Part 5 Programming Languages

Text 1 C

(C Programming Language, 2006)

C was developed in the early 1970s, and it was developed by Dennis Ritchie as a systems programming language for UNIX. C has grown into a very popular language now. C might best be described as a "medium-level language". Like a true high-level language, there is a one-to-many relationship between a C statement and the machine language instructions it is compiled into. However, unlike most high-level languages, C let you easily do chores (such as bit and pointer manipulation) additionally performed by assembly languages. Therefore, C is an especially good tool to use for developing operating systems (such as the UNIX operating system), or other system software.

1. What is C?

The C programming language is a popular and widely used programming language for creating computer programs. It is one of thousands of programming languages currently in use. C has been around for several decades and has won widespread acceptance because it gives programmers maximum control and efficiency. If you are programmer, or if you are interested in becoming a programmer, there are a couple of benefits you gain from learning C:

(1) You will be able to read and write code for a large number of platforms—everything from microcontrollers to the most advanced scientific systems can be written in C, and many modern operating systems are written in C.

(2) The jump to the object oriented C++ language becomes much easier. C++ is an extension of C, and it is nearly impossible to learn C++ without learning C first.

Lexically, C is more cryptic than other languages. In fact, C is an easy language to learn. It is a bit more cryptic in its style than some other languages, but you get beyond that fairly quickly. For example, brackets are often used to obviate the need for keywords. However, underlined characters are allowed in identifiers, which can make them more understandable.

There are a number of monadic and addict (called binary) operators. Some have unexpected precedence. Brackets may be ignored by the compiler, with occasionally surprising results. There are shift operations. Overflows on integer arithmetic may be ignored. There are some composite symbols with special meanings: for example "&&" means "and then" and "||" means "or else".

There are several integer types of different sizes and there are floating point numbers, * pointers (C talks of indirection), arrays and structures. C is not strongly typed: for example,

some compilers do not insert run-time checks on array subscripts, etc. Type conversion is permissive. Address arithmetic can be performed on pointers; Null is demoted by a zero value.

C has procedures and functions. There are few features (apart from procedures and functions) to support modularization, however; separate (strictly independent) compilation is allowed.

C is what is called a compiled language. This means that once you write your C program, you must run it through a C compiler to turn your program into an executable that the computer can run (execute). The C program is the human-readable form, while the executable that comes out of the compiler is the machine-readable and executable form. What this means is that to write and run a C program, you must have access to a C compiler. If you are using a UNIX machine (for example, if you are writing CGI scripts in C on your host's UNIX computer, or if you are a student working on a lab's UNIX machine), the C compiler is available for free. It is called either "cc" or "gcc" and is available on the command line. If you are a student, then the school will likely provide you with a compiler—find out what the school is using and learn about it. If you are working at home on a Windows machine, you are going to need to download a free C compiler or purchase a commercial compiler. A widely used commercial compiler is Microsoft's Visual C++ environment (it compiles both C and C++ programs). Unfortunately, this program costs several hundred dollars. If you do not have hundreds of dollars to spend on a commercial compiler, then you can use one of the free compilers available on the Web.

We will start at the beginning with an extremely simple C program and build up from there. I will assume that you are using the UNIX command line and gcc as your environment for these examples; if you are not, all of the code will still work fine—you will simply need to understand and use whatever compiler you have available.

2．The Simplest C Program

The best way to get started with C is to write, compile, and execute a simple program. Now, let's start with the simplest possible C program and use it both to understand the basics of C and the C compilation process. Type the following program into a standard text editor. Then save the program to a file named samp.c. If you leave off .c, you will probably get some sort of error when you compile it, so make sure you remember the .c. Also, make sure that your editor does not automatically append some extra characters (such as .txt) to the name of the file. Here's the first program:

```
#include<stdio.h>
int main()
{
  printf("This is output from my first program!\n");
  return 0;
}
```

When executed, this program instructs the computer to print out the line "This is output

from my first program!"—then the program quits. You can't get much simpler than that!

Position: When you enter this program, position #include so that the pound sign is in column 1 (the far left side). Otherwise, the spacing and indentation can be any way you like it. The spacing and indentation shown above is a good example to follow.

To compile this code, take the following steps:

(1) On a UNIX machine, type gcc samp.c -o samp (if gcc does not work, try cc). This line invokes the C compiler called gcc, asks it to compile samp.c and asks it to place the executable file it creates under the name samp. To run the program, type samp.

(2) On a DOS or Windows machine using DJGPP, at an MS-DOS prompt type gcc samp.c-o samp.exe. This line invokes the C compiler called gcc, asks it to compile samp.c and asks it to place the executable file it creates under the name samp.exe. To run the program, type samp.

(3) If you are working with some other compiler or development system, read and follow the directions for the compiler you are using to compile and execute the program.

You should see the output "This is output from my first program!" when you run the program. If you mistyped the program, neither will it compile, nor will it run. You should edit it again and see where you went wrong in your typing. Fix the error and try again.

Let's walk through this program and start to see what the different lines are doing.

(1) This C program starts with #include<stdio.h>. This line includes the "standard I/O library" into your program. The standard I/O library lets you read input from the keyboard(called "standard in"), write output to the screen (called "standard out"), process text files stored on the disk, and so on. It is an extremely useful library. C has a large number of standard libraries like stdio, including string, time and math libraries. A library is simply a package of code that someone else has written to make your life easier.

(2) The line int main() declares the main function. Every C program must have a function named main somewhere in the code. At run time, program execution starts at the first line of the main function.

(3) In C, the "{" and "}" symbols mark the beginning and end of a block of code. In this case, the block of code making up the main function contains two lines.

(4) The printf statement in C allows you to send output to standard out (for us, the screen). The portion in quotes is called the format string and describes how the data is to be formatted when printed. The format string can contain string literals, symbols for carriage returns (\n), and operators as placeholders for variables. If you are using UNIX, you can type man 3 printf to get complete documentation for the printf function. If not, see the documentation included with your compiler for details about the printf function.

(5) The return 0; line causes the function to return an error code of 0 (not error) to the shell that started execution.

3. Variables

As a programmer, you will frequently want your program to "remember" a value. For example, if your program requests a value from the user, or if it calculates a value, you will want

to remember it somewhere so you can use it later. The way your program remembers things is by using variables. For example:

```
int b;
```

This line says, "I want to create a space called b that is able to hold one integer value." A variable has a name (in this case, b) and a type (in this case, int, an integer). You can store a value in b by saying something like:

```
b=5;
```

You can use the value in b by saying something like:

```
printf("%d",b);
```

In C, there are several standard types for variables:

(1) int—integer (whole number) values.

(2) float—floating point values.

(3) char—single character values (such as "m" or "Z").

4．Printf

The printf statement allows you to send output to standard out. For us, standard out is generally the screen. Here is another program that will help you learn more about printf:

```
#include<stdio.h>
int main()
{
   int a,b,c;
   a=5;
   b=7;
   c=a+b;
   printf("%d+%d=%d\n",a,b,c);
   return 0;
}
```

Type this program into a file and save it as add.c. Compile it with the line gcc add.c –o add and then run it by typing add (or ./add). You will see the line "5+7=12" as output.

Here is an explanation of the different lines in this program:

(1) The line int a,b,c; declares three integer variables named a,b and c.

(2) The next line initializes the variable named a to the value 5.

(3) The next line sets b to 7.

(4) The next line adds a and b and "assigns" the result to c. The computer adds the value in a (5) to the value in b (7) to form the result 12, and then places that new value (12) into the variable c. The variable c is assigned the value 12. For this reason, the = in this line is called "the assignment operator."

(5) The printf statement then prints the line "5+7=12" The %d placeholders in the printf

statement act as placeholders for values. There are three %d placeholders, and at the end of the printf line there are the names for three variables: a,b and c. C matches up the first %d with a and substitutes 5 there. It matches the second %d with b and substitutes 7. It matches the third %d with c and substitutes 12.

Then it prints the completed line to the screen: 5+7=12. The +, the = and the spacing are a part of the format line and get embedded automatically between the %d operators as specified by the programmer.

New Words and Expressions

leverage *n.* 杠杆作用，影响力
binary *adj.* 双重的，双的；二进制的
subscript *adj.* 写在下面的 *n.* 脚注，下标，下角数码
modularization *n.* 模块化
executable *adj.* 可执行的，可实行的，可以做成的
integer *n.* 整数；完整的东西
automatically *adv.* 自动地；机械地

Exercises to the Text

1．Translate the following words and phrases into English.

（1）系统编程语言 （2）中级语言 （3）高级语言 （4）机器语言 （5）汇编语言 （6）系统软件 （7）免费编译器 （8）间距和缩进 （9）标准输入输出库 （10）标准类型

2．Translate the following paragraphs into Chinese.

(1) C has been around for several decades and has won widespread acceptance because it gives programmers maximum control and efficiency. If you are programmer, or if you are interested in becoming a programmer, there are a couple of benefits you gain from learning C.

(2) There are some composite symbols with special meanings: for example “&&” means “and then” and “||” means “or else”. “==” is used for equality to avoid confusion with “=” in assignments, and “!=” is used for inequality.

(3) C is what is called a compiled language. This means that once you write your C program, you must run it through a C compiler to turn your program into an executable that the computer can run (execute). The C program is the human-readable form, while the executable that comes out of the compiler is the machine-readable and executable form. What this means is that to write and run a C program, you must have access to a C compiler.

(4) The standard I/O library lets you read input from the keyboard(called “standard in”), write output to the screen (called “standard out”), process text files stored on the disk, and so on. It is an extremely useful library. C has a large number of standard libraries like stdio, including string, time and math libraries. A library is simply a package of code that someone else has written to make your life easier.

Text 2　C++

(C++ Primer, 2005)

C++ is a general-purpose programming language with high-level and low-level capabilities. It is a statically typed, free-form, multi-paradigm, usually compiled language supporting procedural programming, data abstraction, object-oriented programming, and generic programming.

1．Origins of the C++ Language

The C++ programming languages can be thought of as the C programming language with classes (and other modern features added). The C programming language was developed by Dennis Ritchie of AT&T Bell Laboratories in the 1970s. It was first used for writing and maintaining the UNIX operating system (up until that time, UNIX systems programs were written either in assembly language or in language called B, a language developed by Ken Thompson, the originator of UNIX). C is a general-purpose language that can be used for writing any sort of program, but its success and popularity are closely tied to the UNIX operating system. If you wanted to maintain your UNIX system, you need to use C. C and UNIX fit together so well that soon not just systems programs but almost all commercial programs that ran under UNIX were written in the C language. C became so popular that versions of the language were written for other popular operating systems; its use is thus not limited to computers that use UNIX. However, despite its popularity, C was not without its shortcomings.

The C language is peculiar because it is a high-level language with many of the features of a low-lever language. C is somewhere in between the two extremes of a very high-level language and a low-level language, and therein lies both its strengths and its weakness. Like (low-level) assembly language, C language programs can directly manipulate the computer's memory. On the other hand, C has the features of a high-level language, which makes it easier to read and write than assembly language. This makes C an excellent choice for writing systems programs, but for other programs (and in some sense even for systems programs) C is not as easy to understand as other languages; also, it does not have as many automatic checks as some other high-level languages.

To overcome these and other shortcomings of C, Bjarne Stroustrup of AT&T Bell Laboratories developed C++ in the early 1980s. Stroustrup designed C++ to be a better C. Most of C is a subset of C++, and so most C programs are also C++ programs (the reverse is not true; many C++ programs are definitely not C programs). The basic syntax and semantics of C and C++ are the same. If you are familiar with C, you can program in C++ immediately. C++ has the same types, operators, and other facilities defined in C that usually correspond directly to computer architecture. Unlike C, C++ has facilities for classes and so can be used for object-oriented programming (OOP).

2．C++ and Object-Oriented Programming

Object-oriented programming is a programming technique that allows you to view concepts as a variety of objects. By using objects, you can represent the tasks that are to be performed, their interaction, and any given conditions that must be observed. A data structure often forms the basis of an object; thus, in C or C++, the struct type can form an elementary object. Communicating with objects can be done through the use of messages. Using messages is similar to calling a function in a procedure-oriented program. When an object receives a message, methods contained within the object respond. Methods are similar to the functions of procedure-oriented programming. However, methods are part of an object.

The main characteristics of object-oriented programming are encapsulation, inheritance, and polymorphism. Encapsulation is a form of information hiding or abstraction. Inheritance has to do with writing reusable code. Polymorphism refers to a way that a single name can have multiple meanings in the context of inheritance. C++ accommodates OOP by providing classes, a kind of data type combining both data and algorithms. C++ is not what some authorities would call a "pure OOP language." C++ tempers its OOP features with concerns for efficiency and what some might call "practicality." This combination has mad C++ currently the most widely used OOP language, although not all of its usage strictly follows the OOP philosophy.

3．The Character of C++

C++ allows the programmer to create classes, which are somewhat similar to C structures. In C++, it can be assigned methods, functions associated to it, of various prototypes, which can access and operate within the class, somewhat like C functions often operate on a supplied handler pointer. The C++ class is an extension of the C language structure. Because the only difference between a structure and a class is that structure members have public access by default and a class member has private access by default, you can use the keywords class or struct to define equivalent classes.

The C++ class is an extension of the C and C++ struct type and forms the required abstract data type for object-oriented programming. The class can contain closely related items that share attributes. Stated more formally, an object is simply an instance of a class.

Ultimately, there should emerge class libraries containing many object types. You could use instances of those object types to piece together program code.

Typically, an object's description is part of a C++ class and includes a description of the object's internal structure, how the object relates with other objects, and some form of protection that isolates the functional details of the object from outside the class. The C++ class structure does all of this.

In a C++ class, you control functional details of the object by using private, public, or protected descriptors. In object-oriented programming, the public section is typically used for the interface information (methods) that makes the class reusable across applications. If data or methods are contained in the public section, they are available outside the class. The private section of a class limits the availability of data or methods to the class itself. A protected section

containing data or methods is limited to the class and any derived subclasses.

C++'s connection to the C language gives it a more traditional look than newer object-oriented languages, yet it has more powerful abstraction mechanisms than many other currently popular languages. C++ has a template facility that allows for full and direct implementation of algorithm abstraction. C++ templates allow you to code using parameters for types. The newest C++ standard, and most C++ compilers, allow multiple namespaces to accommodate more reuse of class and function names. The exception handling facilities in C++ are similar to what you would find in other programming languages. Memory management in C++ is similar to that in C. The programmer must allocate his or her own memory and handle his or her own garbage collection. Most compilers will allow you to do C-style memory management in C++, since C is essentially a subset of C++. However, C++ also has its own syntax for a C++ style of memory management, and you are advised to use the C++ style of memory management when coding in C++.

Inheritance in object-oriented programming allows a class to inherit properties from a class of objects. The parent class serves as a pattern for the derived class and can be altered in several ways. If an object inherits its attributes from multiple parents, it is called multiple inheritance. Inheritance is an important concept since it allows reuse of a class definition without requiring major code changes. Inheritance encourages the reuse of code since child classes are extensions of parent classes.

Another important object-oriented concept that relates to the class hierarchy is that common messages can be sent to the parent class objects and all derived subclass objects. In formal terms, this is called polymorphism.

Polymorphism allows each subclass object to respond to the message format in a manner appropriate to its definition. Imagine a class hierarchy for gathering data. The parent class might be responsible for gathering the name, social security number, occupation, and number of years of employment for an individual. You could then use child classes to decide what additional information would be added based on occupation. In one case a supervisory position might include yearly salary, while in another case a sales position might include an hourly rate and commission information. Thus, the parent class gathers general information common to all child classes while the child classes gather additional information relating to specific job descriptions. Polymorphism allows a common data-gathering message to be sent to each class. Both the parent and child classes respond in an appropriate manner to the message.

Polymorphism gives objects the ability to responds to messages from routines when the object's exact type is not known. In C++ this ability is a result of late binding. With late binding, the addresses are determined dynamically at run time, rather than statically at compile time, as in traditional compiled languages. This static method is often called early binding. Function names are replaced with memory addresses. You accomplish late binding by using virtual functions. Virtual functions are defined in the parent class when subsequent derived classes will overload the function by redefining the function's implementation.

Virtual functions utilize a table for address information. The table is initialized at run time by using a constructor. A constructor is invokes whenever an object of its class is created. The job of the constructor here is to link the virtual function with the table of address information. During the compile operation, the address of the virtual function is not known; rather, it is given the position in the table of addresses that will contain the address for the function.

4. The C++ Program

All procedure-like entities are called functions in C++. Things that are called procedures, methods, functions, or subprograms in other languages are all called functions in C++. A C++ program is basically just a function called main; when you run a program, the run-time system automatically invokes the function named main. Other C++ terminology is pretty much the same as most other programming languages. Here's the C++ program.

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5    int numberOfLanguage;
6    cout<<"Hello reader.\n"
7        <<"Welcome to C++.\n";
8    cout<<"How many programming languages have you used?";
9    cin>>numberOfLanguages;
10    if(numberOfLanguages<1)
11      cout<<"Read the preface. You may prefer\n"
12         <<"a more elementary book by the same author.\n";
13    else
14      cout<<"Enjoy the book.\n";
15  return 0;
16  }
```

A C++ program is really a function definition for a function named main. When the program is run, the function named main is invoked. The body of the function main is enclosed in braces, {}. When the program is run, the statements in the braces are executed. Here are two possible screen displays that might be generated when a user runs the program.

Dialogue 1

```
Hello reader.
Welcome to C++.
How many programming languages have you used? 0← User types in 0 on the keyboard.
Read the preface. You may prefer a more elementary book by the same author.
```

Dialogue 2

```
Hello reader.
Welcome to C++.
How many programming languages have you used? 1← User types in 1 on the keyboard.
Enjoy the book
```

Variable declarations in C++ are similar to what they are in other programming languages. The fifth line declares the variable numberOfLanguages. The type int is one of the C++ types for whole numbers (integers).

If you have not programmed in C++ before, then the use of cin and cout for console I/O is likely to be new to you. But the general idea can be observed in this sample program. For example, consider the eighth line and the ninth lines. The eighth line outputs the text within the quotation marks to the screen. The ninth line reads in a number that the user enters at the keyboard and sets the value of the variable numberOfLanguages to this number.

The eleventh line and the twelfth output two strings instead of just one string. The symbolism \n is the new line character, which instructs the computer to start a new line of output.

New Words and Expressions

architecture *n.* 建筑学
capability *n.* 能力
enhance *v.* 改善，提高，增进
semantics *n.* 语义学，语义论
concept *n.* 概念，观念，思想
interaction *n.* 交互作用，相互作用
extension *n.* 伸展，扩大，延长，延期，电话分机
equivalent *adj.* 等价的，相等的，同意义的 *n.* 等价物，相等物
description *n.* 描述，描写，说明书，类型
subclass *n.* 子类，亚类
hierarchy *n.* 层级，等级制度
inherit *vt.* 继承，遗传而得 *vi.* 成为继承人
inheritance *n.* 遗产，继承，遗传
hourly *adv.* 频繁地，随时，每小时地 *adj.* 每小时的，以钟点计算的，频繁的

Exercises to the Text

1．Translate the following words and phrases into English.

（1）编程语言 （2）面向对象程序设计 （3）抽象机制 （4）代码重用 （5）接口信息 （6）地址信息表 （7）虚函数 （8）附加信息 （9）函数定义 （10）变量声明

2．Translate the following paragraphs into Chinese.

(1) C++ is a general-purpose programming language with high-level and low-level capabilities. It is a statically typed, free-form, multi-paradigm, usually compiled language supporting procedural programming, data abstraction, object-oriented programming, and generic programming.

(2) C is a general-purpose language that can be used for writing any sort of program, but its success and popularity are closely tied to the UNIX operating system. If you wanted to maintain your UNIX system, you need to use C. C and UNIX fit together so well that soon not just