**WORKPLACE:**

**Using pointer we can increase the size of an array or create a dynamic array in C programming although it is fixed.**

/\*INTEREST DRIVING THREE TIMES\*/

#include <stdio.h>

int main()

{

int p,n,count;

float r,si;

count=1;

while(count<=3)

{

printf("\nEnter values p,n and r=");

scanf("%d%d%f",&p,&n,&r);

si=p\*n\*r/100;

printf("Smple interest=%f",si);

count=count+1;

}

return 0;

}

/\*\*\* DEALING WITH STRING \*\*\*/

#include<stdio.h>

#include<string.h>

char main()

{

// create a string of 30 characters

char student[30] = "Manish Yadav";

printf (student); // will print "Schitt's Creek"

printf ("\n"); // break the line, "\n" is also a string

student[6] = 0; // terminate the string earlier

printf (student); // will print Manis

printf ("\n");

student[5] = 'a'; // change the first letter to 'S'

printf (student); // will print "schitt's Creek"

printf ("\n");

/\*

the reason the computer prints `"Schitt's Creek"`,

and not the entire 30 characters, is because it encounters the `0` and stops there.

\*/

strncpy (student, "MANISH", 8); // will completely replace the string

/\* example of a loop with strings \*/

int i = 0;

while (student[i] != 0)

{

printf ("%c", student[i]);

i += 1;

}

return 0;

}

INPUT SCORE DECISION OF RESULT

#include <stdio.h>

#include <stdlib.h>

int main()

{

int m1,m2,m3,m4,m5,per;

printf("Enter your marks");

scanf("%d%d%d%d%d",&m1,&m2,&m3,&m4,&m5);

per=(m1+m2+m3+m4+m5)/5;

if (per>=90)

printf("first division\n");

else

{

if(per>=80)

printf("second division\n");

else

{

if(per>=70)

printf("third division\n");

else

printf("fail");

}

}

printf(per);

return 0;

}

/\*PRINTING ARRAYS ELEMENT WITH REVERSE ORDER\*/

#include <stdio.h>

int main()

{

int A[] = {3,7,23,18,9};

for (int i= 4; i>=0; i=i-1)

{

printf("%d ", A[i]);

}

return 0;

}

/\*CHAR IF CONDITION\*/

#include <stdio.h>

#include <stdlib.h>

int main()

{

char sex,ms;

int age;

printf("Enter age,ms(M/U),sex(m/f)");

scanf("%d%c%c",&age,&sex,&ms);

if ((ms=='M')||(age>30&&ms=='U'&&sex=='m')||(ms=="U"&&sex="f"&&age>25))

printf("approved");

else printf("not approved");

return 0;

}

/\* FORMING A TRINGLE WITH THE INPUT NUMBER \*/

#include <stdio.h>

int main()

{ int n;

printf("\n");

printf("Enter a positive integer ");

scanf("%d",&n); /\*command used to take user input\*/

printf("%d",n);

printf("\n");

/\*implementation of nested loops \*/

int row;

for (row=1; row<=n; row=row+1)

{

int i;

for (i=0; i<row; i=i+1)

{

printf(" %d ", i);

}

printf("\n");

}

return 0; /\*end of program\*/

}

/\*\*\* CHANGING STRINGS \*\*\*/

#include <stdio.h>

int main()

{

char name[30]="RAHUL";

unsigned char age=20; // 0 TO 255

long money=-10;

char name2[30]="MANISH";

unsigned char age2=19;

long money2=100000;

printf("%s is %u years old has %lu rupees\n",name, age, money);

printf("%s is %u years old has %lu rupees\n",name2, age2, money2);

return 0;

}

#include <stdio.h>

int main()

{

struct Person

{

char name[30]; //30 bytes

unsigned int age; // 4 bytes

long money; // 8 bytes

unsigned short weight; // 2 bytes, 0 to 65536

unsigned short height; // 2 bytes

};

struct Person person[3];

for(int i=0;i<3; i++)

{

printf("Please enter details of the person %d\n",i);

printf("name:");

scanf("%s",person[i].name);

printf("age:");

scanf("%u",person[i].age);

printf("money:");

scanf("%ld",person[i].money);

}

for(int i=0;i<3;i+=1)

{

printf("%s is %u years old has %ld rupees\n",person[i].name,person[i].age,person[i].money);

}

/\*struct Person person1={"Manish",19,-10,100};

struct Person person2={"Manish",20,2};

struct Person person3={"Manish",21,3};

printf("%s is %u years old has %lu rupees and weight %u\n",person1.name,person1.age,person1.money,person1.weight);

printf("%s is %u years old has %lu rupees\n",person2.name,person2.age,person2.money);\*/

return 0;

}

/\* size of the person classs\*/

#include <stdio.h>

int main()

{

typedef struct

{

char name[30];

unsigned int;

long money;

unsigned short height;

unsigned short weight;

}

Person;

Person persons[10];

printf("size of char is %lu bytes\n",sizeof(char));

printf("size of int is %lu bytes\n",sizeof(int));

printf("size of short is %lu bytes\n",sizeof(short));

printf("size of long is %lu bytes\n",sizeof(long));

printf("size of Person of struc is %lu bytes\n",sizeof(persons));

return 0;

}

**ESCAPE SEQUENCE:**

\n NEXT LINE

\b \t SAME BUT NO GAP AMD GAP RESPECTIVELY

\WHATEVER GETS PRINTED EX \"

Declaration statement always comes before action statemen

FORMATE SPECIFIER:

%d int , %f float, %c char, %lf double

CFJ Terminal Commands and Basics of C

### **Useful Timestamps**

Basic Overview - 0:00

Shell commands - 1:50

Hello World in C - 12:33

Compilation and execution in C - 19:35

Primitive Datatypes in C - 24:48

Loops and Conditionals - 32:07

### **Quick Notes**

#### **Shell commands**

Every shell command is of the form: program option(s) argument(s).

In some commands, option(s) and/or argument(s) are absent.

Every option is preceded by a hyphen symbol “-”.

Directory - to crudely put it, a directory is a folder in your computer.

1. **pwd** - stands for *print working directory*. Will print the path of the current directory with respect to the root directory. For me, the root directory was *rootfs*, and thus my path was *home/prashanthi*.
2. **ls** - stands for *list*. Will print a list of files and directories present in the current working directory.
3. **ls -l** - This is the list command with an added option. The option, -l stands for *long* format. Will print a list of files and directories present in the current working directory, but in a long format.
4. **-cd** - stands for *change directory*. Will help you move in and out of directories and change your current working directory.

E.g. if Blockchain is a directory in my current working directory,

*cd Blockchain* will change my current directory to Blockchain. To go back to the previous directory - *cd ..* (basically, cd dot dot).

If SastaICP is a directory within Blockchain, this is how I can cd into it from the home directory: *cd Blockchain/SastaICP*

E.g.: *cd ../../* will take me back to the home directory from SastaICP

1. **mkdir** - stands for *make directory*. Will help you create directories**.**

E.g.: *mkdir SastaICP* will create a new directory inside the current working directory.

1. **clear**: will *clear* your console. Very satisfying, no?
2. **rmdir**: stands for *remove directory* (surprise, surprise). Will delete a directory if it is empty.

E.g.: *rmdir SastaICP* will remove/delete the directory SastaICP if it is empty.

1. **rm -r** or **rm - rf**  - program (rm) followed by options (-r or -rf). rm stands for remove. It can be used to remove files and directories.

E.g.: *rm -r(f) SastaICP* will remove the directory SastaICP regardless of whether it is empty or not.

1. **touch** - create files in your current working directory

E.g.: touch hello.c

touch test.txt

touch test.py

1. **nano** - edit files on your terminal

E.g.: nano hello.c

Some other useful basic shell commands here:

<https://www.geeksforgeeks.org/basic-shell-commands-in-linux/>

#### **C vs Python**

#### More like why is C still relevant with all its disadvantages? (Hint: high performance, among other things)

|  |  |
| --- | --- |
| C | Python |
| Low-level language  (closer to the computer language\* than human language) | High-level language  (closer to natural languages than computer language) |
| You need to declare variables with their types. | No need to declare variables. |
| It is a compiled language  (it run in two steps: compilation and execution) | It is an interpreted language  (It executes code line by line and terminates when it finds an error; no extra compilation step) |
| Readability: very low for people not familiar with the syntax. | Very readable, accessible, user-friendly, short learning curve. |
| High performance; consumes less energy and is very, very fast (since it’s easily understood by the computer). | Poor performance; consumes a lot of energy and is very slow (not easily understood by your computer; has to be broken down into a simpler language for the computer to understand). |
| Preferred for complex computations and hardware-related applications.  (computations that involve large numbers and complex operations, e.g.: cryptography) | Not preferred for complex computations because of its poor performance.  (preferred for machine learning etc. because of its ease of use and abstraction+it’s many useful libraries) |
| Limited number of libraries available. You may have to implement most of the functionalities yourself sometimes. | A large repository of libraries available, a lot of data structures; basically makes life easy for programmers. |
| You need to be very specific with regards to datatypes of variables. | Not very stringent when it comes to datatypes of variables. |

\* By computer language, I mean 0s and 1s (binary).

Conclusion: As you may have guessed, both are important and very relevant in different areas of Computer Science and Tech! Your choice of language depends on what you want to do.

#### **Hello World in C**

.c is the extension for C codes, just as .py is for Python.

Code attached: testhello.c

<https://www.programiz.com/c-programming/examples/print-sentence>

Some resources to understand stdio.h and other headerfiles:

<https://fresh2refresh.com/c-programming/c-function/stdio-h-library-functions/>

<https://www.tutorialspoint.com/cprogramming/c_header_files.htm>

<https://www.tutorialspoint.com/standard-header-files-in-c>

#### **Compilation and Execution**

The only things you need to know about the compilation process:

1. The C compiler spots the syntax or semantic errors that you might have made in your code.
2. The C compiler also allocates space in the memory for the variables you have created in your code if there are no errors.
3. Your code can run into errors during execution itself even if there are no syntax errors in your code. Such errors are called run time errors.

Think of an executable this way: You want to share your program with A, let’s say a computer game. But you don’t want to share your code with them, because you fear they might misuse your code. You can just share your code’s executable with A. A won’t be able to retract your original code (unless, of course, they know how your game works and they write their own code for your game).

Compile your C code hello.c -

1. gcc hello.c // in this case, your executable is stored in the default executable a.out
2. gcc hello.c -o hello // here, you are naming your executable. It won’t be stored in a.out. Now, your executable is hello.
3. gcc -o hello hello.c //another way to do the above command. They mean the same things

[Note: In case you get a “permission denied” error, use “sudo” before gcc in the above commands.]

Run/Execute your C code hello.c -

1. If you used method 1 above to compile, run your code as:

./a.out

1. If you used methods 2 or 3 above to compile, run your code as:

./hello

#### **Primitive Datatypes in C**

Int, char, float, double.

<https://www.geeksforgeeks.org/data-types-in-c/>

Be very careful about datatypes of variables. C is very strict about types.

Code attached: basic.c

#### **Input and Output in C**

stdio.h has two important functions among others:

printf: a function to print to the console

scanf: a function that is used to take input from the user.

<https://www.programiz.com/c-programming/c-input-output>

<https://codeforwin.org/2015/05/list-of-all-format-specifiers-in-c-programming.html>

#### **Loops and Conditionals**

Code attached: loops.c

Loops:

Very beautifully explained here:

For Loops: <https://www.programiz.com/c-programming/c-for-loop>

Do while and while loops: <https://www.programiz.com/c-programming/c-do-while-loops>

Conditionals:

If-else in C: <https://www.programiz.com/c-programming/c-if-else-statement>

Read about other stuff (break, continue, switch statements) under C - Flow Control section here: <https://www.programiz.com/c-programming>

I did not cover **makefiles** in this class, but that is something important to know:

<https://opensource.com/article/18/8/what-how-makefile>

<https://dev.to/nerfur/easy-make-and-makefile-for-beginners-476a>

<https://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/>

### **Some other important things to know**

(Feel free to contact me with questions regarding this section)

#### **Operators: Unary, Binary and Ternary**

<https://tutorials.webencyclop.com/c-language/c-operators/>

#### **Functions in C**

<https://www.programiz.com/c-programming/c-functions>

### **Some corrections in the recording**

11:38: I meant “not empty” instead of “empty”

18:23: scanf is an inbuilt function present in the headerfile stdio.h function to take input from the user - I realise I said “keyword”.

18:30: Yes, I meant percent (%), and not dollar ($), jeez.

44:38: My enthu south Indian parents and guests :)

### 

***Contact Prashanthi (***[***prashanthi.r@alumni.edu.in***](mailto:prashanthi.r@alumni.edu.in)***; +91 9500662485) or any other instructor, should you face any difficulties with these questions. Please try to utilise the doubts clarification sessions (every Wednesday, 6:30-7:30 PM) for getting your doubts clarified as much as possible.***

**Lecture - 1: Terminal Commands and Basics of C**

[Source: <https://www.w3resource.com/c-programming-exercises/>. ]

/\*\*\* FACTORIAL \*\*\*/

#include <stdio.h>

int main()

{

printf("\n"); /\*to go to the next line\*/

int x; /\*initializing variables which can take integer values\*/

printf("NUMBER WHOSE FACTORIAL YOU WANT TO FIND:");

scanf("%d", &x);

int i;

for(i=x-1; i>0;i=i-1)

{

int y=x\*(x-1);

return y;

x=x-1;

}

printf("FACTORIAL OF %d IS %d ",x,y);

return 0;

}

/\*\*\* [Loops] SUM OF N NATURAL NUMBERS \*\*\*/

#include <stdio.h>

int main()

{

int a;

scanf("%d",&a);

printf(" LIST OF PRECEDING NATURAL NUMBERS ARE:");

int A[]={};

for(int i=a; i>0; i=i-1)

{

printf("%d,",i);

int y=i+(i-1);

}

return 0;

}

/\*\*\* [Loops] SUM OF N NATURAL NUMBERS \*\*\*/

#include <stdio.h>

int main()

{

int a;

scanf("%d",&a);

printf(" LIST OF PRECEDING NATURAL NUMBERS ARE:");

int A[]={};

for(int i=a; i>0; i=i-1)

{

printf("%d,",i);

int y=i+(i-1);

}

return y;

printf("%d",y);

return 0;

}

/\*\*\* [If-else Conditional statements]

Write a C program to read the age of a candidate and determine whether it is eligible for casting his/her own vote (using if-else statements).

Write a C program to find the largest of three numbers. (take the three numbers as input from the user)

/\*\*\* DEALING WITH STRING \*\*\*/

#include<stdio.h>

#include<string.h>

char main()

{

int a,b,c;

printf("ENTER THREE NUMBERS");

scanf("%d%d%d",a,b,c);

if (a>b or )

return 0;

}

**Lecture - 2: Signs, Strings, Structures & Arrays**

[Strings]

1. Write a program in C to separate the individual characters from a string.

Sample Output:

Input the string : hello

The characters of the string are : h e l l o

Take a look at the functions that string.h comes with here before attempting questions 2 through: <https://www.programiz.com/c-programming/string-handling-functions>

(Also, look up a function called **strrev()** in Con Google that is not mentioned here^)

1. Now, use a function from string.h to reverse an inputted string.

Sample Output:

Input the string : hello there

The reversed string is : ereht olleh

1. Write a program to check if the string taken as input from the user is a [palindrome](https://examples.yourdictionary.com/palindrome-examples.html).

[Hint: Try to use strrev and strcmp from string.h for this question]

Sample Output 1:

Input the string : madam

Yes, the word “madam” is a palindrome.

Sample Output 2:

Input the string : world

No, the word “world” is not a palindrome.

1. Write a simple program to convert any given input string to uppercase.

Sample Output:

Input the string : hello there!

The reversed string is : HELLO THERE!

[Arrays and Manipulation]

1. Write a program in C to store elements in an array and print it.

Sample Output:

Input 10 elements in the array :

element - 0 : 1

element - 1 : 2

element - 2 : 4

*Expected Output* :

Elements in array are: 1 2 4

1. Write a program in C to find the sum of all elements of the array.

Sample Output:

Input the number of elements to be stored in the array :3

Input 3 elements in the array :

element - 0 : 2

element - 1 : 5

element - 2 : 8

Sum of all elements stored in the array is : 15

1. Write a program in C to count a total number of duplicate elements in an array.

Sample Output:

Input the number of elements to be stored in the array :3

Input 3 elements in the array :

element - 0 : 5

element - 1 : 1

element - 2 : 1

*Expected Output* :

Total number of duplicate elements found in the array is : 1

1. Write a program in C to read n number of values in an array and display it in reverse order.

Sample Output:

Input the number of elements to store in the array :3

Input 3 number of elements in the array :

element - 0 : 2

element - 1 : 5

element - 2 : 7

The values store into the array are :

2 5 7

The values store into the array in reverse are :

7 5 2

[Structures]

1. Write a program to store and print the roll number, name, age, and marks of a student using structures.

1. Write a program to add, subtract and multiply two complex numbers using structures to function.

[Hint: Think of your complex number in the form: a+bi where a and b are float values.]