Algorithms and Data Structures

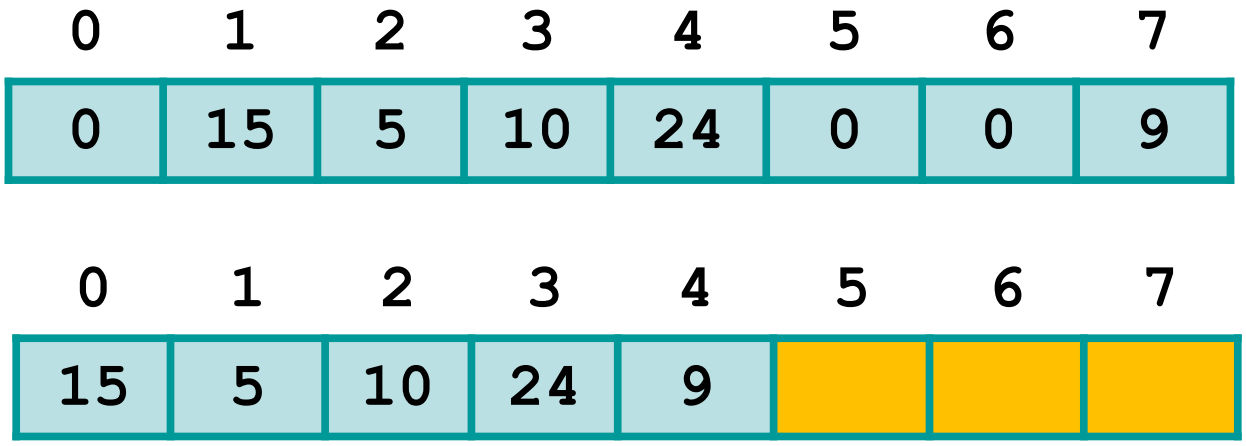
Problem Solving: Warm Up

Stefano Quer
Department of Control and Computer Engineering
Politecnico di Torino

Exercise

- Write a program which
  - Reads an array of DIM integer values
  - Eliminates all elements equal to zero from the array and shift all other values on the left, i.e., toward the beginning of the array.
  - Prints-out the resulting array (only meaningful elements)

DIM is a pre-defined constant



Solution 1

0	1	2	3	4	5	6	7
0	15	5	10	24	0	0	9

```
#define DIM 10
...
int i, j, n;
int array[DIM];
...
n = DIM;
i = 0;
while (i < n) {
  if (array[i] == 0) {
    for (j=i+1; j<n; j++) {
      array[j-1] = array[j];
    }
    n = n - 1;
  } else {
    i = i + 1;
  }
}
```

Solution 2

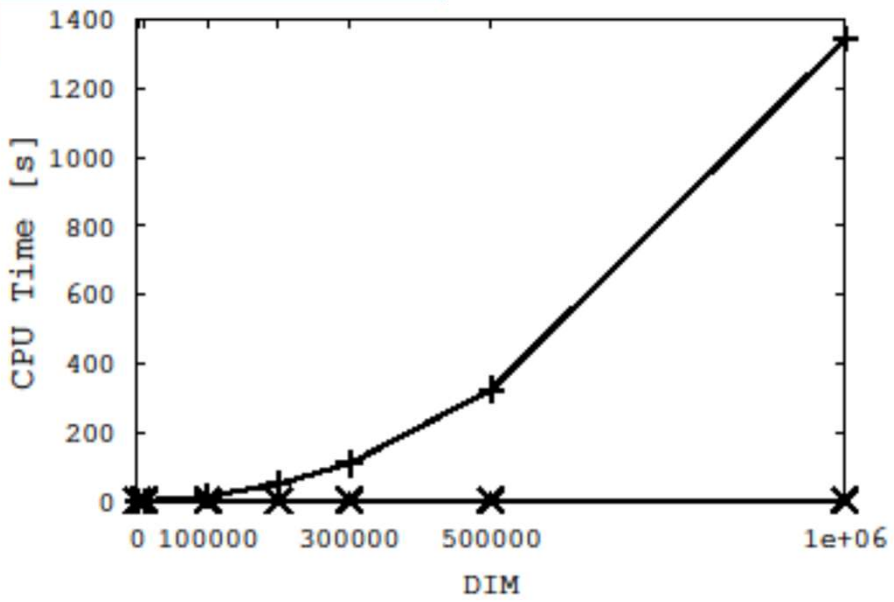
0	1	2	3	4	5	6	7
0	15	5	10	24	0	0	9

```
#define DIM 10

...

int i, j, n;
int array[DIM];

...


for (i=0, j=0; i<DIM; i++) {
  if (array[i] != 0) {
    array[j] = array[i];
    j++;
  }
}
```

Solution 1 versus 2

```
n = DIM;
i = 0;
while (i < n) {
  if (array[i] == 0) {
    for (j=i+1; j<n; j++) {
      array[j-1] = array[j];
    }
    n = n - 1;
  } else {
    i = i + 1;
  }
}
```

```
for (i=0, j=0; i<DIM; i++) {
  if (array[i] != 0) {
    array[j] = array[i];
    j++;
  }
}
```



Exercise

- Write a program that
  - Reads a sequence of integer numbers until a value strictly negative or larger than 99 is introduced
  - Prints-out an histogram made up of 10 rows of characters '#', where the numbers of characters in the
    - First row indicates the number of values previously introduced in the range [0, 9]
    - Second row indicates the number of values previously introduced in the range [10, 19]
    - …
    - The tenth (and last) row indicates the number of values previously introduced in the range [90, 99]

Exercise

```
1 2 3 4 5 10 11 12 20 30 40 50 60 62 64 70 80 90 98 -1
```

```
 0- 9 #####
10-19 ###
20-29 #
30-29 #
40-29 #
50-59 #
60-69 ###
70-79 #
80-89 #
90-99 ##
```

Solution

```
#include <stdio.h>

#define SIZE 10

int main(void) {
  int i, j, val, flag;
  int classes[SIZE];

  for (i=0; i<SIZE; i++) {
    classes[i] = 0;
  }
```

Solution

```
printf("Input sequence:\n");
flag = 1;
do {
  printf("Value = ");
  scanf("%d", &val);

  if (val<0 || val>=SIZE*SIZE) {
    flag = 0;
  } else {
    classes[val/SIZE]++;
  }
} while (flag == 1);
```

Solution

```
  for (i=0; i<SIZE; i++) {
    printf("[%2d-%2d] ", i*SIZE, (i+1)*SIZE-1);
    for (j=0; j<classes[i]; j++) {
      printf("#");
    }
    printf("\n");
  }

  return 0;
}
```

Exercise

- Write a program able to
  - Read from standard input two integer matrices, m1 of size DIM1xDIM1 and m2 of size DIM2xDIM2 storing only characters '*' and `#'
  - Find in m1 all sub-matrices equivalent to m2
  - Copy all sub-matrices equal to m2 in m1} into a new matrix m3, in which all other characters are white elements (blanks).
  - Print-out m3

*	#	*	*	*
#	#	#	*	*
*	#	*	#	*
*	*	#	#	#
*	*	*	#	*

*	#	*
#	#	#
*	#	*

*	#	*		
#	#	#		
*	#	*	#	*
		#	#	#
		*	#	*

Solution

```
#include <stdio.h>

#define DIM1 5
#define DIM2 3

int main(void) {
  int i, j, r, c, flag;
  char m1[DIM1][DIM1], m2[DIM2][DIM2], aux[DIM1][DIM1];

  /* read the first matrix */
  ...

  /* read the second matrix */
  ...
```

Solution

```
for (i=0; i<=DIM1-DIM2; i++) {
    for (j=0; j<=DIM1-DIM2; j++) {
      flag = 1;
      for (r=0; r<DIM2 && flag==1; r++) {
        for (c=0; c<DIM2 && flag==1; c++) {
          if (m1[i+r][j+c] != m2[r][c]) {
            flag = 0;
          }
        }
      }
      if (flag == 1) {
        for (r=0; r<DIM2; r++) {
          for (c=0; c<DIM2; c++) {
            aux[i+r][j+c] = 1;
          }
        }
      }
    }
  }
```

Exercise

- A file stores the set of flights of an airline company
- Given a departure airport, a departure time, and an arrival airport, find all connections with up to one stop (direct and with at most one single stop)
  - For each connection (intermediate stop) the departure time has to (obviously) follow the arrival time

Exercise

```
AZ0A1 TOR ROM 07.00 08.00
AZ0A2 TOR ROM 11.00 12.00
AZ0A3 TOR ROM 17.00 18.00
AZ0A4 TOR ROM 19.00 20.00
AZ0B1 ROM PAL 07.30 08.30
AZ0B2 ROM PAL 11.30 12.30
AZ0B3 ROM PAL 17.30 18.30
AZ0B4 ROM PAL 19.30 20.30
AZ0C1 TOR PAL 07.45 09.45
AZ0C2 TOR PAL 11.45 13.45
AZ0C3 TOR PAL 17.45 19.45
AZ0C4 TOR PAL 19.45 21.45
```

All times are within the same 24 hours

- Flights
  - From TOR
  - To PAL
  - Leaving from 16.00

```
1 stop:
  AZ0A3 TOR ROM 17.00 18.00
  AZ0B4 ROM PAL 19.30 20.30
0 stop:
  AZ0C3 TOR PAL 17.45 19.45
0 stop:
  AZ0C4 TOR PAL 19.45 21.45
```

Solution

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

#define MAX_FLIGHTS 150

typedef struct {
  char code[6];
  char depCity[4];
  char arrCity[4];
  float depTime;
  float arrTime;
} flight_t;

int flight_read (flight_t *flights, char *name);
void connection_search (flight_t *flights, int nf,
  flight_t request);
void info_write (flight_t flight);
```

Solution

```
int main(int argc, char *argv[]) {
  flight_t flights[MAX_FLIGHTS], request;
  int nf;

  nf = flight_read(flights, argv[1]);
  if (nf == 0) {
    return 1;
  }

  printf("Introduce the departure city: ");
  scanf("%s", request.depCity);
  printf("Introduce the arrival city: ");
  scanf("%s", request.arrCity);
  printf("Introduce the departure time: ");
  scanf("%f", &request.depTime);

  connection_search(flights, nf, request);
  return 0;
}
```

Solution

```
int flight_read (flight_t *flights, char *name) {
  char line[100];
  FILE *fp;
  int i=0;
  fp = fopen(name, "r");
  if (fp == NULL) {
    printf("Error opening the input file.\n");
    return 0;
  }
  while (fgets(line, 100, fp)!=NULL && i<MAX_FLIGHTS) {
    sscanf(line, "%s%s%s%f%f", flights[i].code,
                  flights[i].depCity, flights[i].arrCity,
                  &flights[i].depTime, &flights[i].arrTime);
    i++;
  }
  fclose(fp);
  return i;
}
```

Solution

```
void connection_search (flight_t *flights, int nf, flight_t request) {
  int i, j, found;
  for (i=0; i<nf; i++) {
    if (strcmp(request.depCity, flights[i].depCity)==0 &&
        request.depTime<=flights[i].depTime) {
      if (strcmp(request.arrCity, flights[i].arrCity)==0) {
        printf("Direct flight:\n"); info_write(flights[i]);
      } else {
        found = 0;
        for (j=0; j<nf && !found; j++) {
          if (strcmp(flights[j].depCity, flights[i].arrCity)==0 &&
              flights[j].depTime>=flights[i].arrTime &&
              strcmp(request.arrCity, flights[j].arrCity)==0) {
            printf("Flight with one stop:\n");
            info_write(flights[i]); info_write(flights[j]);
            found = 1;
          }
        }
      }
    }
  }
}
```

Solution

```
void info_write (flight_t flight) {
  printf("%s ", flight.code);
  printf("%s %s ", flight.depCity, flight.arrCity);
  printf("%2.2f %2.2f\n", flight.depTime, flight.arrTime);

  return;
}
```