Увод в програмирането

Масиви. Символни низове. Работа с паметта

2017-2018 г.

ФМИ, специалност „Софтуерно инженерство“

Слайдовете в тази презентация са създадени от доц. Атанас Семерджиев

Съдържание

- Основни операции с масиви
- Многомерни масиви
- Работа с паметта. Динамична памет

Основни операции с масиви

- Добавяне на елемент
- Премахване на елемент
- Търсене на елемент
- Сортиране

Добавяне на елемент

```
void ShiftRight(double* pArray,
                size_t Size,
                size_t StartFrom,
                size_t Positions)
{
    size_t write = Size + Positions - 1;
    size_t read  = Size - 1;

    while (read >= StartFrom)
    {
        pArray[write--] = pArray[read--];
    }
}
```

```
void InsertAt(double *pArr,
              size_t Size,
              size_t Index,
              double Element)
{
    ShiftRight(pArr, Size, Index, 1);

    pArr[Index] = Element;
}
```

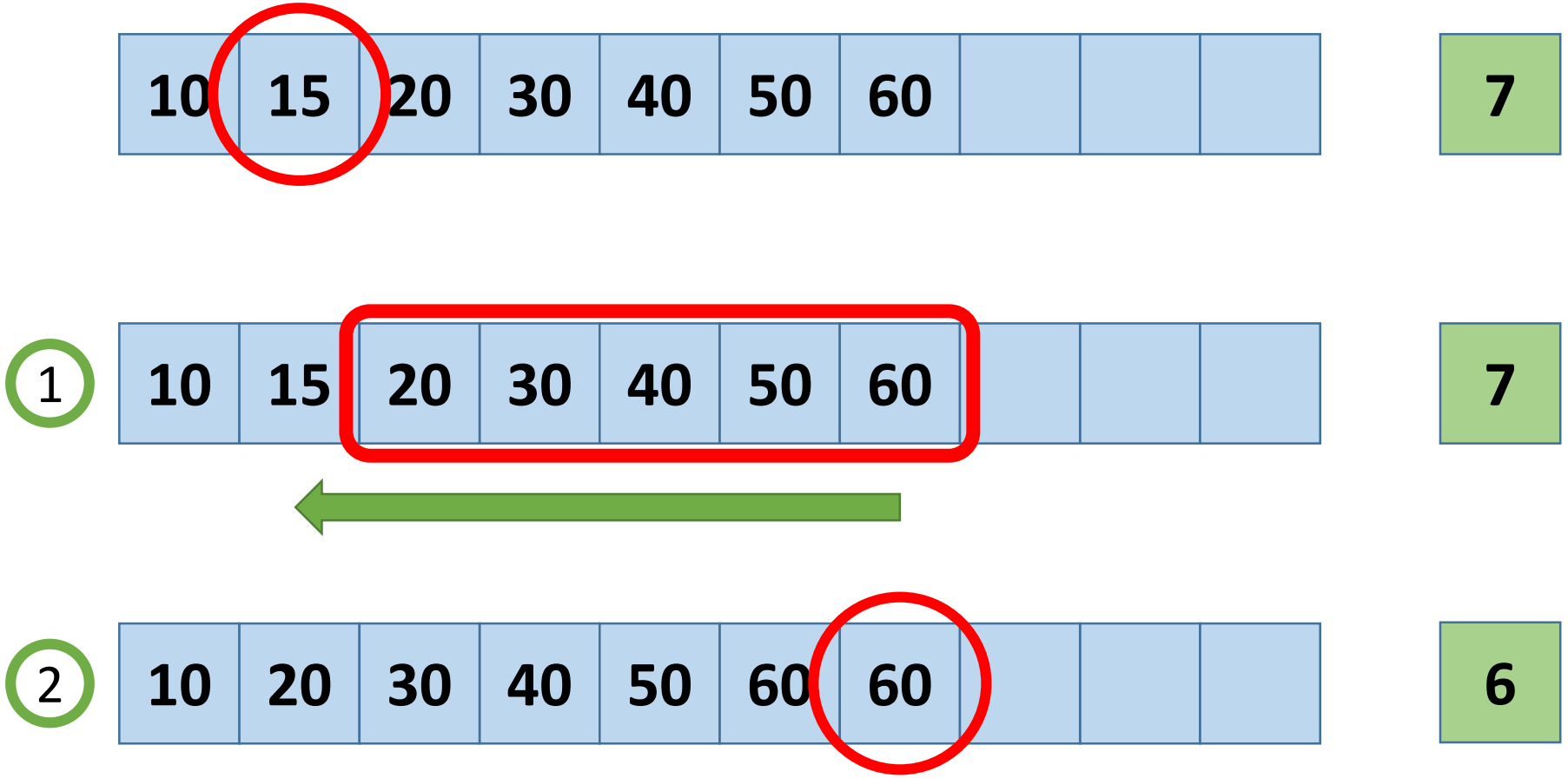
```
void PrintArray(const double * pArr, size_t Size)
{
    for (size_t i = 0; i < Size; i++)
    {
        std::cout << pArr[i] << std::endl;
    }
}

void main()
{
    double data[10] = { 10, 20, 30, 40, 50, 60 };

    InsertAt(data, 6, 1, 15);

    PrintArray(data, 10);
}
```

Премахване на елемент



```
void ShiftLeft(double* pArr,
               size_t Size,
               size_t StartFrom,
               size_t Positions)
{
    size_t write = StartFrom;
    size_t read  = StartFrom + Positions;

    while (read < Size)
    {
        pArr[write++] = pArr[read++];
    }
}
```

```
void RemoveAt(double* pArr, size_t Size, size_t Index)
{
    ShiftLeft(pArr, Size, Index, 1);
}

void main()
{
    double data[10] = { 10, 20, 30, 40, 50, 60 };
    RemoveAt(data, 6, 0);
    PrintArray(data, 10);
}
```

Търсене на елемент

Просто търсене

```
int Find(const double* pArr, size_t Size, double Element)
{
    for (size_t i = 0; i < Size; i++)
    {
        if (pArr[i] == Element)
            return i;
    }

    return -1; // Елементът не е намерен
}
```

Двоично търсене

- Масивът трябва да е сортиран
- Връща се първият намерен елемент

```
int BinarySearch(const double* pArr, size_t Size, double Element)
{
    size_t Left = 0, Right = Size - 1;

    while (Left <= Right) {
        int Middle = (Left + Right) / 2;

        if (pArr[Middle] == Element)
            return Middle;
        else if (Element > pArr[Middle])
            Left = Middle + 1;
        else
            Right = Middle - 1;
    }

    return -1; // Елементът не е намерен
}
```

Сортиране

```
// Пряка селекция

void SelectionSort(double* pArr, size_t Size)
{
    size_t Min;

    for (size_t i = 0; i < Size; i++)
    {
        Min = i;

        for (size_t j = i + 1; j < Size; j++)
        {
            if (pArr[j] < pArr[Min])
                Min = j;
        }

        if (Min != i)
            std::swap(pArr[i], pArr[Min]);
    }
}
```

```
// Метод на мехурчето

void BubbleSort(double* pArr, size_t Size)
{
    size_t Rightmost = Size - 1;

    for (size_t i = 0; i < Rightmost; i++)
    {
        for (size_t j = Rightmost; j > i; j--)
        {
            if (pArr[j - 1] > pArr[j])
                std::swap(pArr[j - 1], pArr[j]);
        }
    }
}
```


```
double data[3][4] = { 10., 11., 12., 13.,
                      20., 21., 22., 23.,
                      30., 31., 32., 33. };

data[0][2] <--> *(data + 0 * 4 + 2)
data[1][2] <--> *(data + 1 * 4 + 2)
data[2][2] <--> *(data + 2 * 4 + 2)
...
```

```
void PrintArray(double arr[][])
{
    for (size_t row = 0; row < 3; row++)
    {
        for (size_t col = 0; col < 5; col++)
        {
            std::cout << arr[row][col] << ' ';
        }

        std::cout << std::endl;
    }
}

void main()
{
    double data[3][5] = { 10., 11., 12., 13., 14.,
                          20., 21., 22., 23., 24.,
                          30., 31., 32., 33., 34. };

    PrintArray(data);
}
```

*(arr + row * ??? + col)

```
void PrintArray(double arr[][5])
{
    for (size_t row = 0; row < 3; row++)
    {
        for (size_t col = 0; col < 5; col++)
        {
            std::cout << arr[row][col] << ' ';
        }

        std::cout << std::endl;
    }
}

void main()
{
    double data[3][5] = { 10., 11., 12., 13., 14.,
                          20., 21., 22., 23., 24.,
                          30., 31., 32., 33., 34. };

    PrintArray(data);
}
```

*(arr + row * 5 + col)

Достъп чрез указател

```
void ZeroArray(double * pArray, size_t Size)
{
   for (size_t i = 0; i < Size; i++)
      pArray[i] = 0;
}

void main()
{
   double data[500];
   ZeroArray(data, 500);
}
```

Достъп чрез указател (const)

```
void PrintArray(const double * pArray, size_t Size)
{
   for (size_t i = 0; i < Size; i++)
      std::cout << pArray[i] << std::endl;
}

void main()
{
   double data[5] = { 1., 2., 3., 4., 5. };
   PrintArray(data, 5);
}
```

```
void PrintArray(const double * pArray, size_t Size)
{
   for (size_t i = 0; i < Size; i++)
      cout << pArray[i] << endl;
}

void main()
{
   double data[3][4] = { 10., 11., 12., 13.,
                         20., 21., 22., 23.,
                         30., 31., 32., 33. };

   PrintArray((const double*)data, 3 * 4);
}
```

```
void PrintArray(const double * pArray,
                size_t Rows,
                size_t Cols)
{
   for (size_t row = 0; row < Rows; row++)
   {
      for (size_t col = 0; col < Cols; col++)
      {
         std::cout << *(pArray + row * Cols + col)
                   << ' ';
      }
   }

   std::cout << std::endl;
}
```

```
void main()
{
    double data[3][4] = { 10., 11., 12., 13.,
                          20., 21., 22., 23.,
                          30., 31., 32., 33. };

    PrintArray((const double*)data, 3, 4);
}
```

```
void PrintArray(const double * pArray,
                size_t Rows,
                size_t Cols)
{
   for (size_t row = 0; row < Rows; row++)
   {
      for (size_t col = 0; col < Cols; col++)
      {
         std::cout << pArray[row * Cols + col]
                   << ' ';
      }

      std::cout << std::endl;
   }
}

// *(pArray + row*Cols + col) <--> pArray[row*Cols + col]
```