INTEGER ARITHMETIC:

**ADDITION**

#include <stdio.h>

int main() {

int num1 = 2;

int num2 = 3;

int sum;

sum = num1 + num2;

printf("The sum of %d and %d is %d\n", num1, num2, sum);

return 0;

}

**SUBTRACTION:**

#include <stdio.h>

int main() {

int num1 = 5;

int num2 = 3;

int difference;

difference = num1 - num2;

printf("The difference between %d and %d is %d\n", num1, num2, difference);

return 0;

}

**MULTIPLICATION:**

#include <stdio.h>

int main() {

int num1 = 5;

int num2 = 3;

int product;

product = num1 \* num2;

printf("The product of %d and %d is %d\n", num1, num2, product);

return 0;

}

**DIVISION:**

#include <stdio.h>

int main() {

int dividend = 10;

int divisor = 2;

int quotient;

quotient = dividend / divisor;

printf("The quotient of %d divided by %d is %d\n", dividend, divisor, quotient);

return 0;

}

**FLOATING POINTS:**

**ADDITION:**

#include <stdio.h>

int main() {

float num1 = 2.5;

float num2 = 3.7;

float sum;

sum = num1 + num2;

printf("The sum of %f and %f is %f\n", num1, num2, sum);

return 0;

}

**SUBTRACTION:**

#include <stdio.h>

int main() {

float num1 = 5.8;

float num2 = 2.3;

float difference;

difference = num1 - num2;

printf("The difference between %f and %f is %f\n", num1, num2, difference);

return 0;

}

**MULTIPLICATION:**

#include <stdio.h>

int main() {

float num1 = 2.5;

float num2 = 3.2;

float product;

product = num1 \* num2;

printf("The product of %f and %f is %f\n", num1, num2, product);

return 0;

}

**DIVISION:**

#include <stdio.h>

int main() {

float dividend = 10.0;

float divisor = 3.0;

float quotient;

quotient = dividend / divisor;

printf("The quotient of %f divided by %f is %f\n", dividend, divisor, quotient);

return 0;

}

**Single-precision representation:**

#include <stdio.h>

#include <stdint.h>

void printBinary(uint32\_t num) {

for (int i = 31; i >= 0; i--) {

printf("%d", (num >> i) & 1);

if (i == 31 || i == 23)

printf(" ");

}

printf("\n");

}

int main() {

float num;

printf("Enter a single-precision floating-point number: ");

scanf("%f", &num);

uint32\_t\* binaryRep = (uint32\_t\*)&num;

printf("Binary representation: ");

printBinary(\*binaryRep);

return 0;

}

**Double-precision representation:**

#include <stdio.h>

#include <stdint.h>

void print\_double\_binary(double num) {

uint64\_t \*ptr = (uint64\_t \*)&num; // Treat the double as a 64-bit unsigned integer

uint64\_t mask = 1ULL << 63; // Start with the most significant bit

printf("Binary representation of %.15lf: ", num);

for (int i = 0; i < 64; i++) {

printf("%d", (\*ptr & mask) ? 1 : 0);

if (i == 0 || i == 11) // Print the sign bit and the exponent

printf(" ");

mask >>= 1; // Move to the next bit

}

printf("\n");

}

int main() {

double num = 3.141592653589793238; // Example double-precision floating-point number

print\_double\_binary(num); // Print the binary representation

return 0;

}

**RESTORING DIVISION:**

#include <stdio.h>

#include <string.h>

void restoring\_division(int dividend[], int divisor[], int quotient[]) {

int partial\_remainder[N+1];

int borrow = 0;

memset(partial\_remainder, 0, sizeof(partial\_remainder));

for (int i = 0; i < N; i++) {

for (int j = N; j > 0; j--)

partial\_remainder[j] = partial\_remainder[j - 1];

partial\_remainder[0] = dividend[i];

for (int j = 0; j < N+1; j++) {

partial\_remainder[j] -= divisor[j];

if (partial\_remainder[j] < 0) {

partial\_remainder[j] += 2;

partial\_remainder[j+1] -= 1;

}

}

quotient[i] = (partial\_remainder[0] >= 0) ? **1** : 0;

if (partial\_remainder[0] < 0) {

for (int j = 0; j < N+1; j++) {

partial\_remainder[j] += divisor[j];

}

}

}

}

int main() {

int dividend[N] = {1, 1, 0, 1, 0, 1, 0, 1}; // Binary representation of dividend (example)

int divisor[N] = {1, 0, 1, 1, 0, 0, 1, 0}; // Binary representation of divisor (example)

int quotient[N]; // Quotient will be of the same size as the dividend

restoring\_division(dividend, divisor, quotient);

printf("Quotient: ");

for (int i = 0; i < N; i++) {

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

}

printf("\n");

return 0;

}

**NON RESTORING:**

#include <stdio.h>

#include <string.h>

void non\_restoring\_division(int dividend[], int divisor[], int quotient[]) {

int partial\_remainder[N+1];

int borrow = 0;

memset(partial\_remainder, 0, sizeof(partial\_remainder));

for (int i = 0; i < N; i++) {

for (int j = N; j > 0; j--)

partial\_remainder[j] = partial\_remainder[j - 1];

partial\_remainder[0] = dividend[i];

if (partial\_remainder[0] == 0) {

for (int j = 0; j < N+1; j++) {

partial\_remainder[j] -= divisor[j];

if (partial\_remainder[j] < 0) {

partial\_remainder[j] += 2;

partial\_remainder[j+1] -= 1;

}

}

} else {

for (int j = 0; j < N+1; j++) {

partial\_remainder[j] += divisor[j];

if (partial\_remainder[j] >= 2) {

partial\_remainder[j] -= 2;

partial\_remainder[j+1] += 1;

}

}

}

quotient[i] = (partial\_remainder[0] >= 0) ? 1 : 0;

if (partial\_remainder[0] < 0) {

for (int j = 0; j < N+1; j++) {

partial\_remainder[j] += divisor[j];

}

}

}

}

int main() {

int dividend[N] = {1, 1, 0, 1, 0, 1, 0, 1};

int divisor[N] = {1, 0, 1, 1, 0, 0, 1, 0};

int quotient[N];

non\_restoring\_division(dividend, divisor, quotient);

printf("Quotient: ");

for (int i = 0; i < N; i++) {

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

}

printf("\n");

return 0;

}

**BOOTH ALGORITHM:**

#include <stdio.h>

void booth\_multiplication(int multiplicand, int multiplier, int \*result) {

\*result = 0;

int multiplier\_bits = 0;

int sign\_bit = multiplier & 0x80000000;

while (multiplier != 0) {

int ls\_bit = multiplier & 0x1;

if (ls\_bit != multiplier\_bits) {

if (ls\_bit == 1) {

\*result += multiplicand;

} else {

\*result -= multiplicand;

}

}

multiplicand <<= 1;

int msb = multiplicand & 0x80000000;

if (msb != 0) {

multiplicand |= 0xFFFFFFFF;

}

multiplier >>= 1;

multiplier\_bits = ls\_bit;

}

if (sign\_bit != 0) {

\*result = -\*result;

}

}

int main() {

int multiplicand, multiplier;

int product;

printf("Enter multiplicand: ");

scanf("%d", &multiplicand);

printf("Enter multiplier: ");

scanf("%d", &multiplier);

booth\_multiplication(multiplicand, multiplier, &product);

printf("Product: %d\n", product);

return 0;

}

CODE CONVERSIONS:-

1. Convert Binary to Octal

#include <stdio.h>

void main()

{

long num, binary\_num, decimal\_num = 0, base = 1, rem;

printf (" Enter a binary number with the combination of 0s and 1s \n");

scanf (" %ld", &num);

binary\_num = num;

while ( num > 0)

{

rem = num % 10;

decimal\_num = decimal\_num + rem \* base;

num = num / 10;

base = base \* 2;

}

printf ( " The binary number is %ld \t", binary\_num);

printf (" \n The decimal number is %ld \t", decimal\_num);

int n=decimal\_num;

printf (" \n The octal number is %o \t", n);

}

1. Convert Binary to Decimal

#include <stdio.h>

void main()

{

long num, binary\_num, decimal\_num = 0, base = 1, rem;

printf (" Enter a binary number with the combination of 0s and 1s \n");

scanf (" %ld", &num);

binary\_num = num;

while ( num > 0)

{

rem = num % 10;

decimal\_num = decimal\_num + rem \* base;

num = num / 10;

base = base \* 2;

}

printf ( " The binary number is %ld \t", binary\_num);

printf (" \n The decimal number is %ld \t", decimal\_num);

int n=decimal\_num;

printf (" \n The decimal number is %x \t", n);

}

1. Convert Binary to Hexa-decimal

#include <stdio.h>

int main() {

long num, binary\_num, decimal\_num = 0, base = 1, rem;

printf("Enter a binary number with the combination of 0s and 1s:\n");

scanf("%ld", &num);

binary\_num = num;

while (num > 0) {

rem = num % 10;

decimal\_num = decimal\_num + rem \* base;

num = num / 10;

base = base \* 2;

}

printf("The hexadecimal number is %X\n", decimal\_num);

return 0;

}

1. Convert Decimal to Binary

#include<stdio.h>

int main()

{

int a[10],n,i;

printf("Enter the number to convert: ");

scanf("%d",&n);

for(i=0;n>0;i++)

{

a[i]=n%2;

n=n/2;

}

printf("\nBinary of Given Number is=");

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

{

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

}

return 0;

}

1. Convert Decimal to Octal

#include<stdio.h>

int main()

{

int a[10],n,i;

printf("Enter the number to convert: ");

scanf("%d",&n);

for(i=0;n>0;i++)

{

a[i]=n%8;

n=n/8;

}

printf("\nOctal of Given Number is=");

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

{

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

}

return 0;

}

1. Convert Decimal to Hexa-decimal

#include<stdio.h>

int main()

{

int n;

printf("enter the decimal number");

scanf("%d",&n);

printf("the hexa decimal value is:%x",n);

return 0;

}

1. Convert Hexa-decimal to Binary

#include <stdio.h>

int main() {

int n, a[10], m, i;

printf("Enter the hexadecimal number: ");

scanf("%x", &n);

m = n; // Save the decimal value in variable m

printf("Decimal value: %d\n", m);

for (i = 0; m > 0; i++) {

a[i] = m % 2;

m = m / 2;

}

printf("Binary of Given Number is: ");

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

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

}

return 0;

}

1. Convert Hexa-decimal to Decimal

#include<stdio.h>

int main()

{

int n;

printf("enter the hex decimal number");

scanf("%x",&n);

printf("the decimal value is:%d",n);

return 0;

}

1. Convert Hexa-decimal to Octal

#include <stdio.h>

int main() {

int n, a[10], m, i;

printf("Enter the hexadecimal number: ");

scanf("%x", &n);

m = n; // Save the decimal value in variable m

printf("Decimal value: %d\n", m);

for (i = 0; m > 0; i++) {

a[i] = m % 8;

m = m / 8;

}

printf("Octal of Given Number is: ");

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

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

}

return 0;

}

1. Convert Octal to Binary

#include <stdio.h>

int main()

{

char octalnum[100];

long i = 0;

printf("Enter any octal number: ");

scanf("%s", octalnum);

printf("Equivalent binary value: ");

while (octalnum[i])

{

switch (octalnum[i])

{

case '0':

printf("000"); break;

case '1':

printf("001"); break;

case '2':

printf("010"); break;

case '3':

printf("011"); break;

case '4':

printf("100"); break;

case '5':

printf("101"); break;

case '6':

printf("110"); break;

case '7':

printf("111"); break;

default:

printf("\n Invalid octal digit ");

return 0;

}

i++;

}

return 0;

}

1. Convert Octal to Hexa-decimal

#include <stdio.h>

int main() {

int octal,hexa;

printf("Enter the octal value : ");

scanf("%o",&octal);

printf("The hexadecial of given octal number is : %x",octal);

}

1. Convert Octal to Decimal

#include <stdio.h>

int main() {

int octal;

printf("Enter the octal value : ");

scanf("%o",&octal);

printf("The decimal of given number is : %d",octal);

}