

```
1 // Created by hfwei on 2024/10/30.
2
3 #include <limits.h>
4 #include <stdio.h>
5
6 int main() {
7     printf("CHAR_MIN = %d\n", CHAR_MIN);
8     printf("CHAR_MAX = %d\n", CHAR_MAX);
9
10    char c = 150;
11    int i = 900;
12
13    printf("i / c = %d\n", i / c);
14
15    return 0;
16 }
```

```
1 // Created by hengxin on 2024/10/30.
2
3 #include <stdio.h>
4
5 int main() {
6     /**
7      * Do not use a counter of type float/double,
8      * although it works on some platforms.
9      *
10     * 0.1 cannot be exactly represented in machines.
11     */
12     for (double x = 0.1; x <= 1.0; x += 0.1) {
13         printf("%.20f\n", x);
14     }
15
16     return 0;
17 }
```

```
1 // Created by hfwei on 2024/10/31.
2
3 #include <stdio.h>
4
5 int main() {
6     // Integer types
7     printf("Size of char: %zu bytes\n", sizeof(char));
8     printf("Size of signed char: %zu bytes\n", sizeof(signed
    char));
9     printf("Size of unsigned char: %zu bytes\n\n", sizeof(
    unsigned char));
10
11     printf("Size of short: %zu bytes\n", sizeof(short));
12     printf("Size of unsigned short: %zu bytes\n\n", sizeof(
    unsigned short));
13
14     printf("Size of int: %zu bytes\n", sizeof(int));
15     printf("Size of unsigned int: %zu bytes\n\n", sizeof(
    unsigned int));
16
17     printf("Size of long: %zu bytes\n", sizeof(long));
18     printf("Size of unsigned long: %zu bytes\n\n", sizeof(
    unsigned long));
19
20     printf("Size of long long: %zu bytes\n", sizeof(long
    long));
21     printf("Size of unsigned long long: %zu bytes\n\n",
    sizeof(unsigned long long));
22
23
24     // (Real) Floating-point types
25     printf("Size of float: %zu bytes\n", sizeof(float));
26     printf("Size of double: %zu bytes\n", sizeof(double));
27     printf("Size of long double: %zu bytes\n\n", sizeof(long
    double));
28
29     // Array types
30     int numbers[] = {0, 1, 2, 3, 4};
31     size_t len = sizeof numbers / sizeof(int);
32     printf("Length of numbers: %zu\n", len);
33
34     return 0;
35 }
```

```
1 // Created by hfwei on 2024/10/31.
2
3 #include <limits.h>
4 #include <stdio.h>
5
6 #define SIZE UINT_MAX
7
8 char string[SIZE] = {'A', 'B', 'C', 'D', 'E', 'F'};
9
10 void Print(const int string[], size_t size);
11
12 int main(void) {
13     Print(string, SIZE);
14
15     return 0;
16 }
17
18 void Print(const int string[], size_t size) {
19     for (int i = 0; i < size; i++) {
20         printf("%d : %d\n", i, string[i]);
21     }
22 }
```

```
1  /**
2   * See
3   * https://randomascii.wordpress.com/2012/02/25/comparing-
   floating-point-numbers-2012-edition/
4   *
5   * Created by hfwei on 2024/10/31.
6   */
7
8  #include <float.h>
9  #include <math.h>
10 #include <stdbool.h>
11 #include <stdio.h>
12
13 #define EPSILON 1e-5
14
15 bool IsEqual(double x, double y);
16
17 int main() {
18     printf("%d\n", IsEqual(DBL_MAX, DBL_MAX - 100));
19
20     printf("%.50f\n", DBL_MAX - (DBL_MAX - 100));
21
22     return 0;
23 }
24
25 bool IsEqual(double x, double y) { return fabs(x - y) <=
    EPSILON; }
```

```
1 # 7-data-types
2
3 ## `int-limits.c`
4
5 ## `unsigned.c`
6
7 ## `timing.c`
8
9 ## `char.c`
10
11 ## `int-overflow.c`
12
13 ## `implicit-inversion.c`
14
15 ## `explicit-inversion.c`
16
17 ## `float-limits.c`
18
19 ## `sums.c`
20
21 ## `loop.c`
22
23 ## `compare.c`
```

```
1 // Created by hfwei on 2024/10/10.
2
3 #include <stdio.h>
4
5 int main() {
6     const int array[] = {0, 1, 2, 3, 4};
7     int i = -1;
8
9     size_t size = sizeof array / sizeof array[0];
10    printf("The size of the array is %zu\n", size);
11
12    if (i <= size) {
13        printf("i <= sizeof array\n");
14    } else {
15        printf("i > sizeof array\n");
16    }
17
18    return 0;
19 }
```

```
1 // Created by hfwei on 2024/10/31.
2
3 #include <limits.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 unsigned int pow2(unsigned int exp);
8
9 int main(void) {
10     unsigned int exp = 30;
11
12     unsigned int pow = pow2(exp);
13     printf("2^%d = %d\n", exp, pow);
14
15     return 0;
16 }
17
18 unsigned int pow2(unsigned int exp) {
19     if (exp >= sizeof(unsigned int) * CHAR_BIT) {
20         printf("Exp is too large!\n");
21         exit(1);
22     }
23
24     return 1 << exp;
25 }
```



```
1 // Created by hfwei on 2024/10/30.
2 // Run on Windows and Linux
3
4 #include <limits.h>
5 #include <stdio.h>
6
7 int main() {
8     // INT_MIN = -2147483648
9     // INT_MAX = 2147483647 (10 digits; ~ 2 Billion)
10    printf("INT_MIN = %d\n", INT_MIN);
11    printf("INT_MAX = %d\n\n", INT_MAX);
12
13    // printf("UINT_MIN = %u\n", 0U);
14    // printf("UINT_MAX = %u\n\n", UINT_MAX);
15
16    printf("LONG_MIN = %ld\n", LONG_MIN);
17    printf("LONG_MAX = %ld\n\n", LONG_MAX);
18
19    // printf("ULONG_MIN = %lu\n", 0UL);
20    // printf("ULONG_MAX = %lu\n\n", ULONG_MAX);
21
22    // long long int: >= 64 bits
23
24    // LLONG_MIN = -9223372036854775808
25    // LLONG_MAX = 9223372036854775807 (19 digits)
26    printf("LLONG_MIN = %lld\n", LLONG_MIN);
27    printf("LLONG_MAX = %lld\n\n", LLONG_MAX);
28
29    // printf("ULONG_LONG_MIN = %llu\n", 0ULL);
30    // printf("ULONG_LONG_MAX = %llu\n\n", ULONG_LONG_MAX);
31    //
32    // printf("ULLONG_MAX = %llu\n\n", ULLONG_MAX);
33
34    return 0;
35 }
```

```
1 // Created by hfwei on 2024/10/31.
2
3 #include <stdint.h>
4
5 int main(void) {
6     int8_t small = -100;
7     int32_t large = 100000;
8
9     return 0;
10 }
```

```
1  /**
2   * file: sums.c
3   * See
4   * https://randomascii.wordpress.com/2012/02/25/comparing-
   floating-point-numbers-2012-edition/
5   *
6   * Created by hengxin on 2024/10/30.
7   */
8
9  #include <stdio.h>
10
11 int main() {
12     // 0.1: 0.0 0011 0011 0011
13     float f = 0.1F;
14
15     float sum = 0.0F;
16     for (int i = 0; i < 10; ++i) {
17         sum += f;
18     }
19
20     float product = f * 10;
21
22     printf("sum = %.15f\nmul = %.30f\n", sum, product);
23
24     return 0;
25 }
```

```
1 # Objects, size, precision, width, limits
2 add_executable(size size.c)
3 add_executable(precision precision.c)
4 add_executable(int-limits int-limits.c)
5 add_executable(exact-width exact-width.c)
6
7 add_executable(unsigned unsigned.c)
8 add_executable(sizet sizet.c)
9 add_executable(timing-primes timing-primes.c)
10
11 add_executable(char char.c)
12
13 add_executable(unsinged-wrap unsigned-wrap.c)
14 add_executable(for-unsigned for-unsigned.c)
15 add_executable(unsigned-wrap-fix unsigned-wrap-fix.c)
16
17 add_executable(signed-overflow-fix signed-overflow-fix.c)
18
19 add_executable(implicit-conversion implicit-conversion.c)
20 add_executable(integer-promotion integer-promotion.c)
21 add_executable(explct-conversion explct-conversion.c)
22
23 add_executable(float-limits float-limits.c)
24
25 add_executable(sum-product sum-product.c)
26 add_executable(loop loop.c)
27
28 add_executable(compare compare.c)
29 target_link_libraries(compare m)
```

```
1 // Created by hfwei on 2024/10/30.
2
3 #include <float.h>
4 #include <stdio.h>
5
6 int main() {
7     // float pi = 3.14F;
8
9     // 3.402823e+38
10    printf("FLT_MAX = %e\n", FLT_MAX);
11    // 1.175494e-38
12    printf("FLT_MIN = %e\n", FLT_MIN);
13    // 1.401298e-45
14    printf("FLT_TRUE_MIN = %e\n", FLT_TRUE_MIN);
15    // 1.192093e-07
16    printf("FLT_EPSILON = %e\n\n", FLT_EPSILON);
17
18    // %lf for scanf
19    // 1.797693e+308
20    printf("DBL_MAX = %e\n", DBL_MAX);
21    // 2.225074e-308
22    printf("DBL_MIN = %e\n", DBL_MIN);
23    // 4.940656e-324
24    printf("DBL_TRUE_MIN = %e\n", DBL_TRUE_MIN);
25    // 2.220446e-16
26    printf("DBL_EPSILON = %e\n\n", DBL_EPSILON);
27
28    return 0;
29 }
```

```
1 // Created by hfwei on 2024/10/31.
2
3 #include <stdio.h>
4 #define LEN 100
5
6 int main(void) {
7     int numbers[LEN] = {0};
8
9     for (unsigned int i = LEN; i >= 0; i--) {
10         printf("%u : %d\n", i, numbers[i]);
11     }
12
13     return 0;
14 }
```

```
1 // Created by hfwei on 2024/10/31.
2
3 #include <stdbool.h>
4 #include <stdio.h>
5 #include <time.h>
6
7 bool IsPrime(int number);
8
9 int main(void) {
10     int max = 0;
11     scanf("%d", &max);
12
13     int count = 0;
14
15     // TODO: return the current time in seconds since the
16     // Unix epoch (January 1, 1970)
17
18     for (int number = 2; number <= max; number++) {
19         if (IsPrime(number)) {
20             count++;
21         }
22     }
23     printf("\ncount = %d\n", count);
24
25     // TODO: return the current time in seconds since the
26     // Unix epoch (January 1, 1970)
27
28     return 0;
29 }
30
31 bool IsPrime(int number) {
32     for (int factor = 2; factor * factor <= number; factor
33         ++ ) {
34         if (number % factor == 0) {
35             return false;
36         }
37     }
38
39     return true;
40 }
```

```
1 // Created by hfwei on 2024/10/30.
2
3 #include <limits.h>
4 #include <stdio.h>
5
6 int main() {
7     printf("UINT_MAX = %u\n", UINT_MAX);
8
9     unsigned int max = UINT_MAX;
10    unsigned int one = 1U;
11    unsigned int two = 2U;
12
13    printf("max + one = %u\n", max + one);
14    printf("one - two = %u\n", one - two);
15
16    return 0;
17 }
```



```
1 // Created by hfwei on 2024/10/31.
2
3 #include <stdio.h>
4
5 int main(void) {
6     signed char left = 100;
7     signed char mid = 3;
8     signed char right = 4;
9
10    signed char result = left * mid / right;
11
12    printf("result = %d\n", result);
13
14    return 0;
15 }
```

```
1 // Created by hfwei on 2024/10/31.
2
3 #include <limits.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 unsigned int Add(unsigned int left, unsigned int right);
8 unsigned int Sub(unsigned int left, unsigned int right);
9 unsigned int Mul(unsigned int left, unsigned int right);
10 unsigned int Div(unsigned int left, unsigned int right);
11 unsigned int Mod(unsigned int left, unsigned int right);
12
13 int main(void) {
14     // addition
15     unsigned int left_add = UINT_MAX / 2 + 1;
16     unsigned int right_add = UINT_MAX / 2 + 1;
17
18     printf("%u + %u = %u\n\n", left_add, right_add, Add(
19         left_add, right_add));
20
21     // subtraction
22     unsigned int left_sub = 1;
23     unsigned int right_sub = 2;
24
25     printf("%u - %u = %u\n\n", left_sub, right_sub, Sub(
26         left_sub, right_sub));
27
28     // multiplication
29     unsigned int left_mul = UINT_MAX;
30     unsigned int right_mul = 2;
31
32     printf("%u * %u = %u\n", left_mul, right_mul, Mul(
33         left_mul, right_mul));
34
35     // division
36     unsigned int left_div = 5;
37     unsigned int right_div = 0;
38
39     printf("%u * %u = %u\n", left_div, right_div, Div(
40         left_div, right_div));
41     printf("%u * %u = %u\n", left_div, right_div, Mod(
42         left_div, right_div));
43
44     return 0;
45 }
```

```
40 }
41
42 unsigned int Add(unsigned int left, unsigned int right) {
43     return left + right;
44 }
45
46 unsigned int Sub(unsigned int left, unsigned int right) {
47     return left - right;
48 }
49
50 unsigned int Mul(unsigned int left, unsigned int right) {
51     return left * right;
52 }
53
54 unsigned int Div(unsigned int left, unsigned int right) {
55     return left / right;
56 }
57
58 unsigned int Mod(unsigned int left, unsigned int right) {
59     return left % right;
60 }
61
62 // Add:
63 // if (left + right > UINT_MAX) {
64 //     printf("Too Big!\n");
65 //     exit(1);
66 // } else {
67 //     return left + right;
68 // }
69
70 // Sub:
71 // if (left - right < 0) {
72 //     printf("The result is negative!\n");
73 //     exit(1);
74 // } else {
75 //     return left - right;
76 // }
77
78 // Mul:
79 // if (left * right > UINT_MAX) {
80 //     printf("The result is negative!\n");
81 //     exit(1);
82 // } else {
83 //     unsigned int mul = left * right;
```

```
84 //     return mul;
85 // }
86
87 // Div:
88 // if (right == 0) {
89 //     printf("Division by zero!\n");
90 //     exit(1);
91 // }
92 //
93 // return left / right;
94
95 // Mod:
96 // if (right == 0) {
97 //     printf("Division by zero!\n");
98 //     exit(1);
99 // }
100 //
101 // return left % right;
```

```
1 // Created by hfwei on 2024/10/30.
2
3 #include <limits.h>
4 #include <stdio.h>
5
6 int main() {
7     double pi = 3.14159;
8
9     // to obtain its fractional part
10    double fraction = 0;
11
12    // to compute num * num
13    int num = 100000000; // (8 zeros)
14
15    printf("LLONG_MAX = %lld\n", LLONG_MAX);
16    long long llint = num * num;
17    printf("i = %lld\n", llint);
18
19    return 0;
20 }
```

```
1 // Created by hfwei on 2024/10/30.
2
3 #include <limits.h>
4 #include <stdio.h>
5
6 int SquareInt(int num);
7 double SquareDouble(double num);
8
9 int main() {
10     // narrowing conversion (still in the range)
11     int i = 3.14159;
12
13     // out of the range: undefined behavior!!!
14     int j = UINT_MAX;
15
16     // arguments; narrowing conversion
17     double pi = 3.14;
18     SquareInt(pi);
19
20     // return value; narrowing conversion
21     int val = SquareDouble(pi);
22
23     // from int to float; narrowing conversion
24     int big = 1234567890;
25     float approx = big;
26
27     printf("big = %d\t approx = %f\t diff = %d\n", big,
28           approx,
29           big - (int)approx);
30     return 0;
31 }
32
33 int SquareInt(int num) { return num * num; }
34
35 double SquareDouble(double num) { return num * num; }
```

```
1 // Created by hfwei on 2024/10/31.
2
3 #include <limits.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 int Add(int left, int right);
8 int Sub(int left, int right);
9 int Mul(int left, int right);
10 int Div(int left, int right);
11 int Mod(int left, int right);
12 int Neg(int left);
13
14 int main(void) {
15     // addition
16     int left_add = INT_MAX / 2 + 1;
17     int right_add = INT_MAX / 2 + 1;
18
19     printf("%d + %d = %d\n\n", left_add, right_add, Add(
20         left_add, right_add));
21
22     // subtraction
23     int left_sub = INT_MIN;
24     int right_sub = 1;
25
26     printf("%d - %d = %d\n\n", left_sub, right_sub, Sub(
27         left_sub, right_sub));
28
29     // multiplication
30     int left_mul = INT_MAX;
31     int right_mul = 2;
32
33     printf("%d * %d = %d\n", left_mul, right_mul, Mul(
34         left_mul, right_mul));
35
36     // division
37     int left_div = INT_MIN;
38     int right_div = -1;
39
40     printf("%d / %d = %d\n", left_div, right_div, Div(
41         left_div, right_div));
42
43     // mod (remainder)
44     int left_mod = INT_MIN;
```

```
41  int right_mod = -1;
42
43  printf("%d %% %d = %d\n", left_mod, right_mod, Mod(
    left_mod, right_mod));
44
45  // negation
46  int left_neg = INT_MIN;
47
48  printf("-%d = %d\n", left_neg, Neg(left_neg));
49
50  return 0;
51 }
52
53 int Add(int left, int right) {
54     return left + right;
55 }
56
57 int Sub(int left, int right) {
58     return left - right;
59 }
60
61 int Mul(int left, int right) {
62     return left * right;
63 }
64
65 int Div(int left, int right) {
66     return left / right;
67 }
68
69 int Mod(int left, int right) {
70     return left % right;
71 }
72
73 int Neg(int left) {
74     return -left;
75 }
76
77 // Add:
78 // if ((left > 0 && right > INT_MAX - left) ||
79 //     (left < 0 && right < INT_MIN - left)) {
80 //     printf("Overflow!\n");
81 //     exit(1);
82 // } else {
83 //     return left + right;
```



```
84 // }
85
86 // Sub
87 //if ((left > 0 && right < INT_MIN + left) ||
88 //(left < 0 && right > INT_MAX + left)) {
89 //printf("Overflow!\n");
90 //exit(1);
91 //} else {
92 //int sub = left - right;
93 //return sub;
94 //}
95
96 // Mul
97 // if (left > 0) {
98 //     if (right > 0) { // left > 0 && right > 0
99 //         if (left > INT_MAX / right) {
100 //             printf("Overflow!\n");
101 //             exit(1);
102 //         }
103 //     } else { // left > 0 && right < 0
104 //         if (right < INT_MIN / left) {
105 //             printf("Overflow!\n");
106 //             exit(1);
107 //         }
108 //     }
109 // } else { // left <= 0
110 //     if (right > 0) { // left <= 0 && right > 0
111 //         if (left < INT_MIN / right) {
112 //             printf("Overflow!\n");
113 //             exit(1);
114 //         }
115 //     } else { // left <= 0 && right <= 0
116 //         if (left != 0 && right < INT_MAX / left) {
117 //             printf("Overflow!\n");
118 //             exit(1);
119 //         }
120 //     }
121 // }
122 //
123 // int mul = left * right;
124 // return mul;
125
126 // Div
127 // if (right == 0 || (left == INT_MIN && right == -1)) {
```

```
128 //    printf("Overflow!\n");
129 //    exit(1);
130 // }
131 //
132 // return left / right;
133
134 // Mod
135 // if (right == 0 || (left == INT_MIN && right == -1)) {
136 //     printf("Overflow!\n");
137 //     exit(1);
138 // }
139 //
140 // return left % right;
141
142 // Neg
143 //if (left == INT_MIN) {
144 //printf("Overflow!\n");
145 //exit(1);
146 //}
147 //
148 //return -left;
```