การเขียนโปรแกรมคอมพิวเตอร์

การโปรแกรมแบบโครงสร้าง และแนะนำภาษาซี

อาจารย์ โอภาส วงษ์ทวีทรัพย์ (อ.โอ๊ต)

ศูนย์ปฏิบัติการวิจัย และพัฒนาระบบสารสนเทศอันชาญฉลาด

ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

E-Mail :: oatcomster@gmail.com

ขอบเขตของเนื้อหา

- การจัดแบ่งประเภทของภาษาโปรแกรม ***(Programming Language)*** ที่ใช้ในการพัฒนาโปรแกรมในปัจจุบัน
- แนวทางของการเขียนโปรแกรมแบบโครงสร้าง ***(Structured Programming)***
- แนะนำภาษาซีเบื้องต้น ทั้งข้อเด่น - ข้อด้อยของภาษา
- การใช้กลุ่มคำสั่งรับค่าและแสดงผลลัพธ์

ภาษาโปรแกรมแบ่งตามระดับชั้น

- **แบ่งได้เป็น 4 ระดับ คือ**
- ยุคที่ 1 ภาษาเครื่อง ***(Machine Language)***
- ยุคที่ 2 ภาษาระดับต่ำหรือภาษาเชิงสัญลักษณ์ ***(Low Level Language or Symbolic Language)*** เช่น Assembly
- ยุคที่ 3 ภาษาระดับสูง ***(High Level Language)*** เช่น C,Pascal
- ยุคที่ 4 ภาษาธรรมชาติ ***(Natural Language)*** เช่น SQL

ภาษาเครื่อง

- ยุคที่ 1 ภาษาเครื่อง ***(Machine Language)*** เป็นภาษาในยุคแรกเริ่มต้นที่มีคอมพิวเตอร์เกิดขึ้นมา
- เป็นภาษาที่เขียนในรูปของคำสั่งที่คอมพิวเตอร์สามารถเข้าใจและตีความ เพื่อการทำงานได้ทันที
- ยากแก่การเข้าใจ เพราะว่าเขียนเป็นรหัสของตัวเลขทั้งหมด ทำให้การเขียนโปรแกรมและแก้ไขนั้นเป็นไปอย่างยากลำบาก ต้องอาศัยความชำนาญสูง

ภาษาระดับต่ำหรือภาษาเชิงสัญลักษณ์

- ยุคที่ 2 ภาษาระดับต่ำหรือภาษาเชิงสัญลักษณ์ ***(Low Level Language or Symbolic Language)*** เกิดขึ้นเพื่อแก้ไขปัญหาของภาษาเครื่องที่เป็นรหัสทำให้เข้าใจยาก
- ภาษาในระดับนี้ จะเป็นภาษาที่แทนรหัสตัวเลขของคำสั่ง ด้วยสัญลักษณ์ของข้อความที่กำหนดขึ้น
- ทำให้มนุษย์อ่านและเข้าใจง่ายมากยิ่งขึ้น ว่าโปรแกรมทำงานอะไร (แต่สุดท้ายก็ต้องแปลงไปเป็นภาษาเครื่องอยู่ดี)

ภาษาระดับสูง

- ยุคที่ 3 ภาษาระดับสูง ***(High Level Language)***
- เป็นกลุ่มภาษาที่มีความใกล้เคียงกับภาษามนุษย์มาก
- ทำให้เขียนและอ่านง่าย ว่าโปรแกรมนั้นทำงานอะไร
- แต่การทำงานของภาษาระดับนี้จะช้า เพราะเนื่องมาจากต้องถูกแปลงไปเป็นภาษาเครื่อง เพราะภาษาเครื่องคือภาษาเดียวเท่านั้นที่คอมพิวเตอร์เข้าใจ และสามารถทำงานได้

ภาษาธรรมชาติ

- ยุคที่ 4 ภาษาธรรมชาติ ***(Natural Language)*** เช่น SQL
- ภาษาในกลุ่มนี้จะแตกต่างกับภาษาในกลุ่มอื่นๆ กล่าวคือ จะเป็นภาษาที่ระบุถึงความต้องการ ว่าต้องการอะไร ซึ่งเป็นภาษาที่พบในการทำงานที่มีการจัดการฐานข้อมูล ***(Database)***

ภาษาโปรแกรมแบ่งตามแนวคิดของภาษา

- **จะสามารถแบ่งได้ออกเป็น 4 กลุ่ม คือ**
- ***Imperative Language*** เช่น Basic , C , Pascal เป็นต้น
- ***Object - Oriented Programming Language(OOP)*** เช่น C++ , Java , SmallTalk เป็นต้น
- ***Logic Programming Language*** เช่น Prolog เป็นต้น
- ***Functional Programming Language*** เช่น LISP เป็นต้น

วิวัฒนาการของแนวคิดการเขียนโปรแกรม

- *Procedural*
  - 1950s-1970s (procedural abstraction)
- *Data - Oriented*
  - early 1980s (data-oriented)
- *Object - Oriented Programming (OOP)*
  - late 1980s (Inheritance and dynamic binding)

การเขียนโปรแกรมแบบโครงสร้าง

- การเขียนโปรแกรมแบบโครงสร้าง หรือ ***Structured Programming*** นั้น เป็นแนวคิดและวิธีการที่ใช้ในการเขียนโปรแกรมคอมพิวเตอร์ ตั้งแต่แรกเริ่มที่มีคอมพิวเตอร์ขึ้นมา
- โดยการเขียนโปรแกรมในลักษณะนี้ จะมีการเขียนในลักษณะของโครงสร้างของคำสั่ง 3 รูปแบบ ด้วยกัน กล่าวคือ
- โครงสร้างแบบเรียงลำดับ ***(Sequence Statement)*** โครงสร้างแบบเลือก ***(Selection Statement)*** และโครงสร้างแบบทำซ้ำ ***(Iteration Statement)***

การเขียนโปรแกรมแบบโครงสร้าง (2)

- สามารถมีโครงสร้างต่างๆ ซ้อนกันได้ ***(Nested Structure)***
- การเขียนโปรแกรมนั้นจะต้องมีทางเข้าและทางออกของโปรแกรมเพียงทางเดียวเท่านั้น
- มีจุดเริ่มต้น และจุดสิ้นสุดของโปรแกรมอย่างชัดเจน
- ไม่มีการใช้คำสั่ง GOTO ถ้าไม่จำเป็น เพราะจะทำให้ขาดความเป็นโครงสร้าง ***(UnStructure)*** อีกทั้งอาจก็ให้เกิดปัญหา ***Spaghetti Code*** ขึ้นมาได้

Structure of C Language

- A C program contains
  - *Preprocessor Directive* *Type Definitions*
  - *Function Prototypes* *Variables*
  - *Functions*
- All program must contain single main() function.
- Return Default of Main Function is Integer.

Structure of C Language(2)

```
#include <stdio.h>      /* Preprocessor directive */
main()                  //Default return Integer
{                       //Begin of main program
    ...                 //Statement
    ...                 //Statement
}                       //End of main program
```


Structure of C Language(3)

```
#include <stdio.h>      //Preprocessor directive
void main()             //Function no return value
{                       //Begin of main program
    ...                 //Statement
    ...                 //Statement
}                       //End of main program
```


Structure of C Language(4)

```
#include <stdio.h>      //Preprocessor directive
int main()              //Function return value
{                       //Begin of main program
    ...                 //Statement
    return(0);          //Statement Return
}                       //End of main program
```


Structure of C Language(5)

```
#include <stdio.h>
float PI = 3.1415927;
int main() {
    float radias,area;
    printf("Input Radias : ");scanf("%d",&radias);
    area = PI * radias * radias;
    printf("Area of Circle : %d",area); return(0);
}
```


Knowledge

Value of PI???

1. PI = 3.1415927 2. PI = 22/7 = 3.1428571

3. PI = 22/7 = 3 (in C Language)

PI = 3.1415927 x 10000 x 10000 = 314159270

PI = 3.1428571 x 10000 x 10000 = 314285710

Error = 314285710 - 314159270 = 126440

Program Comment

- Program Description in each section of program.
- Program Useful for Evolution in next future.
- Compiler don't compile this section(no change to machine code) and it has 2 forms :

  /* comment part */

  // comment part used only borland C compiler

Program Comment (2)

- **For Examples :**

```
/* This is comment */
/*  For produce the good program, should
Write comment every 10-15 line of program */
//Or use line comment for Borland C
```


Preprocessor Directives

- **Preprocessing occurs before a program is compiled. Preprocessor lines are started with # and ANSI C has 12 preprocessor directives as following:**

#include	**#define**	**#if**	**#else**
#elif	**#endif**	**#error**	**#ifdef**
#ifndef	**#line**	**#progma**	**#undef**

Preprocessor Directives (2)

#include

- Treats text in the file specified by filename as if it appeared in the current file.

Syntax:

- #include "filename"
- #include <filename>

Preprocessor Directives (3)

#include "filename"

searches the source path first, then the include path.

#include <filename>

does not search the source directory but searches the include path.

Preprocessor Directives (4)

#define

- The #define directive defines a macro.
- Macros provide a mechanism for token replacement with or without a set of formal, function-line parameters.

Preprocessor Directives (5)

Syntax:

#define <id1>[(<id2>, ...)] <token string>

Example:

```
#define  MAXINT  32
#define  sum(a,b)  (a+b)
#define  sqr(x)   (x*x)
```


Preprocessor Directives (6)

Example:

```
sum ( a , b )      ==>  ( a + b )
    sum ( 1 , 2 )  ==>  ( 1 + 2 )          = 3
sqr ( x )          ==>  ( x * x )
    sqr ( 2 )      ==>  ( 2 * 2 )          = 4
    sqr ( 1 + 1 )  ==>  ( 1 + 1*1 + 1 )    = 3
```


Fundamental Components

- **Character.**
- **Name Identifier.**
- **Reserve Words.**
- **Standard Identifier.**
- **Data Types of C Language.**

Fundamental Components (2)

- **Variables Declaration.**
- **Precedence of Operators.**
- **Each Operators in C Program.**
- **Expression.**
- **Statement and Function.**

Characters

- Character have 3 elements as following :
  - Numerics :: Decimal Number '0' - '9'
  - Alphabetics :: English Character 'A'-'Z','a'-'z'
  - Special Symbols
    - Mathematic Symbols :: + , - , * , /
    - Other Symbols :: # , $, % , & , @ , (,) , { , }

Name Identifiers

- Name of Variables, Name of Function.
- Name length of variable is not limited but it has meaning about only 31 characters.
- First Identifier must be only Alphabet or Underline(_) and next character is Alphabet, Numeric or Underline(_).

Name Identifiers(2)

- None Special Char, None Space in identifier.
- Case - Sensitive Identifier.
- Easy to Understand, Meaning is Clear.
- Don't used Duplicate Identifier.
- None Keywords of C Language (Standard Identifiers or Reserve Words).

Name Identifiers(3)

Right Example

NUM1	Num_of_Student	Area	Radias
Char1	Title	score	grade

Wrong Example

A+B	Num 1	X#1
power	void	return

Name Identifiers(4)

int _ _ _;

Why Not???

Reserve Words/Std Identifiers

auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

Data Types of C Language

- Simple Data Types
  - Standard Data Types :: int , float , char etc.,
  - User-defined Data Types :: Enumeration
- Structured Data Types :: Structure , Union
- Pointer Data Types :: String , Dynamic Array

Simple Data Types

C Type	Size(byte)	Mean	Range
char	1	character	-
unsigned char	1	small numbers	0...255
short int	2	integers	-32768...32767
unsigned short int	2	positive integers	0...65535
(long) int	4	large integer	$-2^{31}...2^{31}-1$
float	4	real numbers	
double	8	large reals	
void	0	no return value	

Variables Declaration

- A variable has 6 characteristics as following: Name, Address, Type, Value, Scope and Lift Time.
- Name of variables are language constructs that create program to easy manipulate the address of variables(No recognized memory address).

Variables Declaration(2)

- Length of variable name is not limited but it has meaning about only 31 characters.
- Name of variables must have meaning for easy memorize from programmer.
- Easy to Understand and Meaning is Clear.

Variables Declaration(3)

Syntax:

Datatype Variables;

Example:

```
int a;          char c='c';          char st = c;
int A,B,C;      int sum=0,dif=0;
char * str="OaTCoM";
```


Variables Declaration(4)

Homeworks

```
int a = b = 0;          int a = 0 , b = 0;
int num = 'c';
```


Statements

- Statement has two forms as following:
  - *Single Statement*

    printf("Welcome to C");
  - *Compound Statement*

    { printf("Welcome to C");
      printf("Welcome to C"); }

Statements

- ***Assignment Statement*** : Mathematics Expression
- ***Logical Statement*** : Boolean Expression
- ***Input / Output Statement*** : printf, scanf
- ***Selection Statement*** : if, if-else, nested-if, switch-case
- ***Control Statement*** : break, continues, goto
- ***Iteration Statement*** : while, do-while, for

Expression

- Expression has two components ==> A + B , (A+4)/sqrt(8)
  - ***Operand*** may be Constant, Variable Name and Function Name(Return Value).
    - Example : A , B , 4 , 8 , sqrt()
  - ***Operator*** is Symbols that replaced process.
    - Example : + , - , * , / , ++ , -- etc.

Math & Boolean Expression

- *Mathematics Expression*
  - contain mathematics symbols : + , - , * , / , ++ , -- etc.,
  - result return is value of calculation.
- *Boolean Expression*
  - contain relational symbols : < , > , <= , >= , == , !=
  - contain logical symbols : ! , && , || , & , |
  - Boolean value : zero(False) , nonzero (True).

Precedence of Operators

- In the following table of operator precedence, the Turbo C++ operators are divided into 16 categories.
- The #1 category has the highest precedence; category #2 (Unary operators) takes second precedence, and so on to the Comma operator, which has lowest precedence.

Precedence of Operators(2)

The Unary (category #2), Conditional (category #14), and Assignment (category #15) operators associate right-to-left; all other operators associate left-to-right.

Unary Operators

!	Logical negation (NOT)	&	Address
~	Bitwise (1's) complement	*	Indirection
+	Unary plus	-	Unary minus
++	Preincrement or postincrement		
- -	Predecrement or postdecrement		
sizeof	(returns size of operand, in bytes)		

Unary Operators

Unary Operator is a statement contain one operand and one operand.

+	Unary plus	Ex z = +x;	
-	Unary minus	Ex z = -x;	
++	Increment	Ex ++x;	x++;
- -	Decrement	Ex --x;	x--;

Multiplication and Additive

Binary Operator is a statement contain one operator and two operands ex Z = X + Y;

*	Multiply	/	Divide
%	Remainder (modulus)		
+	Binary plus	-	Binary minus

Relational and Equality

<	Less than
< =	Less than or equal to
>	Greater than
> =	Greater than or equal to
= =	Equal to
!=	Not equal to

Shift and Bit-wise Operators

<<	Shift left	Ex	X = 10010110;
>>	Shift right		Y = 01001110;
&	Bitwise AND		Z = X << 1; Z = 00101100
|	Bitwise OR		Z = Y >> 2; Z = 00010011
^	Bitwise XOR		Z = X ^ Y; Z = 11011000

Assignment Operators

=	*=	/=
%=	+=	-=
&=	^=	|=
<<=	>>=	

Assignment Operators(2)

```
X = 5;      //Assigned X equal to 5
X *= 5;     //Multiple X by 5      = 25
X /= 5;     //Divide X by 5        = 5
X %= 5;     //Mod X by 5           = 0
X += 20;    //Add X by 20          = 20
X -= 8;     //Sub X by 8           = 12
```


Input/Output Statements

- Input Statements
  - scanf( ) ,getchar( ) ,getch( ) ,gets( )
- Output Statements
  - printf( ) ,putchar( ) ,puts( )

Examples of Input Statement

- scanf(Format Specifier,Variables);
- int getch(void variable);
- int getchar(void variable);
- int gets(char * variable);

Input Statement(2)

- variable = int getch(void);
- variable = int getchar(void);
- variable = int gets(char *);

Output Statement

- printf(Format Specifier,Variables);
- putch(char variable);
- puts(char * variable);

Format Specifiers

- Format Codes
  - %i or %d Signed Decimal Integer
  - %o or %u Unsigned Octal or Decimal Integer
  - %x or %X Unsigned Hexadecimal Integer
  - %h h->short , l->long (length modifiers)
  - %p Pointer Value of Variable

Format Specifiers(2)

- Format Codes
  - %n Number of Character in printf
  - %c or %s Display one Character or String
  - %e,E,f,g,G Real a Floating-point Number
  - %l or %L Convert to Double or LongDouble
  - %% Display % in User Screen

Format Specifiers(3)

- Control Character or Escape Sequence
  - \' Output the single quote
  - \" Output the double quate
  - \? Output the question mark
  - \\ Output the backslash
  - \a Audible(bell) or Visual Alert

Format Specifiers(4)

- Control Character or Escape Sequence
  - \b Blank one position
  - \f Start Next logical page
  - \n Beginning a New line
  - \r Beginning a Current line
  - \t or \v Next Horizontal or Vertical Tab Pos.

Examples of Input Statement

- scanf("%d",&number);
- scanf("%c %c",&ch1,&ch2);
- scanf("%f",&realnum);
- scanf("%s",str);
- scanf("%d%n",&num,&len);

Output Statement

- printf("Number : %d",num);
- printf("String : %s ",str);
- printf("%d + %d = %d",n1,n2,n1+n2);
- printf("%4d %.3f",X,Y);
- putch(ch); puts(str);

Program Examples

```
int main()  { return(0); }

#include <stdio.h>
int main()  {
    printf("Silpakorn University");
    return(0);
}
```

Program Examples(2)

```
#include <stdio.h>

int main()  {
   printf("My name is Opas Wongtaweesap\n");
   printf("Computer Science\n");
   printf("Silpakorn University\n");
   return(0);
}
```


Program Examples(3)

```
#include <stdio.h>
int main()  {
   int num1, num2, sum;
   scanf("%d",&num1);
   scanf("%d",&num2);
   //scanf("%d %d",&num1,&num2);
   sum = num1 + num2;
   printf("Sum ==> %d",sum);
   return(0);
}
```


Program Examples(4)

```
#include <stdio.h>

int main()  {
   int num1, num2, sum;
   printf("Input Number1 :: ");scanf("%d",&num1);
   printf("Input Number2 :: ");scanf("%d",&num2);
   sum = num1 + num2;
   printf("Sum ==> %d",sum);
   return(0);
}
```


Program Examples(5)

```
#include <stdio.h>

int main()  {
   int num1,num2,sum;
   printf("Input Number1 :: ");scanf("%d",&num1);
   printf("Input Number2 :: ");scanf("%d",&num2);
   sum = num1 + num2;
   printf("Sum ==> %d + %d = %d",num1,num2,sum);
   return(0);
}
```


Program Examples(6)

```
#include <stdio.h>
int main()  {
   int num1, num2;
   printf("Input Number1 :: ");scanf("%d",&num1);
   printf("Input Number2 :: ");scanf("%d",&num2);
   printf("Sum ==> %d + %d = %d",num1,num2,
                  num1+num2);
   return(0);
}
```