

```
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  //
2  // Created by hfwei on 2023/11/15.
3  //
4
5  #include <stdio.h>
6
7  #define LEN 7
8
9  /**
10 * @brief Sort numbs[left .. right] using merge sort.
11 * @param nums
12 * @param left
13 * @param right
14 */
15 void MergeSort(int nums[], int left, int right);
16
17 /**
18 * @brief Merge nums[left .. mid] and nums[mid + 1 .. right]
19 * @param nums
20 * @param left
21 * @param mid
22 * @param right
23 */
24 void Merge(int nums[], int left, int mid, int right);
25
26 /**
27 * @brief Copy src[left .. right] to dest[left .. right]
28 * @param src
29 * @param dest
30 * @param left
31 * @param right
32 */
33 void Copy(const int src[], int dest[], int left, int right);
34
35 int main() {
36     int numbers[LEN] = {38, 27, 43, 3, 9, 82, 10};
37     MergeSort(numbers, 0, LEN - 1);
38
39     for (int i = 0; i < LEN; i++) {
40         printf("%d ", numbers[i]);
41     }
42
43     return 0;
44 }
45
46 void MergeSort(int nums[], int left, int right) {
47     if (left == right) {
48         return;
49     }
50
51     int mid = (left + right) / 2;
52     MergeSort(nums, left, mid);    // Call the Mirror
53     MergeSort(nums, mid + 1, right); // Call the Mirror
```

```
54
55 Merge(nums, left, mid, right);
56 }
57
58 void Merge(int nums[], int left, int mid, int right) {
59     static int copy[LEN] = {0};
60
61     int left_index = left;
62     int right_index = mid + 1;
63     int copy_index = left;
64
65     while (left_index <= mid && right_index <= right) {
66         if (nums[left_index] <= nums[right_index]) {
67             copy[copy_index] = nums[left_index];
68             left_index++;
69         } else {
70             copy[copy_index] = nums[right_index];
71             right_index++;
72         }
73
74         copy_index++;
75     }
76
77     while (left_index <= mid) {
78         copy[copy_index] = nums[left_index];
79         left_index++;
80         copy_index++;
81     }
82
83     while (right_index <= right) {
84         copy[copy_index] = nums[right_index];
85         right_index++;
86         copy_index++;
87     }
88
89     Copy(copy, nums, left, right);
90 }
91
92 void Copy(const int src[], int dest[], int left, int right) {
93     for (int i = left; i <= right; ++i) {
94         dest[i] = src[i];
95     }
96 }
```

```
1 // file: limits.h
2
3 #include <stdio.h>
4 #include <limits.h>
5
6 int main() {
7     printf("INT_MIN = %d\n", INT_MIN);
8     printf("INT_MAX = %d\n\n", INT_MAX);
9
10    printf("UINT_MIN = %u\n", 0U);
11    printf("UINT_MAX = %u\n\n", UINT_MAX);
12
13    printf("LONG_MIN = %ld\n", LONG_MIN);
14    printf("LONG_MAX = %ld\n\n", LONG_MAX);
15
16    printf("ULONG_MIN = %lu\n", 0UL);
17    printf("ULONG_MAX = %lu\n\n", ULONG_MAX);
18
19    // long long int: >= 64 bits
20    printf("LLONG_MIN = %lld\n", LLONG_MIN);
21    printf("LLONG_MAX = %lld\n\n", LLONG_MAX);
22
23    printf("ULONG_LONG_MIN = %llu\n", 0ULL);
24    printf("ULONG_LONG_MAX = %llu\n\n", ULONG_LONG_MAX);
25
26    return 0;
27 }
```

```
1 //
2 // Created by hfwei on 2022/11/10.
3 //
4
5 #include <stdio.h>
6
7 int main() {
8     int array[] = {0, 1, 2, 3, 4};
9     int i = -1;
10
11     size_t size = sizeof array;
12     printf("The size of the array is %zu\n", size);
13
14     if (i <= size) {
15         printf("i <= sizeof array\n");
16     } else {
17         printf("i > sizeof array\n");
18     }
19
20     return 0;
21 }
```

```
1 //
2 // Created by hfwei on 2022/11/10.
3 //
4
5 #include <stdio.h>
6 #include <time.h>
7
8 long long Fib(int n);
9
10 int main() {
11     int n;
12     scanf("%d", &n);
13
14     printf("Fib(%d) = %lld\n", n, Fib(n));
15
16     return 0;
17 }
18
19 long long Fib(int n) {
20     if (n <= 1) {
21         return n;
22     }
23
24     return Fib(n - 1) + Fib(n - 2);
25 }
```

File - D:\cpl\2023-cpl-coding-0\7-data-types\char.c

```
1 //
2 // Created by hfwei on 2022/11/10.
3 //
4
5 #include <stdio.h>
6
7 int main() {
8     // (signed) char on my computer: -128 ~ 127
9     // using unsigned char c = 150;
10    unsigned char c = 150;
11    int i = 900;
12
13    printf("i / c = %d\n", i / c);
14
15    return 0;
16 }
```

```
1 //
2 // Created by hfwei on 2022/11/10.
3 //
4
5 #include <stdio.h>
6 #include <limits.h>
7 int main() {
8     printf("UINT_MAX = %u\n", UINT_MAX);
9
10    unsigned int max = UINT_MAX;
11    unsigned int one = 1U;
12    unsigned int two = 2U;
13
14    printf("max + one = %u\n", max + one);
15    printf("one - two = %u\n", one - two);
16
17    return 0;
18 }
```



```
1 //
2 // Created by hfwei on 2022/11/10.
3 //
4
5 #include <limits.h>
6 #include <stdio.h>
7
8 int SquareInt(int num);
9 double SquareDouble(double num);
10
11 int main() {
12     // narrowing conversion (still in the range)
13
14     // out of the range: undefined behavior!!!
15
16     // arguments; narrowing conversion
17
18     // return value; narrowing conversion
19
20     return 0;
21 }
22
23 int SquareInt(int num) {
24     return num * num;
25 }
26
27 double SquareDouble(double num) {
28     return num * num;
29 }
```

```
1 //
2 // Created by hfwei on 2022/11/10.
3 //
4
5 #include <stdio.h>
6 #include <limits.h>
7
8 int main() {
9     double pi = 3.14159;
10
11     // below: obtain its fractional part
12     double fraction = pi - (int) pi;
13
14     int num = 1000000000;
15     printf("LLONG_MAX = %lld\n", LLONG_MAX);
16     // long long llint = num * num;
17     // long long llint = (long long) num * num;
18     long long llint = (long long) (num * num);
19
20     printf("llint = %lld\n", llint);
21
22     return 0;
23 }
```

```
1 //
2 // Created by hfwei on 2022/11/9.
3 //
4
5 #include <stdio.h>
6 #include <float.h>
7
8 int main() {
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  /**
2   * file: sums.c
3   * See https://randomascii.wordpress.com/2012/02/25/comparing-floating-point-numbers-2012-edition/
4   *
5   * Created by hengxin on 11/21/21.
6   */
7
8  #include <stdio.h>
9
10 int main() {
11     // 0.1: 0.0 0011 0011 0011
12     float f = 0.1F;
13     float sum = 0.0F;
14
15     for (int i = 0; i < 10; ++i) {
16         sum += f;
17     }
18
19     float product = f * 10;
20
21     printf("sum = %.15f\nmul = %.30f\n",
22           sum, product);
23
24     return 0;
25 }
```

```
1  /**
2   * file: loop.c
3   *
4   * Created by hengxin on 11/21/21.
5   */
6
7  #include <stdio.h>
8
9  int main() {
10     /**
11      * Do not use a counter of type float/double,
12      * although it works on some platforms.
13      *
14      * 0.1 cannot be exactly represented in machines.
15      */
16     for (double x = 0.1; x <= 1.0; x += 0.1) {
17         printf("%.20f\n", x);
18     }
19
20     return 0;
21 }
```

```
1 /**
2  * file: compare.c
3  *
4  * See https://randomascii.wordpress.com/2012/02/25/comparing-floating-point-numbers-2012-edition/
5  *
6  * Created by hfwei on 2022/11/10.
7  */
8
9 #include <float.h>
10 #include <math.h>
11 #include <stdio.h>
12 #include <stdbool.h>
13
14 bool IsEqual(double x, double y);
15
16 int main() {
17     printf("%d\n", IsEqual(DBL_MAX, DBL_MAX - 100));
18
19     printf("%.50f\n", DBL_MAX - (DBL_MAX - 100));
20
21     return 0;
22 }
23
24 bool IsEqual(double x, double y) {
25     return fabs(x - y) <= DBL_EPSILON;
26 }
```

```
1 //
2 // Created by hfwei on 2022/11/10.
3 //
4
5 #include <stdio.h>
6 #include <time.h>
7
8 long long Fib(int n);
9
10 int main() {
11     int n;
12     scanf("%d", &n);
13
14     time_t start = time(NULL);
15
16     printf("Fib(%d) = %lld\n", n, Fib(n));
17
18     time_t end = time(NULL);
19     printf("Time elapsed: %lld seconds\n", (long long) end - start);
20
21     return 0;
22 }
23
24 long long Fib(int n) {
25     if (n <= 1) {
26         return n;
27     }
28
29     return Fib(n - 1) + Fib(n - 2);
30 }
```

```
1 //
2 // Created by hfwei on 2022/11/10.
3 //
4
5 #include <limits.h>
6 #include <stdio.h>
7
8 int SquareInt(int num);
9 double SquareDouble(double num);
10
11 int main() {
12     // narrowing conversion (still in the range)
13     int i = 3.14159;
14
15     // out of the range: undefined behavior!!!
16     int j = UINT_MAX;
17
18     // arguments; narrowing conversion
19     double k = 3.14159;
20     SquareInt(k);
21
22     // return value; narrowing conversion
23     int m = SquareDouble(k);
24
25     int big = 1234567890;
26     float approx = big;
27     int approx_big = (int) approx;
28     printf("big = %d\t approx = %f\t approx_big = %d\t diff = %d\n",
29           big, approx, approx_big, big - (int) approx);
30
31     return 0;
32 }
33
34 int SquareInt(int num) {
35     return num * num;
36 }
37
38 double SquareDouble(double num) {
39     return num * num;
40 }
```