the hyperbolic cosine of its argument.
`double tanh(double)`	Returns the hyperbolic tangent of its argument.

6.4 Standard Library Functions

- Random Number Generator Functions
  - Random numbers are a series of numbers whose order cannot be predicted
  - ***Pseudorandom*** numbers are not really random, but are sufficiently random for the task at hand
  - All C compilers provide two functions for creating random numbers: ***rand()*** and ***srand()***, defined in the ***stdlib.h*** header file
    - rand() produces random numbers in the range:
    - 0 < rand() < RAND_MAX
    - srand() provides a starting "seed" value for rand()

6.4 Standard Library Functions

➢ Program 6.8 Random Number Generator

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>
4.  #define TOTALNUMBERS 10
5.  int main(){
6.          float randValue;
7.          srand(time(NULL));
8.          for (int i=1; i<= TOTALNUMBERS; i++){
9.                  randValue = rand();
10.                 printf("%6.0f\n", randValue);
11.         }
12.         return 0;
13. }
```

/*this generates the first "seed" value */

The “seed” value will be used by rand( ) as a ramdom starting point so that the results from rand( ) each time will be different!

Jiangxi University of Science and Technology

Ch06 Modularity Using Functions: Part I

Lecture0604 Standard Library Functions_B

6.4 Standard Library Functions

➢Scaling

—The method for adjusting the random numbers produced by a random-number generator to reside within a specified range is called ***scaling***

- To scale a random number as an integer value ***between 1 and N***:
  - **1 + (int)rand() % N**

- To produce a random integer ***between the numbers a and b***:
  - **a + (int)(rand() % (b - a + 1))**

6.4 Standard Library Functions

- Program 6.9 Coin Toss Simulation

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>
4.  int flip(int);   /*prototype for flip*/
5.  void percentages(int, int);//prototype for percentage
6.  int main(){
7.          int numTosses = 100000;
8.          int heads;
9.          heads = flip(numTosses);
10.         percentages(numTosses, heads);
11.         return 0;
12. }
```

6.4 Standard Library Functions

- Program 6.9 Coin Toss Simulation

```
13. int flip(int numTimes){
14.         int randValue;
15.         int heads = 0;
16.         srand(time(NULL));
17.         for (int i = 1; i <= numTimes; i++){
18.                 randValue =  (int)rand() % 2;
19.                 if (randValue >=1)
20.                         heads++;
21.         }
22.         return (heads);
23. }
```

this method tosses the coin numTimes and returns the number of heads

6.4 Standard Library Functions

Program 6.9 Coin Toss Simulation

```
24.  void percentages(int numTosses, int heads){
25.          int tails;
26.          float perheads, pertails;
27.          if (numTosses == 0)
28.                  printf("There were no tosses, so no percentages can be calculated.\n");
29.          else
30.          {
31.                   tails = numTosses - heads;
32.                  printf("Number of coin tosses: %d\n", numTosses);
33.                  printf("   Heads: %d   Tails: %d\n", heads, tails);
34.                 perheads=(float)heads/numTosses *100.0;
35.                 pertails=(float)(numTosses-heads) /numTosses * 100.0;
36.                  printf("Heads came up %6.2f percent of the time.\n", perheads);
37.                 printf("Tails came up %6.2f percent of the time.\n", pertails);
38.          }
39.  }
```

6.4 Standard Library Functions

➢Input/Output Library Functions

- **getchar()** can be used for single character input
- **int** getchar()
- The reason for returning characters in integer format is to allow the End-Of-File (EOF) sentinel to be returned

- **putchar()** expects a single character argument and displays the character passed to it on the terminal
- For example: putchar('a')

6.4 Standard Library Functions

➢ Character Processing Functions (ctype.h)

Table 6.2 Character Functions (require the header file `ctype.h`)

Prototype	Description	Example
`int isalnum(int)`	Returns a non-0 number if the argument is a letter or a digit; otherwise it returns a 0.	`isalnum('9');`
`int isalpha(int)`	Returns a non-0 number if the argument is a letter; otherwise, it returns 0.	`isalpha('a')`
`int iscntrl(int)`	Returns a non-0 number if the argument is a control argument; otherwise, it returns 0.	`iscntrl('a')`
`int isdigit(int)`	Returns a non-0 number if the argument is a digit (0–9); otherwise, it returns 0.	`isdigit('a')`
`int isgraph(int)`	Returns a non-0 value if the argument is a printable character other than a space; otherwise it returns a 0.	`isgraph('@')`
`int islower(int)`	Returns a non-0 number if the argument is lowercase; otherwise, it returns 0.	`islower('a')`

6.4 Standard Library Functions

➢ Character Processing Functions

Table 6.2 Character Functions (require the header file `ctype.h`) (continued)

Prototype	Description	Example
`int isprint(int)`	Returns a non-0 number if the argument is a printable argument; otherwise, it returns 0.	`isprint('a')`
`int ispunct(int)`	Returns a non-0 number if the argument is a punctuation argument; otherwise, it returns 0.	`ispunct('!')`
`int isspace(int)`	Returns a non-0 number if the argument is a space; otherwise, it returns 0.	`isspace(' ')`
`int isupper(int)`	Returns a non-0 number if the argument is uppercase; otherwise, it returns 0.	`isupper('a')`
`int isxdigit(int)`	Returns a non-0 value if the argument is a hexadecimal digit (A–F, a–f, or 0–9).	`isxdigit('b')`
`int tolower(int)`	Returns the lowercase equivalent if the argument is uppercase; otherwise, it returns the argument unchanged.	`tolower('A')`
`int toupper(int)`	Returns the uppercase equivalent if the argument is lowercase; otherwise, it returns the argument unchanged.	`toupper('a')`

6.4 Standard Library Functions

➢Program 6.10 The character is a letter or digit

```
1.  #include <stdio.h>
2.  #include <ctype.h>
3.  int main()
4.  {
5.      char inChar;
6.      do
7.      {
8.          printf("\nPush any key (type an x to stop) ");
9.          inChar = getchar();
10.         inChar = tolower(inChar);
11.         getchar();
12.     if ( isalpha(inChar))
13.         printf("\nThe character entered is a letter.\n");
14.     else if ( isdigit(inChar) )
15.         printf("\nThe character entered is a digit.\n");
16.     }while (inChar != 'x');
17. }
```

/*get and ignore the ENTER key */

/*You can use **EOF** replace **'x'**/

6.4 Standard Library Functions

- Conversion Functions

Table 6.3 String Conversion Functions (require the header file `stdlib.h`)

Prototype[8]	Description	Example
`int atoi(string)`	Converts an ASCII string to an integer. Conversion stops at the first noninteger character.	`atoi("1234")`
`double atof(string)`	Converts an ASCII string to a double-precision number. Conversion stops at the first character that cannot be interpreted as a double.	`atof("12.34")`
`string itoa(int)`	Converts an integer to an ASCII string. The space allocated for the returned string must be large enough for the converted value.	`itoa(1234)`

6.4 Standard Library Functions

➢ Program 6.11 Conversion Functions

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  int main(){
4.  int num;  double dnum;
5.  num = atoi("1234");
6.  printf("The string \"1234\" as an integer number is: %d \n", num);
7.  printf("This number divided by 3 is: %d \n\n", num/3);
8.  dnum = atof("1234.96");
9.  printf("The string \"1234.96\" as a double is: %f \n", dnum);
10. printf("This number divided by 3 is: %f \n", dnum/3);
11. return 0;
12. }
```

6.5 Summary

- A function is called by giving its name and passing any data to it in the parentheses following the name
- The first line of the function is called the function header
- A function's return type is the data type of the value returned by the function
- Functions can directly return at most a single value to their calling functions
- Functions can be declared to all calling functions with a function prototype
- Arguments passed to a function provide a means of evaluating any valid C expression
- A set of preprogrammed functions for
  - mathematical calculations
  - character input and output
  - character processing
  - numerical conversions

Reference

- BOOK
- Some part of this PPT given by Prof 欧阳城添

(Prof: Chengtian Ouyang)

- with special thank
- https://www.codingunit.com/c-tutorial-first-c-program-hello-world