Why Did You Choose CSE?

...

Introduce With

Structural Programming Language

•••

Course Title :- Structured Programming Language

Course Code :- CSE- 121 & 122

Level Term: 1-II-B(G3, G4)

Khandaker Jannatul Ritu, Lecturer(CSE), BAIUST

Content Of This Lecture

1. General Overview of a Simple C Program's Structure:
2. Compilation process in C
3. Differences Between C, C++ and Java
4. Tokens in C
5. C Comments
6. Escape Sequence in C
7. C Variables
8. C Identifiers
9. C Variable Types
10. C Variable Values: Copying Variables
11. Constants & Literals in C : Unchangeable Variable
12. Data Types
13. Size of Data Types in C
14. Integer Data Type Memory Distribution
15. Data Type[Integer, Float, Double, Characters]
16. float vs. double
17. C Decimal Precision
18. Scientific Numbers
19. Basic Input and Output in C
20. I/O Functions
21. ASCII characters & programs
22. Boolean in C
23. Type Conversion in C
24. Online Judge Problems Based On this Lecture!
25. Exercise: Solve this Questions!

Resources will be followed

❑ Books:

1. C: the complete reference
2. Programming in ANSI-C
3. C programming by tamim shahriar subin
4. 52 Programming Problem Tamim Shahriar Subeen

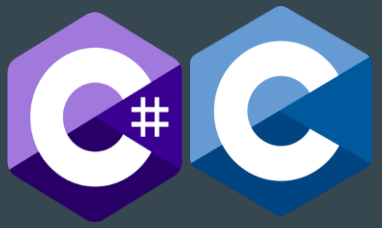
❑ Youtube Tutorial:

C Programming Bangla Tutorials (সবার জন্য সি প্রোগ্রামিং) | Updated in 2023 - YouTube

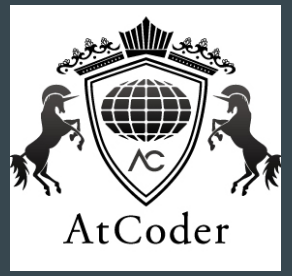
❑ Online Judges:

1. https://codeforces.com/group/MWSDmqGsZm/contests
2. https://judge.beecrowd.com/en/problems/index/1

Popular Programming languages

Competitive Programming OJ

Necessary Websites & Tools Related To Programming

GeeksforGeeks

javaTpoint

programiz

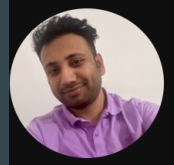
freeCodeCamp

শাফায়েতের ব্লগ
প্রোগ্রামিং, অ্যালগরিদম, ব্যাকএন্ড ইঞ্জিনিয়ারিং

tutorialspoint
SIMPLY EASY LEARNING

w3schools

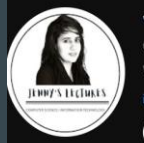
USACO



Anisul Islam
@anisul-islam · 462K subscribers · 2.4K videos
There will be a day when no student will fail just
paypal.me/anisul2022 and 4 more links
Subscribe

Geeky Shows
@geekyshows · 516K subscribers · 3.8K videos
You can learn various Programming Languages,
geekyshows.com and 5 more links
Subscribe

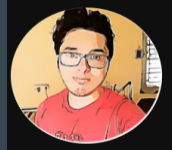
Apna College
@ApnaCollegeOfficial · 5.15M subscribers · 820 videos
I'm Shradha, Ex-Microsoft Software Engineer, DRDO. My
linktr.ee/apnacollege.in and 2 more links
Subscribe



Jenny's Lectures CS IT
@JennyslecturesCSIT · 1.57M subscribers · 718 videos
Warm Welcome to Jenny's Lectures. On This Channel we are foc
instagram.com/jayantikhatrilamba and 3 more links
Subscribe

Gate Smashers
@GateSmashers · 1.86M subscribers · 1.5K videos
Gate Smashers is India's no 1 Free education platforr
facebook.com/gatesmashers and 1 more link
Subscribe

Abdul Bari
@abdul_bari · 946K subscribers · 115 videos
I have started this channel to help Students Commu
facebook.com/mohammed.bari1 and 2 more links
Subscribe



CodeWithHarry
@CodeWithHarry · 5.9M subscribers · 2.3K videos
Code With Harry is my attempt to teach basics and
codewithharry.com and 2 more links
Subscribe

take U forward
@takeUforward · 544K subscribers · 502 videos
Hey, I am a Software Engineer. Where? You can fi
takeuforward.org and 4 more links
Subscribe

GitHub

Programiz
C Online Compiler

Google Scholar

Gmail

RG

STALK

stackoverflow

ideone.com
Your great ideas will be born here

Quora

LinkedIn

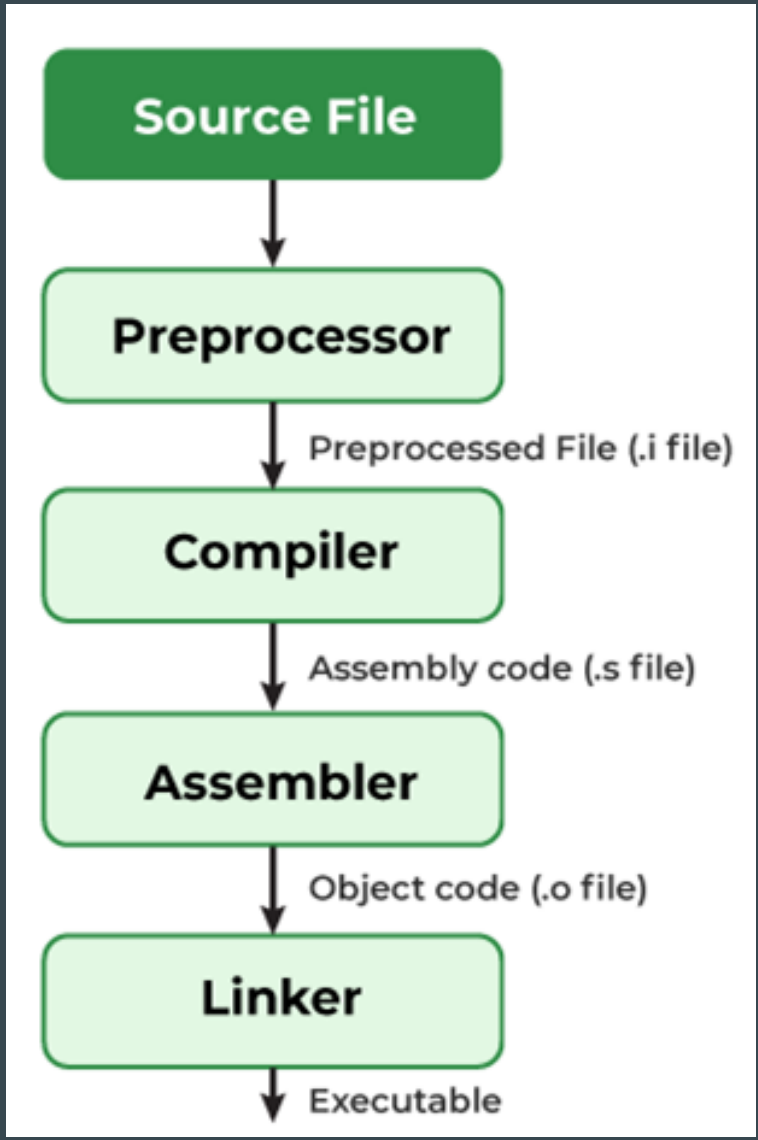
General Overview of a Simple C Program's Structure:

- Header Files
- Main Function
- Variable Declarations
- Statements and Expressions
- Comments
- Return Statement
- Functions
- Standard Input/Output

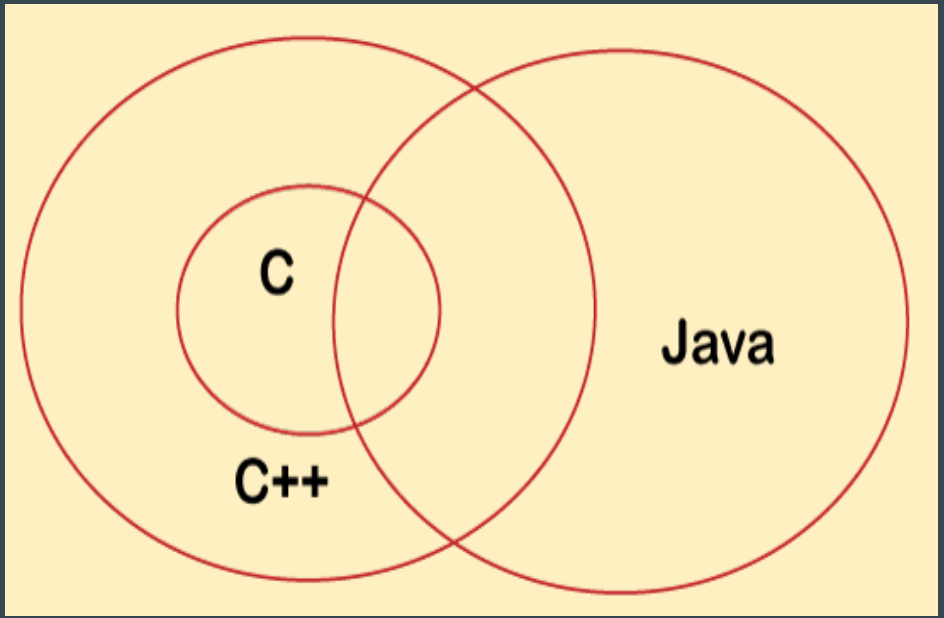
```
#include <stdio.h>
int main()
{
        //My First C Program
        printf("Hello Student's!!!");

        return 0;
}
```

❑ Compilation process in C



❑ Differences Between C, C++ and Java



❑ Tokens in C

A token in C is the **meaningful smallest unit** used in a C program.

❑ Keywords in C

Keywords are predefined or reserved words

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

❑ C Comments

Single-line Comment in C

Syntax of Single Line C Comment
// This is a single line comment

2. Multi-line Comment in C
Syntax of Multi-Line C Comment

```
/*Comment starts
continues
continues
.
.
.
Comment ends*/
```

❑ C Output (Print Text)

```
#include<stdio.h>
int main()
{

   printf("Hello Friends!");
   printf("I am Learning C");
   printf("And It is not awesome!");
}
```

output:

```
Hello Friends!I am Learning CAnd It is not awesome!
```

❑ Escape Sequence in C

Escape Sequence	Name	Description
\n	New Line	It moves the cursor to the start of the next line.
\t	Horizontal Tab	It inserts some whitespace to the left of the cursor and moves the cursor accordingly.
\\	Backlash	Use to insert backslash character.
\’	Single Quote	It is used to display a single quotation mark.
\”	Double Quote	It is used to display double quotation marks.
\?	Question Mark	It is used to display a question mark.

- An escape sequence contains a backslash (\) symbol followed by one of the escape sequence characters or an octal or hexadecimal number.

❑ Escape Sequence in C

C program to illustrate \n escape sequence

```
#include <stdio.h>
int main(void)
{
    printf("Hello\n");
    printf("GeeksforGeeks");
}
```

Output:

```
Hello
GeeksforGeeks
```

C program to illustrate \t escape sequence

```
#include <stdio.h>
int main(void){
    printf("Hello \t GFG");
}
```

Output:

```
Hello      GFG
```

Example to demonstrate how to use \\ escape sequence in C

```
#include <stdio.h>
int main(void){
    printf("Hello\\GFG");
}
```

Output

```
Hello\GFG
```

C program to illustrate \? escape sequence

```
#include <stdio.h>
int main(void){
    printf("\?\?!\n");
}
```

Output

```
??!
```

Example to demonstrate how to use \’ and \” escape sequence in C

```
#include <stdio.h>
int main(void){
    printf("\' Hello Geeks\n");
    printf("\" Hello Geeks");
    return 0;
}
```

Output

```
' Hello Geeks
" Hello Geeks
```

Introduce With

Variables, Constants, Literals

...

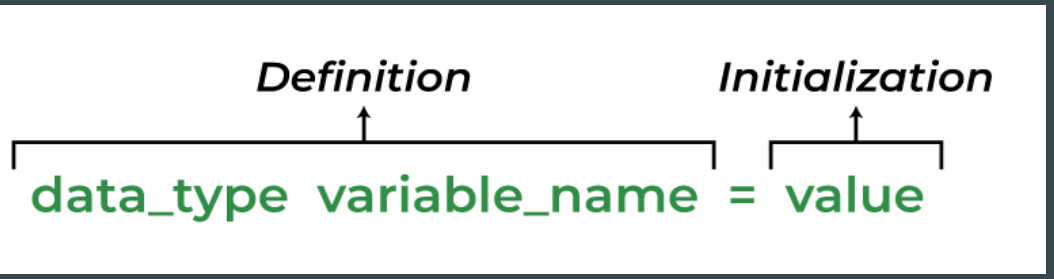
❑ C Variables

Variables are containers for storing data values, like numbers and characters.

In C, there are different types of variables (defined with different keywords), for example:

- **int -** stores integers (whole numbers), without decimals, such as 123 or -123
- **float / double -** stores floating point numbers, with decimals, such as 19.99 or -19.99
- **char/strings -** stores single characters, such as 'a' or 'B'. Characters are surrounded by single quotes

The syntax to declare a variable:




```
int main()
{
   int age = 45;
   float height = 5.5;
   double weight = 56.3;
   char genaration='z';
   char name[50]="programming";


   printf("%d\n", age);
   printf("%f\n", height);
   printf("%lf\n", weight);
   printf("%c\n", genaration);
   printf("%s\n", name);

}
```

Output:

```
45
5.500000
56.300000
z
programming
```

❑ C Identifiers

Identifiers are unique names that are assigned to variables, structs, functions, and other entities.

Rules to Name an Identifier in C

1. An identifier can include letters (a-z or A-Z), and digits (0-9).
2. An identifier cannot include special characters(/,<,>,*,!,~,#,@,$,%,^,&,*) except the '_' underscore.
3. Spaces are not allowed while naming an identifier.
4. An identifier can only begin with an underscore or letters.
5. We cannot name identifiers the same as keywords because they are reserved words to perform a specific task. For example, printf, scanf, int, char, struct, etc.
6. The identifier must be **unique** in its namespace.
7. C language is case-sensitive so, 'name' and 'NAME' are different identifiers.

✅ Number, num=2, val, VAL=23, number1,number2, a1,a2,

❌ 1num, #val, ^list, $name, ~age, &age

✅ _num1, num1_, customer_list, customer_list1

❌ 1num_, customer-list, 1_num,

✅ customer_list

❌ customer list

✅ INT, PRINTF, CHAR

❌ int = 0, printf=0,

✅ name=0, NAME=0, Name=0, NaMe=0, NAMe=0,namE=0

❑ C Variable Types

1. Local Variable
2. Global Variable
3. Static Variable

```
#include<stdio.h>

float cp =10.5; //global variable

static  int age = 34; //static global variable

int main()
{

   float cp = 34; //local variable
   static int myVariable=12;   // static variable

}
```

Q1. If Local & global variable is same, then which variable will be executed?
Q2. If Local & global variable is same, then how to print the global variable in c?
Q3. List Some other variables and learn how they work.

C Variable Values: Copying Variables

- Change Variable Values.
- If you assign a new value to an existing variable,
- it will **overwrite** the previous value

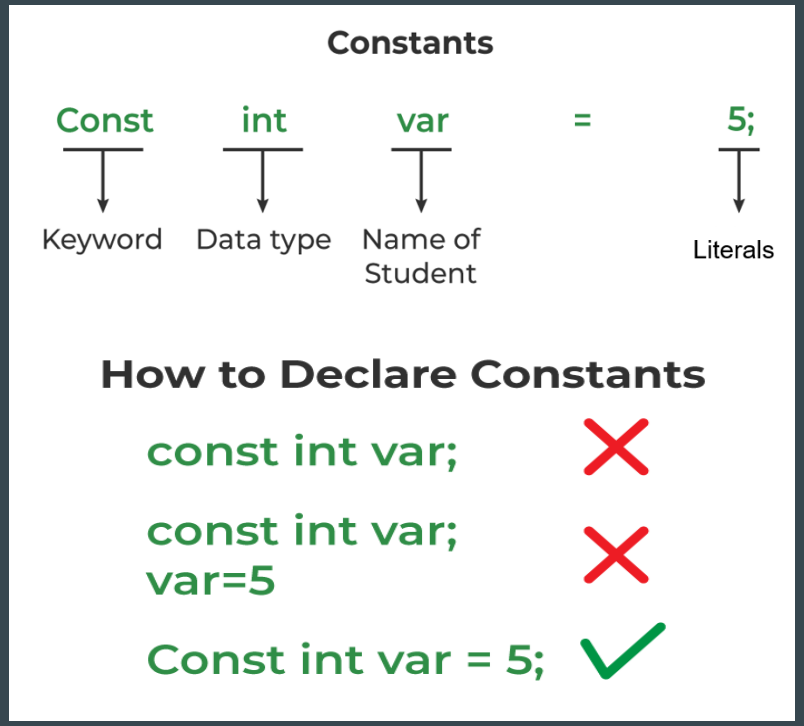
```
int main(){
    int id1, id2, id3;
    int a,b,c,d;
    ///type=1
    id1 = 5;
    id2 = 102;
    id3 = id2;

    id2 = id1;
    id1 = id2;
    printf("id1 = %d\n", id1);
    printf("id2 = %d\n", id2);
    printf("id3 = %d\n\n", id3);

    ///type=2
    a = 61;
    a = 23;
    printf("a = %d\n\n", a);
    a = 78;
    a = 32;
    printf("a = %d\n\n", a);
}
```

❑ Constants & Literals in C : Unchangeable Variable

- If you don't want others (or yourself) to change existing variable values, you can use the const keyword.




```
#include <stdio.h>
int main() {
  const int myNum = 15; // myNum will always be 15
  myNum = 10; // error: assignment of read-only variable 'myNum'
  printf("%d", myNum);
  return 0;
}
prog.c: In function 'main':
prog.c:5:18: error: assignment of read-only variable 'myNum'
    5 |   myNum = 10;
      |         ^
```

Properties of Constant in C

1. Initialization with Declaration
We can only initialize the constant variable in C at the time of its declaration. Otherwise, it will store the garbage value.

2. Immutability
The constant variables in c are immutable after its definition, i.e., they can be initialized only once in the whole program. After that, we cannot modify the value stored inside that variable.

- Defining Constant using #define

#define const_name value

```
#include<stdio.h>
#define month 12
#define PI 3.1416

const int ar=15;
const float age=25.5;
const double d;

int main(){
    const int ar=5;
    //ar = 6; // cannot to change

    printf("constant variable ar : %d\n\n", ar);
    printf("constant variable age : %f\n\n", age);

    scanf("%lf", &d);
    printf("constant variable d : %lf\n\n", d);
    //d =45;// cannot to change

    printf("month : %d\n", month);
    printf("value of pi : %f\n", PI);
}
```

Data Types In ‘C’

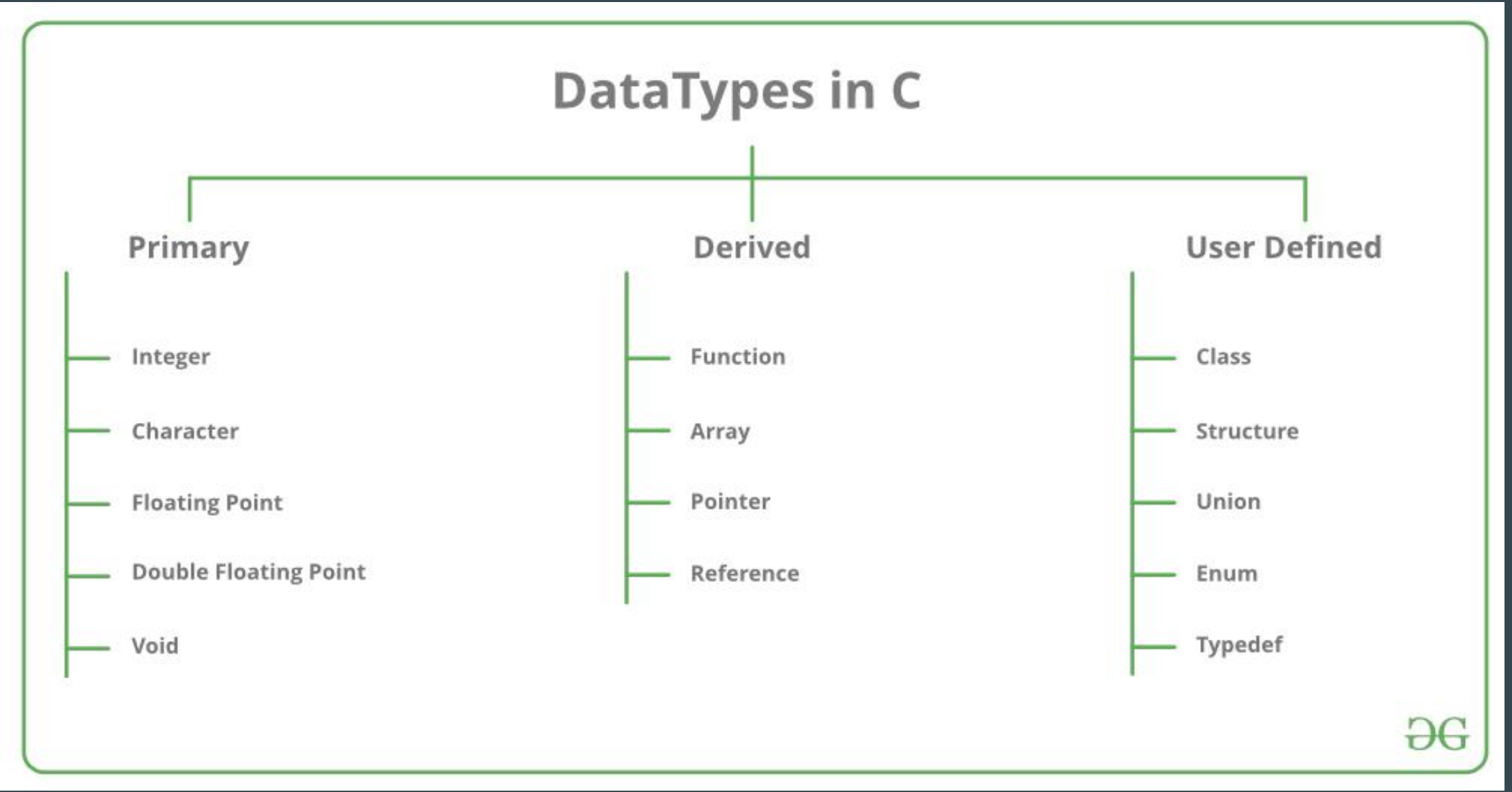
Data Types In ‘C’

Data Types In ‘C’

...

Khandaker Jannatul Ritu, Lecturer(CSE), BAIUST

❑ Data Types


DataTypes in C
Primary
Integer
Character
Floating Point
Double Floating Point
Void
Derived
Function
Array
Pointer
Reference
User Defined
Class
Structure
Union
Enum
Typedef

Data Type	Size (bytes)	Range	Format Specifier
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4	1.2E-38 to 3.4E+38	%f
double	8	1.7E-308 to 1.7E+308	%lf
long double	16	3.4E-4932 to 1.1E+4932	%Lf

❑ Size of Data Types in C

```
int main()
{
   int age = 45;
   float height = 5.5;
   double weight = 56.3;
   char genaration='z';
   char name[50]="programming";

   printf("%d\n", sizeof(age));
   printf("%d\n", sizeof(height));
   printf("%d\n", sizeof(weight));
   printf("%d\n", sizeof(genaration));
   printf("%d\n", sizeof(name));
}
```

Output:
4
4
8
1
50

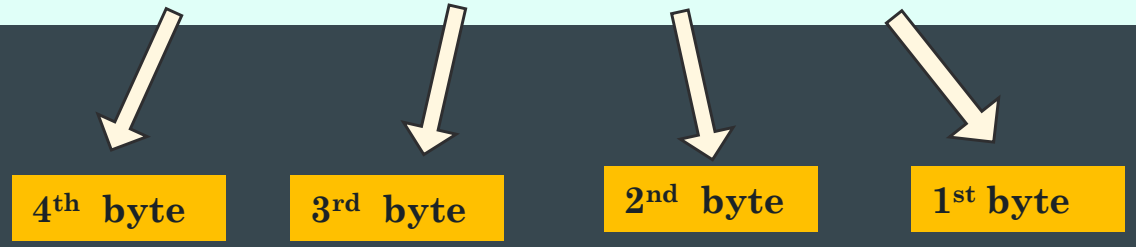
```
int main()
{
   int student[100];
   printf("%d", sizeof(student));
}
```

Output: ??

❑ Integer Data Type Memory Distribution

Integer size 4bytes
1 byte = 8bit
4 byte = 8 * 4 bit = 32bit

If n=3
Memory will be store it as:- 00000000 00000000 00000000 00000011



❑ Data Type[Integer, Float, Double, Characters]

Q1.

```
int main(){
    int age = 45;
    float height = 5.5;
    double weight = 56.3;
    char genaration='z';
    char name[50]="programming";

    printf("%d\n", age);
    printf("%f\n", height);
    printf("%lf\n", weight);
    printf("%c\n", genaration);
    printf("%s\n", name);
}
```

Output:

```
45
5.500000
56.300000
z
programming
```

Q2.

```
#include<stdio.h>
int main()
{
    float value1=34;
    printf("%f", value1);
}
```

Output: ???

Q3.

```
#include<stdio.h>
int main()
{
    int value1=34.555454;
    printf("%d", value1);
}
```

Output: ???

❑ float vs. double

The **precision** of a floating point value indicates how many digits the value can have after the decimal point. The precision of float is six or seven decimal digits, while double variables have a precision of about 15 digits. Therefore, it is often safer to use double for most calculations - but note that it takes up twice as much memory as float (8 bytes vs. 4 bytes).

❑ Scientific Numbers

A floating point number can also be a scientific number with an "e" to indicate the power of 10:

```
#include <stdio.h>
int main()
{
  float f1 = 35e3;
  printf("%f\n", f1);
}
Output: ?????
```

❑ C Decimal Precision

If you want to remove the extra zeros (set decimal precision), you can use a dot (.) followed by a number that specifies how many digits that should be shown after the decimal point:

```
#include <stdio.h>
int main()
{
  float myFloatNum = 3.5;
  printf(" %f   \n", myFloatNum); // Default will show 6 digits
  printf(" %.1f \n", myFloatNum); // Only show 1 digit
  printf(" %.2f \n", myFloatNum); // Only show 2 digits
  printf(" %.4f  ", myFloatNum);   // Only show 4 digits
}
```

Output:

```
3.500000
3.5
3.50
3.5000
```

C Input/Output

C Input/Output

...

C Input/Output

Khandaker Jannatul Ritu, Lecturer(CSE), BAIUST

❑ Basic Input and Output in C

scanf()

Syntax:
scanf("%X", &variable);

printf()

Syntax:
printf("%X", variable);

Integer:
Input: scanf("%d", &intVariable);
Output: printf("%d", intVariable);

Float:
Input: scanf("%f", &floatVariable);
Output: printf("%f", floatVariable);

Double:
Input: scanf("%lf", &doubleVariable);
Output: printf("%lf", doubleVariable);

Character:
Input: scanf("%c", &charVariable);
Output: printf("%c", charVariable);

String word:
Input: scanf("%s", &stringVariable);
Output: printf("%s", stringVariable);

String sentence:
Input: scanf("%[^\n]s", &stringVariable);
Output: printf("%s", stringVariable);

Q1. input and output of Integer, Float, Double

```
int main(){
    int num;  float f; double d;
    printf("Enter the integer: ");
    scanf("%d", &num);
    printf("\nEntered integer is: %d", num);
    printf("\n\nEnter the float: ");
    scanf("%f", &f);
    printf("\nEntered float is: %f", f);
    printf("\n\nEnter the double: ");
    scanf("%lf", &d);
    printf("\nEntered float is: %lf", d);
}
```

Output:

```
Enter the integer: 45
Entered integer is: 45
Enter the float: 4.5
Entered float is: 4.500000
Enter the double: 4.03
Entered float is: 4.030000
```

Q2. input and output of a character

```
#include <stdio.h>
int main(){
    char ch;
    printf("\nEnter the Character: ");
    scanf("%c", &ch);
    printf("\nEntered character is: %c\n\n", ch);
}
```

Output:

```
Enter the Character: ritu
Entered character is: r
```

Q3. input and output of a word

```
#include <stdio.h>
int main(){
    char str[50];
    printf("Enter the Word: ");
    scanf("%s", str);
    printf("\nEntered Word is: %s", str);
}
```

Output:

```
Enter the Word: computer science
Entered Word is: computer
```

Q4. input and output of a sentence

```
#include <stdio.h>
int main(){
    char str[50];
    printf("\nEnter the Sentence: ");
    scanf("%[^\n]s", str);
    printf("\nEntered Sentence is: %s\n\n", str);
}
```

Output:

```
Enter the Sentence: we are learning c-programming language
Entered Sentence is: we are learning c-programming language
```

Formatted I/O Functions

- use format specifiers

- **printf()**

1. **scanf()**
2. **sprintf()**
3. **sscanf()**

```
Q1.
int main()
{
  int num1;
  printf("Enter a integer number: ");
  scanf("%d", &num1);
  printf("You have entered %d", num1);
}
```

Unformatted I/O Functions

- cannot use format specifiers

- getch()
- getche()
- getchar()
- putchar()
- gets()
- puts()
- putch()

```
Q2.
#include <conio.h>
#include <stdio.h>
int main() {
  char name[50];
  puts("Please enter some texts: ");
  gets(name);
  puts(name);
  getch();
}
```

❑ ASCII characters:

1	▯	17	▯	33	!	49	1	65	A	81	Q	97	a	113	q
2		18	▯	34	"	50	2	66	B	82	R	98	b	114	r
3	▯	19	▯	35	#	51	3	67	C	83	S	99	c	115	s
4	▯	20	▯	36	$	52	4	68	D	84	T	100	d	116	t
5	▯	21	▯	37	%	53	5	69	E	85	U	101	e	117	u
6	▯	22	▯	38	&	54	6	70	F	86	V	102	f	118	v
7	▯	23	▯	39	'	55	7	71	G	87	W	103	g	119	w
8	▯	24	▯	40	(	56	8	72	H	88	X	104	h	120	x
9		25	▯	41	)	57	9	73	I	89	Y	105	i	121	y
10		26	▯	42	*	58	:	74	J	90	Z	106	j	122	z
11	▯	27	▯	43	+	59	;	75	K	91	[	107	k	123	{
12	▣	28		44	,	60	<	76	L	92	\	108	l	124	|
13		29		45	-	61	=	77	M	93	]	109	m	125	}
14	▯	30		46	.	62	>	78	N	94	^	110	n	126	~
15	▯	31		47	/	63	?	79	O	95	_	111	o	127	▯
16	▯	32		48	0	64	@	80	P	96	`	112	p	128	€

1. C Program to convert Character → ASCII → Character
2. C Program to Covert Lowercase → Uppercase → Lowercase

Q1. Character → ASCII → Character

```
#include<stdio.h>
int main(){
   /// character --> ascii value
   char ch;
   printf(" Enter a ascii characters: ");
   scanf("%c", &ch);
   printf(" corresponding value : %d\n\n", ch);
   ///ascii value --> character
   int a;
   printf(" Enter a value: ");
   scanf("%d", &a);
   printf(" corresponding ascii character : %c\n", a);

}
```

Output:

```
 Enter a ascii characters: T
 corresponding value : 84

 Enter a value: 83
 corresponding ascii character : S
```

Q2.Lowercase → Uppercase → Lowercase

```
#include<stdio.h>
int main(){
    ///lower to upper
    char lower;
    printf("Enter a letter between (a-z): ");
    scanf("%c", &lower);
    printf("uppercase method-1 : %c\n", (lower - 32));
    printf("uppercase method-2 : %c\n\n", toupper(lower));
    getchar();
    ///lower to upper
    char upper;
    printf("Enter a letter between (A-Z): ");
    scanf("%c", &upper);
    printf("uppercase method-1 : %c\n", (upper + 32));
    printf("uppercase method-2 : %c\n\n", tolower(upper));
}
```

Output:

```
Enter a letter between (a-z): a
uppercase method-1 : A
uppercase method-2 : A

Enter a letter between (A-Z): R
uppercase method-1 : r
uppercase method-2 : r
```

❑ Boolean in C

Syntax: **bool** variable_name;

```
#include <stdbool.h>  // Import the boolean header file
```

Q1.

```
#include <stdio.h>
int main() {
  printf("%d\n", 10 == 10);
  printf("%d\n", 10 == 15);
  printf("%d", 5 == 55);
}
```

Output:

1
0
0

Q2.

```
#include <stdio.h>
int main() {
  int x = 10;
  int y = 9;
  printf("%d", x > y);
 printf("%d", 10 > 9);
}
```

Output:

1 1

Q3.

```
#include <stdio.h>
#include <stdbool.h>
int main()
{
  bool isHamburgerTasty = true;
  bool isPizzaTasty = true;
  printf("%d", isHamburgerTasty == isPizzaTasty);
}
```

Output: 1

Q4.

```
#include <stdio.h>
int main()
{
  int myAge = 25;
  int votingAge = 18;
  printf("%d", myAge >= votingAge);
}
```

Output:

1

Q5.

```
#include <stdio.h>
#include <stdbool.h>
int main() {
  bool isProgrammingFun = true;
  bool isFishTasty = false;
  printf("%d ", isProgrammingFun);
  printf("%d", isFishTasty);
}
```

Output: 1 0

Solve this!!

Q6. Write the output for the following code:

```
int main ()
{
int x = 10;
int y = 9;
bool a = false, b = false;
printf("%d %d %d %d %d %d\n", x > y, x >= y, 13>=13, a==b, a>=b, a<=b );
printf("%d %d %d %d\n", 10 != 10, 10 == 15, x != y, x <= y);
}
```

❑ Type Conversion in C

Priority of Data Type:

bool → char → int → long long → float → double → long double

❑ **Implicit Type Conversion:** the value of one type is automatically converted to the value of another type.

Q1.
```
#include<stdio.h>
int main() {
  double value = 4150.12;
   printf("Double Value: %.2lf\n", value);
   int number = value;
  printf("Integer Value: %d", number);
}
```
Output:
Double Value: 4150.12
Integer Value: 4150

Q3.
```
#include<stdio.h>
int main() {
  char alphabet = 'a';
  printf("Character Value: %c\n", alphabet);
  int number = alphabet;
  printf("Integer Value: %d", number);
}
```
Output:
Character Value: a
Integer Value: 97

Q5.
```
#include <stdio.h>
int main(){
    int x = 10;
    char y = 'a';
    x = x + y;
    float z = x + 1.0;
    printf("x = %d, z = %f", x, z);
}
```
Output
`x = 107, z = 108.000000`

Q2.
```
#include <stdio.h>
int main() {
  int x = 5;
  int y = 2;
  int sum = 5 / 2;
  printf("%d", sum);
}
```
output: 2

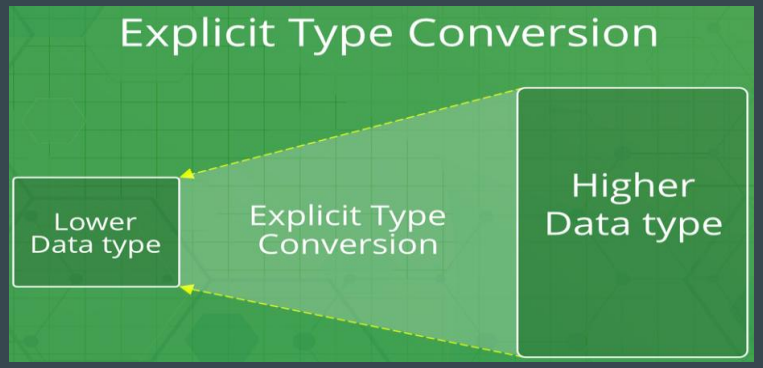
Q4.
```
#include <stdio.h>
int main() {
float myFloat = 9;
printf("%f", myFloat);
}
```

Output: 9.000000

Q6.
```
#include <stdio.h>
int main() {
int myInt = 9.99;
printf("%d", myInt);
}
```

Output: 9

Explicit Type Conversion



Q1.
```
#include <stdio.h>
int main() {
  float a = 1.5;
  int b = (int)a;
  printf("a = %f\n", a);
  printf("b = %d\n", b);
}
```

Q2.
```
#include<stdio.h>
int main() {
  int number = 35;
  printf("Integer Value: %d\n", number);
  double value = (double) number;
  printf("Double Value: %.2lf", value);
}
```
Integer Value: 35
Double Value: 35.00

Q3.
```
#include<stdio.h>
int main(){
double x = 1.2;
int sum = (int)x + 1;
printf("sum = %d", sum);
return 0;
}
```
Output
sum = 2

Q4.
```
#include<stdio.h>
int main() {
   int number = 97;
   printf("Integer Value: %d\n", number);
   char alphabet = (char) number;
   printf("Character Value: %c", alphabet);
}
```
Integer Value: 97
Character Value: a

Q5.
```
#include<stdio.h>
int main()
{
float sum = (float) 5 / 2;
printf("%f", sum);
}
```
Output:
2.500000

Q6.
```
#include <stdio.h>
int main() {
int num1 = 5;
int num2 = 2;
float sum = (float) num1 / num2;
printf("%.1f", sum);
}
```

Q7.
```
#include <stdio.h>
int main() {
int num1 = 7;
int num2 = 4;
float sum = (float) num1 / num2;
printf("%f", sum);
}
```

❖ Online Judge Problems Based On this Lecture!

BeeCrowd:

1. https://judge.beecrowd.com/en/problems/view/1000
2. https://judge.beecrowd.com/en/problems/view/1001
3. https://judge.beecrowd.com/en/problems/view/1002
4. https://judge.beecrowd.com/en/problems/view/1003
5. https://judge.beecrowd.com/en/problems/view/1004
6. https://judge.beecrowd.com/en/problems/view/1005
7. https://judge.beecrowd.com/en/problems/view/1006
8. https://judge.beecrowd.com/en/problems/view/1007
9. https://judge.beecrowd.com/en/problems/view/1008
10. https://judge.beecrowd.com/en/problems/view/1009

CodeForces:

1. https://codeforces.com/group/MWSDmqGsZm/contest/219158/problem/A
2. https://codeforces.com/group/MWSDmqGsZm/contest/219158/problem/B
3. https://codeforces.com/group/MWSDmqGsZm/contest/219158/problem/C
4. https://codeforces.com/group/MWSDmqGsZm/contest/219158/problem/D
5. https://codeforces.com/group/MWSDmqGsZm/contest/219158/problem/E

Exercise: Solve this Questions!

Q1.

```
#include<stdio.h>
int main(){
 char a = 'P';
 char b = 'x';
 char c = (a & b) + '*';
 char d = (a | b) - '-';
 char e = (a ^ b) + '+';
 printf("%c %c %c\n", c, d, e);
 return 0;
}
```

ASCII encoding for relevant characters is given below

A	B	C	...	Z
65	66	67	...	90

a	b	c	...	z
97	98	99	...	122

*	+	–
42	43	45

Q2.

```
int main ()
{
printf ("%d\t", sizeof (6.5));
printf ("%d\t", sizeof (90000));
printf ("%d\t", sizeof ("A"));
}
```

Q3.

```
int main() {
float a = 15.3;
int b = (int) a + 3;
printf("The value of b is: %d", b);
}
```

Q4.

```
int main()
{
int a,b = 2;
float x=9.5,d;
char y='A',
a= x+y;
printf("When stored as integer : %d\n",a);
d=x+b;
printf("When stored as float : %f\n",d);
}
```

Q5.

```
int main ()
{
int a = 3;
int b = 20;
float y;
y= b / (float) a;
printf ("With explicit type conversion: %f\n", y);
}
```