# Explicit Cast(all detected)

- Test File：cast.c

```
#include "stdlib.h"
#include "stdint.h"
#include "stdio.h"
#include "limits.h"
int main()
{
        printf("Test int INT32_MAX to short conversion with value loss\n");
        int a=INT32_MAX;
        short c=(short)a;
        printf("Test long long INT64_MAX to int conversion with value loss\n");
        long long along=INT64_MAX;
        int cint=(int)along;
        printf("Test signed(negative) to unsigned cast\n");
        int sint=-1;
        unsigned int uint=(unsigned int)sint;
        printf("Test unsigned UINT_MAX to int\n");
        unsigned u=UINT_MAX;
        int utoi=(int)u;
        printf("Test int INT32_MAX to char\n");
        char ch=(char)INT32_MAX;
}
```

- Cast Type(all with lost values):
  - Int to short
  - Int to char
  - Long long to int
  - Unsigned int to signed int
  - Signed int to unsigned int
- Result: **all detected**

```
qiqi@qiqi-HP-Pavilion-dm1-Notebook-PC:~/Project/qiqi_test$ clang -fioc-explicit-conversion cast.c -o cast
qiqi@qiqi-HP-Pavilion-dm1-Notebook-PC:~/Project/qiqi_test$ ./cast
Test int INT32_MAX to short conversion with value loss
cast.c:9:17: runtime error: value lost in conversion of '2147483647' from 'int' (int) to 'short' (short)
Test long long INT64_MAX to int conversion with value loss
cast.c:12:16: runtime error: value lost in conversion of '9223372036854775807' from 'long long' (long long) to 'int' (int)
Test signed(negative) to unsigned cast
cast.c:15:34: runtime error: value lost in conversion of '-1' from 'int' (int) to 'unsigned int' (unsigned int)
```

Test unsigned UINT_MAX to int

cast.c:18:16: runtime error: value lost in conversion of '4294967295' from 'unsigned int' (unsigned int) to 'int' (int)

Test int INT32_MAX to char

cast.c:20:16: runtime error: value lost in conversion of '2147483647' from 'int' (int) to 'char' (char)

# Implicit Cast(one of them is not detected)

- Test File：implict_cast.c

```
#include "stdlib.h"
#include "stdint.h"
#include "stdio.h"
#include "limits.h"
#include "string.h"
int main()
{
        printf("Implicit Test:\n");
        printf("Test int INT32_MAX to short conversion with value loss\n");
        int a=INT32_MAX;
        short c=a;
        printf("Test int to char\n");
        char chc=a;
        printf("Test implicit long long to int conversion with value loss\n");
        long long along=INT64_MAX;
        int cint=along;
        printf("Test signed to unsigned cast\n");
        int sint=-1;
        unsigned int uint=sint;
        printf("Test unsigned to signed cast\n");
        unsigned int u=UINT_MAX;
        int utos=(int)u;
        printf("Test malloc error:\n");
        int amalloc=-111;
        char * ch=(char *)malloc(sizeof(char)*amalloc);
        char * cp,*cp1;
        cp=(char *)malloc(sizeof(char)*INT32_MAX);
        cp1=(char *)malloc(sizeof(char)*INT32_MAX);
        if(cp==NULL)
        {
                printf("mallocerror\n");
        }
        if(cp1==NULL)
        {
```

```
                printf("mallocerror1\n");
        }
        printf("Test strncpy:\n");
        int scpy=-123;
        strncpy(cp1,cp,scpy);

}
```

- Cast Type(all with lost values):
  - Int to short
  - Int to char
  - Long long to int
  - Unsigned int to signed int
  - Signed int to unsigned int
  - Malloc error
  - Strncpy error
- Result： **all detected except(implicit unsigned to sign)**

Implicit Test:
Test int INT32_MAX to short conversion with value loss
implicit_cast.c:11:10: runtime error: value lost in conversion of '2147483647' from 'int' (int) to 'short' (short)
Test int to char
implicit_cast.c:13:11: runtime error: value lost in conversion of '2147483647' from 'int' (int) to 'char' (char)
Test implicit long long to int conversion with value loss
implicit_cast.c:16:11: runtime error: value lost in conversion of '9223372036854775807' from 'long long' (long long) to 'int' (int)
Test signed to unsigned cast
implicit_cast.c:19:20: runtime error: value lost in conversion of '-1' from 'int' (int) to 'unsigned int' (unsigned int)
Test unsigned to signed cast
Test malloc error:
implicit_cast.c:25:40: runtime error: value lost in conversion of '-111' from 'int' (int) to 'unsigned int' (unsigned int)//this is malloc error
mallocerror1
Test strncpy:
implicit_cast.c:39:17: runtime error: value lost in conversion of '-123' from 'int' (int) to 'size_t' (unsigned int)//this is strncpy error(here will result in stack overflow, but it is not caused by ioc-clang)

# Implicit Cast(memset)

- Test File：memset.c

```
#include "stdlib.h"
#include "stdint.h"
#include "stdio.h"
#include "limits.h"
#include "string.h"
int main()
{
        printf("Implicit Memset Cast Test:\n");
        char * cp,*cp1;
        cp=(char *)malloc(sizeof(char)*10000);
        if(cp==NULL)
        {
                printf("mallocerror\n");
        }
        if(cp1==NULL)
        {
                printf("mallocerror1\n");
        }
        printf("Test memset:\n");
        int scpy=-124;
        memset(cp1,'a',scpy);

}
```

- Cast Type: Memset
- Result: **Detected**

  Test memset:

  memset.c:21:17: runtime error: value lost in conversion of '-124' from 'int' (int) to 'size_t' (unsigned int)

# Implicit Cast(memcpy)

- Test File：memcpy.c

```
#include "stdlib.h"
#include "stdint.h"
#include "stdio.h"
#include "limits.h"
#include "string.h"
int main()
{
```

```
        char * cp,*cp1;
        cp=(char *)malloc(sizeof(char)*INT32_MAX);
        cp1=(char *)malloc(sizeof(char)*INT32_MAX);
        if(cp==NULL)
        {
                printf("mallocerror\n");
        }
        if(cp1==NULL)
        {
                printf("mallocerror1\n");
        }
        printf("Test strncpy:\n");
        int scpy=-123;
        memcpy(cp1,cp,scpy);
}
```

- Cast Type: Memcpy
- Result: **Detected**

Test strncpy:

memcpy.c:21:16: runtime error: value lost in conversion of '-123' from 'int' (int) to 'size_t' (unsigned int)

# Overflows:

- Test File：flow.c

```
#include "stdlib.h"
#include "stdint.h"
#include "stdio.h"
#include "limits.h"
int main()
{
        printf("Test Integer Underflow\n");
        int a1=INT32_MIN;
        int b1=a1-1;


        printf("Test Integer Overflow:\n");
        int a=INT32_MAX;
        int b=a+1;

        printf("Test unsigned underflow\n");
        unsigned c1=0;
        unsigned c11=c1-1;
```

```
        printf("Test unsigned overflow\n");
        unsigned cd1=UINT_MAX;
        unsigned cd2=cd1+1;
}
```

- Test Type:
  - Signed overflow
  - Signed underflow
  - Unsigned overflow
  - Unsigned underflow
- Result: ALL DETECTED

```
qiqi@qiqi-HP-Pavilion-dm1-Notebook-PC:~/Project/qiqi_test$ clang -fioc-signed -f
ioc-unsigned flow.c -o flow
qiqi@qiqi-HP-Pavilion-dm1-Notebook-PC:~/Project/qiqi_test$ ./flow
Test Integer Underflow
flow.c:9:18: runtime error: signed subtraction overflow [ expr = '-', lval = (sint32) -2147483648,
rval = (sint32) 1 ]
Test Integer Overflow:
flow.c:14:16: runtime error: signed addition overflow [ expr = '+', lval = (sint32) 2147483647, rval
= (sint32) 1 ]
Test unsigned underflow
flow.c:18:17: runtime error: unsigned subtraction overflow [ expr = '-', lval = (uint32) 0, rval =
(uint32) 1 ]
Test unsigned overflow
flow.c:22:18: runtime error: unsigned addition overflow [ expr = '+', lval = (uint32) 4294967295,
rval = (uint32) 1 ]
```

# Divide by zero:

Exception will generated, but not caused by IOC(by clang or else?)

# Shift more than bitwidth:

- Test File:

```
#include "stdlib.h"
#include "stdint.h"
#include "stdio.h"
#include "limits.h"
int main()
{
        int a=1;
        printf("shift an integer by 32 and 33 bits\n");
        int b=a<<32;
```

```
        int c=a<<33;
        printf("shift an unsigned integer by 32 and 33 bits\n");
        unsigned int a1=1;
        unsigned int b1=a1<<32;
        unsigned int c1=a1<<33;
        printf ("shift a char by 33 and 17\n");
        char ch='a';
        char ch1=ch<<33;
        printf("%c\n",ch1);
        char ch2=ch<<17;
        printf("%c\n",ch2);
        printf("shift a short by 32\n");
        short s=-1;
        short s2=s<<32;
        return 0;
}
```

Test Type: shift more than bitwidth

Test Result: Detected(clang will detect it when compiling, IOC will also dynamically detect it at runtime)

```
qiqi@qiqi-HP-Pavilion-dm1-Notebook-PC:~/Project/qiqi_test$ clang -fioc-shifts shift.c -o shift
shift.c:9:9: warning: shift count >= width of type [-Wshift-count-overflow]
        int b=a<<32;
              ^ ~~
shift.c:10:9: warning: shift count >= width of type [-Wshift-count-overflow]
        int c=a<<33;
              ^ ~~
shift.c:13:20: warning: shift count >= width of type [-Wshift-count-overflow]
        unsigned int b1=a1<<32;
                        ^ ~~
shift.c:14:20: warning: shift count >= width of type [-Wshift-count-overflow]
        unsigned int c1=a1<<33;
                        ^ ~~
shift.c:17:13: warning: shift count >= width of type [-Wshift-count-overflow]
        char ch1=ch<<33;
                 ^ ~~
shift.c:23:12: warning: shift count >= width of type [-Wshift-count-overflow]
        short s2=s<<32;
                 ^ ~~
6 warnings generated.
qiqi@qiqi-HP-Pavilion-dm1-Notebook-PC:~/Project/qiqi_test$ ./shift
shift an integer by 32 and 33 bits
shift.c:9:9: runtime error: left shift by amount >= bitwidth [ expr = '<<', lval = (sint32) 1, rval = (sint32) 32 ]
```

```
shift.c:10:9: runtime error: left shift by amount >= bitwidth [ expr = '<<', lval = (sint32) 1, rval = (sint32) 33 ]
shift an unsigned integer by 32 and 33 bits
shift.c:13:20: runtime error: left shift by amount >= bitwidth [ expr = '<<', lval = (uint32) 1, rval = (uint32) 32 ]
shift.c:14:20: runtime error: left shift by amount >= bitwidth [ expr = '<<', lval = (uint32) 1, rval = (uint32) 33 ]
shift a char by 33 and 17
shift.c:17:13: runtime error: left shift by amount >= bitwidth [ expr = '<<', lval = (sint32) 97, rval = (sint32) 33 ]
w

shift a short by 32
shift.c:23:12: runtime error: left shift by amount >= bitwidth [ expr = '<<', lval = (sint32) -1, rval = (sint32) 32 ]
```

# Strict Shift:

- Test File:undefined_shift.c

```
#include "stdint.h"
#include "stdio.h"
#include "limits.h"
int main()
{
        int a=1;
        printf("integer shift by an negative number:\n");
        a<<-1;

}
```

- Test Type: undefined shift behavior in C99
- Test Result:

```
qiqi@qiqi-HP-Pavilion-dm1-Notebook-PC:~/Project/qiqi_test$ clang -fioc-strict-shifts undefined_shift.c -o undefined_shift
undefined_shift.c:9:3: warning: expression result unused [-Wunused-value]
        a<<-1;
        ~^ ~~
undefined_shift.c:9:3: warning: shift count is negative [-Wshift-count-negative]
        a<<-1;
          ^ ~~
2 warnings generated.
qiqi@qiqi-HP-Pavilion-dm1-Notebook-PC:~/Project/qiqi_test$ ./undefined_shift
integer shift by an negative number:
undefined_shift.c:9:3: runtime error: left shift by negative amount [ expr = '<<', lval = (sint32) 1,
```

rval = (sint32) -1 ]
undefined_shift.c:9:3: runtime error: left shift into or beyond sign bit [ expr = '<<', lval = (sint32) 1,
rval = (sint32) -1 ]