ПРИЛОЖЕНИЕ

СРАВНЕНИЕ СОБЫТИЙНОГО И АВТОМАТНОГО ПОДХОДОВ К ПРОГРАММИРОВАНИЮ УПРАВЛЕНИЯ

Пример 1. **Программная реализация *R*-триггера**

1.1. Событийный подход

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
int R,S,Z=0;
void main()
{
for(;;){
clrscr();
cout << "\n\n\r Задайте R,S";
cout << "\n Z="  << Z;
cout << "\n R="; cin >> R; if(R>1) break;
cout << " S=";   cin >> S; if(S>1) break;
        if(S)    Z=1;
        if(R)    Z=0;
       }
}
```

1.2. Автоматный подход. Используется модель автомата без входного преобразователя

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
int R, S, Z=0;
void main()
{
for(;;){
clrscr();
cout << "\n\n\r Задайте R,S";
cout << "\n Z="  << Z;
cout << "\n R="; cin >> R; if(R>1) break;
cout << " S=";   cin >> S; if(S>1) break;
switch(Z){
         case 0: if(!R&S) Z=1;
                 break;
         case 1: if(R)    Z=0;
                 break;
```

```
        }
      }
}
```

Пример 2. **Программная реализация счетного триггера**

2.1. Событийный подход

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
int x,y=0,z=0;
void main()
{
for(;;){
clrscr();
cout << "\n\n\r Задайте x";
cout << "\n z=" << z;
cout << "\n y=" << y;
cout << "\n x="; cin >> x; if(x>1) break;
switch(x){
         case 0: if(z) y=1;
                 else  y=0;
                 break;
         case 1: if(y) z=0;
                 else  z=1;
                 break;
         }
       }
}
```

2.2. Автоматный подход. Используется модель автомата Мура второго рода

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
int x, Y=0, z=0;
void main()
{
for(;;){
clrscr();
cout <<  "\n\n\r Задайте x";
cout <<  "\n Y=" << Y;
cout <<  "\n z=" << z;
cout <<  "\n x="; cin >> x; if(x>1) break;
switch(Y){
         case 0: if(x) {Y=1;z=1;}
                 break;
         case 1: if(!x){Y=2;z=1;}
                 break;
```

```
          case 2: if(x) {Y=3;z=0;}
                  break;
          case 3: if(!x){Y=0;z=0;}
                  break;
          }
        }
}
```

Пример 3. **Фрагмент программы, реализующей некоторые функции управления элементом графического пользовательского интерфейса «тулбар»**

Н.И.Туккель, А.А.Шалыто

3.1. Событийный подход

```
//====== toolbars.h ======
#ifndef _TOOLBARS_H_INCLUDED_
#define _TOOLBARS_H_INCLUDED_

#include <Pt.h>              // Подключение библиотеки
                             // Photon.
#define TBMAXCOUNT 10        // Максимальное количество
                             // тулбаров.

//---
// Кнопка тулбара
//
typedef struct
{
  int toggle ;
  int function_index ;
} toolbar_btn_t ;

//---
// Тулбар
//
typedef struct
{
  toolbar_btn_t buttons[30] ; // Массив кнопок тулбара
  int           btn_count ;   // Количество кнопок в
                              // тулбаре
  PtWidget_t    *wgt, *pane ; // wgt - окно тулбара,
                              // pane - виджет PtPane
                              // внутри окна
  PhPoint_t     pos ;         // Положение тулбара
  int           menu ;        // Переменная состояния
                              // тулбара
  PhPoint_t     drag_pos ;    // Положение курсора мыши
                              // в начале движения
  int           cfg_index ;   // Индекс тулбара
                              // в массиве конфигураций
} toolbar_t ;
#endif
```

```
//====== toolbars.c ======
// массив toolbars[] описан как
// toolbar_t toolbars[TBMAXCOUNT] ;
//======
// Обработчик события "нажатие правой кнопки мыши
// при нахождении ее курсора на тулбаре"
//
int toolbar_btn_press( PtWidget_t *widget,
                       ApInfo_t *apinfo,
                       PtCallbackInfo_t *cbinfo )
{
  // Используется для получения номера тулбара
  PtArg_t          args[1] ;
  // Указатель на номер тулбара
  int              *number ;
  // Координаты курсора мыши
  PhRect_t         *rect ;
  // Произошедшее событие
  PhEvent_t        *event = cbinfo->event ;
  // Дополнительная информация о событии
  PhPointerEvent_t *edata = PhGetData(event) ;

  // Получить номер тулбара, для которого был вызван
  // обработчик события
  PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
  PtGetResources( widget, 1, args ) ;

  if( edata->buttons&Ph_BUTTON_MENU )
    // Была нажата правая кнопка мыши
  {
    rect = PhGetRects( event ) ;

    // Запомнить координаты курсора мыши
    toolbars[*number].drag_pos = rect->ul ;

    // Переместить окно тулбара выше всех остальных
    PtWindowToFront( toolbars[*number].wgt ) ;

    // Запомнить, что была нажата правая кнопка мыши
    toolbars[*number].menu = 1 ;
  } ;
  return( Pt_CONTINUE ) ;
} ;
```

```
//======
// Обработчик события "отпускание правой кнопки мыши при
// нахождении ее курсора на тулбаре"
//
int toolbar_btn_release( PtWidget_t *widget,
                         ApInfo_t *apinfo,
                         PtCallbackInfo_t *cbinfo )
{
  // Используется для получения номера тулбара
  PtArg_t          args[1] ;
  // Указатель на номер тулбара
  int              *number ;
  // Произошедшее событие
  PhEvent_t        *event = cbinfo->event ;
  // Дополнительная информация о событии.
  PhPointerEvent_t *edata = PhGetData( event ) ;

  //Получить номер тулбара, для которого был вызван
  //обработчик события
  PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
  PtGetResources( widget, 1, args ) ;

  if(   event->subtype==Ph_EV_RELEASE_REAL
     && edata->buttons&Ph_BUTTON_MENU )
     // Была отпущена правая кнопка мыши

    if( toolbars[*number].menu == 1 )
      // Перед этим была нажата правая кнопка мыши
    {
      // Отобразить меню
      ApCreateModule( ABM_toolbar_menu, NULL, NULL ) ;
    } ;

  toolbars[*number].menu = 0 ;

  return( Pt_CONTINUE ) ;
} ;

//======
// Обработчик события "перемещение мыши с нажатой правой
// кнопкой при нахождении ее курсора на тулбаре"
//
int toolbar_btn_move( PtWidget_t *widget,
                      ApInfo_t *apinfo,
                      PtCallbackInfo_t *cbinfo )
{
  // Используется для получения номера тулбара
  PtArg_t          args[1] ;
  // Указатель на номер тулбара
  int              *number ;
```

```
  // Координаты события
  PhRect_t         *rect ;
  // Произошедшее событие
  PhEvent_t        *event = cbinfo->event ;
  // Дополнительная информация о событии
  PhPointerEvent_t *edata = PhGetData( event ) ;

  // Если не нажата правая кнопка мыши - завершить
  // выполнение функции
  if(!edata->buttons&Ph_BUTTON_MENU) return Pt_CONTINUE;

  //Получить номер тулбара, для которого был вызван
  //обработчик события
  PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
  PtGetResources( widget, 1, args ) ;

  // Переместить тулбар вслед за курсором мыши
  rect = PhGetRects( event ) ;
  toolbars[*number].menu = 0 ;
  toolbar_move( &toolbars[*number],
                rect->ul.x - tb->drag_pos.x,
                rect->ul.y - tb->drag_pos.y ) ;
  return( Pt_CONTINUE ) ;
} ;

//======
// Обработчик события "пересечение курсором мыши
// границы тулбара"
//
int toolbar_boundary( PtWidget_t *widget,
                      ApInfo_t *apinfo,
                      PtCallbackInfo_t *cbinfo )
{
  // Используется для получения номера тулбара
  PtArg_t   args[1] ;
  // Указатель на номер тулбара
  int       *number ;

  //Получить номер тулбара, для которого был вызван
  //обработчик события
  PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
  PtGetResources( widget, 1, args ) ;

  toolbars[*number].menu = 0 ;

  return( Pt_CONTINUE ) ;
} ;
```

3.2. Автоматный подход. Используется модель смешанного автомата

```
//====== toolbars.h ======

#ifndef _TOOLBARS_H_INCLUDED_
#define _TOOLBARS_H_INCLUDED_

#include <Pt.h>

#define TBBTNSIZE 24
#define TBMAXCOUNT 10

//======
// Кнопка тулбара
//
typedef struct
{
  int toggle ;
  int function_index ;
} toolbar_btn_t ;

//======
// Тулбар
//
typedef struct
{
  toolbar_btn_t buttons[30] ; // Массив кнопок тулбара
  int           btn_count ;   // Количество кнопок
                              // в тулбаре
  PtWidget_t    *wgt, *pane ; // wgt - окно тулбара
                              // pane - виджет PtPane
                              // внутри окна
  PhPoint_t     pos ;         // Положение тулбара
  int y0 ;                    // Переменная состояния
                              // тулбара
  PhPoint_t     drag_pos ;    // Положение курсора мыши
                              // в начале движения
  int           cfg_index ;   // Индекс тулбара
                              // в массиве конфигураций
} toolbar_t ;

int toolbars_create() ;
void toolbars_config_read() ;
void toolbars_ontop() ;

#endif
```

```
//====== toolbars.c ======

//
// Модуль, реализующий управление тулбаром
//

#include "photon_stuff.h"

#include "toolbars.h"
#include "config.h"
#include "status.h"
#include "functions.h"

toolbar_t toolbars[TBMAXCOUNT] ;

//======
// Перемещение указанного тулбара
// на заданное расстояние
//
void toolbar_move( toolbar_t *tb, int inc_x, int inc_y )
{
  PtArg_t   args[1] ;
  PhPoint_t *pos = NULL, new_pos = {0, 0} ;
  short     abs_x = 0, abs_y = 0 ;

  PtSetArg( args, Pt_ARG_POS, &pos, 0 ) ;
  PtGetResources( tb->wgt, 1, args ) ;
  new_pos.x = pos->x + inc_x ;
  new_pos.y = pos->y + inc_y ;
  PtSetArg( args, Pt_ARG_POS, &new_pos, 0 ) ;
  PtSetResources( tb->wgt, 1, args ) ;

  PtGetAbsPosition( ABW_base, &abs_x, &abs_y ) ;
  cfg[tb->cfg_index].value.INT = new_pos.x - abs_x ;
  cfg[tb->cfg_index+1].value.INT = new_pos.y - abs_y ;
} ;

//======
// Обработчик события "нажатие правой кнопки мыши при
// нахождении ее курсора на тулбаре"
//
int toolbar_bpress( PtWidget_t *widget,
                    ApInfo_t *apinfo,
                    PtCallbackInfo_t *cbinfo )
{
  PtArg_t          args[1] ;
  int              *number = NULL ;
  toolbar_t        *tb = NULL ;
  PhEvent_t        *event = cbinfo->event ;
  PhPointerEvent_t *edata = PhGetData( event ) ;
```

```
if( edata->buttons == Ph_BUTTON_MENU )
  {
    PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
    PtGetResources( widget, 1, args ) ;
    tb = &toolbars[*number] ;

    // Вызвать управляющий автомат
    A0( 10, tb, event ) ;
  } ;

  return( Pt_CONTINUE ) ;
} ;

//======
// Обработчик события "отпускание правой кнопки мыши при
// нахождении ее курсора на тулбаре"
//
int toolbar_brelease( PtWidget_t *widget,
                      ApInfo_t *apinfo,
                      PtCallbackInfo_t *cbinfo )
{
  PtArg_t          args[1] ;
  int              *number = NULL ;
  toolbar_t        *tb = NULL ;
  PhEvent_t        *event = cbinfo->event ;
  PhPointerEvent_t *edata = PhGetData( event ) ;

  if(   event->subtype == Ph_EV_RELEASE_REAL
     && edata->buttons == Ph_BUTTON_MENU )
  {
    PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
    PtGetResources( widget, 1, args ) ;
    tb = &toolbars[*number] ;

    // Вызвать управляющий автомат
    A0( 20, tb, event ) ;
  } ;

  return( Pt_CONTINUE ) ;
} ;

//======
// Обработчик события "перемещение мыши
// с нажатой правой кнопкой при
// нахождении ее курсора на тулбаре"
//
int toolbar_bmove( PtWidget_t *widget, ApInfo_t *apinfo,
                   PtCallbackInfo_t *cbinfo )
{
  PtArg_t          args[1] ;
  int              *number = NULL ;
```

```
  toolbar_t        *tb = NULL ;
  PhEvent_t        *event = cbinfo->event ;
  PhPointerEvent_t *edata = PhGetData( event ) ;

  if( edata->buttons == Ph_BUTTON_MENU )
  {
    PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
    PtGetResources( widget, 1, args ) ;
    tb = &toolbars[*number] ;

    // Вызвать управляющий автомат
    A0( 30, tb, event ) ;
  } ;

  return( Pt_CONTINUE ) ;
} ;

//======
// Обработчик события "пересечение курсором мыши границы
// тулбара"
//
int toolbar_ptrleave( PtWidget_t *widget,
                      ApInfo_t *apinfo,
                      PtCallbackInfo_t *cbinfo )
{
  PtArg_t   args[1] ;
  int       *number = NULL ;
  toolbar_t *tb = NULL ;
  PhEvent_t *event = cbinfo->event ;

  if( event->subtype == Ph_EV_PTR_LEAVE )
  {
    PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
    PtGetResources( widget, 1, args ) ;
    tb = &toolbars[*number] ;

    // Вызвать управляющий автомат
    A0( 40, tb, event ) ;
  } ;

  return( Pt_CONTINUE ) ;
} ;

//======
// Функция, реализующая граф переходов
// алгоритма управления тулбаром
//
void A0( int e, toolbar_t *tb, PhEvent_t *event )
{
```

```
  // Запомнить текущее состояние
  int y_old = tb->y0 ;

  #ifdef GRAPH_EVENTS_LOGGING
    log_exec( "A0", y_old, e ) ;
  #endif

// Проверить условия переходов, если необходимо выполнить
// переход и действия на дуге
//
switch( tb->y0 )
{
  case 0:
    if( e == 10 )             tb->y0 = 1 ;
  break ;

  case 1:
    if( e == 20 ) { z30(tb) ; tb->y0 = 0 ; }
    else
    if( e == 30 )             tb->y0 = 2 ;
    else
    if( e == 40 )             tb->y0 = 0 ;
  break ;

  case 2:
    if( e == 30 ) { z20(tb, event) ; }
    else
    if( e != 30 )             tb->y0 = 0 ;
  break ;

  default:
    #ifdef GRAPH_ERRORS_LOGGING
      log_write( LOG_GRAPH_ERROR,
      "ОШИБКА В A0: неизвестное состояние", 0 ) ;
    #endif
  break ;
} ;

// Если состояние не изменилось -
// завершить выполнение функции
if( y_old == tb->y0 ) goto A0_end ;

#ifdef GRAPH_TRANS_LOGGING
  log_trans( "A0", y_old, tb->y0 ) ;
#endif
// Выполнить действия в состоянии
switch( tb->y0 )
{
  case 1:
    z10(tb, event) ; z40(tb) ;
  break ;
```

```
    case 2:
      z20(tb, event) ;
    break;
  } ;

  A0_end:
  #ifdef GRAPH_ENDS_LOGGING
    log_end( "A0", tb->y0, e ) ;
  #endif
  ;
} ;

//======
// Запомнить координаты курсора мыши
//
void z10( toolbar_t *tb, PhEvent_t *event )
{
  PhRect_t         *rect = NULL ;

  #ifdef ACTIONS_LOGGING
    log_write( LOG_ACTION,
               "z10. Запомнить координаты курсора
                мыши", 0 ) ;
  #endif

  rect = PhGetRects( event ) ;
  tb->drag_pos = rect->ul ;
} ;

//======
// Переместить тулбар
//
void z20( toolbar_t *tb, PhEvent_t *event )
{
  PhRect_t         *rect = NULL ;

  #ifdef ACTIONS_LOGGING
    log_write( LOG_ACTION,
               "z20. Переместить тулбар", 0 ) ;
  #endif

  rect = PhGetRects( event ) ;
  toolbar_move( tb, rect->ul.x - tb->drag_pos.x,
                rect->ul.y - tb->drag_pos.y ) ;
} ;
```

```
//======
// Вызвать меню тулбара
//
void z30( toolbar_t *tb )
{
  #ifdef ACTIONS_LOGGING
    log_write( LOG_ACTION,
               "z30. Вызвать меню тулбара", 0 ) ;
  #endif

  ApCreateModule( ABM_toolbar_menu, NULL, NULL ) ;
} ;

//======
// Переместить окно тулбара выше остальных
//
void z40( toolbar_t *tb )
{
  #ifdef ACTIONS_LOGGING
    log_write( LOG_ACTION, "z40. Переместить окно
               тулбара выше остальных", 0 ) ;
  #endif

  PtWindowToFront( tb->wgt ) ;
} ;
```

«Полный» протокол проверки всех переходов автомата, реализующего алгоритм управления тулбаром

Обработка события «нажатие правой кнопки мыши при нахождении ее курсора на тулбаре»

```
16:44:58.543{ A0:  в состоянии 0 запущен с событием e10
16:44:58.543T A0:  перешел из состояния 0 в состояние 1
16:44:58.543* z10. Запомнить координаты курсора мыши
16:44:58.543* z40. Переместить окно тулбара выше остальных
16:44:58.543} A0:  завершил обработку события e10 в состоянии 1
```

Обработка события «отпускание правой кнопки мыши при нахождении ее курсора на тулбаре» — вызов меню тулбара

```
16:44:59.903{ A0:  в состоянии 1 запущен с событием e20
16:44:59.903* z30. Вызвать меню тулбара
16:44:59.903T A0:  перешел из состояния 1 в состояние 0
16:44:59.903} A0:  завершил обработку события e20 в состоянии 0
```

Обработка события «выход курсора мыши за границу тулбара»

```
16:44:59.903{ A0:  в состоянии 0 запущен с событием e40
16:44:59.903} A0:  завершил обработку события e40 в состоянии 0
```

Обработка события «нажатие правой кнопки мыши при нахождении ее курсора на тулбаре»

```
16:45:03.963{ A0:  в состоянии 0 запущен с событием e10
```

```
16:45:03.963T A0:  перешел из состояния 0 в состояние 1
16:45:03.963* z10. Запомнить координаты курсора мыши
16:45:03.963* z40. Переместить окно тулбара выше остальных
16:45:03.963} A0:  завершил обработку события e10 в состоянии 1
```

Обработка события «выход курсора мыши за границу тулбара»

```
16:45:05.933{ A0:  в состоянии 1 запущен с событием e40
16:45:05.933T A0:  перешел из состояния 1 в состояние 0
16:45:05.933} A0:  завершил обработку события e40 в состоянии 0
```

Обработка события «нажатие правой кнопки мыши при нахождении ее курсора на тулбаре»

```
16:45:10.482{ A0:  в состоянии 0 запущен с событием e10
16:45:10.482T A0:  перешел из состояния 0 в состояние 1
16:45:10.482* z10. Запомнить координаты курсора мыши
16:45:10.482* z40. Переместить окно тулбара выше остальных
16:45:10.482} A0:  завершил обработку события e10 в состоянии 1
```

Обработка события «перемещение мыши с нажатой правой кнопкой при нахождении ее курсора на тулбаре»

```
16:45:12.812{ A0:  в состоянии 1 запущен с событием e30
16:45:12.812T A0:  перешел из состояния 1 в состояние 2
16:45:12.812* z20. Переместить тулбар
16:45:12.812} A0:  завершил обработку события e30 в состоянии 2
16:45:12.852{ A0:  в состоянии 2 запущен с событием e30
16:45:12.852* z20. Переместить тулбар
16:45:12.852} A0:  завершил обработку события e30 в состоянии 2
```

Обработка события «отпускание правой кнопки мыши при нахождении ее курсора на тулбаре»

```
16:45:15.812{ A0:  в состоянии 2 запущен с событием e20
16:45:15.812T A0:  перешел из состояния 2 в состояние 0
16:45:15.812} A0:  завершил обработку события e20 в состоянии 0
```

Обработка события «выход курсора мыши за границу тулбара»

```
16:45:16.742{ A0:  в состоянии 0 запущен с событием e40
16:45:16.742} A0:   завершил обработку события e40 в состоянии 0
```

Обработка события «нажатие правой кнопки мыши при нахождении ее курсора на тулбаре»

```
16:45:18.992{ A0:  в состоянии 0 запущен с событием e10
16:45:18.992T A0:  перешел из состояния 0 в состояние 1
16:45:18.992* z10. Запомнить координаты курсора мыши
16:45:18.992* z40. Переместить окно тулбара выше остальных
16:45:18.992} A0:  завершил обработку события e10 в состоянии 1
```

Обработка события «перемещение мыши с нажатой правой кнопкой при нахождении ее курсора на тулбаре»

```
16:45:20.472{ A0:  в состоянии 1 запущен с событием e30
16:45:20.472T A0:  перешел из состояния 1 в состояние 2
16:45:20.472* z20. Переместить тулбар
```

```
16:45:20.472} A0:  завершил обработку события e30 в состоянии 2
16:45:21.192{ A0:  в состоянии 2 запущен с событием e30
16:45:21.192* z20. Переместить тулбар
16:45:21.192} A0:  завершил обработку события e30 в состоянии 2
```

Обработка события «выход курсора мыши за границу тулбара» — прекращение перемещения

```
16:45:21.232{ A0:  в состоянии 2 запущен с событием e40
16:45:21.232T A0:  перешел из состояния 2 в состояние 0
16:45:21.232} A0:  завершил обработку события e40 в состоянии 0
```