**Variable Names**

**Writer:** Sayak Haldar

Before go into deep for variable names, let’s learn the basics:

**Identifiers are names for entities in a C program, such as variables, arrays, functions, structures, unions and labels.** An identifier can be composed only of uppercase, lowercase letters, underscore and digits, but should start only with an alphabet or an underscore. If the identifier is not used in an external link process, then it is called as internal. Example: Local variable. If the identifier is used in an external link process, then it is called as external. Example: Global variable

**Rules for constructing identifiers**

1. The first character in an identifier must be an alphabet or an underscore and can be followed only by any number alphabets, or digits or underscores.

2. They must not begin with a digit.

3. Uppercase and lowercase letters are distinct. That is, identifiers are case sensitive.

4. Commas or blank spaces are not allowed within an identifier.

5. Keywords cannot be used as an identifier.

6. Identifiers should not be of length more than 31 characters.

7. Identifiers must be meaningful, short, quickly and easily typed and easily read.

**kinds of identifiers**

C defines two kinds of identifiers:

Internal

External

**Internal identifier**

If the identifier is used in an external link process, then it is called as external. These identifiers are also known as external names; include function names and global variable names that are shared between source files. It has at least 63 significant characters.

**External identifier**

If the identifier is not used in an external link process, then it is called as internal. These identifiers are also known as internal names; includes the names of local variables. It has at least 31 significant characters.

**Differentiate between Keywords words and identifiers**

|  |  |
| --- | --- |
| **Keyword** | **Identifier** |
| Predefined-word | User defined word |
| Must be written in lowercase only Can written in lowercase and uppercase | Can written in lowercase and uppercase |
| Has fixed meaning | Must be meaningful in the program |
| Whose meaning has already been explained to the C compiler | Whose meaning not explained to the C compiler |
| Combination of alphabetic characters | Combination of alphanumeric characters |
| Used only for it intended purpose | Used for required purpose |
| Underscore character is not considered as a letter | Underscore character is considered as a letter |

**1. C99 standard guarantees uniqueness of \_\_\_\_ characters for internal names.**

a) 31

b) 63

c) 12

d) 14

**Answer: b**

Explanation: ISO C99 compiler may consider only first 63 characters for internal.

**2. C99 standard guarantees uniqueness of \_\_\_\_\_ characters for external names.**

a) 31

b) 6

c) 12

d) 14

**Answer: a**

Explanation: ISO C99 compiler may consider only first 31 characters for external

variables having 31 characters due to which it may not be unique.

3. **Which of the following is not a valid variable name declaration?**

a) int \_\_a3;

b) int \_\_3a;

c) int \_\_A3;

d) None of the mentioned

**Answer: d**

**4. Which of the following is not a valid variable name declaration?**

a) int \_a3;

b) int a\_3;

c) int 3\_a;

d) int \_3a

**Answer: c**

Explanation: Variable name cannot start with a digit.

**5. Variable names beginning with underscore is not encouraged. Why?**

a) It is not standardized

b) To avoid conflicts since assemblers and loaders use such names

c) To avoid conflicts since library routines use such names

d) To avoid conflicts with environment variables of an operating system

Answer: c

**6. All keywords in C are in**

a) LowerCase letters

b) UpperCase letters

c) CamelCase letters

d) None of the mentioned

Answer: a

Explanation: None.

**7. Variable name resolving (number of significant characters for uniqueness of variable) depends on**

a) Compiler and linker implementations

b) Assemblers and loaders implementations

c) C language

d) None of the mentioned

Answer: a

Explanation: It depends on the standard to which compiler and linkers are adhering to.

**8. Which of the following is not a valid C variable name?**

a) int number;

b) float rate;

c) int variable\_count;

d) int $main;

Answer: d

Explanation: Since only underscore and no other special character is allowed in a variable name, it results in an error.

**9. Which of the following is true for variable names in C?**

a) They can contain alphanumeric characters as well as special characters

b) It is not an error to declare a variable to be one of the keywords(like goto, static)

c) Variable names cannot start with a digit

d) Variable can be of any length

**Answer: c**

**10. Which is valid C expression?**

a) int my\_num = 100,000;

b) int my\_num = 100000;

c) int my num = 1000;

d) int $my\_num = 10000;

Answer: b

Explanation: Space, comma and $ cannot be used in a variable name.

**11. What is the output of this C code?**

#include <stdio.h>

int main()

{

printf("Hello World! %d \n", x);

return 0;

}

a) Hello World! x;

b) Hello World! followed by a junk value

c) Compile time error

d) Hello World!

**Answer: c**

Explanation: It results in an error since x is used without declaring the variable x.

Output:

$ cc pgm1.c

pgm1.c: In function ‘main’:

pgm1.c:4: error: ‘x’ undeclared (first use in this function)

pgm1.c:4: error: (Each undeclared identifier is reported only once

pgm1.c:4: error: for each function it appears in.)

**12. What is the output of this C code?**

#include <stdio.h>

int main()

{

int y = 10000;

int y = 34;

printf("Hello World! %d\n", y);

return 0;

}

a) Compile time error

b) Hello World! 34

c) Hello World! 1000

d) Hello World! followed by a junk value

**Answer: a**

Explanation: Since y is already defined, redefining it results in an error.

Output:

$ cc pgm2.c

pgm2.c: In function ‘main’:

pgm2.c:5: error: redefinition of ‘y’

pgm2.c:4: note: previous definition of ‘y’ was here

**13. Which of the following is not a valid variable name declaration?**

a) float PI = 3.14;

b) double PI = 3.14;

c) int PI = 3.14;

d) #define PI 3.14

**Answer: d**

Explanation: #define PI 3.14 is a macro preprocessor, it is a textual substitution.

**14. What will happen if the below program is executed?**

#include <stdio.h>

int main()

{

int main = 3;

printf("%d", main);

return 0;

}

a) It will cause a compile-time error

b) It will cause a run-time error

c) It will run without any error and prints 3

d) It will experience infinite looping

**Answer: c**

Explanation: A C program can have same function name and same variable name.

$ cc pgm3.c

$ a.out

3

**15. What is the problem in following variable declaration?**

float 3Bedroom-Hall-Kitchen?;

a) The variable name begins with an integer

b) The special character ‘-‘

c) The special character ‘?’

d) All of the mentioned

Answer: d

Explanation: A variable name cannot start with an integer, along with that the C compiler

interprets the ‘-‘ and ‘?’ as a minus operator and a question mark operator respectively.

**16. Comment on the output of this C code?**

#include <stdio.h>

int main()

{

int ThisIsVariableName = 12;

int ThisIsVariablename = 14;

printf("%d", ThisIsVariablename);

return 0;

}

a) The program will print 12

b) The program will print 14

c) The program will have a runtime error

d) The program will cause a compile-time error due to redeclaration

**Answer: b**

Explanation: Variable names ThisIsVariablename and ThisIsVariableName are both distinct as C is case sensitive.

Output:

$ cc pgm4.c

$ a.out

14

**17.Which of the following cannot be a variable name in C?**

a) volatile

b) true

c) friend

d) export

**Answer: a**

Explanation: volatile is C keyword.