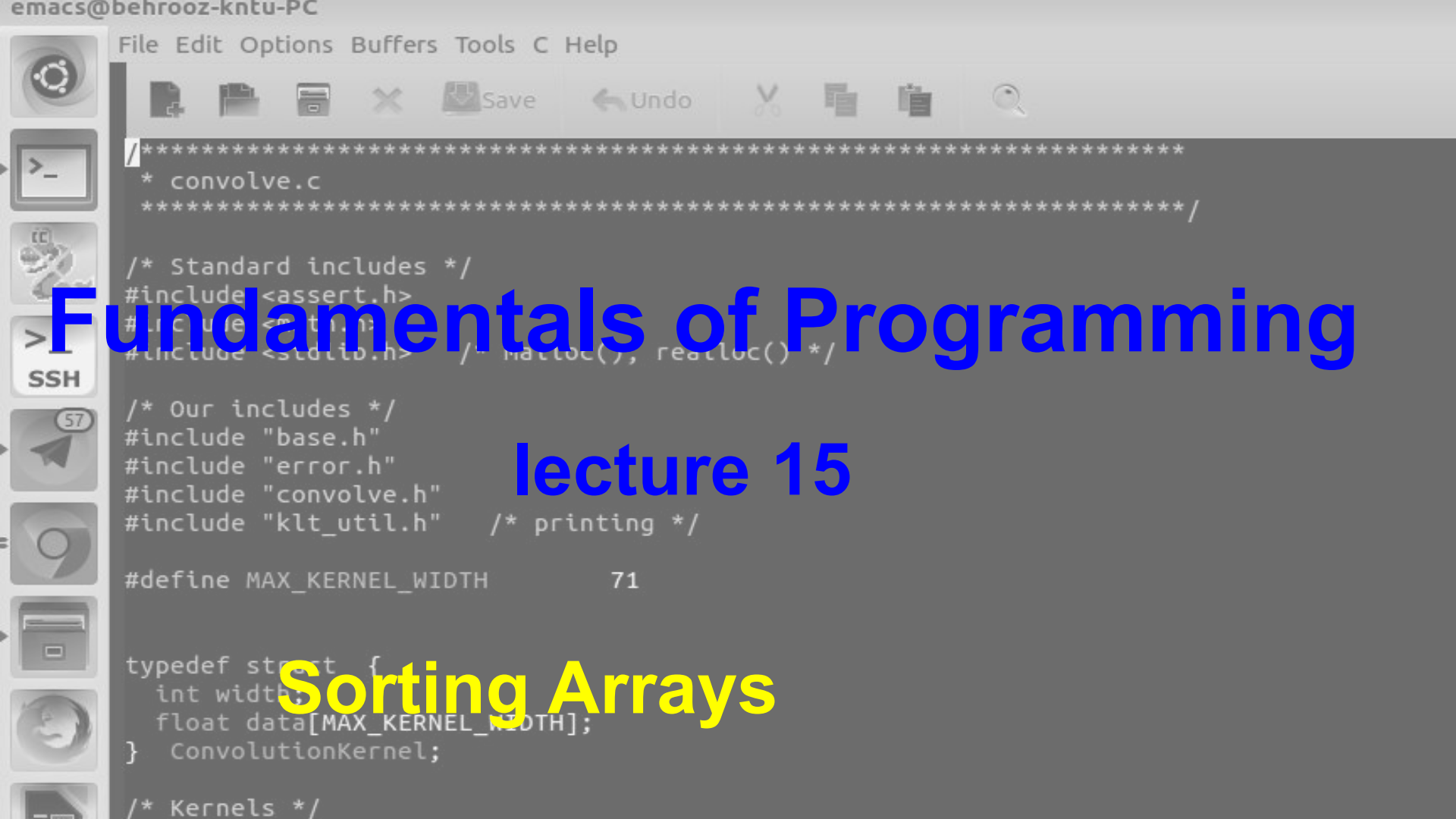
emacs@behrooz-kntu-PC
File Edit Options Buffers Tools C Help
Save
Undo
/* Standard includes */
#include <assert.h>
/* Our includes */
#include "base.h"
#include "error.h"
#include "convolve.h"
#include "klt_util.h" /* printing */
#define MAX_KERNEL_WIDTH 71
typedef struct {
int width;
float data[MAX_KERNEL_WIDTH];
} ConvolutionKernel;
/* Kernels */
Fundamentals of Programming
lecture 15
Sorting Arrays

Sorting Arrays

```
#define SIZE  11

int array[11] = {25, 19, 14, 17, 27,  6, 18, 20,  1, 21, 32};
```

25	19	14	18	27	6	32	18	20	1	21

Sorting Arrays

```
#define SIZE  11

int array[11] = {25, 19, 14, 17, 27,  6, 18, 20,  1, 21, 32};
```

25	19	14	18	27	6	32	18	20	1	21

1	6	14	18	18	19	20	21	25	27	32

Sorting Arrays

```c
void printArray(int a[], int n);
void sort(int a[], int n);

int main() {
  int a[] = {25, 19, 14, 18, 27, 6, 32, 18, 20, 1, 21};
  int n = sizeof(a) / sizeof(a[0]);

  printArray(a,n);

  sort(a,n);

  printArray(a,n);

  return 0;
}

void sort(int a[], int n) {
  // ??
}
```

sortarray.c

Sorting Arrays

```
void sort(int a[], int n) {
  int m;

  m = n-1;
  for (int i = 0; i < m; i++) {

    if (a[i] > a[i+1]) {
      int temp = a[i];
      a[i] = a[i+1];
      a[i+1] = temp;
    }


  }


}
```

sortarray2.c

25	19	14	18	27	6	32	18	20	1	21

Sorting Arrays

```
void sort(int a[], int n) {
  int m;

  m = n-1;

  for (int j = 0; j < m; j++) {

    for (int i = 0; i < m; i++) {

      if (a[i] > a[i+1]) {
        int temp = a[i];
        a[i] = a[i+1];
        a[i+1] = temp;
      }

    }
  }


}
```

sortarray2.5.c

25	19	14	18	27	6	32	18	20	1	21

Sorting Arrays

```c
void sort(int a[], int n) {
  int m;

  for (m = n-1; m > 0; m--) {
    for (int i = 0; i < m; i++) {

      if (a[i] > a[i+1]) {
        int temp = a[i];
        a[i] = a[i+1];
        a[i+1] = temp;
      }


    }
  }

}
```

sortarray3.c

25	19	14	18	27	6	32	18	20	1	21

Sorting Arrays

```
void sort(int a[], int n) {
  int m;

  for (m = n-1; m > 0; m--) {
    for (int i = 0; i < m; i++) {

      if (a[i] > a[i+1]) {
        int temp = a[i];
        a[i] = a[i+1];
        a[i+1] = temp;
      }


    }
  }

}
```

sortarray3.c

descending order?

25	19	14	18	27	6	32	18	20	1	21

Mean, median and mode

Write a program printing mean, median, and mode of an arrays of integers. You can assume that array elements are integers between 0 and 19.

stats.c

```c
void printArray(const int a[], int n);
void sort(int a[], int n);
double mean(const int a[], int n);
double median(int a[], int n);
int mode(const int a[], int n);

int main() {
  int a[] = {1,18,2,3,4,5,7,7,4,19,18,12,13, 0, 18,
             14,17,1,0,8,9,9,9,10, 11, 0,19,18,
             18,1,3,4,6,7,8,3,2,15,1,0,7,13,14,
             10,12,16,17,18,1,4,6,8,4,0,5,8,7,5,
             6,4,9,16,15,12,13,14,12,13,16,18,9,10};

  int n = sizeof(a) / sizeof(a[0]);

  printArray(a,n);

  printf("mean=%.2f, mode= %d, median=%.1f\n", mean(a,n), mode(a,n), median(a,n));

  return 0;
}
```

Mean, median and mode

Write a program printing mean, median, and mode of an arrays of integers. You can assume that array elements are integers between 0 and 19.

```c
double mean(const int a[], int n) {

  int sum = 0;
  for (int i = 0; i < n; i++)
    sum += a[i];

  return sum / (double) n;
}
```

stats.c

Mean, median and mode

Assume that the array elements are integers between **0 and 19**.

```
int mode(const int a[], int n) {
  int count[20];

  for (int i = 0; i < 20; i++)
    count[i] = 0;

  for (int i = 0; i < n; i++)
    count[a[i]]++;

  int maxi = 0;
  for (int i = 1; i < 20; i++)
    if (count[i] > count[maxi])
      maxi = i;

  return maxi;
}
```

stats.c

Static vs automatic arrays

```c
int mode(const int a[], int n) {
  static int count[20];

  for (int i = 0; i < 20; i++)
    count[i] = 0;

  for (int i = 0; i < n; i++)
    count[a[i]]++;

  int maxi = 0;
  for (int i = 1; i < 20; i++)
    if (count[i] > count[maxi])
      maxi = i;

  return maxi;
}
```

stats2.c

Median

```c
double median(int a[], int n) {
  sort(a,n);

  if (n % 2 == 0)
    return (a[n/2] + a[n/2-1]) / 2.0;
  else
    return a[n/2];

}
```

stats.c

Median

```c
double median(int a[], int n) {
  sort(a,n);

  if (n % 2 == 0)
    return (a[n/2] + a[n/2-1]) / 2.0;
  else
    return a[n/2];

}
```

stats.c

side effect?

Median

```c
double median(const int a[], int n) {
  int b[n];

  for (int i = 0; i < n; i++)
    b[i] = a[i];

  sort(b,n);

  if (n % 2 == 0)
    return (b[n/2] + b[n/2-1]) / 2.0;
  else
    return b[n/2];

}
```

stats2.c