Keywords, Identifier, Literals, Operators and Expression Assignment

Mandatory:

1. Choose all valid identifiers

a. int int

b. int \_numvalue

c. float price\_money

d. char name1234567890123456789012345678901234567890

e. char name value

f. char $name

valid identifiers are

int \_numvalue,

float price\_money,

char name1234567890123456789012345678901234567890 ,

char $name

2. What is the meaning of the following keywords, show the usage

a. auto

The auto keyword is used to declare automatic (local) variables in C. By default, all local variables are automatic in C, so the auto keyword is not usually needed explicitly. However, it was once used to declare variables that are automatically allocated and deallocated, typically on the stack. In modern C, auto is often redundant, but it is still valid.

Auto int a=10;

b. extern

The extern keyword is used to declare variables or functions that are defined in another file

#include <stdio.h>

extern int count;

int main() {

count = 10;

printf("Count is %d\n", count);

return 0;

}

// file2.c

int count;

c. volatile

The volatile keyword is used to tell the compiler that a variable's value may be changed by something outside the current program flow, such as hardware, an interrupt, or a different thread. This prevents the compiler from optimizing the variable, ensuring that every read or write to the variable happens as expected.

#include <stdio.h>

volatile int flag = 0;

void interrupt\_handler()

{

flag = 1;

}

int main() {

while (flag == 0) {

printf("Interrupt received!\n");

return 0;

}

d. sizeof

The sizeof operator is used to determine the size, in bytes, of a data type or variable

#include <stdio.h>

int main() {

int x = 10;

double y = 20.5;

char z = 'A';

printf("Size of int: %zu bytes\n", sizeof(x)); // Size of variable x

printf("Size of double: %zu bytes\n", sizeof(y)); // Size of variable y

printf("Size of char: %zu bytes\n", sizeof(z)); // Size of variable z

printf("Size of int type: %zu bytes\n", sizeof(int)); // Size of int type

printf("Size of double type: %zu bytes\n", sizeof(double)); // Size of double type

return 0;

}

e. const

The const keyword is used to declare variables whose values cannot be modified after they are initialized.

#include <stdio.h>

int main() {

const int MAX\_VALUE = 100; // Declare constant variable

printf("Max value is %d\n", MAX\_VALUE);

// MAX\_VALUE = 200; // This would cause an error, as MAX\_VALUE is constant

return 0;

}

3. Explain the difference between the following variables.

a. char \*ptr = “ABC”;

b. char arr[]=”ABC”;

Can you manipulate the contents of ptr? Why?

Can you manipulate the contents of arr? Why?

Which one of the above is a string literal?

a.In this declaration, ptr is a pointer to a char. The string "ABC" is a **string literal** (a constant string), and the pointer ptr is assigned the address of the first character of the string literal "ABC".

b.In this declaration, arr is a character array initialized with the string "ABC". Here, the **string literal "ABC" is copied into the array arr**, and the array arr is **modifiable** because it resides in the **data segment** (not read-only memory like the string literal).

4. Predict the output of the following code .

void main()

{

//set a and b both equal to 5.

int a=5, b=5;

//Print them and decrementing each time.

//Use postfix mode for a and prefix mode for b.

printf("\n%d %d",a--,--b);

printf("\n%d %d",b++,--b);

}

5 4

4 4

5. Refer the code snippet. It fails with error. Fix it.

Refer the code snippet. It fails with error. Fix it.

#include<stdio.h>

int main()

{

int i,k;

const int num;

/\* for(i = 0;i < 9;i++)

{

k = k + 1;

} \*/

num = num + k; /\* Compiler gives the error here \*/

printf("final value of k:%d\n",k);

printf("value of num:%d\n",num);

return 0;

}

A screen shot of a computer program

Description automatically generated

6. Consider the following code snippet. Evaluate the value of f1, f2 and f3.

int main()

{

int i = 10;

int j = 3;

float f1 = i / j;

float f2 = (float ) i / j;

float f3 = (float ) (i / j);

}

