Day11.C

Arrays (2)
Applications

Young W. Lim

December 12, 2017

1 Applications of Arrays

1.1 Find max, min, sum, avg values

```
::::::::::::::
find.c
::::::::::::::
#include <stdio.h>

int find_sum(int a[], int len) {
  int i, S=0;;

  for (i=0; i<len; ++i) S += a[i];

  return S;
}

int find_max(int a[], int len) {
  int i, cmax=(1L<<31); // the least negative int
  printf("%x %+15d\n", cmax, cmax);

  for (i=0; i<len; ++i)
    if (cmax < a[i]) cmax = a[i];

  return cmax;
}

int find_min(int a[], int len) {
  int i, cmin=(1L<<31)-1; // the greatest positive int
  printf("%x %+15d\n", cmin, cmin);

  for (i=0; i<len; ++i)
    if (cmin > a[i]) cmin = a[i];

  return cmin;
}

double find_avg(int a[], int len) {
  double avg;

  avg = find_sum(a, len);
  avg /= len;

  return avg;

}


int main(void) {
  int array[10] = {1, 9, 3, 5, 4, 2, 6, 8, 7, 10};
```

```
  int sum_val, max_val, min_val, avg_val;

  sum_val = find_sum(array, 10);
  max_val = find_max(array, 10);
  min_val = find_min(array, 10);
  avg_val = find_avg(array, 10);

  printf("sum_val= %d \n", sum_val);
  printf("max_val= %d \n", max_val);
  printf("min_val= %d \n", min_val);
  printf("avg_val= %d \n", avg_val);

}
```

```
::::::::::::::
find.out
::::::::::::::
80000000     -2147483648
7fffffff     +2147483647
sum_val= 55
max_val= 10
min_val= 1
avg_val= 5
```

cmax

- to find a maximum, `cmax` must set to the least possible negative number.
- for the type `int` has a 4-byte storage, 0x80000000 is such a number.
- this number can be obtained by shifting 1 to the left by 31 bit positions.
- to avoid warning messages (overflow), use the `long int` type
- `(1L << 31)` is an 8-byte integer, 0x0000000080000000 (positive)
- this 8-byte integer is assigned to the 4-byte integer variable `cmax`
- the value of `cmax` is now 0x80000000.
- the value of `cmax` is the least negative number for the `int` type integer

cmin

- to find a minimum, `cmin` must set to the greatest possible positive number.
- for the type `int` has a 4-byte storage, 0x7FFFFFFF is such a number.
- this number can be obtained

  - by shifting 1 to the left by 31 bit positions,
  - then subtracting by one,
  - 0x7FFFFFFF is less than 0x80000000,
  - considering them as positive numbers.

- to avoid warning messages (overflow), use the `long int` type
- `(1L << 31)` is an 8-byte integer, 0x0000000080000000 (positive)
- `(1L << 31)-1` is an 8-byte integer, 0x000000007FFFFFFF (positive)
- this 8-byte integer is assigned to the 4-byte integer variable `cmin`
- the value of `cmin` is now 0x7FFFFFFF.
- the value of `cmin` is the greatest positive number for the `int` type integer

1.2 Linear Search

```
::::::::::::::
search.c
::::::::::::::
#include <stdio.h>

int find_max(int a[], int len) {
  int i, cmax=(1L<<31);

  for (i=0; i<len; ++i)
    if (cmax < a[i]) cmax = a[i];

  return cmax;
}

int find_min(int a[], int len) {
  int i, cmin=(1L<<31)-1;

  for (i=0; i<len; ++i)
    if (cmin > a[i]) cmin = a[i];

  return cmin;
}

int search(int a[], int key, int len) {
  int i;

  for (i=0; i<len; ++i)
    if (key == a[i]) return i;

  return -1;
}
```

```
int main(void) {
  int array[10] = {1, 9, 3, 5, 4, 2, 6, 8, 7, 10};
  int max_val, min_val, max_index, min_index;

  max_val = find_max(array, 10);
  min_val = find_min(array, 10);

  max_index = search(array, max_val, 10);
  min_index = search(array, min_val, 10);

  printf("max_val= %d \n", max_val);
  printf("max_index= %d \n", max_index);
  printf("min_val= %d \n", min_val);
  printf("min_index= %d \n", min_index);

}
```

```
::::::::::::::
search.out
::::::::::::::
max_val= 10
max_index= 9
min_val= 1
min_index= 0
```

searching the array

- the content of the array

index	0	1	2	3	4	5	6	7	8	9
value	1	9	3	5	4	2	6	8	7	10

- `max_val` (=10) is stored in `array[0]` (key=10, index=0)
- `min_val` (=1) is stored in `array[9]` (key=1, index=9)

the search function

- in the `for` loop,

```
for (i=0; i<len; ++i)
  if (key == a[i]) return i;
```

- whenever the input `key` is matched with any element of `array`
- the function immediately returns with the index `i`.
- however, when no element of the `array` matches with `key`,
- the next statement `return -1` will be executed.
- normally, the index values are non-negative.
- the index value of -1 represents there is no matching key in `array`.

1.3 Passing individual elements of arrays

```
::::::::::::::
swap.c
::::::::::::::
#include <stdio.h>

void swap(int *a, int *b) {
  int tmp;

  tmp = *a;
  *a = *b;
  *b = tmp;

}

void pr_element(int index, int val) {
  printf("array[%d] = %d \n", index, val);
}


int main(void) {
  int array[10] = {1, 9, 3, 5, 4, 2, 6, 8, 7, 10};
  int i;

  for (i=0; i<10; ++i)
    printf("array[%d]= %d \n", i, array[i]);

  swap(&array[0], &array[9]);

  printf("------------------\n");
  pr_element(0, array[0]);
  pr_element(9, array[9]);



}



::::::::::::::
swap.out
::::::::::::::
array[0]= 1
array[1]= 9
array[2]= 3
array[3]= 5
array[4]= 4
array[5]= 2
```

```
array[6]= 6
array[7]= 8
array[8]= 7
array[9]= 10
------------------
array[0] = 10
array[9] = 1
```

passing the address of an array element

- `swap(&array[0], &array[9]);`
- `swap(array, array+9);`

passing the elements of an array

- `pr_element(0, array[0]);`
- `pr_element(9, array[9]);`

1.4 Historgram

```
::::::::::::::
h2.c
::::::::::::::
#include <stdio.h>
#include <stdlib.h>

#define SIZE 20

int main(void) {
  int A[SIZE];
  int H[10];
  int i, j;

  // H[0]= number of students 0  <= score < 10   score/10 = 0
  // H[1]= number of students 10 <= score < 20   score/10 = 1
  // H[2]= number of students 20 <= score < 30   score/10 = 2
  // H[3]= number of students 30 <= score < 40   score/10 = 3
  // H[4]= number of students 40 <= score < 50   score/10 = 4
  // H[5]= number of students 50 <= score < 60   score/10 = 5
  // H[6]= number of students 60 <= score < 70   score/10 = 6
  // H[7]= number of students 70 <= score < 80   score/10 = 7
  // H[8]= number of students 80 <= score < 90   score/10 = 8
  // H[9]= number of students 90 <= score <=100  score/10 = 9,10

  srand(1);

  for (i=0; i<SIZE; ++i) A[i] = rand() % 101;

  for (i=0; i<SIZE; ++i) printf("A[%d]= %d \n", i, A[i]);
```

```
  for (j=0; j<10; ++j) H[j]= 0;

  for (i=0; i<SIZE; ++i) {
     j =A[i] / 10;  // integer division = quotient
     if (j==10) j=9;
     H[j]++;
  }

  for (j=0; j<10; ++j)
    printf("H[%d]= %d students\n", j, H[j]);

}
::::::::::::::
h2.out
::::::::::::::
A[0]= 32
A[1]= 32
A[2]= 54
A[3]= 12
A[4]= 52
A[5]= 56
A[6]= 8
A[7]= 30
A[8]= 44
A[9]= 94
A[10]= 44
A[11]= 39
A[12]= 65
A[13]= 19
A[14]= 51
A[15]= 91
A[16]= 1
A[17]= 5
A[18]= 89
A[19]= 34
H[0]= 3 students
H[1]= 2 students
H[2]= 0 students
H[3]= 5 students
H[4]= 2 students
H[5]= 4 students
H[6]= 1 students
H[7]= 0 students
H[8]= 1 students
H[9]= 2 students
```

Histogram with 10 bins

- there are 10 bins
- all the scores have the range of [0, 100]
- `score / 10` : this integer division gives a bin index with a range of [0, 9]
- if score is equal to 100, the bin index 9 is assigned
- `H[bin_size]` is intialized with zero
- `H[j]` is incremented whenever a student's `score / 10` is equal to `j`

Histogram with 5 bins

```
#include <stdio.h>
#include <stdlib.h>

#define SIZE 20
#define BINSZ 5

//--------------------------------
// return 0   0 <= score < 20
// return 1  20 <= score < 40
// return 2  40 <= score < 60
// return 3  60 <= score < 80
// return 4  80 <= score < 50
//--------------------------------
int bin(int A[], int i) {
   int j;
   int score = A[i];

   if (score < 20) j= 0;
   else if (score < 40) j= 1;
   else if (score < 60) j= 2;
   else if (score < 80) j= 3;
   else j= 4;

   return j;
}

//--------------------------------
// bubble sort
//--------------------------------
void bubbleSort(int a[], int size)
{
  int p, j;
  int tmp;

  for (p=1; p< size; ++p) {
    for (j=0; j< size-1; ++j) {
      if ( a[j] < a[j+1] ) {
        tmp = a[j];
```

```
        a[j] = a[j+1];
        a[j+1] = tmp;
      }
    }
  }
}


int main(void) {
  int A[SIZE];
  int H[BINSZ];
  int i, j;

  // H[0]= number of students 0  <= score < 20
  // H[1]= number of students 20 <= score < 40
  // H[2]= number of students 40 <= score < 60
  // H[3]= number of students 60 <= score < 80
  // H[4]= number of students 80 <= score < 50


  srand(1);

  for (i=0; i<SIZE; ++i) A[i] = rand() % 101;

  for (i=0; i<SIZE; ++i) printf("A[%d]= %d \n", i, A[i]);

  for (j=0; j<10; ++j) H[j]= 0;

  for (i=0; i<SIZE; ++i) {
     j = bin(A, i);
     H[j]++;
  }

  printf("\nSorting in decreasing order \n\n");

  bubbleSort(A, SIZE);

  for (i=0; i<SIZE; ++i) printf("A[%d]= %d \n", i, A[i]);

  printf("\n");
  for (j=0; j<BINSZ; ++j)
    printf("H[%d]= %d students\n", j, H[j]);

}
```