



Fundamentals of C Programming (Part 2)

By Noor Hazlini binti Borhan

Overview

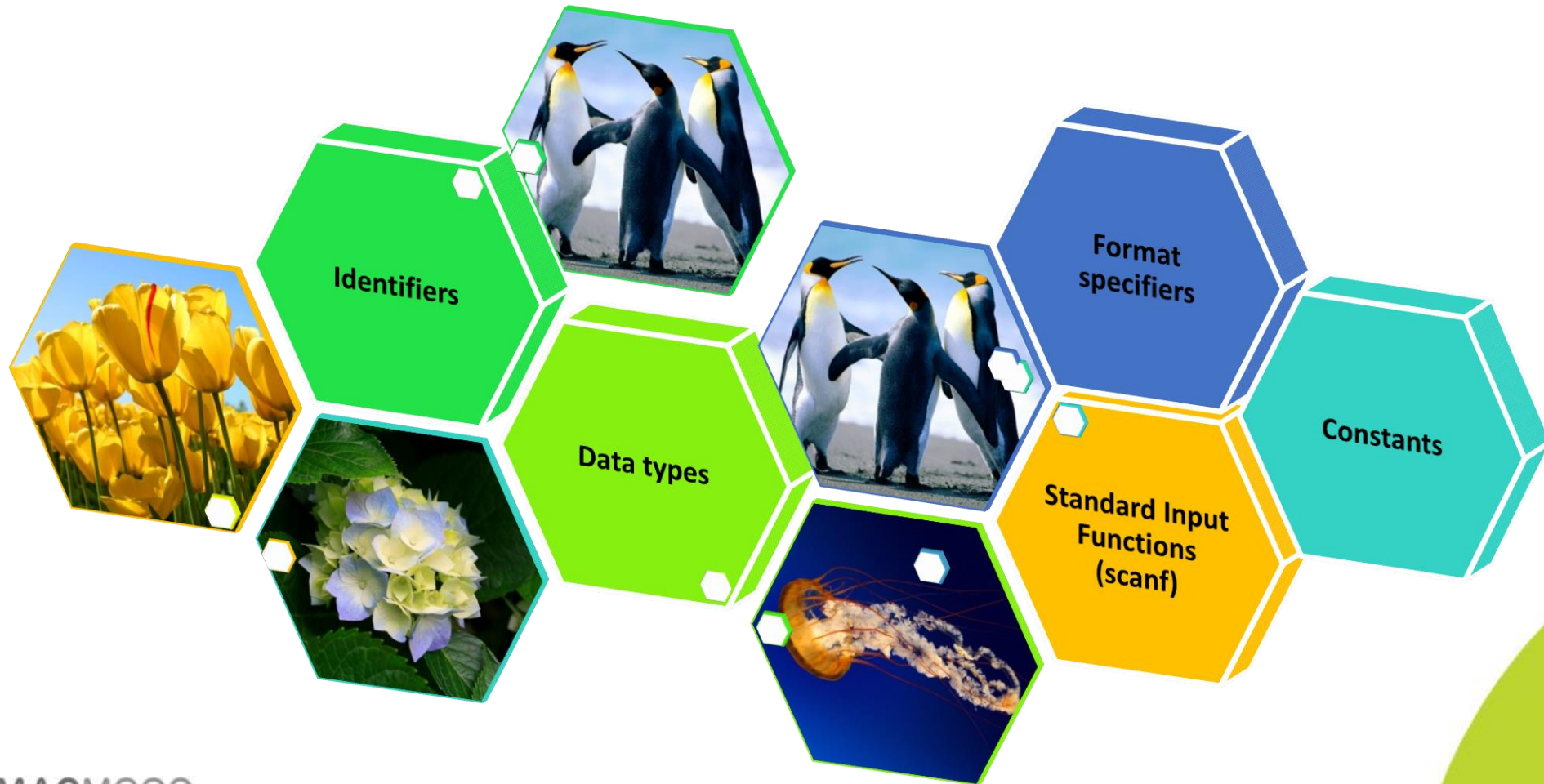
- ✓ Fundamental components in a program code:
 - identifiers
 - data types
 - format specifiers
 - constants
 - standard input functions (scanf)
- ✓ Simple C programming example.

Learning Outcomes

On completion of this unit, you should be able to:

- ✓ Define the components in a program code.
- ✓ Declare the variables.
- ✓ Write a simple C program.
- ✓ Apply the format specifiers and scanf().

In **general**, a C Program consists of the following **components**:



Simple C Program-Version 2

```
/*Program Name:Student Details Program
Author Name:Noor Hazlini
Descriptions:This is a simple C program. In this program, some basic components of C such as
constants,variables,scanf(),format specifiers and data types will be applied.*/
#include <stdio.h>                                //pre-processor directives
#define YEAR 2016                                // constant declarations
int main()                                        //int main()
{                                                  //open curly brace

    int student_id,year_born,student_age;        //variable declarations

    printf("Welcome to C programming Course!!\n"); //printf() statement
    printf("The purpose of this program is to save the student details\n");

    printf("Enter your student ID:\n");
    scanf("%d",&student_id);                    //scanf() with format specifiers

    printf("Enter your year of born:");
    scanf("%d",&year_born);

    student_age = YEAR - year_born;

    printf("Your student ID is %d and your age is %d",student_id,student_age); //printf() with %d format specifiers
    return 0;
}                                                  //close curly brace
```

Identifiers

Also known as **“variables”**

Words to **represent & reference** certain program entities

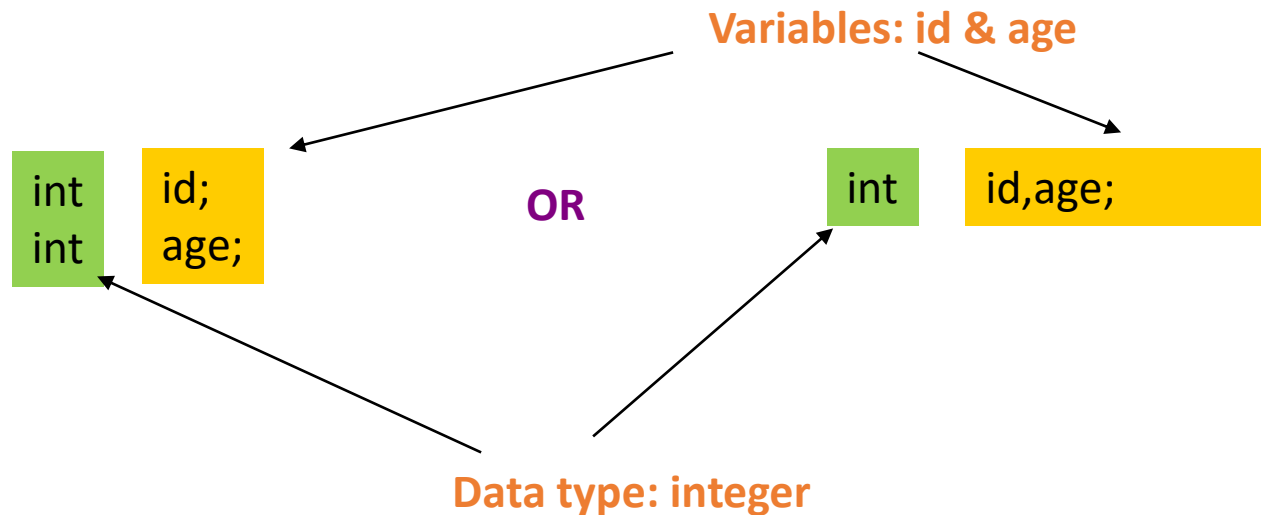
Identifiers give **unique names** to various objects in a program.

Refer to **location in memory** where value can be stored

Case sensitive: differentiates between **lowercase & uppercase** letter

Declarations

- **A declaration** is a syntactical element that associates a type with a program entity, such as a variable.
- Variables must be declared before they can be used.



Initializing Variables

- We can assign an **initial value** to a variable when we declare it.
For example:

```
int no=10;
```

- Sets the int variable to ten as soon as it is created. This is just the same as:

```
int no;
```

```
no=10;
```

Assignment operator

Rules for constructing Identifiers:

- ☐ A to Z , a to z , 0 to 9 , and the underscore “_”
- ☐ The first character must be a letter or an underscore
- ☐ Only the first 32 characters as significant
- ☐ There can be no embedded blanks
- ☐ Reserved words cannot be used as identifiers
- ☐ Identifiers are case sensitive

Example: Identifiers

Valid identifiers	Invalid identifiers
• student_ID, • Item_12, • count, • perimeter_of_circle • _age	• student gender //embedded blank • 28thMarch //the first character is a digit • Width*Length //special character* • \$money //special character \$ • break //reserved word

Tips to choose the name for identifier:

- Avoid excessively short and cryptic names such as x or yz.
- Instead, to add to the readability of your program, use reasonably **descriptive names**, such as student_id and student_age.
- Use **underscore or capital letters** to separate words in identifiers that consist of two or more words, such as student_program or StudentProgram are much easier to read than studentprogram .

Data types

- A data type is a set of data values and a set of operations on those values.
- A data type is used to **identify the type** of a variable when the variable is declared.
- Its also can be used to identify the type of the **return value** of a function.

2 types of data type:

Built-in data types

- **1. Fundamental data types:**
 - Example: int, char, double, float, void
 - Some of these data types have short, long, signed, and unsigned variants.
- **2. Derived data**
 - Examples: arrays, strings and structures.

Programmer-defined data types

- **(covered in Structures)**

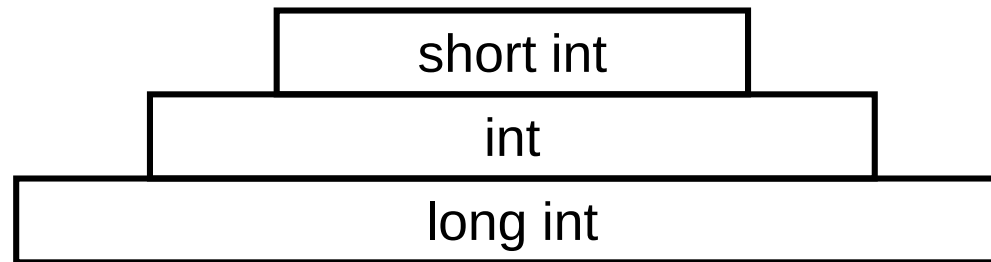
Example: Data types

Data type	Description	Size (no of bytes)	Range
int	numeric program variables of integer type.	2	-32768 to +32767
char	character variables.	1	0 - 255
double	floating-point numbers	8	Approximately 15 digits of Precision
float	floating-point numbers	4	-2,147,483,648 to +2,147,483,647
unsigned short	short unsigned integer	3	0-65535

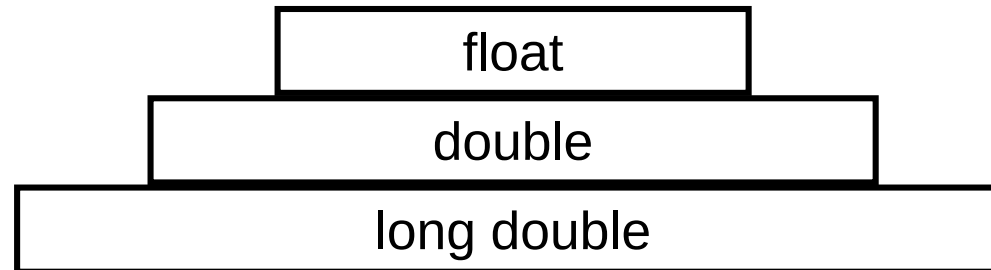
Data type	Description	Size (no of bytes)	Range
signed long	Long signed integer	4	-2,147,483,648 to +2,147,483,647
unsigned char	Unsigned character	1	0 to 255
signed char	Character	1	-128 to 127
signed short/short	Short signed integer	2	-32768 to +32767
void	No data type	0	

- **Unsigned** means the number is **always zero or positive, never negative**.
- **Signed** means the number **may be negative or positive** (or zero).
- If you don't specify signed or unsigned, the data type is presumed to be **signed**.
- Thus, signed short and short are the same.

- An integer type is a number without a fractional part.
- It is also known as an integral number.
- C supports three different sizes of the integer data type: **short int**, **int** and **long int**.
- $\text{sizeof}(\text{short int}) \leq \text{sizeof}(\text{int}) \leq \text{sizeof}(\text{long int})$



- A floating-point type is a number with a fractional part, such as 43.32.
- The C language supports three different sizes of floating-point: **float**, **double** and **long double**.
- $\text{sizeof}(\text{float}) \leq \text{sizeof}(\text{double}) \leq \text{sizeof}(\text{long double})$



Constants

- **Constants** refer to fixed values that may not be altered by the program.
- All the data types we have previously covered can be defined as **constant data types** if we wish to do so.
- The **constant** data types must be defined before the main function.
- The use of **constants** is mainly for making your programs easier to be understood and modified by others and yourself in the future.

- We can define constants in a C program in the following ways:
- By “const” keyword
- By “#define” preprocessor directive
- The format is as follows:
#define CONSTANTNAME value
- The **constant** name is normally written in capitals and does not have a semi-colon at the end.

Types of C Constant

- Integer constants
- Real or Floating point constants
- Octal & Hexadecimal constants
- Character constants
- String constants

Constant type	data type	Example
Integer constants	int unsigned int long int	53, 762, -478 etc 5000u, 1000U etc
Real or Floating point constants	float double	10.456789
Octal constant	int	013 /* starts with o */
Hexadecimal constant	int	0x90 /* starts with 0x */
character constants	char	'A' , 'B', 'C'
string constants	char	"ABCD" , "Hi"

Example: Constants

Constant Writing Format

#define CONSTANTNAME value

Examples: **#define YEAR 2016**

#define MAX 100

#define PI 3.142

“const” keyword

Examples:

const int height = 100; /*int constant*/

const float number = 3.14; /*Real constant*/

const char letter = 'A'; /*char constant*/

const char letter_sequence[10] = "ABC"; /*string constant*/

```
/*Program Name:Student Details Program
```

```
Descriptions:This is a simple C program. In this program, some basic components of C such as constants,variables,scanf(),format specifiers and data types will be applied.*/
```

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

```
#define YEAR 2016
```

```
int main()
```

```
{
```

```
    int student_id,year_born,student_age;
```

```
    const char letter = 'A';
```

```
    printf("Welcome to C programming Course!!\n");
```

```
    printf("The purpose of this program is to save the student details\n");
```

```
    printf("Enter your student ID:\n");
```

```
    scanf("%d",&student_id);
```

```
    printf("Enter your year of born:");
```

```
    scanf("%d",&year_born);
```

```
    student_age = YEAR - year_born;
```

```
    printf("Your student ID is %d and your age is %d\n",student_id,student_age);
```

```
    printf("value of letter : %c \n", letter );
```

```
    return 0;
```

```
}
```

```
    //close curly brace
```

```
//pre-processor directives
```

```
//define constant declarations
```

```
//int main()
```

```
//open curly brace
```

```
//variable declarations
```

```
//char constant
```

```
//printf() statement
```

```
//scanf() with format specifiers
```

Define constant

Keyword const

Standard Input Function scanf()

- The declaration of the standard library function `scanf()` is contained in the header file `stdio.h`.
- Therefore , the preprocessor directive `#include <stdio.h>` must appear earlier in the program.
- `scanf()` **reads** data entered via the keyboard.
- Many common format specifiers, which indicate the type of data expected from the keyboard.
- Data argument identifiers **must** be preceded with an ampersand (&).
- The (&) ahead of the variable signifies the address in memory where the data will be stored (its a pointer to the variable).

scanf(format string, input list);

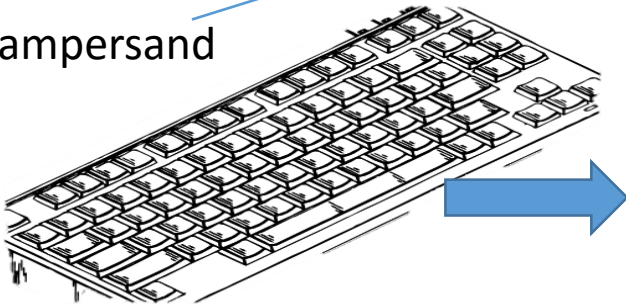
Format specifiers

"%d%f"

&id, &mark

scanf("%d%f", &id, &mark);

ampersand



.. 125 88..
Input stream

w

125
(id)

x

88
(mark)

Format Specifiers

- In C programming we need lots of format specifier to work with various data types.
- Format specifiers define the type of data, whether to print formatted output (printf) or to take formatted input (scanf).

Example: Format Specifiers

Format specifier	Description	Supported data types
%c	Character	char unsigned char
%d	Signed Integer	short unsigned short int long
%e or %E	Scientific notation of float values	float double
%f	Floating point	float
%i	Signed Integer	short unsigned short int long
%l or %ld or %li	Signed Integer	long
%o	Octal representation of Integer.	short unsigned short int unsigned int long

Format specifier	Description	Supported data types
%p	Address of pointer to void void *	void *
%s	String	char *
%u	Unsigned Integer	unsigned int unsigned long
%x or %X	Hexadecimal representation of Unsigned Integer	short unsigned short int unsigned int long
%lf	Floating point	Double
%n	Prints nothing	
%%	Prints % character	

Example: format specifiers for printf() & scanf() -%d

*/*Program Name:Student Details Program*

Descriptions:This is a simple C program. In this program, some basic components of C such as constants,variables,scanf(),format specifiers and data types will be applied.//*

`#include <stdio.h>`

//pre-processor directives

`#define YEAR 2016`

//constant declarations

`int main()`

//int main()

`{`

//open curly brace

`int student_id,year_born,student_age;`

//variable declarations

`printf("Welcome to C programming Course!!\n");`

//printf() statement

`printf("The purpose of this program is to save the student details\n");`

`printf("Enter your student ID:\n");`

`scanf("%d",&student_id);`

//scanf() with %d format specifiers

`printf("Enter your year of born:");`

`scanf("%d",&year_born);`

`student_age = YEAR - year_born;`

`printf("Your student ID is %d and your age is %d",student_id,student_age);` *//printf() with %d format specifiers*

`return 0;`

//close curly brace