

C made easy

C programming fking easy

# Introduction:

Developed at AT&T bell laboratories in 1972 by Dennis Ritchie. No one can beat C when it comes to fact of performance. Windows, linux, unix all written in C

*#include<stdio.h> -* **printf(), scanf(), cout,cin.**

*#include<conio.h> -* **clrscr(), getch(), getche() etc.**

*int main() - crogram starts from main*

*{printf(“hello world”);*

*getch(); /\**to make screen still upto given a key, if it not, the screen vanishes instanly\*/

*return 0; - return 0 to os*

*}*

1. **Identifiers**- All elements which c program can identifies, such as variables, functions, constant, keywords, array and user defined items
2. **Keywords**- All predefined meaning words. 44 keywords in C.(just remember how many keywords that C has)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| auto | break | case | char | const | Continue | default | do | Double |
| extern | Float | for | goto | If | Inline | Int | long | register |
| short | Signed | sizeof | static | Struct | switch | typedef | union | unsigned |

1. **Datatypes**- Predefined memory requirement and variable declaration. Int, char, float, double.

a) Primary: 1.Integeral- int (1byte), char (2byte)

2. Floating- float (4byte), double (8byte)

b) Secondary:

|  |  |  |  |
| --- | --- | --- | --- |
| **Datatype** | **Memory** | **Range** | **Format** |
| Signed char | 1 byte | -128 to 127 | %c |
| Unsigned char | 1 byte | 0 to 255 | %c |
| Short signed int | 2 byte | -32,768 to 32,768 | %d |
| Short unsigned int | 2 byte | 0 to 65535 | %u |
| Signed int | 2 byte | -32,768 to 32,768 | %d |
| Unsigned int | 2 byte | 0 to 65535 | %u |
| Long signed int | 4 byte | -2,147,483,648 to 2,147,483,648 | %Ld |
| Long unsigned int | 4 byte | 0 to 4294967295 | %lu |
| Float | 4 byte | -3.4e38 to 3.4e38 | %f |
| Double | 8 byte | -1.7e308 to -1.7e308 | %lf |
| Long double | 10 byte | -1.7e4938 to 1.7e4932 | %lf |

\*float value=4e10 //4 power10

\*sizeof(value)// *it will print bytes occupied by value*

*\**null character // ’\0’

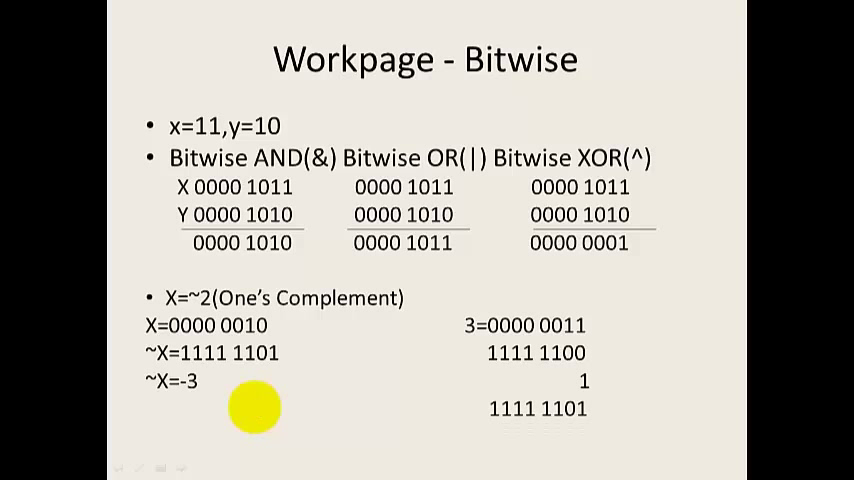
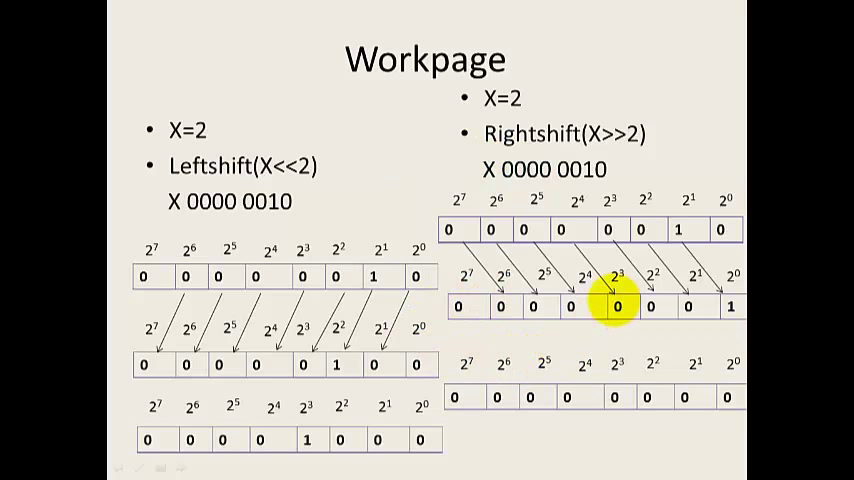
**Typedef:** rename a datatype, syntax: typedef int rajesh; rajesh a=10;//*rajesh become int*

1. **Variables**- straight name for data(memory location) given by user. Type of variable is type of constant that variable can store
2. **Constant**- non changeable value by user
   1. Integer-23,454,566665
   2. Real constant-25.8694, 299.4945
   3. Character constant-’a ’ ’A ’ ’z ’ ’9’ ‘=’
   4. String constant- ‘Rajesh’

**Typecasting:** Way to convert a variable from one data type to another data type

Byte->short->int->long->float->double

1. **Operators-**It is an symbol that specifies an math operation
   1. Arithmetic operators +-\*/%
   2. Relational operators <, <=,>,>=, ==,! =,Take !a>=400 as a<=400.
   3. Logical operators &&(and),||(or),!(not) *is for bytes*
   4. Assignment operators x+=y is x=x+y
   5. Increment decrement operators ++,--
   6. Conditional operators(ternary) condition?exp1:exp2;
   7. Bitwise operators &(and),|(or),^(xor),~(1s compliment)

 ****

* 1. Special operators comma, sizeof, pointer, member selection
  2. Typecasting z=x(float)/y(float) *where x and y is int*

1. **Input/output**
   1. Unformatted-getc(variable);getchar(variable);gets(variable); putc(variable); putchar(variable) puts(variable);

*Char raj; raj=getchar(); putchar(raj);*

*Char raj[20]; gets(raj); puts(raj);*

* 1. Formatted- scanf(); printf(); fscan(); fprint();

**Conditional statements**

%c character

%d integer

%s string

%f float

%p pointer

1. **Decisions making**
   1. **if-** if(condition){exp1;}
   2. **if else-**if(condition){exp1;}else{exp2;}
   3. if else if

if(condition){exp 1;}else if(condition){ exp2;} else {exp3;}

* 1. **nested if-** if(condition1){if(condition2){exp1}else{exp2}}else{exp3}}
  2. **tenary operator-**expersion1 ? expersion 1: expersion 2;

1. **Looping**
   1. **while** while(condition){body;}
   2. **do while** do{body}while(condition)
   3. **for** for(intialise counter;test condition;inc or dec counter){body;}

for(; i<=5; i++); print last loop only be aware

* 1. **nesting of for** for(conditions){for(conditions){body}body;}
  2. **switch statement** switch(*varible*){case *constant*: operation; break;…..default: operation;break;}

**\***u can combine two cases so make use of it

* 1. **break statement** break; (complete teriminate)
  2. **continue statement** continue;(back to start of loop)
  3. **goto statement** goto label; label:{exp}

1. **Array-**Collection of similar data items under a common name
2. one dimensional, b) two dimensional, c) three dimensional

char raj[a]; u should give length int raj[]; no need

1. **String-** group of character, digits, symbols enclose within quotation

*Don’t confuse putchar %s printf, its easy only*

strcpy(a,b);//*copy b string to a*

strcat(a,b);//*add or join b string with a string*

1. **Pointers-**It is a variable that contains address of the variable
2. **‘\*’ is Indirection operator(\*) b. ‘&’ is Address of operator(0026)**

%p variable -print address

%s “variable” -print string

%d variable -print value

%p pointer -pointer address of variable

\*rajesh=9; value of rajesh is 9

int \*\*p; p=&q; \*q=a; a=4//*pointer to pointer*

**Array and pointers:** In a program the array has ascending address, so it is easy to predict the address of the array, int a[]={1,2,3,4,5,6};

Printf(”%d”,a); //so a is gives address of first value address

Printf(“%d”,a+1);//gives address of next value

Printf(“%d”,\*(a+1));//value of integer in that address ie.2

1. **Function-**

a.predefined- printf() scanf() b.userdefined

return function: a=function();

function(){return(x);}

1. **Pass by value-** normal passing, the value copied to function paramter
2. **Pass by reference-** address get copied

void display(int \*){--}

main(){ int a; display(&a);}

void display(int \*x){--}

**note:***while passing array just type the variable alone the & is not needed*

1. **Structure**-different datatype in same

struct rajesh{

----};

int maint(){

struct rajesh a1={raj,21,048}; printf(“%d”,a1.name);}

Int main(){a1=a2; // *To copy to structures*

Printf(“%d”,a2.name);}//*will print value of a1 in a2*

U cant copy a1.name=a2.name, then u have to use strcpy();

**Nested structure:**

**s**truct address{-int number;-}; struct student{-int age;-struct address addr};

main(){ struct student a1; a1.age=2; a1.addr.number=4;}

**Array of structures:**

main(){struct student a[i]; int i=4;}// *we can control the number of entries*

**Structure pointer:**

main(){strcut student \*ptr; ptr=&a; printf(“%d”,ptr->age);}//*this opertr and normal dot operator not works for it*

**Passing structure to function by reference**

main(){struct student a1; display(&a1);}display(\*ptr){printf(“%d”,ptr->age)}

**Typedef Structure:**

Typedef struct{int age;}rajesh; main(){rajesh a1; a1.age =18;}

1. **Union-**Same as structure, different datatypes are stored in same memory location, thus display only one datatype. It occupy memory nearly half of structure
2. **Enum-**Its like array, which should sequence like days months etc.

*enum=[mon=1,tue,wed,thu];main() enum day, day\_count;//*declaration

1. **Typedef-** give another name to datatype *typedef struct student{}object;*

*main(){object temp; Temp.name;temp.regno;}*

1. **Storage classes** auto int rajesh=1;

1.auto(*default value*)

2. register (*use cpu register instead of ram*)

3. static ( *keep a local variable in existence, instead of creating and destroying it each time, stored in heap, value is 0 not garbage,*)

4. external(*can use global variables, u can give value outside main or fucntion*)

1. **Memory allocation-** Memory is allocated dynamically during program execution from the heap(set of unused memory) segment

int \*a;

a=(int \*)malloc(n\*sizeof(int));

b=(int \*)calloc(z,n\*sizeof(int));

a=(int \*)realloc(int,sizeof(int));

Free(a);

1. **Preprocessor**
2. define

**#**define macroname chractername *ex.#define LIMIT 10 if 10<=LIMIT*

#define macroname (variable) instructions /\*macro used as function\*/

ex: #define Rajesh(a) if (a<10){print();}else{printf();} main(){ Rajesh(9); }

1. Preprocessor directives

#include “rajesh.c” //*to include a file in same folder*

1. conditional compilation

#IF #ELSE #ELIF #ENDIF #IFDEF #IFNDEF

1. Standard library function

#include<stdio.h> - prinf,scanf,getchar,putchar

#include<stdlib.h> number rconversion, memory alloc, exit and system, quick sort

#include<float.h> system liit for flot type

#include<math.h> mathematical function

#include<assert.h> assertions

#include<ctype.h> character class tests

#include<limits.h> system limit for integral types

#include<setjmp.h> non local jumps

#include<signal.h> signal and error handling

#include<stdarg.h> variable length parameter list

#include<string.h> string function

#include<time.h> date and time functions

#include<conio.h> clrscr(), getch(), getche() etc

1. **FILE handling**

File handling 1.Sequenctial 2.Random

main(){

FILE \*Fpointer;//*Fpointer maybe any user defined name*

Fpointer=fopen(“rajesh.txt”,”w”); //r-read, w-write, a-edit

If(Fpointer == Null){printf(“unable to open file;”)}

else{ getc(a);

gets(a);//*checking file*

fputc(a[i],Fpointer);//*to write a char in file*

fputs(a[i],Fpointer);//*string*

fscanf(F pointer,”%d”,&num)

fprintf(Fpointer,”%d”,num);//*to perform print in file*

fclose(Fpointer); }}

rename(“rajesh.txt”,”muthu.txt”);//*rename text file in that folder*

remove(“muthu.txt”);// *file removed from that folder*

remove(“folder/muthu.txt”);//*remove if file from a folder*

22.Time

#include <time.h>

int main(){

struct tm \*local,\*gm;

time\_t t;

t=time(NULL);

local=localtime(&t);

printf("time--- %d:%d:%d",local->tm\_hour,local->tm\_min,local->tm\_sec);

printf("month--- %d / %d / %d",local->tm\_year+1900,local->tm\_mon+1,local->tm\_mday);

gm=gmtime(&t);//*greenwhich merdian time*

printf("time--- %d:%d:%d",gm->tm\_hour,gm->tm\_min,gm->tm\_sec);

ctime(&t);

puts(ctime(&t));

asctime(localtime(&t));

puts(asctime(localtime(&t)));

strftime()

%A full weekday name

%B full month name

%C standard date time string

%H hour(0-23)

%I hout(0-12)

%P local AM PM

%M minute(0-59)

%S secound (0-60)

%X standard time string

%Z time zone name

Main(){

Struct tm \*ptr;

Time\_t t;

Char str[100];

t=time(NULL);

ptr=localtime(&t);

strftime(str,100,”it is now %I %P%M……);

puts(str);}

# Important facts:

* Linkers invokes objects in program
* Escape sequence using backslash \n \t \s etc
* ASCII *American standard code for information interchange*
* To make infinite loop for(;;) for(;1;)

# Character test function

if(isalpha(ch)) -is character is alphabet

if(isdigit(ch)) -is character is digit

if(islower(ch)) -if it is lowercasenn

if(isupper(ch)) -if it is uppercase

if(ispunct(ch)) -if it is punctuation

if(isspace(ch)) -if it is space

if(isprint(ch)) -if is printable character(a,b,c)

if(iscntrl(ch)) -if it is control character(/n/t..)

abs(variable) -become non negative value absolute

tolower(ch)-convert it to lower case

toupper(ch)-convert it to upper case

# String standard function

strlen() –to find string length

strcpy() –copy two strings

strcat() –combine two strings strcat(variable, newname);

strcmp() –compare two strings

strlwr() –use to change upper case

strupr() –use to change lower case

strdup() –use to duplicate string

strrev() –use to reverse string

int a=atoi(string); -to convert string into integer

int n=div(10,3); n.rem=remainder, n.quot=quotient

sqrt(var);

cbrt(var);

ceil(9.4); -give smallest whole number not less than 9.4 so give 10

floor(9.6) -give largest whole number not greater than 9.6 so give 9

round(9.4) -give nearest whole number so give 9

pow(base,expression) –give the base^expression

memcmp(str1,str2,minimumlenght); compare strings same or not

memcpy(str1,str2,lenght); copy one string to another

memmov(str1,str2,lenght); move one string from another

exit(0); just stop the program and still

printf(strerror(0)); “No error”

printf(strerror(1)); “Operation not permitted”

printf(strerror(2)); “No such file or directory”