**Fundamentals of Computer programming**

**ASSIGNMENT**

**IV**

**Prepared by: Allen Nembang**

**Shift: Morning**

**Roll Number: 2**

**BscCSIT**

**Theoretical Assignment**

1. **Explain in detail about storage class. Write in detail about 4 different storage class.**

A storage class defines the scope (visibility) and life-time of variables and/or functions within a C Program. They precede the type that they modify.

The four different storage classes in a C program are:-

* **Auto**
* **Register**
* **Static**
* **Extern**

**Auto Storage Class**: - The **auto** storage class is the default storage class for all local variables.

{

int mount;

auto int month;

}

The example above defines two variables with in the same storage class. 'auto' can only be used within functions, i.e., local variables.

**Register Storage Class**: - The register storage class is used to define local variables that should be stored in a register instead of **RAM**. This means that the variable has a maximum size equal to the register size (usually one word) and can't have the unary '&' operator applied to it (as it does not have a memory location).

{

register int miles;

}

The register should only be used for variables that require quick access such as counters. It should also be noted that defining 'register' does not mean that the variable will be stored in a register. It means that it MIGHT be stored in a register depending on hardware and implementation restrictions.

**Static Storage Class: -** The static storage class instructs the compiler to keep a local variable in existence during the life-time of the program instead of creating and destroying it each time it comes into and goes out of scope. Therefore, making local variables static allows them to maintain their values between function calls.

The static modifier may also be applied to global variables. When this is done, it causes that variable's scope to be restricted to the file in which it is declared.

In C programming, when static is used on a class data member, it causes only one copy of that member to be shared by all the objects of its class.

#include <stdio.h>

/\* function declaration \*/

void func(void);

static int count = 5; /\* global variable \*/

main() {

while(count--) {

func();

}

return 0;

}

/\* function definition \*/

void func( void ) {

static int i = 5; /\* local static variable \*/

i++;

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

}

**Result:**

i is 6 and count is 4

i is 7 and count is 3

i is 8 and count is 2

i is 9 and count is 1

i is 10 and count is 0

**Extern Storage Class: -** The extern storage class is used to give a reference of a global variable that is visible to ALL the program files. When you use 'extern', the variable cannot be initialized however, it points the variable name at a storage location that has been previously defined.

When you have multiple files and you define a global variable or function, which will also be used in other files, then extern will be used in another file to provide the reference of defined variable or function. Just for understanding, extern is used to declare a global variable or function in another file.

The extern modifier is most commonly used when there are two or more files sharing the same global variables or functions

1. **Write one simple example of for-loop. Explain about the steps how it works.**

Example of for-loop: -

#include<stdio.h>

Int main(){

/\* for loop execution \*/

for(int i=1;i<5;i=i+1){

printf(“Value of a: %d\n”,i);

}

return 0;

}

**Result**:-

Value of i: 1

Value of i: 2

Value of i: 3

Value of i: 4

**Syntax of for-loop:-**

**for ( init; condition; increment )**

**{**

**statement(s);**

**}**

Following steps describe the working of for-loop:-

* The **init step** is executed first, and only once. This step allows us to declare and initialize any loop variables. We are not required to put a statement here, as long as a semicolon appears.
* Next, the **condition** is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and the flow of control jumps to the next statement just after the ‘for’ loop.
* After the body of the ‘for’ loop executes, the flow of control jumps back up to the **increment** statement. This statement allows us to update any loop control

variables. This statement can be left blank, as long as a semicolon appears after the condition.

1. **Write one simple example of while-loop. Explain about the steps how it works.**

Example of while-loop:-

#include <stdio.h>

int main ()

{

/\* local variable definition \*/

int a = 5;

/\* while loop execution \*/

while( a < 10 )

{

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

a++;

}

return 0;

}

**Result:-**

value of a: 5

value of a: 6

value of a: 7

value of a: 8

value of a: 9

**Syntax of while-loop:-**

while(condition)

{

statement(s);

}

**Steps for working of while-loop:-**

* **condition**, may be any expression , and true is any nonzero value. The loop iterates while the condition is true.
* **statement(s)** may be a single statement or a block of statements. If the condition is true, statement(s) executes.

1. **Write a paragraph and an example of each about strlen(), strcpy(), strcat(), strcmp(), strrev().**

**strlen():-**

The strlen() function calculates the length of a given string.

**C strlen() Prototype**:-

size\_t strlen(const char \*str);

The function takes a single argument, i.e, the string variable whose length is to be found, and returns the length of the string passed.

The strlen() function is defined in <string.h> header file.

**Example:**

#include <stdio.h>

#include <string.h>

int main()

{

char a[20]="Program";

char b[20]={'P','r','o','g','r','a','m','\0'};

char c[20];

printf("Enter string: ");

gets(c);

printf("Length of string a = %d \n",strlen(a));

//calculates the length of string before null charcter.

printf("Length of string b = %d \n",strlen(b));

printf("Length of string c = %d \n",strlen(c));

return 0;

}

**Output:**

Enter string: String

Length of string a = 7

Length of string b = 7

Length of string c = 6

**strcpy():-**

The strcpy() function copies the string to the another character array.

**strcpy() Function prototype:-**

char\* strcpy(char\* destination, const char\* source);

The strcpy() function copies the string pointed by source (including the null character)

to the character array destination. This function returns character array destination.

The strcpy() function is defined in string.h header file.

**Example**:

#include <stdio.h>

#include <string.h>

int main()

{

char str1[10]= "awesome";

char str2[10];

char str3[10];

strcpy(str2, str1);

strcpy(str3, "well");

puts(str2);

puts(str3);

return 0;

}

**Output:**

awesome

well

**strcat(s1,s2):-**

This c library function concatenates string s2 onto the end of string s1.

**strcat() Function Prototype:-**

This function use string.h as a header file.

**Example**:

#include <stdio.h>

#include <string.h>

int main ()

{

char str1[12] = "Hello";

char str2[12] = "World";

/\* concatenates str1 and str2 \*/

strcat( str1, str2);

printf("strcat( str1, str2): %s\n", str1 );

return 0;

}

**Result**:

HelloWorld

**strcmp(s1,s2):-**

The C library function int strcmp(const char \*str1, const char \*str2) compares the string

pointed to, by str1 to the string pointed to by str2.

**Return Value:-**

* if Return value < 0 then it indicates str1 is less than str2.
* if Return value > 0 then it indicates str2 is less than str1.
* if Return value = 0 then it indicates str1 is equal to str2.

**Example:-**

#include<stdio.h>

#include<string.h>

int main ()

{

char str1[15],char str2[15];

int ret;

printf("Enter first string:");

scanf("%s",str1);

printf("Enter second string:");

scanf("%s",str2);

ret = strcmp(str1,str2);

if(ret<0){

printf("str1 is less than str2");

}else if(ret>0){

printf("str is greater than str2");

}else{

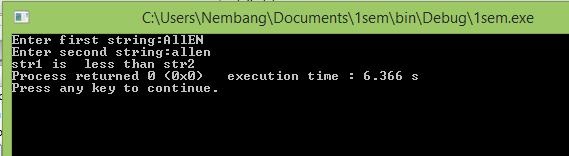
printf("str1 is equal to str2");

}

return(0);

}

**Output:-**

****

**strrev**():-

This function reverses a given string in C language. Syntax for strrev( ) function is given

below. char \*strrev(char \*string);

**Example:-**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int main ()

{

char str[10];

printf("Enter string:");

scanf("%s",str);

strrev(str);

printf("Reverse of entered string is:%s",str);

return(0);

}

**Result:-**

****

1. **What is an array? How array is different from ordinary variable?**

Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type.

An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

*For eg: - to declare a 10-element array called balance of type double, use this statement −* double balance[10];

Array is different from ordinary variable due to following points:-

* Array is known as the collection of multiple variables with its value whereas variable can store only single value at a time.
* Array is always declared, initialized, and accessed using subscript whereas ordinary variable don’t have any subscript.
* Syntax for array variable is :- data\_type v1[N1],v2[N2],…, where v1,v2 are name of variable and N1, N2 are the integer constants indicating the maximum size of array whereas syntax for ordinary variable is:- data\_type v1, v2,….

1. **How can you prepare a student mark sheets software using the things you’ve learned till now? Give me the list of the things you’ve learned and how can you use it to do what kind of operation to prepare a mark sheet of a student.**

For a mark sheets containing basic features of marks secured in each subject, total marks, percentage and division, I will use the following things:-

* Operator