

## Coding C

RUN

MENU

Auto saved at 18:06:18

```

1
2
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <string.h>
7
8 struct Employee {
9     int* ratings;
10    int total_score;
11 };
12
13 void inputEmployees(struct Employee* employees, int numEmployees, int numPeriods) {
14     for (int i = 0; i < numEmployees; i++) {
15         employees[i].ratings = (int*)malloc(numPeriods * sizeof(int));
16         printf("Enter ratings for employee %d:\n", i + 1);
17         for (int j = 0; j < numPeriods; j++) {
18             printf("Period %d: ", j + 1);
19             scanf("%d", &employees[i].ratings[j]);
20         }
21     }
22 }
23
24 void displayPerformance(struct Employee* employees, int numEmployees, int numPeriods) {
25     for (int i = 0; i < numEmployees; i++) {
26         printf("Ratings for employee %d:\n", i + 1);
27         for (int j = 0; j < numPeriods; j++) {
28             printf("Period %d: %d\n", j + 1, employees[i].ratings[j]);
29         }
30     }
31 }
32
33 int findEmployeeOfYear(struct Employee* employees, int numEmployees, int numPeriods) {
34     int max = employees[0].ratings[0];
35     int employeeIndex = 0;
36     for (int i = 0; i < numEmployees; i++) {
37         for (int j = 0; j < numPeriods; j++) {
38             if (employees[i].ratings[j] > max) {
39                 max = employees[i].ratings[j];
40                 employeeIndex = i;
41             }
42         }
43     }
44     return employeeIndex;
45 }
46
47 int main() {
48     int numEmployees, numPeriods;
49     printf("Enter number of employees: ");
50     scanf("%d", &numEmployees);
51     printf("Enter number of periods: ");
52     scanf("%d", &numPeriods);
53
54     struct Employee* employees = (struct Employee*)malloc(numEmployees * sizeof(struct
55
56     inputEmployees(employees, numEmployees, numPeriods);
57     displayPerformance(employees, numEmployees, numPeriods);
58
59     int employeeOfYear = findEmployeeOfYear(employees, numEmployees, numPeriods);
60     printf("Employee of the year: %d\n", employeeOfYear + 1);
61
62     // Free allocated memory
63     for (int i = 0; i < numEmployees; i++) {
64         free(employees[i].ratings)
65     }
66     free(employees);
67     return 0;
68 }
69 ...

```

Tab

{ }

“ ”

;

↶

↑

↷

=

\

&amp;

,

↵

↓

⇨

```

2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <string.h>
6
7 typedef struct {
8     int* ratings;
9     int totalScore;
10 } Employee;
11 void inputRatings(Employee* employee, int numPeriods) {
12     employee->ratings = malloc(numPeriods * sizeof(int));
13     for (int i = 0; i < numPeriods; i++) {
14         printf("Enter rating for period %d: ", i + 1);
15         scanf("%d", &employee->ratings[i]);
16         employee->totalScore += employee->ratings[i];
17     }
18 }
19 void displayPerformance(Employee* employee, int numPeriods) {
20     printf("Ratings: ");
21     for (int i = 0; i < numPeriods; i++) {
22         printf("%d ", employee->ratings[i]);
23     }
24     printf("\nTotal Score: %d\n", employee->totalScore);
25 }
26 int main() {
27     int numEmployees, numPeriods;
28     printf("Enter number of employees: ");
29     scanf("%d", &numEmployees);
30     printf("Enter number of evaluation periods: ");
31     scanf("%d", &numPeriods);
32
33     Employee* employees = malloc(numEmployees * sizeof(Employee));
34     for (int i = 0; i < numEmployees; i++) {
35         printf("Employee %d:\n", i + 1);
36         inputRatings(&employees[i], numPeriods);
37         displayPerformance(&employees[i], numPeriods);
38     }
39     for (int i = 0; i < numEmployees; i++) {
40         free(employees[i].ratings);
41     }
42     free(employees);
43
44     return 0;
45 }

```

```
1
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5
6 int validateEmail(char* email) {
7     int atCount = 0, dotCount = 0;
8     for (int i = 0; email[i]; i++) {
9         if (email[i] == '@') atCount++;
10        else if (email[i] == '.' && atCount == 1) dotCount++;
11    }
12    return atCount == 1 && dotCount > 0;
13 }
14
15 int main() {
16     char* email = (char*)malloc(100 * sizeof(char));
17     printf("Enter an email address: ");
18     fgets(email, 100, stdin);
19     email[strlen(email)] = 0;
20
21     if (validateEmail(email)) printf("Valid Email\n");
22     else printf("Invalid Email\n");
23
24     free(email);
25     return 0;
26 }
```

```

1 #include <stdio.h>
2 #include <string.h>
3
4 #define BALLS 12
5
6 struct Player {
7     char playerName[50];
8     int ballScores[BALLS];
9     int totalScore;
10 };
11
12 int validateScore(int score) {
13     return (score >= 0 && score <= 6);
14 }
15
16 void playGame(struct Player *player) {
17     printf("Enter scores for %s:\n", player->playerName);
18     player->totalScore = 0;
19     for (int i = 0; i < BALLS; i++) {
20         int score;
21         printf("Ball %d: ", i + 1);
22         scanf("%d", &score);
23         if (validateScore(score)) {
24             player->ballScores[i] = score;
25             player->totalScore += score;
26         } else {
27             printf("Invalid score! Ball is marked, but no runs added.\n");
28             player->ballScores[i] = 0;
29         }
30     }
31 }
32
33 void findWinner(struct Player player1, struct Player player2) {
34     printf("\nMatch Result:\n");
35     if (player1.totalScore > player2.totalScore) {
36         printf("%s wins with %d runs!\n", player1.playerName, player1.totalScore);
37     } else if (player2.totalScore > player1.totalScore) {
38         printf("%s wins with %d runs!\n", player2.playerName, player2.totalScore);
39     } else {
40         printf("It's a tie! Both players scored %d runs.\n", player1.totalScore);
41     }
42 }
43
44 void displayMatchScoreboard(struct Player player1, struct Player player2) {
45     printf("\n%s's Performance:\n", player1.playerName);
46     for (int i = 0; i < BALLS; i++) printf("%d ", player1.ballScores[i]);
47     printf("\nTotal Score: %d\n", player1.totalScore);
48     printf("Average Score: %.2f\n", (double)player1.totalScore / BALLS);
49     printf("\n%s's Performance:\n", player2.playerName);
50     for (int i = 0; i < BALLS; i++) printf("%d ", player2.ballScores[i]);
51     printf("\nTotal Score: %d\n", player2.totalScore);
52     printf("Average Score: %.2f\n", (double)player2.totalScore / BALLS);
53 }
54
55 int main() {
56     struct Player player1, player2;
57     printf("Enter Player 1's name: ");
58     fgets(player1.playerName, sizeof(player1.playerName), stdin);
59     player1.playerName[strcspn(player1.playerName, "\n")] = 0;
60     printf("Enter Player 2's name: ");
61     fgets(player2.playerName, sizeof(player2.playerName), stdin);
62     player2.playerName[strcspn(player2.playerName, "\n")] = 0;
63     playGame(&player1);
64     playGame(&player2);
65     displayMatchScoreboard(player1, player2);
66     findWinner(player1, player2);
67
68     return 0;

```



```

3 #include <stdio.h>
4 #include <stdlib.h>
5
6 typedef struct {
7     int* ratings;
8     int totalScore;
9 } Employee;
10
11 void inputRatings(Employee* employee, int numPeriods) {
12     employee->ratings = malloc(numPeriods * sizeof(int));
13     for (int i = 0; i < numPeriods; i++) {
14         printf("Enter rating for period %d: ", i + 1);
15         scanf("%d", &employee->ratings[i]);
16         employee->totalScore += employee->ratings[i];
17     }
18 }
19
20 int main() {
21     int numEmployees, numPeriods;
22     printf("Enter number of employees: ");
23     scanf("%d", &numEmployees);
24     printf("Enter number of evaluation periods: ");
25     scanf("%d", &numPeriods);
26
27     Employee* employees = malloc(numEmployees * sizeof(Employee));
28
29     for (int i = 0; i < numEmployees; i++) {
30         inputRatings(&employees[i], numPeriods);
31     }
32
33     for (int i = 0; i < numEmployees; i++) {
34         free(employees[i].ratings);
35     }
36     free(employees);
37
38     return 0;
39 }

```

```

1
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 typedef struct {
6     char type[50];
7 } Crop;
8
9 typedef struct {
10     int temperature;
11 } Weather;
12
13 typedef struct {
14     char type[50];
15 } Equipment;
16
17 typedef struct {
18     int soilNutrients;
19 } Sensor;
20
21 typedef struct {
22     Crop* crops;
23     Weather* weatherForecasts;
24     Equipment* equipment;
25     Sensor* sensorData;
26 } Field;
27
28 typedef struct {
29     Field** fields;
30 } RegionalHub;
31
32 int main() {
33     RegionalHub hub;
34     hub.fields = malloc(2 * sizeof(Field*));
35
36     for (int i = 0; i < 2; i++) {
37         hub.fields[i] = malloc(sizeof(Field));
38         hub.fields[i]->crops = malloc(sizeof(Crop));
39         hub.fields[i]->weatherForecasts = malloc(sizeof(Weather));
40         hub.fields[i]->equipment = malloc(sizeof(Equipment));
41         hub.fields[i]->sensorData = malloc(sizeof(Sensor));
42     }
43
44     for (int i = 0; i < 2; i++) {
45         free(hub.fields[i]->crops);
46         free(hub.fields[i]->weatherForecasts);
47         free(hub.fields[i]->equipment);
48         free(hub.fields[i]->sensorData);
49         free(hub.fields[i]);
50     }
51     free(hub.fields);
52
53     return 0;
54 }

```

```

1
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 typedef struct {
7     int* scores;
8 } Profile;
9
10 typedef struct {
11     int resolution;
12     int playbackPosition;
13 } Device;
14
15 typedef struct {
16     char title[50];
17     int rating;
18 } Content;
19
20 int main() {
21     int numRows = 2;
22     int numCols = 3;
23     int** engagement = malloc(numRows * sizeof(int*));
24     for (int i = 0; i < numRows; i++) engagement[i] = malloc(numCols * sizeof(int));
25
26     Profile* profiles = malloc(numRows * sizeof(Profile));
27     for (int i = 0; i < numRows; i++) profiles[i].scores = engagement[i];
28
29     int numDevices = 2;
30     Device** devices = malloc(numRows * sizeof(Device*));
31     for (int i = 0; i < numRows; i++) devices[i] = malloc(numDevices * sizeof(Device));
32
33     int numContents = 3;
34     Content** contents = malloc(numCols * sizeof(Content*));
35     for (int i = 0; i < numCols; i++) contents[i] = malloc(numContents * sizeof(Content));
36
37     for (int i = 0; i < numRows; i++) {
38         free(engagement[i]);
39         free(devices[i]);
40     }
41     free(engagement);
42     free(devices);
43     for (int i = 0; i < numCols; i++) free(contents[i]);
44     free(contents);
45     free(profiles);
46
47     return 0;

```