FIT ICT Software Development

Lecture 4

Working with Variables

Lecturer : Charles Tyner

FIT ICT Software Development

Creating Program Variables

- A variable has a name and a value
- The name references an address in memory where the value is stored
- The name can thus be used to access the value
- Variable names are **case sensitive** so variables named VAR, Var and var would be treated as three separate variables

FIT ICT Software Development

Creating Program Variables

Variable names must comply with the naming conventions set out below:

Naming Rule	Example
CANNOT contain C keywords	register
CANNOT contain arithmetic operators	x+y*z
CANNOT contain punctuation characters	%$£#@!
CANNOT contain any spaces	user name
CANNOT start with a number	2bdone
CAN contain numbers elsewhere	good1
CAN contain mixed case	UserName
CAN contain underscores	Is_ok

FIT ICT Software Development

Creating Program Variables

- Good practice to choose meaningful names for variables
- A variable has to be **declared** using the syntax *data-type variable-name;*
- Multiple variables of same data type can be created in a single declaration as a comma-separated list:

data-type variable_name-1, variable-name-2;

FIT ICT Software Development

Data Types in C

Data type	Description	Example
char	A single byte. Capable of storing just one character	‘A’
int	An integer whole number	134
float	A floating-point number correct to six decimal places	0.012345
double	A floating-point number correct to tendecimal places	0.0123456789

FIT ICT Software Development

Data Types in C

- Four data types allocate different amounts of memory for storing data
- Smallest is **char**, which uses a single byte
- Largest is **double**, which uses 8 bytes
- **double** should be used sparingly, only if absolutely necessary
- **float** is allocated 4 bytes
- **int** depends on whether it is **long** or **short** and ranges from 2 to 4 bytes

FIT ICT Software Development

More on Variables

- Variables should be declared **before** any executable code appears in a function
- Variables are *initialised* by having a value assigned to them
- Optionally, a variable may be initialised at the time of declaration
- IMPORTANT: values of *char* data type must always be enclosed in single quotes, and strings in double quotes

FIT ICT Software Development

Code snippet below declares and, in some cases, initialises variables with appropriate values, as described in the code comments:

```
int num1, num2;          /*Declares 2 integer variables*/
char letter;             /*Declares a character variable*/
float decimal = 7.5;     /*Declares & initialises a floating-
                           point variable*/
num1 = 100;              /*Initialises integer variables*/
num2 = 150;

Letter = 'A';            /*Initialises the character variable*/
```

FIT ICT Software Development

Displaying Variable Values

- Value of variables can be displayed using *printf()* function
- Format is specified as an argument of function

Specifier	Description	Example
%d	Integer -32768 to +32768	100
%ld	Long integer -2^{31} to $+2^{31}$	123456789
%f	Floating point number	0.123456
%c	Single character	'X'
%s	String of characters	"Hello World!"
%p	Machine memory address	0x0022FF34

FIT ICT Software Development

Displaying Variable Values

Format Specifiers

- Can ensure output occupies specific minimum number of spaces
- To achieve this, place digit(s) representing space to occupy immediately after % character (**%7d)**
- To fill blank spaces with leading zeros, place a zero between % character and specified number **(%07d**)

FIT ICT Software Development

Displaying Variable Values

Precision Specifiers

- Can be used to specify how many decimal places to display
- Used with %f specifier
- Consists of a dot followed by a number (**%.2f**)
- Can be combined with format specifier
- To display seven spaces including two decimal places, and empty spaces filled with zeros: (**%07.2f**)

FIT ICT Software Development

Displaying Variable Values

1. Begin a new program with a preprocessor instruction to include standard input/output library:

```
#include <stdio.h>
```

2. Add main function that declares and initialises two variables:

```
int main() {
        int num = 100 ;
        double pi =  3.1415926536 ;
}
```

FIT ICT Software Development

Displaying Variable Values

3. After variable declarations, insert statements to produce output using format and precision specifiers:

```
printf( "Integer is %d \n",  num);
printf( "Values are %d and %f \n", num, pi );
printf( "%%7d displays %7d \n", num);
printf("%%07d displays %07d \n", num );
printf( "Pi is approximately %1.10f \n", pi);
printf( "Right-aligned %20.3f rounded pi \n",  pi);
printf( "Left-aligned %-20.3f rounded pi \n",  pi);
```

FIT ICT Software Development

Displaying Variable Values

4. At end of main function block, insert final statement to return a zero integer value, as required by the function declaration:

```
return 0;
```

5. Save file (with .c extension). Then, at command prompt, compile and execute program to see variables output in specified format. Study carefully!!

FIT ICT Software Development

Inputting Variable Values

The function ***scanf()*** is provided in the library file **stdio.h**

Used to get user input

Requires two arguments in its call:

1. type of data to be input
2. location where input will be stored

First argument must be one of the format specifiers e.g. "%d"

FIT ICT Software Development

Inputting Variable Values

Second argument must be a variable name preceded by the & ampersand character

Ampersand character has several uses in C

In current context, it should be understood as the "addressof" operator

Input data will therefore be stored at the memory location reserved for that variable

The ampersand character is not required if input data is a string of text

FIT ICT Software Development

Inputting Variable Values

scanf() function can be used to assign values to multiple variables

First argument will then contain a list of format specifiers, each separated by a space, and entire list must be enclosed in double quotes

Second argument will contain comma-separated list of variable names, each preceded by & character

The & *addressof* operator can be used to return the (hex) address where the data is stored

Combining ***printf()*** and ***scanf()*** functions creates basic interactivity between user and program

FIT ICT Software Development

Inputting Variable Values

1. Begin new program with preprocessor instruction to include standard i/o library functions

```
#include <stdio.h>
```

2. Add main function that declares three variables

```
int main()
{ char letter;
  int num1, num2;
}
```

FIT ICT Software Development

Inputting Variable Values

3. After varaible declarations, insert statements to get input from user:

```
printf("Enter any single keyboard character: ");
scanf("%c", &letter);
printf("Enter two nums separated by a space: ");
scanf("%d %d", &num1, &num2);
```

FIT ICT Software Development

Inputting Variable Values

4. Add code to output stored data:

```
printf("Numbers input: %d and %d \n", num1,num2);
printf("Letter input: %c",  letter);
printf(" Stored at: %p \n ", &letter);
```

FIT ICT Software Development

Inputting Variable Values

5. At end of main function block, add final statement to return a zero integer value, as required by function declaration

```
return 0;
```

6. Save and compile program. Execute it and input data when requested. Examine output.

NB: Format specifier for an address in memory is **%p**

FIT ICT Software Development

Casting

Casting is the process of converting a variable from one data type to another

Compiler will implicitly cast if logical to do so

E.g. where an integer is assigned to a **float** variable, the compiler will add .00

Any data stored in a variable can be explicitly "coerced" into a variable of a different type

var_name2 = (modifiers data-type) var_name1

FIT ICT Software Development

Casting

Cast command states data type to which variable should be converted in parentheses before the name of the variable storing the original data

Should also include any modifiers, if appropriate e.g. **unsigned**

Casting does not change original data type but copies that value into as a different data type

FIT ICT Software Development

Casting

When casting from **float** to **int**, value is truncated, not rounded

When casting **char** to **int**, result is numerical ASCII code for that character

Uppercase in range ASCII 65-90; lowercase in range 97-120

Casting **int** in above ranges into **char** will produce equivalent letter

FIT ICT Software Development

Casting

Often necessary to cast **int** to **float** for accuracy

Otherwise, division can produce truncated result.

E.g. int x=7, y=5

Statement float z = x/y assigns value 1.000000 to z

Statement float z = (float)x / (float)y assigns 1.400000

Casting long floating-point **double** to shorter **float** type rounds up or down as appropriate

double decimal = 0.1234569

(float) decimal = 0.123457

FIT ICT Software Development

Casting

1. Begin new program with preprocessor instruction

```
#include <stdio.h>
```

2. Add main function to declare and initialise variables

```
int main()
{ float num = 5.75;
  char letter = 'A';
  int zed = 90;
  int x = 7, y = 5;
  double decimal = 0.1234569;
}
```

FIT ICT Software Development

Casting

3. Insert statements to output cast equivalent of each variable value

```
printf("Float cast to int: %d\n", (int)num);
printf("Char cast to int: %d\n", (int)letter);
printf("Int cast to char: %c\n", (char)zed);
printf("Float arithmetic: %f\n", (float)x/(float)y);
printf("Double cast to float: %f\n", (float)decimal);
```

4. Add statement to return a zero

```
return 0;
```

5. Save, compile, execute and study output

FIT ICT Software Development

Creating Array Variables

Array variables declared as normal but with addition of size (number of elements) in square brackets

data-type array-name [number-of-elements] ;

Optionally, can be initialised at declaration by assigning values to each element as comma-separated list within curly brackets

int arr[3] = {1,2,3};

FIT ICT Software Development

Creating Array Variables

Arrays can be used to store strings of text

Each element in array of **char** data type stores single character

Adding special **\0** null character escape sequence in array's final element "promotes" array to string status

char str[4] = {'C','a','t','\0'};

Entire string can then be referenced using just array name and displayed using **printf()** and **%s** specifier

printf("%s", str);

FIT ICT Software Development

Creating Array Variables

1. Start new program with preprocessor instruction

```
#include <stdio.h>
```

2. Add main function that declares array of three elements

```
int main()
{
int arr[3];
}
```

FIT ICT Software Development

Creating Array Variables

3. Insert code to initialise integer array elements

```
arr[0] = 100;
arr[1] = 200;
arr[2] = 300;
```

4. Insert statement to create and initialise character array to hold string of text

```
char str[10] = {'C', ' ', 'P', 'r', 'o', 'g', 'r', 'a', 'm', '\0' };
```

FIT ICT Software Development

Creating Array Variables

5. Insert code to display values of elements of integer array and string of character array

```
Printf("First element value: %d\n", arr[0]);
Printf("Secondelement value: %d\n", arr[1]);
Printf("Third element value: %d\n", arr[2]);
Printf("String: %s\n", str);
```

6. At end of function, return zero integer

```
return 0;
```

7. Save, compile, run, study

FIT ICT Software Development

Describing Multiple Dimensions

Element numbers more correctly called “array index”

Index numbers start at 0

Content	A	B	C	D	E
Index	[0]	[1]	[2]	[3]	[4]

Arrays can also be multi-dimensional. These arrays will have multiple indices

First Index [0]	A	B	C	D	E
[1]	F	G	H	I	J
Second Index	[0]	[1]	[2]	[3]	[4]

FIT ICT Software Development

Describing Multiple Dimensions

1. Start new program with preprocessor instruction

```
#include <stdio.h>
```

2. Add main function that declares and initialises a two-dimwensional integer array

```
int main()
{
int matrix[2][3] = { { 'A','B','C'}, {1,2,3}};
}
```

FIT ICT Software Development

Describing Multiple Dimensions

3. Insert code to display all elements of first index

```
printf("Element [0][0] contains %c\n", matrix[0][0]);
printf("Element [0][1] contains %c\n", matrix[0][1]);
printf("Element [0][2] contains %c\n", matrix[0][2]);
```

4. Insert code to display all elements of second index

```
printf("Element [1][0] contains %c\n", matrix[1][0]);
printf("Element [1][1] contains %c\n", matrix[1][1]);
printf("Element [1][2] contains %c\n", matrix[1][2]);
```

FIT ICT Software Development

Describing Multiple Dimensions

5. At end of function block return a zero integer

```
return 0;
```

6. Save, compile, execute, study