```c
// Created by hfwei on 2024/10/30.

#include <limits.h>
#include <stdio.h>

int main() {
  printf("CHAR_MIN = %d\n", CHAR_MIN);
  printf("CHAR_MAX = %d\n", CHAR_MAX);

  char c = 150;
  int i = 900;

  printf("i / c = %d\n", i / c);

  return 0;
}
```

```c
// Created by hengxin on 2024/10/30.

#include <stdio.h>

int main() {
  /**
   * Do not use a counter of type float/double,
   * although it works on some platforms.
   *
   * 0.1 cannot be exactly represented in machines.
   */
  for (double x = 0.1; x <= 1.0; x += 0.1) {
    printf("%.20f\n", x);
  }

  return 0;
}
```

```c
// Created by hfwei on 2024/10/31.

#include <stdio.h>

int main() {
  // Integer types
  printf("Size of char: %zu bytes\n", sizeof(char));
  printf("Size of signed char: %zu bytes\n", sizeof(signed char));
  printf("Size of unsigned char: %zu bytes\n\n", sizeof(unsigned char));

  printf("Size of short: %zu bytes\n", sizeof(short));
  printf("Size of unsigned short: %zu bytes\n\n", sizeof(unsigned short));

  printf("Size of int: %zu bytes\n", sizeof(int));
  printf("Size of unsigned int: %zu bytes\n\n", sizeof(unsigned int));

  printf("Size of long: %zu bytes\n", sizeof(long));
  printf("Size of unsigned long: %zu bytes\n\n", sizeof(unsigned long));

  printf("Size of long long: %zu bytes\n", sizeof(long long));
  printf("Size of unsigned long long: %zu bytes\n\n",
         sizeof(unsigned long long));

  // (Real) Floating-point types
  printf("Size of float: %zu bytes\n", sizeof(float));
  printf("Size of double: %zu bytes\n", sizeof(double));
  printf("Size of long double: %zu bytes\n\n", sizeof(long double));

  // Array types
  int numbers[] = {0, 1, 2, 3, 4};
  size_t len = sizeof numbers / sizeof(int);
  printf("Length of numbers: %zu\n", len);

  return 0;
}
```

```c
// Created by hfwei on 2024/10/31.

#include <limits.h>
#include <stdio.h>

#define SIZE UINT_MAX

char string[SIZE] = {'A', 'B', 'C', 'D', 'E', 'F'};

void Print(const int string[], size_t size);

int main(void) {
  Print(string, SIZE);

  return 0;
}

void Print(const int string[], size_t size) {
  for (int i = 0; i < size; i++) {
    printf("%d : %d\n", i, string[i]);
  }
}
```

```c
/**
 * See
 * https://randomascii.wordpress.com/2012/02/25/comparing-floating-point-numbers-2012-edition/
 *
 * Created by hfwei on 2024/10/31.
 */

#include <float.h>
#include <math.h>
#include <stdbool.h>
#include <stdio.h>

#define EPSILON 1e-5

bool IsEqual(double x, double y);

int main() {
  printf("%d\n", IsEqual(DBL_MAX, DBL_MAX - 100));

  printf("%.50f\n", DBL_MAX - (DBL_MAX - 100));

  return 0;
}

bool IsEqual(double x, double y) { return fabs(x - y) <= EPSILON; }
```

```markdown
# 7-data-types

## `int-limits.c`

## `unsigned.c`

## `timing.c`

## `char.c`

## `int-overflow.c`

## `implicit-inversion.c`

## `explicit-inversion.c`

## `float-limits.c`

## `sums.c`

## `loop.c`

## `compare.c`
```

```c
// Created by hfwei on 2024/10/10.

#include <stdio.h>

int main() {
  const int array[] = {0, 1, 2, 3, 4};
  int i = -1;

  size_t size = sizeof array / sizeof array[0];
  printf("The size of the array is %zu\n", size);

  if (i <= size) {
    printf("i <= sizeof array\n");
  } else {
    printf("i > sizeof array\n");
  }

  return 0;
}
```

```c
// Created by hfwei on 2024/10/31.

#include <limits.h>
#include <stdio.h>
#include <stdlib.h>

unsigned int pow2(unsigned int exp);

int main(void) {
  unsigned int exp = 30;

  unsigned int pow = pow2(exp);
  printf("2^%d = %d\n", exp, pow);

  return 0;
}

unsigned int pow2(unsigned int exp) {
  if (exp >= sizeof(unsigned int) * CHAR_BIT) {
    printf("Exp is too large!\n");
    exit(1);
  }

  return 1 << exp;
}
```

```c
// Created by hfwei on 2024/10/30.
// Run on Windows and Linux

#include <limits.h>
#include <stdio.h>

int main() {
  // INT_MIN = -2147483648
  // INT_MAX = 2147483647 (10 digits; ~ 2 Billion)
  printf("INT_MIN = %d\n", INT_MIN);
  printf("INT_MAX = %d\n\n", INT_MAX);

  // printf("UINT_MIN = %u\n", 0U);
  // printf("UINT_MAX = %u\n\n", UINT_MAX);

  printf("LONG_MIN = %ld\n", LONG_MIN);
  printf("LONG_MAX = %ld\n\n", LONG_MAX);

  // printf("ULONG_MIN = %lu\n", 0UL);
  // printf("ULONG_MAX = %lu\n\n", ULONG_MAX);

  // long long int: >= 64 bits

  // LLONG_MIN = -9223372036854775808
  // LLONG_MAX = 9223372036854775807 (19 digits)
  printf("LLONG_MIN = %lld\n", LLONG_MIN);
  printf("LLONG_MAX = %lld\n\n", LLONG_MAX);

  // printf("ULONG_LONG_MIN = %llu\n", 0ULL);
  // printf("ULONG_LONG_MAX = %llu\n\n", ULONG_LONG_MAX);
  //
  // printf("ULLONG_MAX = %llu\n\n", ULLONG_MAX);

  return 0;
}
```

```c
// Created by hfwei on 2024/10/31.

#include <stdint.h>

int main(void) {
  int8_t small = -100;
  int32_t large = 100000;

  return 0;
}
```

```c
/**
 * file: sums.c
 * See
 * https://randomascii.wordpress.com/2012/02/25/comparing-floating-point-numbers-2012-edition/
 *
 * Created by hengxin on 2024/10/30.
 */

#include <stdio.h>

int main() {
  // 0.1: 0.0 0011 0011 0011
  float f = 0.1F;

  float sum = 0.0F;
  for (int i = 0; i < 10; ++i) {
    sum += f;
  }

  float product = f * 10;

  printf("sum = %.15f\nmul = %.30f\n", sum, product);

  return 0;
}
```

```
# Objects, size, precision, width, limits
add_executable(size size.c)
add_executable(precision precision.c)
add_executable(int-limits int-limits.c)
add_executable(exact-width exact-width.c)

add_executable(unsigned unsigned.c)
add_executable(sizet sizet.c)
add_executable(timing-primes timing-primes.c)

add_executable(char char.c)

add_executable(unsinged-wrap unsigned-wrap.c)
add_executable(for-unsigned for-unsigned.c)
add_executable(unsigned-wrap-fix unsigned-wrap-fix.c)

add_executable(signed-overflow-fix signed-overflow-fix.c)

add_executable(implicit-conversion implicit-conversion.c)
add_executable(integer-promotion integer-promotion.c)
add_executable(explict-conversion explict-conversion.c)

add_executable(float-limits float-limits.c)

add_executable(sum-product sum-product.c)
add_executable(loop loop.c)

add_executable(compare compare.c)
target_link_libraries(compare m)
```

```c
// Created by hfwei on 2024/10/30.

#include <float.h>
#include <stdio.h>

int main() {
  // float pi = 3.14F;

  // 3.402823e+38
  printf("FLT_MAX = %e\n", FLT_MAX);
  // 1.175494e-38
  printf("FLT_MIN = %e\n", FLT_MIN);
  // 1.401298e-45
  printf("FLT_TRUE_MIN = %e\n", FLT_TRUE_MIN);
  // 1.192093e-07
  printf("FLT_EPSILON = %e\n\n", FLT_EPSILON);

  // %lf for scanf
  // 1.797693e+308
  printf("DBL_MAX = %e\n", DBL_MAX);
  // 2.225074e-308
  printf("DBL_MIN = %e\n", DBL_MIN);
  // 4.940656e-324
  printf("DBL_TRUE_MIN = %e\n", DBL_TRUE_MIN);
  // 2.220446e-16
  printf("DBL_EPSILON = %e\n\n", DBL_EPSILON);

  return 0;
}
```

```c
// Created by hfwei on 2024/10/31.

#include <stdio.h>
#define LEN 100

int main(void) {
  int numbers[LEN] = {0};

  for (unsigned int i = LEN; i >= 0; i--) {
    printf("%u : %d\n", i, numbers[i]);
  }

  return 0;
}
```

```c
// Created by hfwei on 2024/10/31.

#include <stdbool.h>
#include <stdio.h>
#include <time.h>

bool IsPrime(int number);

int main(void) {
  int max = 0;
  scanf("%d", &max);

  int count = 0;

  // TODO: return the current time in seconds since the Unix epoch (January 1, 1970)

  for (int number = 2; number <= max; number++) {
    if (IsPrime(number)) {
      count++;
    }
  }
  printf("\ncount = %d\n", count);

  // TODO: return the current time in seconds since the Unix epoch (January 1, 1970)

  return 0;
}

bool IsPrime(int number) {
  for (int factor = 2; factor * factor <= number; factor++) {
    if (number % factor == 0) {
      return false;
    }
  }

  return true;
}
```

```c
// Created by hfwei on 2024/10/30.

#include <limits.h>
#include <stdio.h>

int main() {
  printf("UINT_MAX = %u\n", UINT_MAX);

  unsigned int max = UINT_MAX;
  unsigned int one = 1U;
  unsigned int two = 2U;

  printf("max + one = %u\n", max + one);
  printf("one - two = %u\n", one - two);

  return 0;
}
```

```c
// Created by hfwei on 2024/10/31.

#include <stdio.h>

int main(void) {
  signed char left = 100;
  signed char mid = 3;
  signed char right = 4;

  signed char result = left * mid / right;

  printf("result = %d\n", result);

  return 0;
}
```

```c
// Created by hfwei on 2024/10/31.

#include <limits.h>
#include <stdio.h>
#include <stdlib.h>

unsigned int Add(unsigned int left, unsigned int right);
unsigned int Sub(unsigned int left, unsigned int right);
unsigned int Mul(unsigned int left, unsigned int right);
unsigned int Div(unsigned int left, unsigned int right);
unsigned int Mod(unsigned int left, unsigned int right);

int main(void) {
  // addition
  unsigned int left_add = UINT_MAX / 2 + 1;
  unsigned int right_add = UINT_MAX / 2 + 1;

  printf("%u + %u = %u\n\n", left_add, right_add, Add(
left_add, right_add));

  // subtraction
  unsigned int left_sub = 1;
  unsigned int right_sub = 2;

  printf("%u - %u = %u\n\n", left_sub, right_sub, Sub(
left_sub, right_sub));

  // multiplication
  unsigned int left_mul = UINT_MAX;
  unsigned int right_mul = 2;

  printf("%u * %u = %u\n", left_mul, right_mul, Mul(
left_mul, right_mul));

  // division
  unsigned int left_div = 5;
  unsigned int right_div = 0;

  printf("%u * %u = %u\n", left_div, right_div, Div(
left_div, right_div));
  printf("%u * %u = %u\n", left_div, right_div, Mod(
left_div, right_div));

  return 0;
```

```c
}

unsigned int Add(unsigned int left, unsigned int right) {
   return left + right;
}

unsigned int Sub(unsigned int left, unsigned int right) {
    return left - right;
}

unsigned int Mul(unsigned int left, unsigned int right) {
    return left * right;
}

unsigned int Div(unsigned int left, unsigned int right) {
  return left / right;
}

unsigned int Mod(unsigned int left, unsigned int right) {
  return left % right;
}

// Add:
//  if (left + right > UINT_MAX) {
//    printf("Too Big!\n");
//    exit(1);
//  } else {
//    return left + right;
//  }

// Sub:
//  if (left - right < 0) {
//    printf("The result is negative!\n");
//    exit(1);
//  } else {
//    return left - right;
//  }

// Mul:
//  if (left * right > UINT_MAX) {
//    printf("The result is negative!\n");
//    exit(1);
//  } else {
//    unsigned int mul = left * right;
 //    return mul;
 //  }

 // Div:
 //  if (right == 0) {
 //    printf("Division by zero!\n");
 //    exit(1);
 //  }
 //
 //  return left / right;

 //  Mod:
 //  if (right == 0) {
 //    printf("Division by zero!\n");
 //    exit(1);
 //  }
 //
 //  return left % right;
 // Created by hfwei on 2024/10/30.

 #include <limits.h>
 #include <stdio.h>

 int main() {
   double pi = 3.14159;

   // to obtain its fractional part
   double fraction = 0;

   // to compute num * num
   int num = 100000000; // (8 zeros)

   printf("LLONG_MAX = %lld\n", LLONG_MAX);
   long long llint = num * num;
   printf("i = %lld\n", llint);

   return 0;
 }
```

```c
// Created by hfwei on 2024/10/30.

#include <limits.h>
#include <stdio.h>

int SquareInt(int num);
double SquareDouble(double num);

int main() {
  // narrowing conversion (still in the range)
  int i = 3.14159;

  // out of the range: undefined behavior!!!
  int j = UINT_MAX;

  // arguments; narrowing conversion
  double pi = 3.14;
  SquareInt(pi);

  // return value; narrowing conversion
  int val = SquareDouble(pi);

  // from int to float; narrowing conversion
  int big = 1234567890;
  float approx = big;

  printf("big = %d\t approx = %f\t diff = %d\n", big, approx,
         big - (int)approx);

  return 0;
}

int SquareInt(int num) { return num * num; }

double SquareDouble(double num) { return num * num; }
```

```c
// Created by hfwei on 2024/10/31.

#include <limits.h>
#include <stdio.h>
#include <stdlib.h>

int Add(int left, int right);
int Sub(int left, int right);
int Mul(int left, int right);
int Div(int left, int right);
int Mod(int left, int right);
int Neg(int left);

int main(void) {
  // addition
  int left_add = INT_MAX / 2 + 1;
  int right_add = INT_MAX / 2 + 1;

  printf("%d + %d = %d\n\n", left_add, right_add, Add(
left_add, right_add));

  // subtraction
  int left_sub = INT_MIN;
  int right_sub = 1;

  printf("%d - %d = %d\n\n", left_sub, right_sub, Sub(
left_sub, right_sub));

  // multiplication
  int left_mul = INT_MAX;
  int right_mul = 2;

  printf("%d * %d = %d\n", left_mul, right_mul, Mul(
left_mul, right_mul));

  // division
  int left_div = INT_MIN;
  int right_div = -1;

  printf("%d / %d = %d\n", left_div, right_div, Div(
left_div, right_div));

  // mod (remainder)
  int left_mod = INT_MIN;
  int right_mod = -1;

  printf("%d %% %d = %d\n", left_mod, right_mod, Mod(
left_mod, right_mod));

  // negation
  int left_neg = INT_MIN;

  printf("-%d = %d\n", left_neg, Neg(left_neg));

  return 0;
}

int Add(int left, int right) {
  return left + right;
}

int Sub(int left, int right) {
  return left - right;
}

int Mul(int left, int right) {
  return left * right;
}

int Div(int left, int right) {
  return left / right;
}

int Mod(int left, int right) {
  return left % right;
}

int Neg(int left) {
  return -left;
}

// Add:
//  if ((left > 0 && right > INT_MAX - left) ||
//      (left < 0 && right < INT_MIN - left)) {
//    printf("Overflow!\n");
//    exit(1);
//  } else {
//    return left + right;
//  }

// Sub
//if ((left > 0 && right < INT_MIN + left) ||
//(left < 0 && right > INT_MAX + left)) {
//printf("Overflow!\n");
//exit(1);
//} else {
//int sub = left - right;
//return sub;
//}

// Mul
//  if (left > 0) {
//    if (right > 0) { // left > 0 && right > 0
//      if (left > INT_MAX / right) {
//        printf("Overflow!\n");
//        exit(1);
//      }
//    } else { // left > 0 && right < 0
//      if (right < INT_MIN / left) {
//        printf("Overflow!\n");
//        exit(1);
//      }
//    }
//  } else {           // left <= 0
//    if (right > 0) { // left <= 0 && right > 0
//      if (left < INT_MIN / right) {
//        printf("Overflow!\n");
//        exit(1);
//      }
//    } else { // left <= 0 && right <= 0
//      if (left != 0 && right < INT_MAX / left) {
//        printf("Overflow!\n");
//        exit(1);
//      }
//    }
//  }
//
//  int mul = left * right;
//  return mul;

// Div
//  if (right == 0 || (left == INT_MIN && right == -1)) {
```

```c
//    printf("Overflow!\n");
//    exit(1);
//  }
//
//  return left / right;

//  Mod
//  if (right == 0 || (left == INT_MIN && right == -1)) {
//    printf("Overflow!\n");
//    exit(1);
//  }
//
//  return left % right;

// Neg
//if (left == INT_MIN) {
//printf("Overflow!\n");
//exit(1);
//}
//
//return -left;
```