Лекция 7

1 марта

Оператор do-while

```
int pcount_do(unsigned x) {
  int result = 0;
  do {
    result += x & 0x1;
    x >>= 1;
  } while (x);
  return result;
}
```

```
int pcount_do(unsigned x) {
  int result = 0;
loop:
  result += x & 0x1;
  x >>= 1;
  if (x)
    goto loop;
  return result;
}
```

Оператор do-while

Регистр	Значение
edx	**x**
ecx	**result**

```
int pcount_do(unsigned x) {
  int result = 0;
loop:
  result += x & 0x1;
  x >>= 1;
  if (x)
    goto loop;
  return result;
}
```

```
   mov ecx, 0     ;   result = 0
.L2:              ; loop:
   mov eax, edx
   and eax, 1     ;   t = x & 1
   add ecx, eax   ;   result += t
   shr edx, 1     ;   x >>= 1
   jne .L2        ;   If !0, goto loop
```

Оператор while

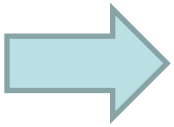
```
int pcount_while(unsigned x) {
  int result = 0;
  while (x) {
    result += x & 0x1;
    x >>= 1;
  }
  return result;
}
```

```
int pcount_do(unsigned x) {
  int result = 0;
loop:
  if (!x) goto done;
  result += x & 0x1;
  x >>= 1;
  goto loop;
done:
  return result;
}
```

```
int pcount_do(unsigned x) {
  int result = 0;
  if (!x) goto done;
loop:
  result += x & 0x1;
  x >>= 1;
  if (x)
    goto loop;
done:
  return result;
}
```

Оператор for

```
#define WSIZE 8*sizeof(int)

int pcount_for(unsigned x) {
  int i;
  int result = 0;
  for (i = 0; i < WSIZE; i++) {
    unsigned mask = 1 << i;
    result += (x & mask) != 0;
  }
  return result;
}
```



```
int pcount_for_gt(unsigned x) {
  int i;
  int result = 0;
  i = 0;
  if (!(i < WSIZE))
    goto done;
 loop:
  {
    unsigned mask = 1 << i;
    result += (x & mask) != 0;
  }
  i++;
  if (i < WSIZE)
    goto loop;
 done:
  return result;
}
```

```c
int fib(int x) { // x >= 1
  int i;
  int p_pred = 0;
  int pred = 1;
  int res = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = p_pred + pred;
     p_pred = pred;
     pred = res;
  }
  return res;
}
```

Регистр	Значение
ecx	**x**
edx	**p_pred**
ebx	**pred**
eax	**res**

```
fib:
   push  ebp
   mov   ebp, esp
   push  ebx

   mov   ecx, dword [ebp + 8] ; x
   xor   edx, edx             ; p_pred
   mov   ebx, 1               ; pred
   mov   eax, 1               ; res
   dec   ecx

   jecxz .end
.loop:
   lea   eax, [edx + ebx]
   mov   edx, ebx
   mov   ebx, eax
   loop .loop

.end:
   pop ebx
   pop ebp
   ret
```

```c
int fib(int x) { // x >= 1
  int i;
  int p_pred = 0;
  int pred = 1;
  int res = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = p_pred + pred;
     p_pred = pred;
     pred = res;
  }
  return res;
}
```

Регистр	Значение
ecx	x
edx	p_pred
ebx	pred
eax	res

```
fib:
   push  ebp
   mov   ebp, esp
   push  ebx

   mov   ecx, dword [ebp + 8] ; x
   xor   edx, edx             ; p_pred
   mov   ebx, 1               ; pred
   mov   eax, 1               ; res
   dec   ecx

   jecxz .end
.loop:
   lea   eax, [edx + ebx]
   mov   edx, ebx
   mov   ebx, eax
   loop .loop

.end:
   pop ebx
   pop ebp
   ret
```

```c
int fib(int x) { // x >= 1
  int i;
  int p_pred = 0;
  int pred = 1;
  int res = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = p_pred + pred;
     p_pred = pred;
     pred = res;
  }
  return res;
}
```

Регистр	Значение
ecx	x
edx	p_pred
ebx	pred
eax	res

```
fib:
   push  ebp
   mov   ebp, esp
   push  ebx

   mov   ecx, dword [ebp + 8] ; x
   xor   edx, edx             ; p_pred
   mov   ebx, 1               ; pred
   mov   eax, 1               ; res
   dec   ecx

   jecxz .end
.loop:
   lea   eax, [edx + ebx]
   mov   edx, ebx
   mov   ebx, eax
   loop .loop

.end:
   pop ebx
   pop ebp
   ret
```

```c
int fib(int x) { // x >= 1
  int i;
  int p_pred = 0;
  int pred = 1;
  int res = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = p_pred + pred;
     p_pred = pred;
     pred = res;
  }
  return res;
}
```

Регистр	Значение
ecx	x
edx	p_pred
ebx	pred
eax	res

```
fib:
   push  ebp
   mov   ebp, esp
   push  ebx

   mov   ecx, dword [ebp + 8] ; x
   xor   edx, edx             ; p_pred
   mov   ebx, 1               ; pred
   mov   eax, 1               ; res
   dec   ecx

   jecxz .end
.loop:
   lea   eax, [edx + ebx]
   mov   edx, ebx
   mov   ebx, eax
   loop .loop

.end:
   pop ebx
   pop ebp
   ret
```

Обратная задача

```
f:
   ...
   mov edx, dword [ebp+8] ; (1)
   mov eax, 0             ; (2)
   test edx, edx          ; (3)
   je .L7                 ; (4)
 .L10:                    ;
   xor eax, edx           ; (5)
   shr edx, 1             ; (6)
   jne .L10               ; (7)
 .L7:                     ;
   and eax, 1             ; (8)
   ...
```

```
int f(unsigned x) {
   int val = 0;
   while (________) {
      ____________;
   }
   return ________;
}
```

Передача управления

Си

- `if`
- `if-else`
- **`switch`**
- `do-while`
- `while`
- `for`
- `goto`
- `break`
- `contunue`
- `return`

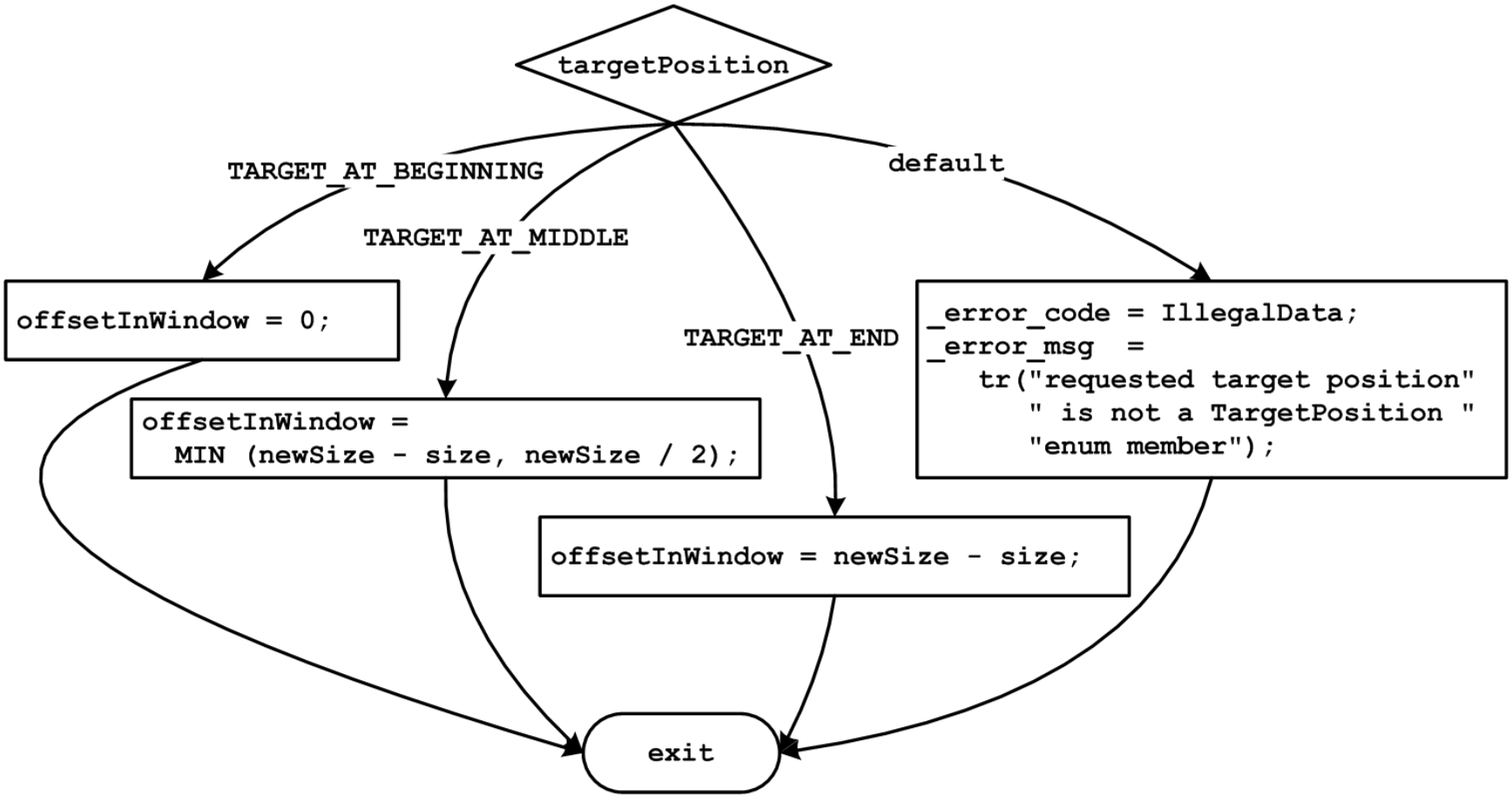
Ассемблер

- JMP
- J*cc*

- CALL
- RET

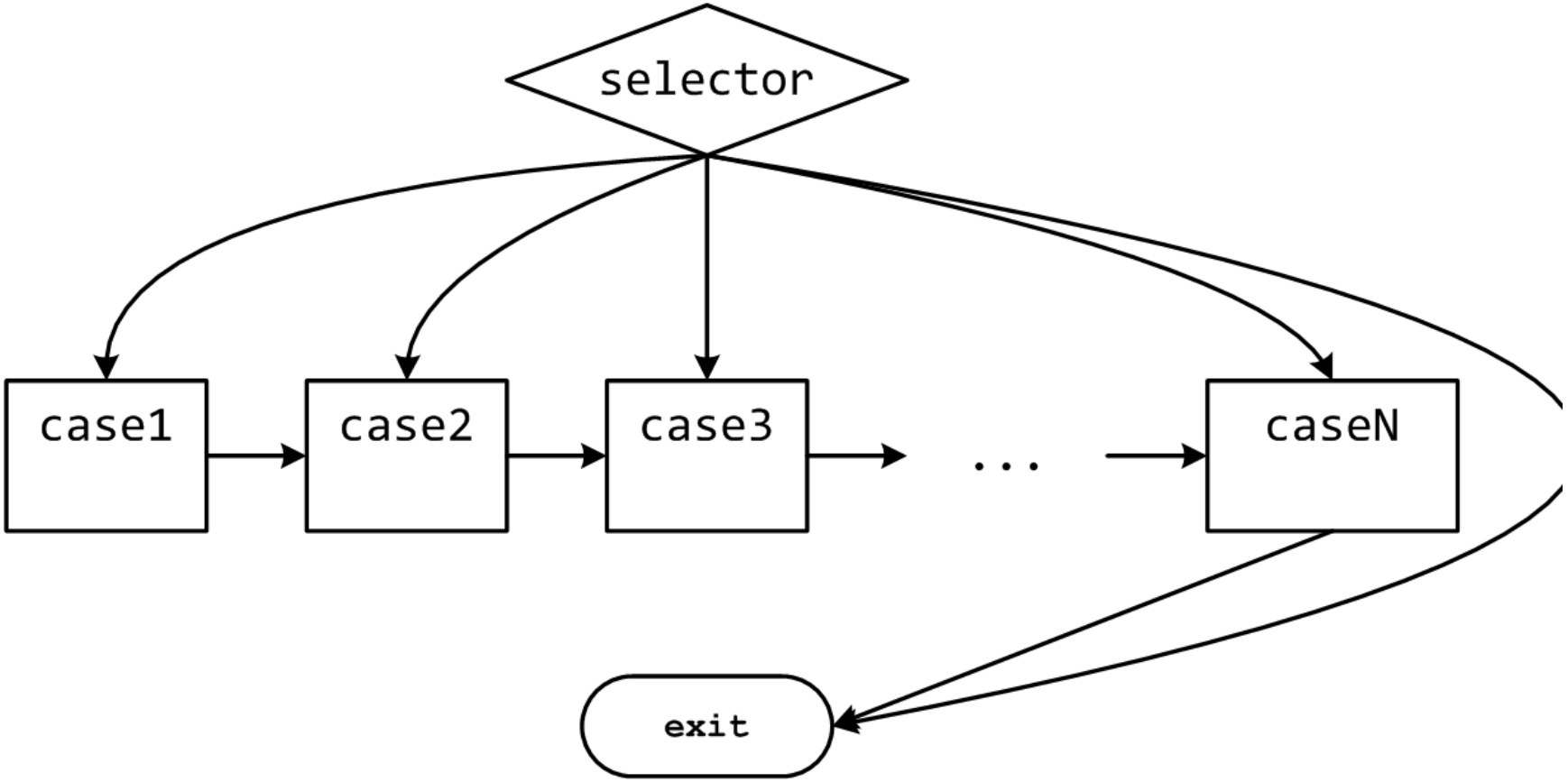
- CMOV*cc*

```
enum TargetPosition {
   TARGET_AT_BEGINNING,
   TARGET_AT_MIDDLE,
   TARGET_AT_END
};

switch (targetPosition){

case TARGET_AT_BEGINNING:
   offsetInWindow = 0;
   break;
case TARGET_AT_MIDDLE:
   offsetInWindow = MIN (newSize - size, newSize / 2);
   break;
case TARGET_AT_END:
   offsetInWindow = newSize - size;
   break;
default:
   _error_code = IllegalData;
   _error_msg  = tr("requested target position"
                    " is not a TargetPosition "
                    "enum member");
}
```

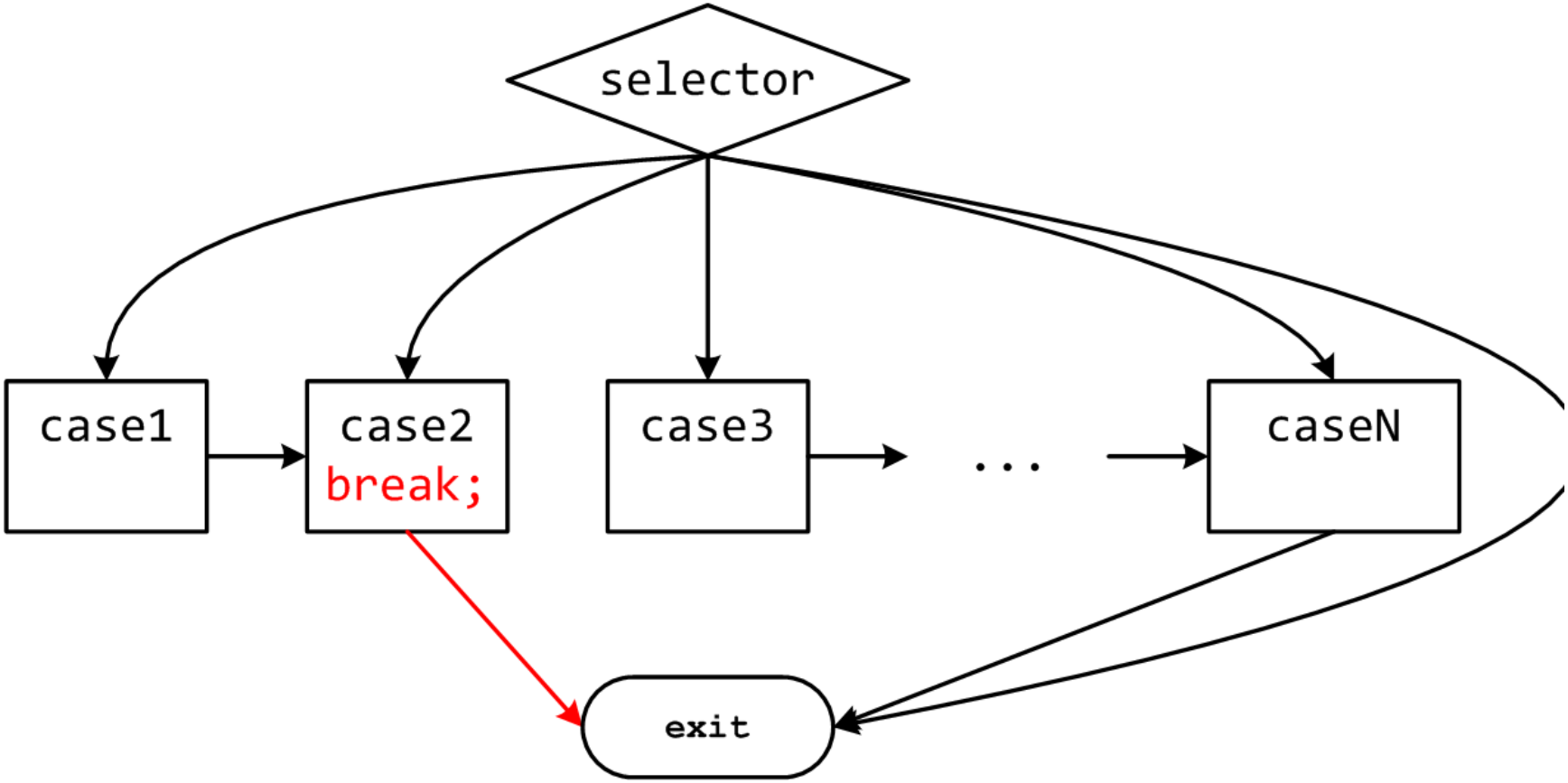


```
enum TargetPosition {
   TARGET_AT_BEGINNING,
   TARGET_AT_MIDDLE,
   TARGET_AT_END
};

if (TARGET_AT_BEGINNING == targetPosition) {
   offsetInWindow = 0;
} else if (TARGET_AT_MIDDLE == targetPosition) {
   offsetInWindow = MIN (newSize - size, newSize / 2);
} else if (TARGET_AT_END == targetPosition) {
   offsetInWindow = newSize - size;
} else {
   _error_code = IllegalData;
   _error_msg  = tr("requested target position"
                    " is not a TargetPosition "
                    "enum member");
}
```

```
; в edx помещено значение управляющего выражения
; т.е. targetPosition

 cmp  edx, TARGET_AT_BEGINNING
 jne  .comp2
; код для case TARGET_AT_BEGINNING:
 jmp .switch_exit

.comp2:
 cmp  edx, TARGET_AT_MIDDLE
 jne  .comp3
; код для case TARGET_AT_MIDDLE:
 jmp .switch_exit

.comp3:
 cmp  edx, TARGET_AT_END
 jne  .default
; код для case TARGET_AT_END:
 jmp .switch_exit

.default:
; код для default:
.switch_exit:
```



Вспоминаем пример из курса «АиАЯ»:
подсчет количества дней, прошедших с первого января.



Duff's Device

```
void duffs_device(char *to, char *from, int count) {

   register n = (count + 7) / 8; /* count > 0 assumed */

   switch (count % 8) {
      case 0:   do {  *to = *from++;
      case 7:         *to = *from++;
      case 6:         *to = *from++;
      case 5:         *to = *from++;
      case 4:         *to = *from++;
      case 3:         *to = *from++;
      case 2:         *to = *from++;
      case 1:         *to = *from++;
                   } while (--n > 0);
   }
}
```