Лекция 9. Обработка ошибок.

НИЖЕГОРОДСКИЙ ИНСТИТУТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
НИИТ

Типы ошибок

- **Software Errors**

(классические ошибки программирования)

- **Run-Time Errors**

(ошибки связанные с нехваткой или недоступностью ресурсов ОС)

Традиционные техники

1. прекратить выполнение
2. возвратить значение «ошибка»
3. возвратить допустимое значение и оставить программу в ненормальном состоянии
4. вызвать функцию обработки ошибок

Вариант 1. Прекратить выполнение.

```
#include <cassert>;

char Stack::pop()
{
  assert( top != 0 );
  return store[--top];
}
```

```
#ifdef NDEBUG
#define assert(exp)
#else
#define assert(exp)
        ((exp) ? 0 : __assert_fail(__STRING(expr),
                                   __FILE__,
                                   __LINE__))
#endif
```

Результат работы assert

```
stack_assert: simple_stack.cpp:61:
char Stack::pop (): Assertion 'top != 0' failed.
Abort (core dumped)
```

```
(gdb) where
#0  0x400b8b01 in __kill () from /lib/i686/libc.so.6
#1  0x400b88da in raise (sig=6) at ../sysdeps/posix/raise.c:27
#2  0x400ba082 in abort () at ../sysdeps/generic/abort.c:88
#3  0x400b2220 in __assert_fail () at assert.c:74
#4  0x0804a22d in Stack::pop ()
#5  0x0804a49a in Stack_Interface::pop_and_print_char ()
#6  0x0804a713 in main ()
#7  0x400a6647 in __libc_start_main (main=0x804a59c <main>, argc=1,
```

Вариант 2. Возвратить «ошибку».

```
char Stack::pop() {
  return top ? store[--top] : 0;
}
```

```
enum tRC {
  OK,
  BAD_SIZE,
  OVERFLOW,
  UNDERFLOW
};
```

```
tRC Stack::pop(char* c)
{
  if (!top) return UNDERFLOW;
  *c=store[--top];
  return OK;
}
```

```
char Stack::pop(tRC* rc) {
  if (top) {
    *rc=OK;
    return store[--top];
  }
  *rc=UNDERFLOW;
  return 0;
}
```

Вариант 2. Возвратить «ошибку».

- Может не быть подходящего значения
- Результат каждого вызова должен проверяться на «ошибку»

Вариант 3. Оставить программу в ненормальном состоянии.

```
// error_handle.h
enum tError {
  OK,
  BAD_SIZE,
  OVERFLOW,
  UNDERFLOW,
  ...
} g_Error;
```

```
// stack.h
class Stack {
  Stack();
  ~Stack();
  void  push(char);
  char  pop();
  // ...
};
```

```
// stack.cpp
#include "error_handle.h"

char Stack::pop() {
  if (top!=0) {
   return store[--top];
  }
  g_Error=UNDERFLOW;
  return 0;
}
```

Вариант 3. Оставить программу в ненормальном состоянии.

```
#include "stack.h"
#include "error_handle.h"

int main()
{
  Stack stack;
  char c = stack.pop();
  if (g_Error != OK)  {
    // error occured
  }
  else  {
    // OK
  }
}
```

Вызывающая функция может не заметить ненормального состояния

Вариант 4. Вызвать функцию обработки ошибок.

```
void* new (size_t size)
{
  for(;;)
  {
    if (void* p = malloc(size))
      return p;
    if (!find_memory_somewhere())
      return 0;
  }
}
```

У функции обработки ошибок есть только три первые альтернативы, как обрабатывать ошибку

Конец