Проектиране и Тестиране на Софтуер
ТУ, кат. КС, летен семестър 2012

Лекция 7

Тема:

Валидация на Входни Данни

Съдържание:

- Въведение в проблема
- Видове входни потоци
- Случаи, налагащи валидация на входни данни
- Средства и ресурси за валидация на входни данни

Валидация на входни данни

ISO17799 IS AN INFORMATION SECURITY MANAGEMENT STANDARD

Section 10: Systems Development and Maintenance

- 10.1 ...
- ***10.2 BUILD SECURITY INTO APPLICATION SYSTEMS***
  - ***10.2.1 BUILD INPUT DATA VALIDATION INTO YOUR SYSTEMS***
  - 10.2.2 BUILD PROCESSING CONTROLS INTO YOUR SYSTEMS
  - 10.2.3 BUILD MESSAGE AUTHENTICATION INTO YOUR SYSTEMS
  - 10.2.4 BUILD OUTPUT DATA VALIDATION INTO YOUR SYSTEMS
- 10.3 ...
- 10.4 ...
- 10.5 ...

ISO 17799 2000 is now OBSOLETE.see ISO IEC 17799 2005!

ISO17799-2000 IS AN INFORMATION SECURITY MANAGEMENT STANDARD

10.2.1 BUILD INPUT DATA VALIDATION INTO YOUR SYSTEMS

- Build input data validation controls into application systems.
- Make sure that your validation controls are able to verify that your input data is correct and appropriate.
- Make sure that validation controls are used to verify that the input of business transactions is correct and appropriate.
- Make sure that validation controls are used to verify that input of standing data and parameter tables is correct and appropriate.

ISO17799 -2000 IS AN INFORMATION SECURITY MANAGEMENT STANDARD

10.2.1 BUILD INPUT DATA VALIDATION INTO YOUR SYSTEMS (continued)

- Use input validation controls to detect out-of-range values.
- Use input validation controls to detect missing or incomplete data.
- Use input validation controls to detect invalid characters in data fields.
- Use input validation controls to detect when upper and lower data volumes have been exceeded.
- Review periodically the content of key fields in order to verify their integrity and validity.

ISO17799 -2000 IS AN INFORMATION SECURITY MANAGEMENT STANDARD

10.2.1 BUILD INPUT DATA VALIDATION INTO YOUR SYSTEMS (continued)

- Review periodically the content of data files in order to verify their integrity and validity.
- Inspect hardcopy input documents in order to detect unauthorized changes to input data.
- Develop procedures to respond to data validation errors.
- Develop procedures to test the plausibility of input data.
- Define the responsibilities of all data input personnel.

Валидация на входни данни

ISO IEC 17799 2005 INFORMATION SECURITY STANDARD

- ***12. Information Systems Security Management***
  - ***12.2 Make sure applications process information correctly***
    - ***12.2.1 Validate data input into your applications***
    - 12.2.2 Use validation checks to control processing
    - 12.2.3 Protect message integrity and authenticity
    - 12.2.4 Validate your applications' output data

Валидация на данни

Data Validation е процес, който потвърждава, че една програма борави с коректни данни (clean, correct and useful data). Отделни ПП проверяват коректността и смислеността на входните данни за софтуерния пакет.

Data Validation проверява коректността на данните преди тяхната същинска обработка.

Една примитивна data validation проверява дали символите от входния поток принадлежат на множество валидни/допустими символи.

Валидация на данни

Пример: входен поток като последователност от телефонни номера.

Множество допустими символи:

цифри: 0 ... 9

символи: + - ( )

Валидация на данни

Методи, прилагани за валидация на входни данни:

- Проверка по шаблон (Format or picture check)
- Проверка по диапазон (Range check)
- Проверка за наличие (Presence check)
- Контролни цифри (Check digits)
- Обща стойност (Batch totals)
- Hash totals

Методи …

- **Проверка по шаблон (Format or picture check)**

Проверка дали данните са в определен формат,

например за дата DD/MM/YYYY

например за време HH:MM:SS

Методи …

- **Проверка по диапазон (Range check)**

Проверка дали данните са в определен диапазон от стойности,
например месец в интервала 1 .. 12
дни от месеца 1 .. 28 (29, 30, 31)
дни от седмицата 1 .. 7 (0 .. 6)

Методи …

- **Проверка за наличие (Presence check)**

Проверка дали не са пропуснати важни съществени записи с данни,

например в БД Студенти наличие на поле (запис) за факултетен номер.

Методи …

Контролни цифри (Check digits)

Приложим за числени данни. Допълнителна цифра се прикачва към число, като се пресмята по цифрите на числото. При вход тази цифра се пресмята и проверява с постъпилата,

например кодът ISBN (10 цифри). Последната 10-та цифра се изчислява като сума по modulus 11 над първите 9 цифри.

Методи …

- **Обща стойност (Batch totals)**

Проверка за липсващи записи. Числени полета се натрупват в пакет. Общият брой (batch total) се въвежда и програмата следи дали входният поток съдържа указания брой стойности.

- **Hash totals**

Модификация на batch total.

Валидация на данни

Опасност: некоректна неправилна валидация на данни води до:

Разрушаване на данни (data corruption)

Пробив в сигурността (security vulnerability)

Разрушаване на данни

Data corruption се отнася до грешки в computer data, които възникват при предаване и извличане на данни и които водят до неочаквани (unintended) изменения на оригиналните данни. Необходими са мерки за **data integrity**.

Input Data Validation е една възможна мярка в тази посока.

Пробив в сигурността

Уязвимостта на ПО (**security vulnerability**) се отнася до слабости и недостатъци, които допускат attacker да пробие integrity, confidentiality, access control, availability до приложението и данните, които то обработва

Vulnerabilities са резултат от бъгове bugs или грешки в проектирането на ПО

Vulnerability се проявява чрез програмен код експлоит exploit.

Какво е експлоит?

An **exploit** is a piece of software that takes advantage of a bug, glitch or vulnerability, leading to privilege escalation or denial of service on a computer system.

There are several methods of classifying exploits. Most common is by how the exploit contacts the vulnerable software.

A 'remote exploit' works over a network and exploits the security vulnerability without any prior access to the vulnerable system.

A 'local exploit' requires prior access to the vulnerable system and usually increases the privileges of the person running the exploit past those granted by the system administrator.

Exploits against client applications also exist, usually consisting of modified servers that send an exploit if accessed with client application.

Основни х-ки

- **Input Data Validation - :**
- Валидация на входни данни означава повтаряемо интерактивно въвеждане на данни, докато във входния поток постъпи стойност от допустим диапазон (valid range) или постъпи стойност в съответствие с обявен шаблон (intended template format).

Видове цикли

- Цикъл – управляваща структура, която повтаря група оператори от програма

Цикъл с брояч	**Цикъл с условие**
	Спец. Стойност Sentinel
	Край на данни EOF
	Валидация на данни
	Флаг управляем или условие от общ х-р

Цикъл за валидация на данни

- Пример 1: цикъл за въвеждане на брой наблюдения с вградена валидация за въведена положителна стойност.

  Програмен код: InputDatValid1.cpp (сл. слайд)

  Изпълним код: InputDatValid1.exe

InputDatValid1.cpp

```
void main()
{
    int numobs;
    cout << "\n Enter number of observed values:";
    cin >> numobs;
    while (numobs <=0)          {
        cout << "\nNegative or zero values invalid. try again:";
        cin >> numobs;
                                }
    cout << "\nYou entered " << numobs << " observed values\n";
}
```

Вход, базиран на RTL ф-ии

Си: Всяка I/O ф-ия трябва да се следва от о-р за проверка на вр. стойност и при необходимост и стойността на глобалната променлива **errno**

C++: Същите Си RTL ф-ии са достъпни и за C++.

errno е цяла стойност, а ф-ията strerror(errno) връща текст, описващ типа на грешката.

Пример 2: илюстрира errno.

Програмен код:InputDatValid2.cpp (сл. слайд)

Изпълним код: InputDatValid2.exe

InputDatValid2.cpp

```
void main()
    {      extern int errno;          int fh;
    if (  (fh = _open("Test.txt", _O_RDONLY)) == -1 )   {
          /* Two ways to create error message: */
                perror("\nperror says open failed" );
          printf("\nerrno = %d   %s", errno, strerror(errno));
          printf("\n File can not open\n\n");                    }
     else  { printf("\n File exists and can open\n\n"); _close( fh); }
//=================================
    for (int i=0; i<=45; i++)
    {
       printf("\n %d    %s", i, strerror(i));
    }
}
```

Видове входни потоци

Откъде идват входни данни?

- Конзола-входен поток клавиатура;
- GUI контроли;
- Файл(ове) – текстов/двоичен;
- Мрежова връзка(и) – socket;
- База Данни (Data Base);
- XML входен поток.

Случаи

- Ситуации, които налагат валидация на данни, водят до следните последователно настъпващи случаи:
  - Разделяне на част от входния поток, например въведен ред, на отделни полета.
  - Преобразуване на поле от входния поток (символен низ) в стойност от даден тип.
  - Проверка допустима ли е преобразуваната на предишната стъпка стойност.

Случай Валидация

Първи: Разделяне на част от входния поток, например въведен ред, на отделни полета.	Необходими проверки: •Правилен брой полета; •Валидно множество разделители.

Случай	Валидация
Втори: Преобразуване на поле от входния поток (символен низ) в стойност от даден тип.	Необходими проверки: •Полето съдържа допустими символи за съответния тип данни; •Възможно конвертиране.

Случай	Валидация
Трети: Проверка допустима ли е преобразуваната на предишната стъпка стойност.	Необходими проверки: •Въведената стойност е в определен диапазон (certain valid range).

Средства за валидация на входни данни

A. I/O ф-ии за прочитане на низ от символи
B. I/O ф-ии за прочитане на данни
C. I/O ф-ии за четене на стойност от даден тип
D. String ф-ии за разделяне стринг на полета (при форматиран входен поток)
E. String ф-ии за конвертиране към стойност от даден тип
F. Ф-ии за работа с регулярни изрази

A. I/O ф-ии за прочитане на низ от символи

Тези функции четат целия входен поток или част от него. Данните се въвеждат ред по ред (‘\n’ delimiter).

C:gets(),fgets() C++:класове istream,ifstream

Типични проверки:

- Броят въведени редове съвпада ли с очаквания, ако е известен предварително?
- Прочетеният ред празен ли е?
- Дължината на въведения ред в допустим диапазон ли е?

А. I/O ф-ии за четене на низ от символи

- Пример(и). Демо програми 3 и 4

  Програмен код: InputDatValid3.cpp (виж сл. слайд)

  Изпълним код: InputDatValid3.exe

InputDatValid3.cpp

```
// This program uses fgets to display a line from a file on the screen.
void main( void )  {
    int linecount=0, charcount=0;   FILE *stream;   char line[100];
  if( (stream = fopen( "eof.c", "r" )) != NULL )   {
    if( fgets( line, 100, stream ) == NULL)    printf( "fgets error\n" );
    else {printf( "\n%s", line);   linecount++;  charcount+=strlen(line);}
    while (fgets(line, 100, stream) != NULL)
          {printf( "\n%s", line);  linecount++; charcount+= strlen(line);}
    printf("\n\nTotal lines read        = %d", linecount);
    printf("\n\nTotal characters read = %d", charcount); fclose( stream );
                                                          }
  else  printf("\nCan't open file");
}
```

А. I/O ф-ии за четене на низ от символи

- Пример(и). Демо програми 3 и 4

  Програмен код: InputDatValid4.cpp (виж сл. слайд)

  Изпълним код: InputDatValid4.exe

InputDatValid4.cpp

```
// Uses feof to get the end file. It also checks for errors with ferror.
void main( void ) {
   int  count, total = 0;   char buffer[100];   FILE *stream;
   if( (stream = fopen( "eof.c", "r" )) == NULL )      exit( 1 );
   while( !feof( stream ) ) /* Cycle until end of file reached: */
   {
      count = fread( buffer, sizeof( char ), 100, stream );
      if( ferror( stream ) )      {        perror( "Read error" );        break;    }
      total += count; /* Total up actual bytes read */
   }
   printf( "\nNumber of bytes read = %d\n", total );
   fclose( stream );
                        }
```

B. I/O ф-ии за четене на данни

Тези функции обикновено прочитат определен брой байтове от вх. поток.

Functions: _open(), _read(), _close()

Единствената възможна проверка, (ако има смисъл да се прилага) е дали са прочетени зададения брой байтове или по-малко.

В. I/O ф-ии за четене на данни

- Пример(и). Демо програма 5

  Програмен код: InputDatValid5.cpp (сл. слайд)

  Изпълним код: InputDatValid5.exe

InputDatValid5.cpp

```
// reads data from a file 10 bytes at a time until EOF or an error occurs
void main( void )
{   int  fh, count, total = 0;   char buf[10];
   if( (fh = _open( "eof.c", _O_RDONLY )) == - 1 )
   {       perror( "Open failed");       exit( 1 );   }
   while( !_eof( fh ) ) /* Attempt to read in 10 bytes: */
   {      if( (count = _read( fh, buf, 10 )) == -1 )
                          {  perror( "Read error" );  break;  }
          total += count; /* Total actual bytes read */
   }
   printf( "\n\n\nNumber of bytes read = %d\n", total );
   _close( fh );
}
```

C. I/O ф-ии за четене на стойност от даден тип

Повечето (асемблерни) ПЕ не предоставят функции от този клас. Задача на програмиста е да създаде такива ф-ии с вградени проверки за невалидни входни данни.

C/C++ предоставят сл. възможности:

C: scanf(), fscanf(), sscanf()

C++: cin >>, cin.fail()

C. I/O ф-ии за четене на стойност от даден тип

C: scanf(), fscanf(), sscanf()

```
int x;
int result = scanf("%d", &x);
CheckErrno(result);   // error check if
                      // result == -1
if (result != 1) error();
```

С. I/O ф-ии за четене на стойност от даден тип

C++: cin >>, cin.fail()

```
#include<iostream>
using namespace std;
. . .
int x;
cin >> x;   // fail bit switched on in case of
            // invalid conversion to integer
if ( cin.fail() ) error;
```

C. I/O functions to read certain data type value in C#

Problem: How To test a string to contain only numeric characters

C#:

Use **`isNumber( )`** method of **`char`** class

```
string aV = "1234";
if (char.isNumber(aV[0]) …
```

C. I/O functions to read certain data type value in C#

```
static bool isNumericData(string par)
      {
         bool flag=true;
         for (int i = 0; i < par.Length; i++)
            if (!char.IsNumber(par[i])) { flag = false; break; }
         return flag;
      }

//============================
string pom;
pom = Console.ReadLine();
if (isNumericData(pom)) Console.WriteLine("Only digits typed");
   else          Console.WriteLine("Digits and other chars typed");
```

C. I/O functions to read certain data type value in Java

Problem: You can use a Scanner, which gives you access to ready-made methods which parse numbers of various types (including boolean test methods which allow you to avoid exceptions if that's a problem):

How To test a string to contain only numeric characters Java:

```
String inputString = " 345.9888  ";
Scanner scanner = new Scanner(inputString);
if (scanner.hasNextDouble()) {
    double d = scanner.nextDouble();
    doStuffWith(d);
}
```

C. I/O functions to read certain data type value in Java

Problem: You can use a Scanner, which gives you access to ready-made methods which parse numbers of various types (including boolean test methods which allow you to avoid exceptions if that's a problem):

How To test a string to contain only numeric characters

Java:

```
Scanner cin = new Scanner(System.in);

    int pom = cin.nextInt();
    float pom2 = cin.nextFloat();
    double pom3 = cin.nextDouble();

    int pom4;

    if (cin.hasNextInt()) pom4=cin.nextInt();
```

C. I/O functions to read certain data type value in Java

Problem: You can use a Scanner, which gives you access to ready-made methods which parse numbers of various types (including boolean test methods which allow you to avoid exceptions if that's a problem):

How To test a string to contain only numeric characters

Java:

See also Daniel Liang, pp337

The `Character` class

C. I/O functions to read certain data type value in Java

How To test a string to contain only numeric characters Java:

```
public boolean isInteger( String input )
{
    try
    {
        Integer.parseInt( input );
        return true;
    }
    catch( Exception e)
    {
        return false;
    }
}
```

C. I/O functions to read certain data type value in Java

Problem: How To test a string to contain only numeric characters

Java:

```
String pom5="123456789";
String pom6="2345ggg789";

  if(isInteger(pom5)==true) System.out.println(pom5 + "  is numeric
      string");
    else  System.out.println(pom5 + "  is not numeric string");

  if(isInteger(pom6)) System.out.println(pom6 + "  is numeric string");
    else  System.out.println(pom6 + "  is not numeric string");
```

D. String ф-ии за разделяне символен низ на полета

Разделянето на полета става чрез задаване на множество от разделители.

Ф-ията връща последователност от полета (поднизове).

C/C++: strtok() – неопределн брой полета и специфичен набор разделители

sscanf() – фиксиран брой полета и набор разделители (‘ ‘,\n, \r, \t)

D. String ф-ии за разделяне символен низ на полета

- Пример(и). Демо програми 6 и 7

  Програмен код: InputDatValid6.cpp (виж сл. слайд)

  Изпълним код: InputDatValid6.exe

InputDatValid6.cpp

```
// InputDatValid6.cpp Това е демо програма за strtok RTL функция
char string[] = "A string\tof ,,tokens\nand some  more tokens";
char seps[]   = " ,\t\n";       char *token;
void main( void )
{   printf( "%s\n\nTokens:\n", string );
   /* Establish string and get the first token: */
   token = strtok( string, seps );
   while( token != NULL ) /* While there are tokens in "string" */
   {
     printf( " %s\n",token ); token=strtok( NULL, seps ); /* Get token: */
   }
}
```

D. String ф-ии за разделяне символен низ на полета

- Пример(и). Демо програми 6 и 7

  Програмен код: InputDatValid7.cpp (виж сл. 3 слайда)

  Изпълним код: InputDatValid7.exe

InputDatValid7.cpp: Тази програма ползва *sscanf* да чете данни от низ *tokenstring* и ги извежда на екран

```
void main( void )
{ char tokenstring[]="15 12 14..."; char s[81]; char c; int i; float f;
  /* Input various data from tokenstring: */
  sscanf( tokenstring, "%s", s );
  sscanf( tokenstring, "%c", &c );
  sscanf( tokenstring, "%d", &i );
  sscanf( tokenstring, "%f", &f );

  /* Output the data read */
  printf( "String    = %s\n", s );
  printf( "Character = %c\n", c );
  printf( "Integer:  = %d\n", i );
  printf( "Real:     = %f\n\n", f );
```

InputDatValid7.cpp: Тази програма ползва *sscanf* да чете данни от низ *tokenstring* и ги извежда на екран

```
/* Input various data from tokenstring: */
   sscanf( tokenstring, "%s", s );
   sscanf( tokenstring+1, "%c", &c );
   sscanf( tokenstring+3, "%d", &i );
   sscanf( tokenstring+6, "%f", &f );

   /* Output the data read */
   printf( "String    = %s\n", s );
   printf( "Character = %c\n", c );
   printf( "Integer:  = %d\n", i );
   printf( "Real:     = %f\n\n", f );
```

InputDatValid7.cpp: Тази програма ползва *sscanf* да чете данни от низ *tokenstring* и ги извежда на екран

```
 /* Input various data from tokenstring: */
  sscanf( tokenstring, "%s%d%f", s , &i, &f);

  /* Output the data read */
  printf( "String    = %s\n", s );
  printf( "Integer:  = %d\n", i );
  printf( "Real:     = %f\n\n", f );

    printf("\nPress any key to continue");
    getchar();
}
```

E. String ф-ии за конвертиране към стойност от даден тип

Тези ф-ии се ползват при прехвърляне на полета данни от вх. Поток във вътр. Структури данни на програмата. Етапите са следните:

- Премахване на празни символи отляво/дясно
- Тест допустими символи <cctype>, <ctype.h>
- Конвертиране
- Тест дали конвертираната стойност е в допустими граници

C: is*(), ato*(), sscanf

C++: STL класове: istrstream, istringstream

E. String ф-ии за конвертиране към стойност от даден тип

- Пример(и). Демо програми 8 и 9

  Програмен код: InputDatValid8.cpp (виж сл. слайд)

  Изпълним код: InputDatValid8.exe

InputDatValid8.cpp

```
#include <iostream>
using namespace std;
void main( )  {   bool result1 = isalpha ( 'L');
  bool result2 = isalpha ( '@' );   bool result3 = isalpha ( '3' );

  if ( result1 )  cout << "The character 'L' is "<< "alphabetic." << endl;
  else    cout << "The character 'L' is "<< " not alphabetic." << endl;

  if ( result2 )  cout << "The character '@' is "<< "alphabetic." << endl;
  else    cout << “Character '@' "<< " not alphabetic." << endl;

  if ( result3 )  cout << "The character '3' is "<< "alphabetic." << endl;
  else      cout << “Character '3' is "<< " not alphabetic." << endl;
}
```

E. String ф-ии за конвертиране към стойност от даден тип

- Пример(и). Демо програми 8 и 9

Програмен код: InputDatValid9.cpp

Изпълним код: InputDatValid9.exe (виж сл. слайд)

InputDatValid9.cpp

```
// shows how numbers as strings can be converted to numeric values
// using the atof, atoi, and atol functions
#include <stdlib.h>
#include <stdio.h>
void main( void ){   char *s; double x; int i; long l;
   s = "  -2309.12E-15";      x = atof( s );
   printf( "atof test: ASCII string: %s\t\tfloat:  %e\n", s, x );
   s = "7.8912654773d210";    x = atof( s );
   printf( "atof test: ASCII string: %s\tfloat:  %e\n", s, x );
   s = "  -9885 pigs";         i = atoi( s );
   printf( "atoi test: ASCII string: %s\t\tinteger: %d\n", s, i );
   s = "98854 dollars";      l = atol( s );
   printf( "atol test: ASCII string: %s\t\tlong: %ld\n", s, l );
                        }
```

F. Ф-ии за работа с регулярни изрази

РИ се използват за:

- тестване с определен шаблон (match);
- търсене със заместване (search/replace)

РИ служат както за валидиране на входни данни, така и за разделяне на стринг на поднизове

C: доп. Библиотеки поддържат РИ

C++: Boost regex library, CRETA

F. Functions based on regular expressions

- Пример(и). Демо програми под Linux

  Програмен код: InputDatValid10.c (сл. слайд)

  Изпълним код: a.out

InputDatValid10.cpp

```
// InputDatValid10.c – Regular Expressions
#include <sys/types.h>
#include <regex.h>
void main()
{
  regex_t regex;                int reti;
  reti = regcomp( &regex, "^a[[:alnum:]]", 0); // compile RE
  if(reti) { printf("\n Could not compile regex\n"); exit(1); }

  reti = regexec( &regex, "abc", 0, NULL, 0); // execute RE
  if (reti)  puts("\nNo Match");
     else     puts("\n Match");

  regfree( &regex);
}
```

Валидация на вх. Данни в Си

Източник:

http://gd.tuwien.ac.at/languages/c/programming-bbrown/c_032.htm

Основни правила:

- Филтрирайте невалидни данни. Don't pass invalid data onwards.
- Валидирайте данни при вход. Validate data at input time.
- Информирайте потребителя смислено. Always give the user meaningful feedback
- Дайте указания какъв тип данни се очаква да постъпят като вход. Tell the user what you expect to read as input

Пример 1, валидация в цикъл

```
#include <stdio.h>
#include <ctype.h>
 void main(){
     int     valid_input;    /* when 1, data is valid and loop is exited */
     char    user_input;   /* handles user input, single cha menu choice */
     valid_input = 0;
     while( valid_input == 0 ) {
            printf("Continue (Y/N)?\n");  scanf("  %c", &user_input );
            user_input = toupper( user_input );
            if((user_input == 'Y') || (user_input == 'N') )  valid_input = 1;
            else  printf("\007Error: Invalid Choice\n");
                                        }
 }
```

Пример 1, валидация в цикъл

Sample Program Output

Continue (Y/N)?

b

Error: Invalid Choice

Continue (Y/N)?

N

Пр. 2, четене и валидиране на избор

```
#include <stdio.h>
#include <ctype.h>
main(){
    int  exit_flag = 0, valid_choice;                    char    menu_choice;
    while( exit_flag == 0 ) { valid_choice = 0;
            while( valid_choice == 0 ) {
            printf("\nC = Copy File\nE = Exit\nM = Move File\n");
            printf("Enter choice:\n");          scanf("   %c", &menu_choice );
    if((menu_choice=='C') || (menu_choice=='E') || (menu_choice=='M'))
                                    valid_choice = 1;
    else    printf("\007Error. Invalid menu choice selected.\n");
            }
    switch( menu_choice ) {
            case 'C' : .....();    break;
            case 'E' : exit_flag = 1;  break;
            case 'M' : .....();  break;
            default : printf("Error--- Should not occur.\n"); break;
                            }       }
}
```

Пр. 2, четене и валидиране на избор

Sample Program Output
C = Copy File
E = Exit
M = Move File
Enter choice:
X
Error. Invalid menu choice selected.
C = Copy File
E = Exit
M = Move File
Enter choice:
E

Валидация на вх. Данни в Си

scanf() води до проблеми, когато вместо цифри за цяло >> постъпи например низ от букви във входния пток. Начин да се преодолее това е, ако целият входен поток се прочете като низ (use *getchar()*), с последващо конвертиране на низа в съответния тип данни.

Пример 3,
четене на думи от входния поток
to read a word at a time

```
#include <stdio.h>      /* example three, to read a word at a time */
#include <ctype.h>
#define MAXBUFFERSIZE   80
void cleartoendofline( void ) { char ch;  ch=getchar(); while( ch != '\n' ) ch=getchar(); }
main()  {
      char    ch;  /* handles user input */
      char    buffer[MAXBUFFERSIZE];  /* sufficient to handle one line */
      int     char_count;           /* number of characters read for this line */
      int     exit_flag = 0;              int     valid_choice;
      while( exit_flag  == 0 ) {
              printf("Enter a line of text (<80 chars)\n");ch = getchar();char_count = 0;
              while( (ch != '\n')  &&  (char_count < MAXBUFFERSIZE)) {
                      buffer[char_count++] = ch;   ch = getchar();        }
              buffer[char_count] = 0x00;      /* null terminate buffer */
              printf("\nThe line you entered was:\n");        printf("%s\n", buffer);
              valid_choice = 0;
              while( valid_choice == 0 ) {
                  printf("Continue (Y/N)?\n");  scanf(" %c", &ch );  ch = toupper( ch );
                      if((ch == 'Y') || (ch == 'N') )         valid_choice = 1;
                      else  printf("\007Error: Invalid choice\n");
                  cleartoendofline();       }
              if( ch == 'N' ) exit_flag = 1;
```

Пример 4,
четене на число като низ
reading a number as a string

```
#include <stdio.h>      /* example four, reading a number as a string */
#include <ctype.h>
#include <stdlib.h>
#define MAXBUFFERSIZE   80
void cleartoendofline( void ) { char ch;  ch=getchar(); while( ch != '\n' ) ch=getchar(); }
main()
{
      char    ch;                 /* handles user input */
      char    buffer[MAXBUFFERSIZE];  /* sufficient to handle one line */
      int     char_count;         /* number of characters read for this line */
      int     exit_flag = 0, number, valid_choice;
      while( exit_flag  == 0 ) {      valid_choice = 0;
               while( valid_choice == 0 ) {
                        printf("Enter a number between 1 and 1000\n"); ch = getchar();
                        char_count = 0;
                        while( (ch != '\n')  &&  (char_count < MAXBUFFERSIZE)) {
                                 buffer[char_count++] = ch;        ch = getchar();   }
                        buffer[char_count] = 0x00;      /* null terminate buffer */
                        number = atoi( buffer );
                        if( (number < 1) || (number > 1000) )
                                 printf("\007Error. Number outside range 1-1000\n");
                        else         valid_choice = 1;
```

```
/* example four, reading a number as a string, continued */\
        printf("\nThe number you entered was:\n");
        printf("%d\n", number);

        valid_choice = 0;
        while( valid_choice == 0 ) {
                printf("Continue (Y/N)?\n");   scanf(" %c", &ch );
                ch = toupper( ch );
                if((ch == 'Y') || (ch == 'N') )
                        valid_choice = 1;
                else
                        printf("\007Error: Invalid choice\n");
                cleartoendofline();
                                        }
        if( ch == 'N' ) exit_flag = 1;
    }
}   // end of main()
```

IDE & GUI

- Валидация на вх. Данни
  - Елемент на потребителски GUI интерфейс е контролата EditTextBox, служи за вход.
  - Описание като ресурс без валидация

```
EDITTEXT   IDC_NR, 30, 30, 70, 16
```

  - Описание като ресурс с валидация

```
EDITTEXT   IDC_NR, 30, 30, 70, 16, ES_NUMBER
```

- Други свойства на контролата
  - ES_MULTILINE
  - ES_READONLY
  - ES_PASSWORD
  - WS_TABSTOP

JavaScript валидация на форми

JavaScript служи за валидация на вх. Данни в HTML форми преди съдържанието на формата да се изпрати към сървер.

Примери за типични данни от HTML форма, които да се валидират с JavaScript:

- has the user left required fields empty?
- has the user entered a valid e-mail address?
- has the user entered a valid date?
- has the user entered text in a numeric field?

Валидация
на
Празни полета за вход

Празни полета за вход

Функцията от сл. Слайд проверява дали едно поле от формата е оставено празно преди submit. При празно поле, an alert box alerts a message и функцията връща false. Ако полето е заредено, функцията връща true (т.е. Данните са ОК):

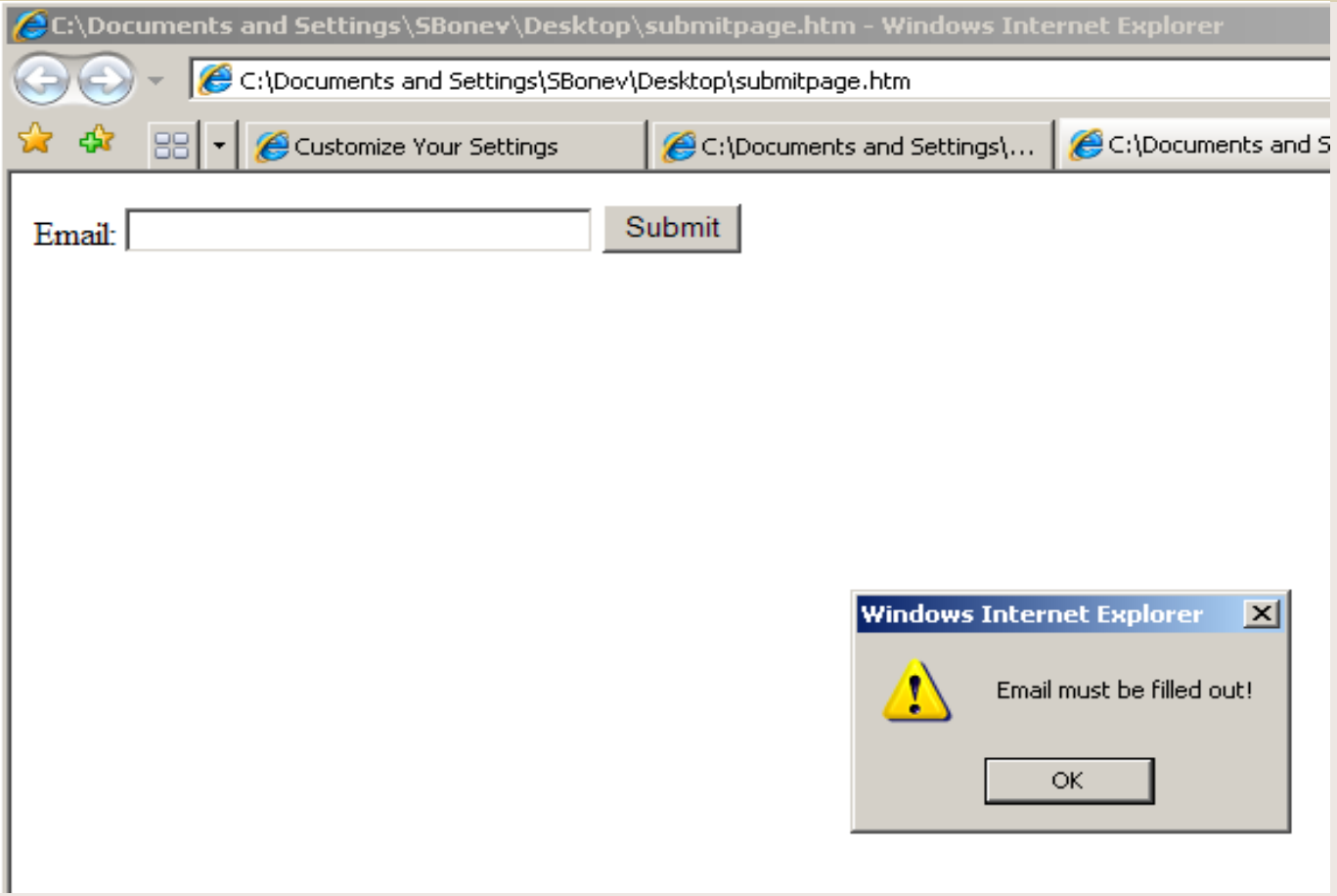
Празни полета за вход

```
function
  validate_required(field,alerttxt)
{
  with (field)
  {
   if (value==null || value=="")
     {
        alert(alerttxt);
        return false
     }
   else {return true}
  }
}
```

Празни полета за вход
Целият скрипт, Формата в HTML ф-т:

```
<html>
<head>
<script type="text/javascript">
function validate_required(field,alerttxt)
{
with (field) {
   if (value==null||value=="")  {alert(alerttxt);return false}
   else {return true}
            }
}
function validate_form(thisform)
{
with (thisform) {
if (validate_required(email,"Email must be filled out!")==false)
  {email.focus();return false}
                 }
}
</script>
</head>
<body>
<form action="submitpage.htm"
onsubmit="return validate_form(this)"
method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Празно вх. поле, Submit бутон



Валидация на E-mail адрес

Валидация на E-mail адрес

Функцията от сл. Слайд проверява дали едно поле от формата съдържа email с правилен синтаксис.

- Следва да съдържа @ sign and at least 1 dot (.).
- @ не трябва да е първи символ,
- Последният dot знак следва да е най-малко 1 символ след знака @ sign:

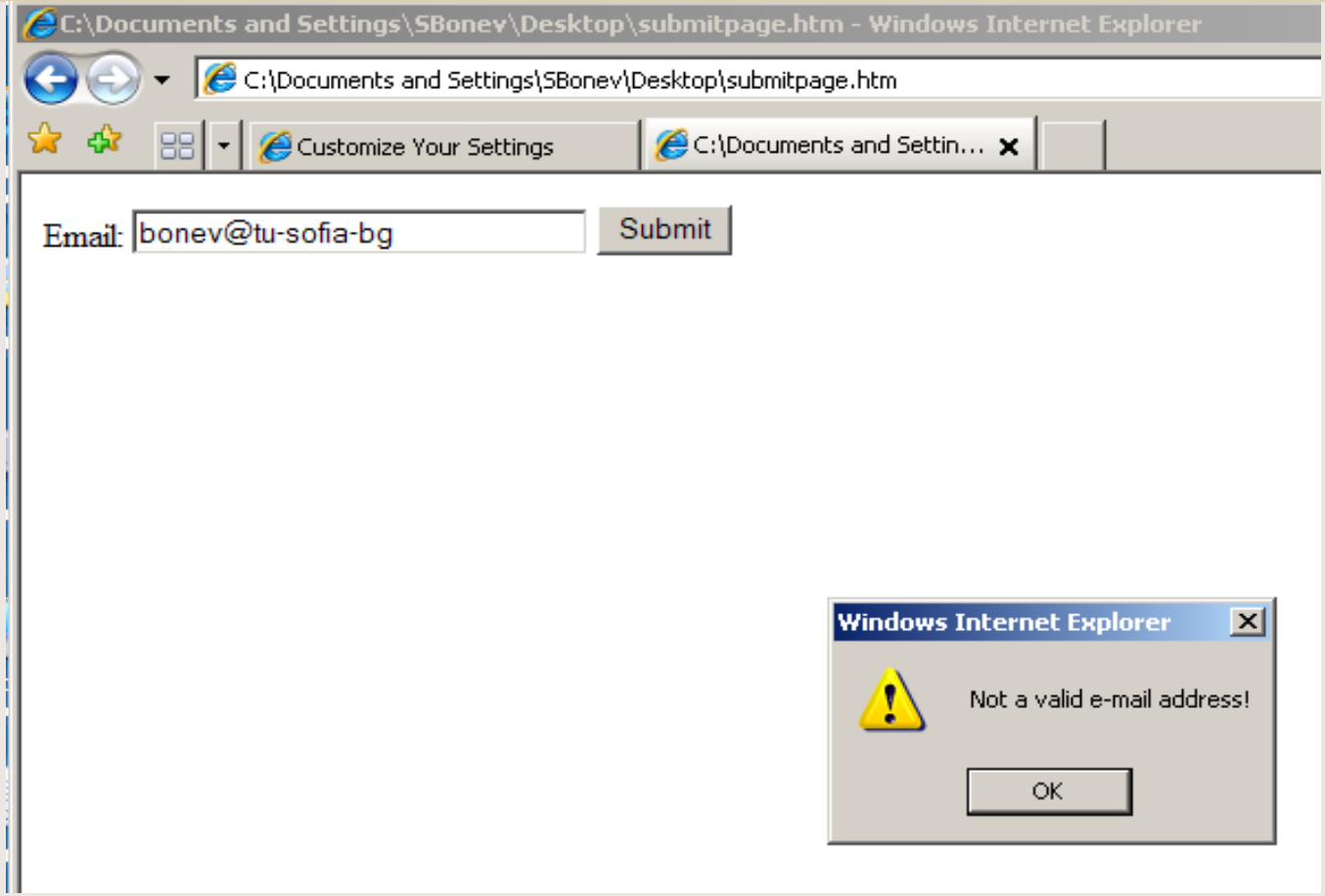
Валидация на E-mail адрес

```
function validate_email(field,alerttxt)
{
  with (field)
  {
   apos=value.indexOf("@")
   dotpos=value.lastIndexOf(".")
   if (apos<1||dotpos-apos<2)
     {
      alert(alerttxt);    return false
     }
   else {return true}
  }
}
```

Валидация на E-mail адрес
Целият скрипт, Формата в HTML ф-т:

```
<html>
<head>
<script type="text/javascript">
function validate_email(field,alerttxt)
{
with (field)  {
apos=value.indexOf("@")
dotpos=value.lastIndexOf(".")
if (apos<1||dotpos-apos<2)  {alert(alerttxt);return false}
else {return true}
              }
}
function validate_form(thisform)
{
with (thisform)  {
if (validate_email(email,"Not a valid e-mail address!")==false)
  {email.focus();return false}
                 }
}
</script>
</head>
<body>
<form action="submitpage.htm"
onsubmit="return validate_form(this);"
method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Грешен e-mail, Submit бутон



Digression on JavaScript functionality: Communicate with the User

Digression on JavaScript functionality:

- Communicate with the User
  - Display an Alert Message
    - An alert box
    - A confirm box
    - A user-input prompt box
  - Write Text to the Browser Window
    - Document.write(…)

Display an Alert Message

- JS provides three types of pop-up dialog boxes for use in applications
  - An alert box
  - A confirm box
  - A user-input prompt box

an Alert box

- Basic alert messages are displayed using the built-in alert function: Example:

```
function AlertBoxTest()
{
 alert("First message displayed in alert box");
}

AlertBoxTest();
```

A Confirm box

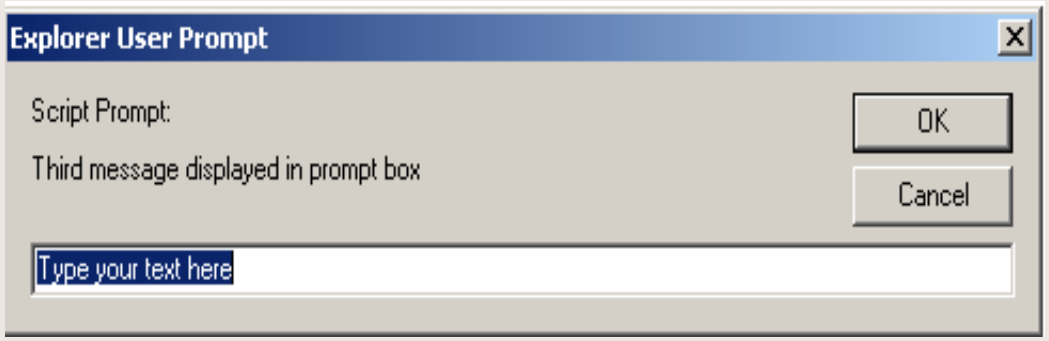
- A confirm box acts as alert box, except it displays both OK and Cancel buttons to the user. The confirm box tells the program which button the user chose, allowing two different outcomes. Example:

```
function ConfirmBoxTest()
{
 if (confirm("Second message displayed in confirm box") == true)
     { document.write("OK pressed“) }
  else
     { document.write( "Cancel pressed“)  }
}


ConfirmBoxTest();
```

A User-input prompt box

- The user-input prompt box allows the program to ask for a typed response to a message. Example:

```
function PromptBoxTest()
{
 var a;
 a = prompt("Third message displayed in prompt box","Type
    your text here");
 document.write(a)
}

PromptBoxTest();
```



Write Text to the Browser Window

- Many JS programs use **`document.write()`** function to print text into the web browser window. This may be used for error reporting, to support dynamic functions, etc.
- Example:

```
document.write("<br>Text sent to browser window")
var a = 218
document.write(“<br><b>a = </b>" + a)
```

Благодаря
За
Вниманието