BİL-142 Bilgisayar Programlama II (C/C++)

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Konular

- Giriş
- Pointer Değişken Tanımı
- Pointer Operatörleri
- Pointer ile Fonksiyonlara Parametre Gönderme
- Pointer ile `const` Kullanımı
- `sizeof` Operatörleri
- Pointer İfadeleri ve Pointer Aritmetiği
- Pointer'larla Diziler Arasındaki İlişki
- Pointer Dizileri
- Pointer Tabanlı String İşlemleri

Giriş

- Pointer'lar hem pass-by-value hemde pass-by-reference şeklinde kullanılabilir.
- Pointer'lar boyutu dinamik olarak büyüyüp küçülebilen yapılar (bağlı listeler, kuyruklar, yığınlar ve ağaçlar) oluşturmak için kullanılabilir.

3

Konular

- Giriş
- Pointer Değişken Tanımı
- Pointer Operatörleri
- Pointer ile Fonksiyonlara Parametre Gönderme
- Pointer ile **`const`** Kullanımı
- **`sizeof`** Operatörleri
- Pointer İfadeleri ve Pointer Aritmetiği
- Pointer'larla Diziler Arasındaki İlişki
- Pointer Dizileri
- Pointer Tabanlı String İşlemleri

Pointer Değişken Tanımı

- Pointer değişkenler değer yerine hafıza adreslerini tutar.
- Bir değişken bir değeri doğrudan gösterir, pointer bir değeri dolaylı olarak gösterir.
- Pointer tanımlarken değişken adının önüne * konulur.

```
int *countPtr, count;

double *xPtr, *yPtr;
```

- Bir pointer tanımlandığında 0, NULL veya bir adresle başlatılabilir.
- NULL <iostream> başlık dosyasında tanımlıdır.

5

Konular

- Giriş
- Pointer Değişken Tanımı
- Pointer Operatörleri
- Pointer ile Fonksiyonlara Parametre Gönderme
- Pointer ile **`const`** Kullanımı
- **`sizeof`** Operatörleri
- Pointer İfadeleri ve Pointer Aritmetiği
- Pointer'larla Diziler Arasındaki İlişki
- Pointer Dizileri
- Pointer Tabanlı String İşlemleri

Pointer Operatörleri

- Adres operatörü (&) bir değişkenin adresini döndürür.

```
int y = 5;
int *yPtr;

yPtr = &y;
```

&yPtr = 500000

***yPtr = 5**
y = 5

&y = 600000
yPtr = 600000

- **`*&yPtr`** (600000 - pointer adresinin gösterdiği) ile **`&*yPtr`** (600000 - pointer'ın gösterdiğinin adresi) aynı değeri döndürür.

7

Pointer Operatörleri

- Örnek

```
1  #include <iostream>
2  using std::cout;
3  using std::endl;
4
5  int y = 5;
6  int *yPtr;
7
8
9  int main()
10 {
11     yPtr = &y;
12
13     cout << "y = " << y << endl;
14     cout << "&y = " << &y << endl;
15     cout << "yPtr = " << yPtr << endl;
16     cout << "*yPtr = " << *yPtr << endl;
17     cout << "&yPtr = " << &yPtr << endl;
18     cout << "&*yPtr = " << &*yPtr << endl;
19     cout << "*&yPtr = " << *&yPtr << endl;
20
21     system("PAUSE");
22     return 0;
23 }
```

```
c:\Documents and Settings\m.ali\My Docume
y = 5
&y = 00419004
yPtr = 00419004
*yPtr = 5
&yPtr = 00419150
&*yPtr = 00419004
*&yPtr = 00419004
Press any key to continue . . .
```

8

Pointer Operatörleri

- Örnek

```
1  // Fig. 8.4: fig08_04.cpp
2  // Using the & and * operators.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  int main()
8  {
9     int a; // a is an integer
10    int *aPtr; // aPtr is an int * -- pointer to an integer
11
12    a = 7; // assigned 7 to a
13    aPtr = &a; // assign the address of a to aPtr
14
15    cout << "The address of a is " << &a
16       << "\nThe value of aPtr is " << aPtr;
17    cout << "\n\nThe value of a is " << a
18       << "\nThe value of *aPtr is " << *aPtr;
19    cout << "\n\nShowing that * and & are inverses of "
20       << "each other.\n&*aPtr = " << &*aPtr
21       << "\n*&aPtr = " << *&aPtr << endl;
22    system("PAUSE");
23    return 0; // indicates successful termination
24 } // end main
```

```
c:\Documents and Settings\m.ali\My Documents\Visual Studio
The address of a is 0012FF60
The value of aPtr is 0012FF60

The value of a is 7
The value of *aPtr is 7

Showing that * and & are inverses of each other.
&*aPtr = 0012FF60
*&aPtr = 0012FF60
Press any key to continue . . .
```

9

Konular

- Giriş
- Pointer Değişken Tanımı
- Pointer Operatörleri
- Pointer ile Fonksiyonlara Parametre Gönderme
- Pointer ile **`const`** Kullanımı
- **`sizeof`** Operatörleri
- Pointer İfadeleri ve Pointer Aritmetiği
- Pointer'larla Diziler Arasındaki İlişki
- Pointer Dizileri
- Pointer Tabanlı String İşlemleri

Pointer ile Fonksiyonlara Parametre Gönderme

- C++ ile fonksiyonlara 3 farklı yolla parametre gönderilebilir.
  - Pass-by-value
  - Pass-by-reference
  - Pass-by-reference with pointer
- Pass-by-value ile bir değişken fonksiyona gönderilir ve fonksiyondan değer geri döndürülmez.
- Pass-by-reference ile bir değişkenin adresi (&) operatörü ile gönderilir ve fonksiyonda doğrudan değişkenin adresindeki değer değiştirilir.
- Pass-by-reference with pointer ile bir değişkenin adresi (*) operatörü ile gönderilir.

11

Pointer ile Fonksiyonlara Parametre Gönderme

```
// Fig. 6.19: fig06_19.cpp
// Comparing pass-by-value and pass-by-reference with references.
#include <iostream>
using std::cout;
using std::endl;

int squareByValue( int ); // function prototype (value pass)

void squareByReference( int & ); // function prototype (reference pass)

int main()
{
    int x = 2; // value to square using squareByValue
    int z = 4; // value to square using squareByReference

    // demonstrate squareByValue
    cout << "x = " << x << " before squareByValue\n";
    cout << "Value returned by squareByValue: "
         << squareByValue( x ) << endl;
    cout << "x = " << x << " after squareByValue\n" << endl;

    // demonstrate squareByReference
    cout << "z = " << z << " before squareByReference" << endl;

    squareByReference( z );
    cout << "z = " << z << " after squareByReference" << endl;
    system("PAUSE");
    return 0; // indicates successful termination
} // end main
```

12

Pointer ile Fonksiyonlara Parametre Gönderme

```
31 // squareByValue multiplies number by itself, stores the
32 // result in number and returns the new value of number
33 int squareByValue( int number )
34 {
35     return number *= number; // caller's argument not modified
36 } // end function squareByValue
37
38 // squareByReference multiplies numberRef by itself and stores the result
39 // in the variable to which numberRef refers in function main
40 void squareByReference( int &numberRef )
41 {
42      numberRef *= numberRef; // caller's argument modified
43 } // end function squareByReference
```

```
c:\Documents and Settings\m.ali\My Documents\Visual Studio 2008\Pr
x = 2 before squareByValue
Value returned by squareByValue: 4
x = 2 after squareByValue

z = 4 before squareByReference
z = 16 after squareByReference
Press any key to continue . . .
```

13

Pointer ile Fonksiyonlara Parametre Gönderme

- pass-by-value ile parametre gönderimi

```
1  // Fig. 8.6: fig08_06.cpp
2  // Cube a variable using pass-by-value.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  int cubeByValue( int ); // prototype
8
9  int main()
10 {
11    int number = 5;
12
13    cout << "The original value of number is " << number;
14
15    number = cubeByValue( number ); // pass number by value to cubeByValue
16    cout << "\nThe new value of number is " << number << endl;
17    system("PAUSE");
18    return 0; // indicates successful termination
19 } // end main
20
21 // calculate and return cube of integer argument
22 int cubeByValue( int n )
23 {
24    return n * n * n; // cube local variable n and return result
25 } // end function cubeByValue
```

```
c:\Documents and Settings\m.ali\My Documents
The original value of number is 5
The new value of number is 125
Press any key to continue . . .
```

14

Pointer ile Fonksiyonlara Parametre Gönderme

- pass-by-reference with pointer ile parametre gönderimi

```
1  // Fig. 8.7: fig08_07.cpp
2  // Cube a variable using pass-by-reference with a pointer argument.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  void cubeByReference( int * ); // prototype
8
9  int main()
10 {
11    int number = 5;
12
13    cout << "The original value of number is " << number;
14
15    cubeByReference( &number ); // pass number address to cubeByReference
16
17    cout << "\nThe new value of number is " << number << endl;
18    system("PAUSE");
19    return 0; // indicates successful termination
20 } // end main
21
22 // calculate cube of *nPtr; modifies variable number
23 void cubeByReference( int *nPtr )
24 {
25    *nPtr = *nPtr * *nPtr * *nPtr; // cube *nPtr
26 } // end function cubeByReference
27
```

```
int number = 5;
int *numberPtr;
numberPtr = &number;
cubeByReference( numberPtr );
```

```
c:\Documents and Settings\m.ali\My Documer
The original value of number is 5
The new value of number is 125
Press any key to continue . . .
```

15

Konular

- Giriş
- Pointer Değişken Tanımı
- Pointer Operatörleri
- Pointer ile Fonksiyonlara Parametre Gönderme
- Pointer ile **`const`** Kullanımı
- **`sizeof`** Operatörleri
- Pointer İfadeleri ve Pointer Aritmetiği
- Pointer'larla Diziler Arasındaki İlişki
- Pointer Dizileri
- Pointer Tabanlı String İşlemleri

Pointer ile `const` Kullanımı

- Bir fonksiyon parametrelerini **`const`** olarak kullanmanın farklı yolları vardır.
- Pointer ile const kullanımı 4 farklı şekilde yapılabilir:
  - Nonconstant pointer to nonconstant data
  - Nonconstant pointer to constant data
  - Constant pointer to nonconstant data
  - Constant pointer to constant data

17

Pointer ile `const` Kullanımı

Nonconstant pointer to nonconstant data

- Nonconstant pointer to nonconstant data kullanımında const kullanılmaz.
- Nonconstant pointer değeri değiştirilebilir.
- Nonconstant data değeri değiştirilebilir.
- Bir karakter dizisinde dizi adı ilk elemanın pointer'ıdır.
- Dizi adını alan pointer değeri artırılarak sonraki elemanlara ulaşılabilir.

18

Pointer ile `const` Kullanımı

Nonconstant pointer to nonconstant data

```
1  // Fig. 8.10: fig08_10.cpp
2  // Converting lowercase letters to uppercase letters
3  // using a non-constant pointer to non-constant data.
4  #include <iostream>
5  using std::cout;
6  using std::endl;
7
8  #include <cctype> // prototypes for islower and toupper
9  using std::islower;
10 using std::toupper;
11
12 void convertToUppercase( char * );
13
14 int main()
15 {
16    char phrase[] = "characters and $32.98";
17
18    cout << "The phrase before conversion is: " << phrase;
19    convertToUppercase( phrase );
20    cout << "\nThe phrase after conversion is:  " << phrase << endl;
21    system("PAUSE");
22    return 0; // indicates successful termination
23 } // end main
24
```


Pointer ile `const` Kullanımı

Nonconstant pointer to nonconstant data - devam

```
25 // convert string to uppercase letters
26 void convertToUppercase( char *sPtr )
27 {
28    while ( *sPtr != '\0' ) // loop while current character is not '\0'
29    {
30       if ( islower( *sPtr ) ) // if character is lowercase,
31          *sPtr = toupper( *sPtr ); // convert to uppercase
32
33       sPtr++; // move sPtr to next character in string
34    } // end while
35 } // end function convertToUppercase
36
```

Sonraki karaktere geçilir

```
c:\Documents and Settings\m.ali\My Documents\Visual Studio 2008\
The phrase before conversion is: characters and $32.98
The phrase after conversion is:  CHARACTERS AND $32.98
Press any key to continue . . .
```

Pointer ile `const` Kullanımı

Nonconstant pointer to constant data

- Nonconstant pointer to constant data kullanımında const ile tanımlama yapılır.
- Nonconstant pointer değeri değiştirilebilir.
- Pointer data üzerinde değişiklik yapamaz.

21

Pointer ile `const` Kullanımı

Nonconstant pointer to constant data

```
1  // Fig. 8.11: fig08_11.cpp
2  // Printing a string one character at a time using
3  // a non-constant pointer to constant data.
4  #include <iostream>
5  using std::cout;
6  using std::endl;
7
8  void printCharacters( const char * ); // print using pointer to const data
9
10 int main()
11 {
12     const char phrase[] = "print characters of a string";
13
14     cout << "The string is:\n";
15     printCharacters( phrase ); // print characters in phrase
16     cout << endl;
17
18     system("PAUSE");
19     return 0; // indicates successful termination
20 } // end main
21
22 // sPtr can be modified, but it cannot modify the character to which
23 // it points, i.e., sPtr is a "read-only" pointer
24 void printCharacters( const char *sPtr )
25 {
26     for ( ; *sPtr != '\0'; sPtr++ ) // no initialization
27     cout << *sPtr; // display character without modification
28 } // end function printCharacters
```

```
c:\Documents and Settings\m.ali\My Docur
The string is:
print characters of a string
Press any key to continue . . .
```

22

Pointer ile `const` Kullanımı

Nonconstant pointer to constant data – devam

- `const` pointer'la gösterilen değer değiştirilemez.

```
1  // Fig. 8.12: fig08_12.cpp
2  // Attempting to modify data through a
3  // non-constant pointer to constant data.
4
5  void f( const int * ); // prototype
6
7  int main()
8  {
9      int y;
10
11     f( &y ); // f attempts illegal modification
12
13     system("PAUSE");
14     return 0; // indicates successful termination
15 } // end main
16
17 // xPtr cannot modify the value of constant variable to which it points
18 void f( const int *xPtr )
19 {
20     *xPtr = 100; // error: cannot modify a const object
21 } // end function f
```

23

Pointer ile `const` Kullanımı

Constant pointer to nonconstant data

- Constant pointer to nonconstant data kullanımında pointer sabit bir adresi gösterir.
- Constant pointer ile gösterilen adres içeriği değiştirilebilir.
- Constant pointer tanımlanırken initiliaze edilmelidir.

24

Pointer ile `const` Kullanımı

Constant pointer to nonconstant data

```
1  // Fig. 8.13: fig08_13.cpp
2  // Attempting to modify a constant pointer to non-constant data.
3
4  int main()
5  {
6      int x, y;
7
8      // ptr is a constant pointer to an integer that can
9      // be modified through ptr, but ptr always points to the
10     // same memory location.
11
12     int * const ptr = &x; // const pointer must be initialized
13
14     *ptr = 7; // allowed: *ptr is not const
15     ptr = &y; // error: ptr is const; cannot assign to it a new address
16
17     system("PAUSE");
18     return 0; // indicates successful termination
19 } // end main
```

```
error C3892: 'ptr' : you cannot assign to a variable that is const
```


Pointer ile `const` Kullanımı

Constant pointer to constant data

- Constant pointer to constant data kullanımında pointer sabit bir adresi gösterir ve adresin içeriğide değiştirilemez.
- Constant pointer tanımlanırken initiliaze edilmelidir.

Pointer ile `const` Kullanımı

Constant pointer to constant data

```
1  // Fig. 8.14: fig08_14.cpp
2  // Attempting to modify a constant pointer to constant data.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  int main()
8  {
9      int x = 5, y;
10
11     // ptr is a constant pointer to a constant integer.
12     // ptr always points to the same location; the integer
13     // at that location cannot be modified.
14
15     const int *const ptr = &x;
16
17     cout << *ptr << endl;
18
19     *ptr = 7; // error: *ptr is const; cannot assign new value
20     ptr = &y; // error: ptr is const; cannot assign new address
21
22     system("PAUSE");
23     return 0; // indicates successful termination
24 } // end main
```

27

Konular

- Giriş
- Pointer Değişken Tanımı
- Pointer Operatörleri
- Pointer ile Fonksiyonlara Parametre Gönderme
- Pointer ile `const` Kullanımı
- `sizeof` Operatörleri
- Pointer İfadeleri ve Pointer Aritmetiği
- Pointer'larla Diziler Arasındaki İlişki
- Pointer Dizileri
- Pointer Tabanlı String İşlemleri

`sizeof` Operatörleri

- **`sizeof`** operatörü bir dizinin toplam boyutunu byte olarak verir.
- **`sizeof(array)`** ifadesi array dizisinin hafızadaki toplam boyutunu byte olarak verir.

```
double realArray[22];

sizeof realArray / sizeof (double); // eleman sayısını verir
```


`sizeof` Operatörleri

```
// Fig. 8.16: fig08_16.cpp
// Sizeof operator when used on an array name
// returns the number of bytes in the array.
#include <iostream>
using std::cout;
using std::endl;

size_t getSize( double * ); // prototype

int main()
{
   double array[ 20 ]; // 20 doubles; occupies 160 bytes on our system

   cout << "The number of bytes in the array is " << sizeof( array );

   cout << "\nThe number of bytes returned by getSize is "
      << getSize( array ) << endl;

   system("PAUSE");
   return 0; // indicates successful termination
} // end main

// return size of ptr
size_t getSize( double *ptr )
{
   return sizeof( ptr );
} // end function getSize
```

```
c:\Documents and Settings\m.ali\My Documents\Visual Stu
The number of bytes in the array is 160
The number of bytes returned by getSize is 4
Press any key to continue . . .
```

`sizeof` Operatörleri

- Örnek: Temel türler, dizi ve pointer boyutunun belirlenmesi.

```
// Fig. 8.17: fig08_17.cpp
// Demonstrating the sizeof operator.
#include <iostream>
using std::cout;
using std::endl;

int main()
{
     char c; // variable of type char
     short s; // variable of type short
     int i; // variable of type int
     long l; // variable of type long
     float f; // variable of type float
     double d; // variable of type double
     long double ld; // variable of type long double
     int array[ 20 ]; // array of int
     int *ptr = array; // variable of type int *

```

31

`sizeof` Operatörleri

- Örnek: Temel türler, dizi ve pointer boyutunun belirlenmesi.

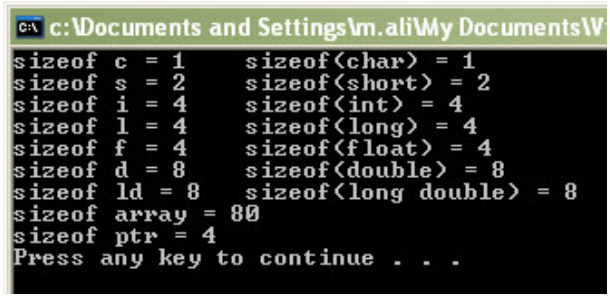
```
     cout << "sizeof c = " << sizeof c
          << "\tsizeof(char) = " << sizeof( char )
          << "\nsizeof s = " << sizeof s
          << "\tsizeof(short) = " << sizeof( short )
          << "\nsizeof i = " << sizeof i
          << "\tsizeof(int) = " << sizeof( int )
          << "\nsizeof l = " << sizeof l
          << "\tsizeof(long) = " << sizeof( long )
          << "\nsizeof f = " << sizeof f
          << "\tsizeof(float) = " << sizeof( float )
          << "\nsizeof d = " << sizeof d
          << "\tsizeof(double) = " << sizeof( double )
          << "\nsizeof ld = " << sizeof ld
          << "\tsizeof(long double) = " << sizeof( long double )
          << "\nsizeof array = " << sizeof array
          << "\nsizeof ptr = " << sizeof ptr << endl;

     system("PAUSE");
     return 0; // indicates successful termination
} // end main
```

32

`sizeof` Operatörleri

- Örnek: Temel türler, dizi ve pointer boyutunun belirlenmesi.



33

Konular

- Giriş
- Pointer Değişken Tanımı
- Pointer Operatörleri
- Pointer ile Fonksiyonlara Parametre Gönderme
- Pointer ile `const` Kullanımı
- `sizeof` Operatörleri
- Pointer İfadeleri ve Pointer Aritmetiği
- Pointer'larla Diziler Arasındaki İlişki
- Pointer Dizileri
- Pointer Tabanlı String İşlemleri

Pointer İfadeleri ve Pointer Aritmetiği

- Pointer'lar birçok aritmetik işlem yapılabilir (++, --).

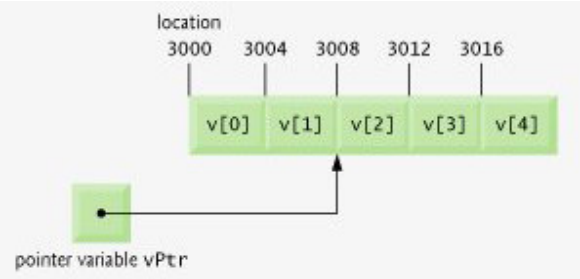
```
int *vPtr = v;

int *vPtr = &v[ 0 ];

vPtr += 2;

vPtr -= 4;

++vPtr;
vPtr++;
```



vPtr += 2; den sonraki durum

35

Konular

- Giriş
- Pointer Değişken Tanımı
- Pointer Operatörleri
- Pointer ile Fonksiyonlara Parametre Gönderme
- Pointer ile **`const`** Kullanımı
- **`sizeof`** Operatörleri
- Pointer İfadeleri ve Pointer Aritmetiği
- Pointer'larla Diziler Arasındaki İlişki
- Pointer Dizileri
- Pointer Tabanlı String İşlemleri

Pointer'larla Diziler Arasındaki İlişki

- Bir dizi adı const pointer olarak düşünülebilir.
- Aşağıda 5 elemanlı dizi ve integer pointer tanımlanmıştır.

```
int b[ 5 ];
int *bPtr;
```

- Dizinin adı ilk elemanı gösteren const pointer'dır.

```
bPtr = b;
```

- İlk elemanın adresini alarak yapılan aşağıdaki ifadeyle aynıdır.

```
bPTr = &b[ 0 ];
```

- Dizinin 4.elemanı aşağıdaki gibi gösterilebilir.

```
*(bPtr + 3)
```

- Dizinin 4.elemanı aşağıdaki gibi de gösterilebilir.

```
&b[ 3 ]
```

37

Pointer'larla Diziler Arasındaki İlişki

- Dizinin 4.elemanı dizi adıylada gösterilebilir.

```
*(b + 3)
```

- Pointer'larda dizilerde olduğu gibi alt indis kullanabilir.

```
bPtr[ 1 ]
```

- Dizi adı const pointer olduğu için aşağıdaki deyim hatalıdır.

```
b+= 3
```

38

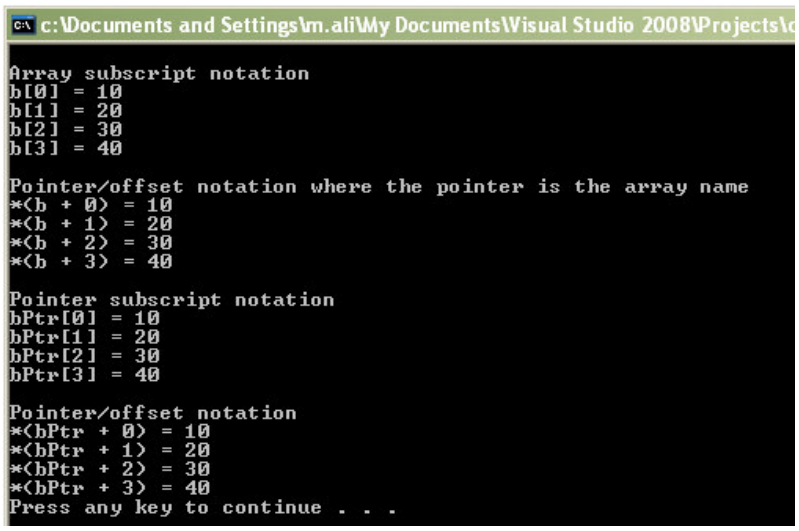
Pointer'larla Diziler Arasındaki İlişki

```
1  // Fig. 8.20: fig08_20.cpp
2  // Using subscripting and pointer notations with arrays.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  int main()
8  {
9      int b[] = { 10, 20, 30, 40 }; // create 4-element array b
10     int *bPtr = b; // set bPtr to point to array b
11
12     // output array b using array subscript notation
13     cout << "Array b printed with:\n\nArray subscript notation\n";
14
15     for ( int i = 0; i < 4; i++ )
16         cout << "b[" << i << "] = " << b[ i ] << '\n';
17
18     // output array b using the array name and pointer/offset notation
19     cout << "\nPointer/offset notation where "
20          << "the pointer is the array name\n";
21
```


Pointer'larla Diziler Arasındaki İlişki

```
22     for ( int offset1 = 0; offset1 < 4; offset1++ )
23         cout << "*(b + " << offset1 << ") = " << *( b + offset1 ) << '\n';
24
25     // output array b using bPtr and array subscript notation
26     cout << "\nPointer subscript notation\n";
27
28     for ( int j = 0; j < 4; j++ )
29         cout << "bPtr[" << j << "] = " << bPtr[ j ] << '\n';
30
31     cout << "\nPointer/offset notation\n";
32
33     // output array b using bPtr and pointer/offset notation
34     for ( int offset2 = 0; offset2 < 4; offset2++ )
35         cout << "*(bPtr + " << offset2 << ") = "
36              << *( bPtr + offset2 ) << '\n';
37
38     system("PAUSE");
39     return 0; // indicates successful termination
40 } // end main
```

Pointer'larla Diziler Arasındaki İlişki



41

Pointer'larla Diziler Arasındaki İlişki

```
1  // Fig. 8.21: fig08_21.cpp
2  // Copying a string using array notation and pointer notation.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  void copy1( char *, const char * ); // prototype
8  void copy2( char *, const char * ); // prototype
9
10 int main()
11 {
12     char string1[ 10 ];
13     char *string2 = "Hello";
14     char string3[ 10 ];
15     char string4[] = "Good Bye";
16
17     copy1( string1, string2 ); // copy string2 into string1
18     cout << "string1 = " << string1 << endl;
19
20     copy2( string3, string4 ); // copy string4 into string3
21     cout << "string3 = " << string3 << endl;
22
23     system("PAUSE");
24     return 0; // indicates successful termination
25 } // end main
26
```

42

Pointer'larla Diziler Arasındaki İlişki

```
27  // copy s2 to s1 using array notation
28  void copy1( char * s1, const char * s2 )
29  {
30      // copying occurs in the for header
31     for ( int i = 0; ( s1[ i ] = s2[ i ] ) != '\0'; i++ )
32     ; // do nothing in body
33  } // end function copy1
34
35  // copy s2 to s1 using pointer notation
36  void copy2( char *s1, const char *s2 )
37  {
38      // copying occurs in the for header
39     for ( ; ( *s1 = *s2 ) != '\0'; s1++, s2++ )
40     ; // do nothing in body
41  } // end function copy2
```

```
c:\Documents and Settings\m.ali\My Documen
string1 = Hello
string3 = Good Bye
Press any key to continue . . .
```

43

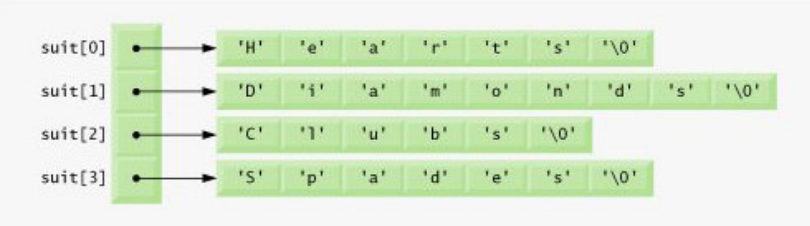
Konular

- Giriş
- Pointer Değişken Tanımı
- Pointer Operatörleri
- Pointer ile Fonksiyonlara Parametre Gönderme
- Pointer ile **`const`** Kullanımı
- **`sizeof`** Operatörleri
- Pointer İfadeleri ve Pointer Aritmetiği
- Pointer'larla Diziler Arasındaki İlişki
- Pointer Dizileri
- Pointer Tabanlı String İşlemleri

Pointer Dizileri

- Pointer tabanlı string'ler (string array) yaygın kullanılmaktadır.

```
const char *suit[ 4 ] =
{ "Hearts", "Diamonds", "Clubs", "Spades" };
```



45

Konular

- Giriş
- Pointer Değişken Tanımı
- Pointer Operatörleri
- Pointer ile Fonksiyonlara Parametre Gönderme
- Pointer ile **`const`** Kullanımı
- **`sizeof`** Operatörleri
- Pointer İfadeleri ve Pointer Aritmetiği
- Pointer'larla Diziler Arasındaki İlişki
- Pointer Dizileri
- Pointer Tabanlı String İşlemleri

Pointer Tabanlı String İşlemleri

- C++ ile bir string karakter dizisi olarak tanımlanabilir.
- Bir string pointer ise karakter dizisi pointer'ı olarak tanımlanabilir.

```
char color[] = "blue";
const char *colorPtr = "blue";
char color[] = { 'b', 'l', 'u', 'e', '\0' };

word[20]

cin >> word;
cin >> setw( 20 ) >> word;

char sentence[ 80 ];
cin.getline( sentence, 80, '\n' );
cin.getline( sentence, 80 );
```

#include <cstring>

47

Pointer Tabanlı String İşlemleri

String fonksiyonları (**<cstring>** başlık dosyası include edilmelidir)

Function prototype	Function description
`char *strcpy( char *s1, const char *s2 );`	
	Copies the string `s2` into the character array `s1`. The value of `s1` is returned.
`char *strncpy( char *s1, const char *s2, size_t n );`	
	Copies at most `n` characters of the string `s2` into the character array `s1`. The value of `s1` is returned.
`char *strcat( char *s1, const char *s2 );`	
	Appends the string `s2` to `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned.
`char *strncat( char *s1, const char *s2, size_t n );`	
	Appends at most `n` characters of string `s2` to string `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned.

48

Pointer Tabanlı String İşlemleri

String fonksiyonları – devam

`int strcmp( const char *s1, const char *s2 );`	
	Compares the string `s1` with the string `s2`. The function returns a value of zero, less than zero (usually `-1`) or greater than zero (usually `1`) if `s1` is equal to, less than or greater than `s2`, respectively.
`int strncmp( const char *s1, const char *s2, size_t n );`	
	Compares up to `n` characters of the string `s1` with the string `s2`. The function returns zero, less than zero or greater than zero if the `n`-character portion of `s1` is equal to, less than or greater than the corresponding `n`-character portion of `s2`, respectively.
`char *strtok( char *s1, const char *s2 );`	
	A sequence of calls to `strtok` breaks string `s1` into "tokens"logical pieces such as words in a line of text. The string is broken up based on the characters contained in string `s2`. For instance, if we were to break the string `"this:is:a:string"` into tokens based on the character `':'`, the resulting tokens would be `"this"`, `"is"`, `"a"` and `"string"`. Function `strtok` returns only one token at a time, however. The first call contains `s1` as the first argument, and subsequent calls to continue tokenizing the same string contain `NULL` as the first argument. A pointer to the current token is returned by each call. If there are no more tokens when the function is called, `NULL` is returned.
`size_t strlen( const char *s );`	
	Determines the length of string `s`. The number of characters preceding the terminating null character is returned.

49

Pointer Tabanlı String İşlemleri

`strcpy` ve **`strncpy`** ile string kopyalama

- **`strcpy`** fonksiyonu ikinci parametresini birinci parametresine kopyalar.
- **`strncpy`** fonksiyonu **`strcpy`** fonksiyonundan farklı olarak kopyalanacak uzunluğu belirler.
- **`strncpy`** fonksiyonunda üçüncü parametre değeri ikinci parametreden büyükse kalanlar null karakter ile doldurulur.

50

Pointer Tabanlı String İşlemleri

```
1  // Fig. 8.31: fig08_31.cpp
2  // Using strcpy and strncpy.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include <cstring> // prototypes for strcpy and strncpy
8  using std::strcpy;
9  using std::strncpy;
10
11 int main()
12 {
13    char x[] = "Happy Birthday to You"; // string length 21
14    char y[ 25 ];
15    char z[ 15 ];
16
17    strcpy( y, x ); // copy contents of x into y
18
19    cout << "The string in array x is: " << x
20       << "\nThe string in array y is: " << y << '\n';
21
22    // copy first 14 characters of x into z
23    strncpy( z, x, 14 ); // does not copy null character
24    z[ 14 ] = '\0'; // append '\0' to z's contents
25
26    cout << "The string in array z is: " << z << endl;
27
28    system("PAUSE");
29    return 0; // indicates successful termination
30 } // end main
```

```
c:\Documents and Settings\m.ali\My Documents\Visual Stu
The string in array x is: Happy Birthday to You
The string in array y is: Happy Birthday to You
The string in array z is: Happy Birthday
Press any key to continue . . .
```

51

Pointer Tabanlı String İşlemleri

`strcat` ve **`strncat`** ile string birleştirme

- **`strcat`** fonksiyonu ikinci parametresini birinci parametresine ekler.
- **`strncat`** fonksiyonu **`strcat`** fonksiyonundan farklı olarak eklenecek uzunluğu belirler ve sonuca null karakter ekler.

52

Pointer Tabanlı String İşlemleri

```
1  // Fig. 8.32: fig08_32.cpp
2  // Using strcat and strncat.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include <cstring> // prototypes for strcat and strncat
8  using std::strcat;
9  using std::strncat;
10
11 int main()
12 {
13    char s1[ 20 ] = "Happy "; // length 6
14    char s2[] = "New Year "; // length 9
15    char s3[ 40 ] = "";
16
17    cout << "s1 = " << s1 << "\ns2 = " << s2;
18
19    strcat( s1, s2 ); // concatenate s2 to s1 (length 15)
20
21    cout << "\n\nAfter strcat(s1, s2):\ns1 = " << s1 << "\ns2 = " << s2;
```

53

Pointer Tabanlı String İşlemleri

```
22
23    // concatenate first 6 characters of s1 to s3
24    strncat( s3, s1, 6 ); // places '\0' after last character
25
26    cout << "\n\nAfter strncat(s3, s1, 6):\ns1 = " << s1
27       << "\ns3 = " << s3;
28
29    strcat( s3, s1 ); // concatenate s1 to s3
30    cout << "\n\nAfter strcat(s3, s1):\ns1 = " << s1
31       << "\ns3 = " << s3 << endl;
32
33    system("PAUSE");
34    return 0; // indicates successful termination
35 } // end main
36
```

```
c:\Documents and Settings\m.ali\My Docu
s1 = Happy
s2 = New Year

After strcat(s1, s2):
s1 = Happy New Year
s2 = New Year

After strncat(s3, s1, 6):
s1 = Happy New Year
s3 = Happy

After strcat(s3, s1):
s1 = Happy New Year
s3 = Happy Happy New Year
Press any key to continue . . .
```

54

Pointer Tabanlı String İşlemleri

`strcmp` ve **`strncmp`** ile string karşılaştırma

- **`strcmp`** fonksiyonu ikinci parametresini birinci parametresiyle karşılaşırır.
- Eğer iki parametre eşitse 0 döndürür.
- Eğer birinci parametre ikinciden küçükse negatif, ikinci birinciden küçükse pozitif değer döndürür.
- **`strncmp`** fonksiyonu **`strcpy`** fonksiyonundan farklı olarak karşılaştırılacak karakter sayısını belirler.
- İki string'ten birisinde null karakter görürse durur.

Pointer Tabanlı String İşlemleri

```
1  // Fig. 8.33: fig08_33.cpp
2  // Using strcmp and strncmp.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include <iomanip>
8  using std::setw;
9
10 #include <cstring> // prototypes for strcmp and strncmp
11 using std::strcmp;
12 using std::strncmp;
13
14 int main()
15 {
16    char *s1 = "Happy New Year";
17    char *s2 = "Happy New Year";
18    char *s3 = "Happy Holidays";
19
```

Pointer Tabanlı String İşlemleri

```
20    cout << "s1 = " << s1 << "\ns2 = " << s2 << "\ns3 = " << s3
21       << "\n\nstrcmp(s1, s2) = " << setw( 2 ) << strcmp( s1, s2 )
22       << "\nstrcmp(s1, s3) = " << setw( 2 ) << strcmp( s1, s3 )
23       << "\nstrcmp(s3, s1) = " << setw( 2 ) << strcmp( s3, s1 );
24
25    cout << "\n\nstrncmp(s1, s3, 6) = " << setw( 2 )
26       << strncmp( s1, s3, 6 ) << "\nstrncmp(s1, s3, 7) = " << setw( 2 )
27       << strncmp( s1, s3, 7 ) << "\nstrncmp(s3, s1, 7) = " << setw( 2 )
28       << strncmp( s3, s1, 7 ) << endl;
29
30    system("PAUSE");
31    return 0; // indicates successful termination
32 } // end main
33
```

```
c:\Documents and Settings\m.ali\My Docum
s1 = Happy New Year
s2 = Happy New Year
s3 = Happy Holidays

strcmp(s1, s2) =  0
strcmp(s1, s3) =  1
strcmp(s3, s1) = -1

strncmp(s1, s3, 6) =  0
strncmp(s1, s3, 7) =  6
strncmp(s3, s1, 7) = -6
Press any key to continue . . .
```

57

Pointer Tabanlı String İşlemleri

`strtok` ile string parçalama

- **`strtok`** ile bir string girilen karaktere göre (delimiter) (;, space, /, ...) parçalara ayrılır.
- Ayırma için kullanılacak karakter ikinci parametre olarak verilir.

58

Pointer Tabanlı String İşlemleri

```
1  // Fig. 8.34: fig08_34.cpp
2  // Using strtok.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include <cstring> // prototype for strtok
8  using std::strtok;
9
10 int main()
11 {
12    char sentence[] = "This is a sentence with 7 tokens";
13    char *tokenPtr;
14
15    cout << "The string to be tokenized is:\n" << sentence
16       << "\n\nThe tokens are:\n\n";
17    // begin tokenization of sentence
18    tokenPtr = strtok( sentence, " " );
19
20    // continue tokenizing sentence until tokenPtr becomes NULL
21    while ( tokenPtr != NULL )
22    {
23       cout << tokenPtr << '\n';
24       tokenPtr = strtok( NULL, " " ); // get next token
25    } // end while
26    cout << "\nAfter strtok, sentence = " << sentence << endl;
27    system("PAUSE");
28    return 0; // indicates successful termination
29 } // end main
```

```
c:\Documents and Settings\m.ali\My Docu
The string to be tokenized is:
This is a sentence with 7 tokens

The tokens are:

This
is
a
sentence
with
7
tokens

After strtok, sentence = This
Press any key to continue . . .
```


Pointer Tabanlı String İşlemleri

`strlen` ile string boyutunu belirleme

```
1  // Fig. 8.35: fig08_35.cpp
2  // Using strlen.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include <cstring> // prototype for strlen
8  using std::strlen;
9
10 int main()
11 {
12    char *string1 = "abcdefghijklmnopqrstuvwxyz";
13    char *string2 = "four";
14    char *string3 = "Boston";
15
16    cout << "The length of \"" << string1 << "\" is " << strlen( string1 )
17       << "\nThe length of \"" << string2 << "\" is " << strlen( string2 )
18       << "\nThe length of \"" << string3 << "\" is " << strlen( string3 )
19       << endl;
20
21    system("PAUSE");
22    return 0; // indicates successful termination
23 } // end main
24
```

```
c:\Documents and Settings\m.ali\My Documents\Visual Studio
The length of "abcdefghijklmnopqrstuvwxyz" is 26
The length of "four" is 4
The length of "Boston" is 6
Press any key to continue . . .
```

Ödev

- İki boyutlu bir labirent programı yazınız. Ekran ilk açıldığında aşağıdaki örnekteki gibi olacaktır.

- x karakteri ilk açıldığında giriş noktasında duracak ve kullanıcıdan gelen komutlarda dikey veya yatay iki yönde "." krakterleri üzerinde hareket edecektir.
- x karakterini hareket ettirmek için klavyedeki yön tuşları kullanılacaktır.

61

Ödev

- x karakteri çıkış noktasına ulaştığında "Tebrikler çıkışı buldunuz." mesajını ekrana yazarak yeni oyun isteyip istemediği sorulacaktır. 'E' girilirse yeni bir labirent oluşturulacaktır.
- **Labirent dikdörtgen şeklinde olacak ve her yeni oyun başlangıcında yatay ve dikey boyutu kullanıcıdan sorulacaktır.**
- **Labirenti rastgele oluşturacak bir fonksiyon yazılacaktır.**
- **Bu fonksiyon bir çıkış ile bir giriş oluşturacak ve bunların arasında en az bir yol oluşturacaktır.**
- Labirentin yatay ve dikey minimum uzunluğu 10 ve maksimum uzunluğu 50 karakter alınacaktır.

62