MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

- **Kullanılacak kaynaklar:**
  http://www.cplusplus.com/doc/tutorial/
  Published by Juan Soulié

- **C++ ile ileri programlama**
  Paul Deitel
  Harvey Deitel

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

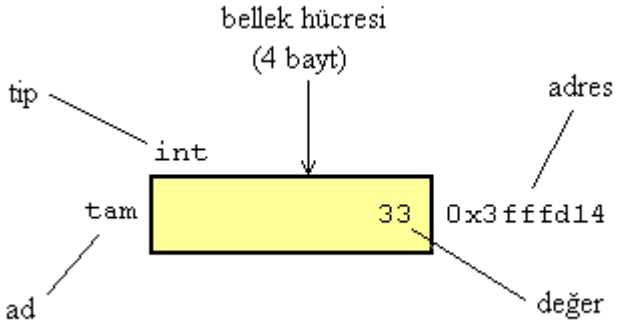
İşaretçiler (Pointers)

İşaretçiler yani pointerlar C++ dilinin en temel yapıtaşlarından biridir. İşaretçiler konusu için C++'ı diğer dillerden ayıran en önemli özelliklerinden biridir. Adından da anlayabileceğiniz üzere C++'ta işaretçiler değişkenlerin bellekteki adreslerini tutabilme ve onlarla işlem yapabilme olanağı sağlar

Belli bir tipte değişken tanımlanıp ve bir değer atandığında, o değişkene dört temel özellik eşlik eder:

- değişkenin adı
- değişkenin tipi
- değişkenin sahip olduğu değer (içerik)
- değişkenin bellekteki adresi

Örnek olarak bir tamsayı değişken tanımlayalım
int tam = 33;
Int tanımlandığındanbellekte 4 bayt büyüklüğünde bir yer ayrılır.



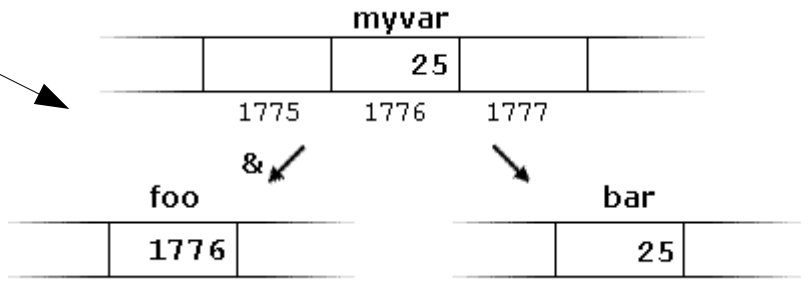
4 baytlık alanda değişkenin binary değeri:
00000000 00000000 00000000 00100001

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

Adres & operatörü

foo = &myvar;
Bu işlem ile myvar değişkeninin adresi foo değişkenine atanır. Örnek olarak myar değişkeninin hafızada 1776 nolu adreste olduğunu düşünelim. Aşağıdaki kod parçasında yapılan işlem

```
myvar = 25;
foo = &myvar;
bar = myvar;
```



Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

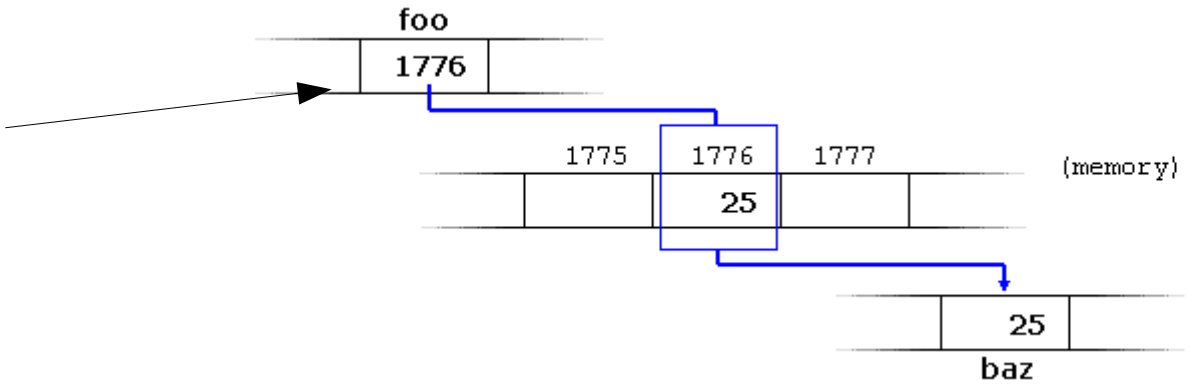
"*"POİNTER (GÖSTERİCİ, İŞARETÇİ)

Bir veri bloğunun bellekte bulunduğu adresi içeren (gösteren) veri tipidir.
Tanımlama biçimi:

veri tipi *p;

p değişkeni <veri tipi> ile belirtilen tipte bir verinin bellekte saklandığı adresi içerir.

Örnek:
baz = *foo;



baz = foo; // baz ve foo eşdeğer (bellek adresleri aynı 1776)
baz = *foo; // baz foo'nun işaret ettiği değere eşit (25)

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

& ——►operetörün işaret ettiği hafızadaki adresi belirtir

* ——► operatörü işaret ettiği hafızadaki değeri

İşaretçi (gösterici), hafıza alanındaki bir adresinin saklandığı değişkendir. İşaretçilere veriler (yani değişkenlerin içeriği) değil de, o verilerin bellekte saklı olduğu yerin başlangıç adresleri atanır. İşaretçi adres tutan bir değişkendir.

Bir işaretçi, diğer değişkenler gibi, sayısal bir değişkendir. Bu sebeple kullanılmadan önce program içinde bildirilmelidir. İşaretçi tipindeki değişkenler şöyle tanımlanır:

```
Tip  *işaretçi_adı;
```

Burada Tip herhangi bir C tip adı olabilir. Değişkenin önünedeki * karakteri yönlendirme (indirection) operatörü olarak adlandırılır ve bu değişkenin veri değil bir adres bilgisi tutacağını işaret eder. Örneğin:

```
char  *ch;            /* tek bir karakter için */

int   *x;             /* bir tamsayı için */

float *deger, sonuc;   /* deger işaretçi tipinde, sonuc sıradan bir gerçel değişkenler
```

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

Örnek:

```
*p++   // == *(p++):  işaretçide artış, ve arttırılmamış  addresi gösterme
*++p   // == *(++p):  işaretçide artış, ve arttırılmış  addresi gösterme
++*p   // ==++(*p): işaretçi, ve onun işaret ettiği değerde  artış
(*p)++ //
```

```
// pointers as arguments:
#include <iostream>
using namespace std;
void increment_all (int* start, int* stop)
{
  int * current = start;
  while (current != stop) {
    ++(*current);  // increment value pointed
    ++current;     // increment pointer
  }
}
void print_all (const int* start, const int* stop)
{
  const int * current = start;
  while (current != stop) {
    cout << *current << '\n';
    ++current;     // increment pointer
  }
}
int main ()
{
  int numbers[] = {10,20,30};
  increment_all (numbers,numbers+3);
  print_all (numbers,numbers+3);
  return 0;
}
```

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

Fonksiyonlar:

Belli işlemleri yapmaya izin veren program yapısıdır. C++ da bir grup işlemlerin belli bir isimde tanımlanması ve programın istendiği yerde çağrılabilir. Fonksiyon sintaksi:

Tür isim (parametre1,parametre2,parametre3,…) { işlemler }

Burda
Tür: fonksiyonlar tarafından yapılacak işlem(ler) sonucunda döndüreceği sonucun türünü belirtir.
İsim: Fonksiyonun çağrılacağı onu tanımlayan ismini belirtir.
Parametreler (gerektiği kadar olabilir): Her parametre belirlenen tipte tanımlıyıcıya sahiptir. Her parametre “,” ile ayrılır ve normal değişkenler giibi tanımlanır.
Işlemler {} arasında belirlenir.

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

Geçen dönemki derslerimizde gördüğümüz gibi birden farkli türde aynı isimde fonksion tanımlayabiliriz. Bunların hepsini tekbir yapıda toplamak daha mantıklı olabilir. Bu tip durumlar için C++'ta bunun için generik tipte fonksiyon tanımlana bilmektedir. Bu tip fonksiyonlara şablon(tamplate) fonksiyon denir. Yapısı:

template <template-parameters> function-declaration

Bir şablon fonksiyon için örnek kalıp olara iki sayının toplamını alan bir şablon fonksiyonu düşünelim bu şablon fonksiyonun yapısı

```
template <class SomeType>

SomeType sum (SomeType a, SomeType b)

{

  return a+b;

}
```

Şeklinde olacaktır.

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

```
#include <iostream>
#include <cstdlib> // contains prototypes for functions srand and
rand
#include <ctime> // contains prototype for function time
using namespace std;
int rollDice(); // rolls dice, calculates and displays sum
int main()
{
  // enumeration with constants that represent the game status
  enum Status { CONTINUE, WON, LOST }; // all caps in
constants
  int myPoint; // point if no win or loss on first roll
  Status gameStatus; // can contain CONTINUE, WON or LOST
  // randomize random number generator using current time
  srand( time( 0 ) );
  int sumOfDice = rollDice(); // first roll of the dice
  // determine game status and point (if needed) based on first
roll
  switch ( sumOfDice )
  {
    case 7: // win with 7 on first roll
    case 11: // win with 11 on first roll
      gameStatus = WON;
      break;
    case 2: // lose with 2 on first roll
    case 3: // lose with 3 on first roll
    case 12: // lose with 12 on first roll
      gameStatus = LOST;
      break;
    default: // did not win or lose, so remember point
      gameStatus = CONTINUE; // game is not over
      myPoint = sumOfDice; // remember the point
      cout << "Point is " << myPoint << endl;
      break; // optional at end of switch
  } // end switch
```

```
// while game is not complete
  while ( gameStatus == CONTINUE ) // not WON or LOST
  {
    sumOfDice = rollDice(); // roll dice again

    // determine game status
    if ( sumOfDice == myPoint ) // win by making point
      gameStatus = WON;
    else
      if ( sumOfDice == 7 ) // lose by rolling 7 before point
        gameStatus = LOST;
  } // end while
  // display won or lost message
  if ( gameStatus == WON )
    cout << "Player wins" << endl;
  else
    cout << "Player loses" << endl;
} // end main
// roll dice, calculate sum and display results
int rollDice()
{
  // pick random die values
  int die1 = 1 + rand() % 6; // first die roll
  int die2 = 1 + rand() % 6; // second die roll
  int sum = die1 + die2; // compute sum of die values
  // display results of this roll
  cout << "Player rolled " << die1 << " + " << die2
    << " = " << sum << endl;
  return sum; // return sum of dice
} // end function rollDice
```

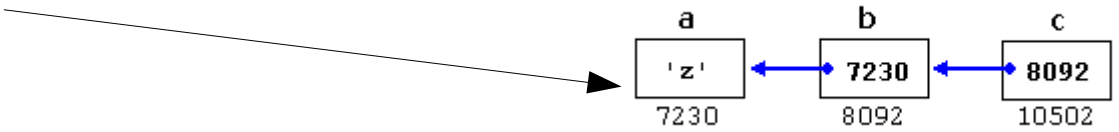
MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

İşaretçinin işaretçisi:

C++ bize işaretçiye işaretçi atama imkani verir. Ör:

```
char a;
char * b;
char ** c;
a = 'z';
b = &a;
c = &b;
```

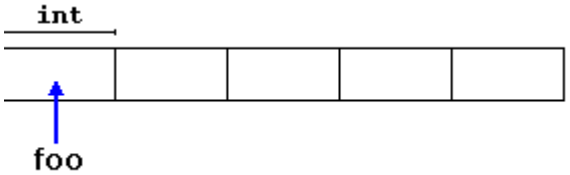


MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

Dinamik Hafıza:
Bundan önceki derslerimizde kullandığımız programlarda bütün değişkenler için gerekli hafıza Miktarları program çalıştırılmaya başlamadan Değişkenler tanımlandığı anda önceden belirlenmişti.
Bir çok durumda gerekli hafıza miktarı program çalıştırıldığı zaman belli olur
(Genelde kullanıcı girdisine göre) Bu gibi durumlarda programlar dinamik olarak gerekli hafiza miktarının belirlenmesi ayrılmasını ve kullanım bittikten sonra silinmesine ihtiyaç duyar.
Bu gibi durumlar için C++ içersinde iki operatör bulunmaktadır (new delete)
Operators new and new[ ]

pointer = new type
pointer = new type [number_of_elements]



C++ iki türlü şekilde dinamik hafızanın ayrılıp ayrılmadığını kontrol eder

- foo = new int [5]; // if allocation fails, an exception is thrown
- foo = new (nothrow) int [5]; bu durumda sorun olduğunda exception atilmaz null (boş 0) işaretçi atanır. Bu durumda atamanın doğruluğunu aşağıdaki gibi kontrol edebiliriz

```
int * foo;
foo = new (nothrow) int [5];
if (foo == nullptr) {
  // error assigning memory. Take measures.
}
```

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

Operatorler delete ve delete[ ]:

Genellikle Dinamik olarak ayrılan hafıza birimleri işlemler bittince tutulmasına gerek kalmaz. Bu hafıza birimlerinin silinmesi-serbest bırakılması ve başka işlemler için kullanılması daha verimli olacaktır. Bunun içinde delete ve delete[ ] operatörleri kullanılır.

delete pointer;

delete [ ] pointer;

Dynamic memory in C

C++ integrates the operators new and delete for allocating dynamic memory. But these were not available in the C language; instead, it used a library solution, with the functions malloc, calloc, realloc and free, defined in the header <cstdlib> (known as <stdlib.h> in C). The functions are also available in C++ and can also be used to allocate and deallocate dynamic memory.

```
// rememb-o-matic
#include <iostream>
#include <new>
using namespace std;
int main ()
{
  int i,n;
  int * p;
  cout << "How many numbers would you like to
type? ";
  cin >> i;
  p= new (nothrow) int[i];
  if (p == nullptr)
    cout << "Error: memory could not be
allocated";
  else
  {
    for (n=0; n<i; n++)
    {
      cout << "Enter number: ";
      cin >> p[n];
    }
    cout << "You have entered: ";
    for (n=0; n<i; n++)
      cout << p[n] << ", ";
    delete[] p;
  }
  return 0;
}
```

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

Data Yapıları (Data structures):

Bir grup datanınn bir isimde toplanıp, gruplandırılmasını çağrılmasını sağlar

```
struct product {
  int weight;
  double price;
} ;
product apple;
product banana, melon;
```

struct type_name {

member_type1 member_name1;

member_type2 member_name2;

member_type3 member_name3;

.

.

} object_names

```
struct product {
  int weight;
  double price;
} apple, banana, melon;
```

```
Yapı elemanlarının çağrılma şekli:
apple.weight
apple.price
banana.weight
banana.price
melon.weight
melon.price
```

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

```
#include <iostream>
#include "maximum.h" // include definition of function template
maximum
using namespace std;

int main()
{
  // demonstrate maximum with int values
  int int1, int2, int3;

  cout << "Input three integer values: ";
  cin >> int1 >> int2 >> int3;

  // invoke int version of maximum
  cout << "The maximum integer value is: "
    << maximum( int1, int2, int3 );

  // demonstrate maximum with double values
  double double1, double2, double3;

  cout << "\n\n Input three double values: ";
  cin >> double1 >> double2 >> double3;

  // invoke double version of maximum
  cout << "The maximum double value is: "
    << maximum( double1, double2, double3 );

  // demonstrate maximum with char values
  char char1, char2, char3;

  cout << "\n\n Input three characters: ";
  cin >> char1 >> char2 >> char3;

  // invoke char version of maximum
  cout << "The maximum character value is: "
    << maximum( char1, char2, char3 ) << endl;
} // end main
```

```
//maximum.h

template < class T >  // or template< typename T >
T maximum( T value1, T value2, T value3 )
{
  T maximumValue = value1; // assume value1 is maximum

  // determine whether value2 is greater than maximumValue
  if ( value2 > maximumValue )
    maximumValue = value2;

  // determine whether value3 is greater than maximumValue
  if ( value3 > maximumValue )
    maximumValue = value3;

  return maximumValue;
} // end function template maximum
```

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

Her değişken için bilgisayar hafızasında bir yer ayrılır. Global değişkenler ve namespaceler için program süresince yer ayrılır. Bu na statik hafıza alanı denir. Yerel değişkenler için ise ayrılan yer sadece programda o değişkenin tanımlandığı blokta oluşturulur. Bu alanlara da dinamik hafızaalanı olarak tanımlanır

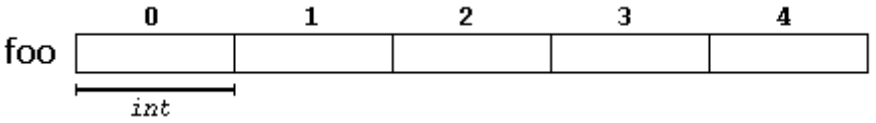
- Global değişkenler ve namespaceler eğer ildeğer verilmez ise otomatik olarak ilk değer olarak 0 atanır.
- Yerel değişkenler ise ilk değerlenidirme otomatik olarak yapılmaz kullanıcı tarafından ilk değer verilmelidir

Örnek:

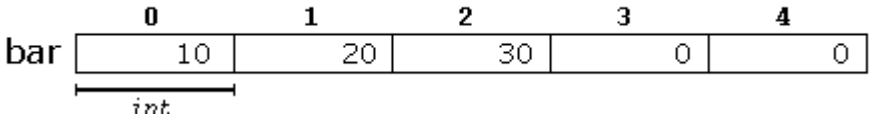
```
// static vs automatic storage
#include <iostream>
using namespace std;
int x;
int main ()
{
  int y;
  cout << x << '\n';
  cout << y << '\n';
  return 0;
}
```

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

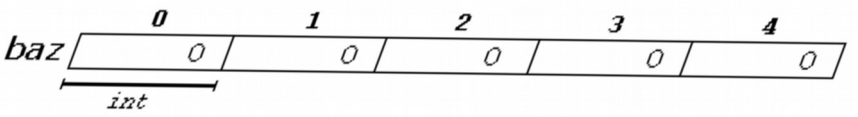
Diziler: Dizi ayni tür değişken serisininsurekli bir hafıza alanında tutulmasıdır. C++ da dizi yapısı:
type isim [eleman sayısı];
Ör: 5 elemanlı integer dizi
int foo [5];




```
int foo [5] = { 16, 2, 77, 40, 12071 };
```

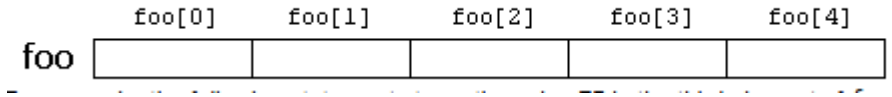



```
int baz [5] = { };
```



Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

Dizinin bir elemanına erişmek:




```
foo [2] = 75;
(Foo nun 3. elemanı)
x = foo[2];
```

```
foo[0] = a;
foo[a] = 75;
b = foo [a+2];
foo[foo[a]] = foo[2] + 5;
```

Örnek:

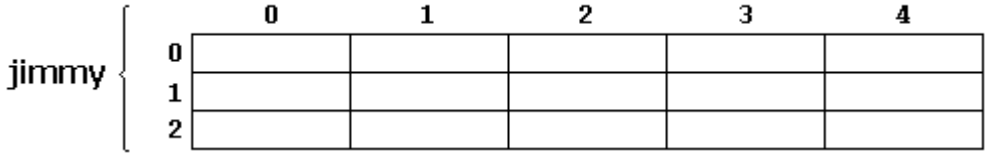
```
// arrays example
#include <iostream>
using namespace std;
int foo [] = {16, 2, 77, 40, 12071};
int n, result=0;
int main ()
{
  for ( n=0 ; n<5 ; ++n )
  {
    result += foo[n];
  }
  cout << result;
  return 0;
}
```

MİMAR SİNAN FINE ARTS UNIVERSITY

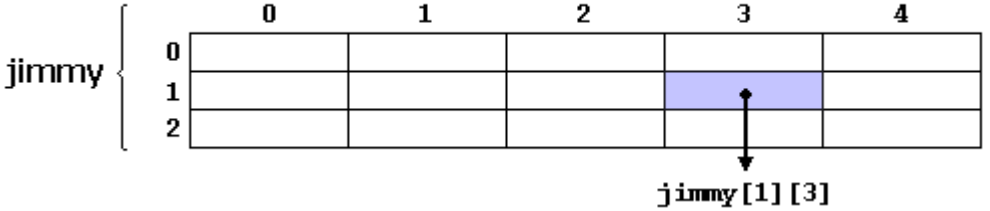
Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

Çok Boyutlu Diziler

```
int jimmy [3][5];
```




```
jimmy[1][3]
```




```
#define WIDTH 5
#define HEIGHT 3

int jimmy [HEIGHT][WIDTH];
int n,m;

int main ()
{
  for (n=0; n<HEIGHT; n++)
    for (m=0; m<WIDTH; m++)
    {
      jimmy[n][m]=(n+1)*(m+1);
    }
}
```

```
#define WIDTH 5
#define HEIGHT 3

int jimmy [HEIGHT * WIDTH];
int n,m;

int main ()
{
  for (n=0; n<HEIGHT; n++)
    for (m=0; m<WIDTH; m++)
    {
      jimmy[n*WIDTH+m]=(n+1)*(m+1);
    }
}
```

jimmy

	0	1	2	3	4
0	1	2	3	4	5
1	2	4	6	8	10
2	3	6	9	12	15

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

Parametre olarak diziler

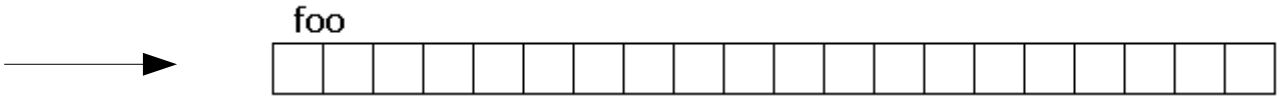
```
// arrays as parameters
#include <iostream>
using namespace std;
void printarray (int arg[], int length)
{
  for (int n=0; n<length; ++n)
    cout << arg[n] << ' ';
  cout << '\n';
}
int main ()
{
  int firstarray[] = {5, 10, 15};
  int secondarray[] = {2, 4, 6, 8, 10};
  printarray (firstarray,3);
  printarray (secondarray,5);
}
```

(int arg[]) bu parametre bilesenleri int olan istenilen uzunlukta olabilecek dizidir

```
   void procedure (int myarray[][3][4])
```

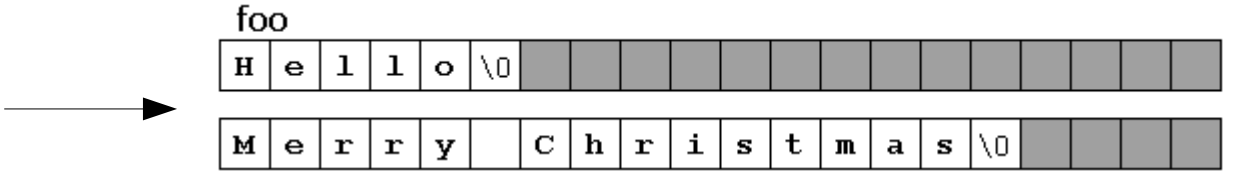
Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK

```
char foo [20];
```



Değeri

"Hello" ve "Merry Christmas"



`\0` ifadesi karakter katarının sonuna gelindiğini gösteriri

```
char myword[] = { 'H', 'e', 'l', 'l', 'o', '\0' };
char myword[] = "Hello";
```

Diziler üste tanımlandıktan sonra yeni değer atamak istenirse

```
myword = "Bye";
myword[] = "Bye";
myword = { 'B', 'y', 'e',
'\0' };
```

YANLIŞ!!!

```
myword[0] = 'B';
myword[1] = 'y';
myword[2] = 'e';
myword[3] = '\0';
```

DOĞRU(GEÇERLİ)!!

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 1
MSGSU Fizik Bölümü
Ferhat ÖZOK