یادآوری: تمامی تمرینات و اطلاعات مربوط به تحویل آنها در سایت درس قرار داده میشوند:

http://ele.aut.ac.ir/~btaheri/cpp/

وکتور ها:

- تعریف وکتور ها:

با استفاده از کتابخانه ی :

```
#include <vector>
```

می توان به صورت زیر تعریف کرد:

```
vector<type> vectorName;
```

تعریف فوق دیده میشود عددی برای معرفی کردن تعداد خانههای وکتور استفاده نشدهاست. این نشان دهندهی دینامیکبودن طول وکتور است. بطوریکه در قسمتهای بعدی خواهیددید، توابع عضوی را از کلاس `vector` معرفی خواهیمکرد که بتوانیم به هر تعداد که بخواهیم عضو به خانههای وکتور اضافه کنیم. برای تعریف مجموعهای از ۵ خانهی `int` به دنبال هم میتوان نوشت .

```
vector<int> gradeForStudent(5);
```

برای دسترسی به مقادیر خانههای وکتور باید از عملگر `[ ]` استفاده کنیم. در اینصورت برای مشخص کردن هر خانه، از شمارندهی اندیس استفاده می - کنیم؛ برای مثال:

```
gradeForStudent[0] = 0;
gradeForStudent[1] = 20;
.
.
.
gradeForStudent[4] = -10;
```

توجه کنید که در مثال معرفی شده، مانند آرایهها، اندیس مجاز از صفر تا ۴ است.

میتوان وکتورها را زمانی که تعریف میشوند، مقداردهی کرد. برای مثال در کد زیر وکتور `chVector` با تعداد ۵ خانه معرفی شده و تمام خانههای آن با `'a'` مقداردهی شدهاند همچنین `intvector` وکتوری با ۵ خانه و با مقادیر اولیهی ۱ است.

```
vector<char> chVector(5, 'a');
vector <int> intVector(5, 1);
```

نکته: زمانی که یک وکتور تعریف میشود، قبل از مقداردهی به آن، مقادیر تمامی خانههای آن بصورت پیشفرض صفر است. لذا اگر در برنام ه‌ای مقادیر اولیهی متغیرها مهم باشد، لازم است به این مسئله توجه کنید.

- **سوال:** نکته ی قبل را نشان دهید!
- **سوال:** نشان دهید عملگرهای زیر به طور پیشفرض برای وکتورها تعریف شدهاند:

```
=< ,< ,=! ,=> ,> ,==
```

- **سوال:** در صورتی که از محدودهی اندیس وکتورها خارج شوید چه اتفاقی رخ می دهد؟؟! (خطای منطقی؟ خطای سینتکسی؟...)
- **نکته:** وکتورها مانند متغیرهای عادی که در فصلهای گذشته خواندهایم، میتوانند بصورت سراسری [1] یا محلی[2] تعریف شوند.
- **سوال:** نکته ی قبل را نشان دهید!
- **تابع `size()`** : تابعی است عضو کلاس `vector` که به ازای هر شی از جنس `vector`، تعداد خانههای آن را در خروجی، نشان خواهد داد. برای مثال در برنامهی زیر یک `vector` با ۵ خانه تعریف شدهاست. لذا `gradeForStudents.size()`، دارای مقدار `5` خواهدبود.

```
vector<int> gradeForStudent(5);
cout << gradeForStudent.size();
```

[1] Global
[2] Local

- **تابع `push_back()`:** یک تابع عضو از خانوادهی کلاس وکتور است. این تابع، مقداری جدید را به انتهای وکتور اضافه میکند و اندازهی وکتور را به اندازهی یک واحد افزایش میدهد.

```
void push_back(const T &val);
```

- مثال:

```
1.  //File:Push_back.cpp
2.  #include <iostream>
3.  #include <vector>
4.  using namespace std;
5.
6.  int main ()
7.  {
8.     vector<int> intVector(1, 5);
9.     vector<char> charVector(1,'a');
10.    intVector.push_back(2);
11.    charVector.push_back('b');
12.    cout << "intVector[0]: " << intVector[0] << endl;
13.    cout << "intVector[1]: " << intVector[1] << endl;
14.    cout << "charVector[0]: " << charVector[0] << endl;
15.    cout << "charVector[1]: " << charVector[1] << endl;
16.    return 0;
17. }
```

- **سوال:** عملکرد برنامه ی فوق را توضیح دهید!
- **تابع `pop_back()`:** این تابع مانند `push_back()` است، با این تفاوت که `pop_back()` آخرین عضو وکتور را پاک میکند
- مثال:

```
1.   //File:Pop_back.cpp
2.   #include <iostream>
3.   #include <vector>
4.   using namespace std;
5.
6.   int main ()
7.   {
8.      vector<int> myvector;
9.      myvector.push_back (100);
10.     myvector.push_back (200);
11.     myvector.push_back (300);
12.     cout << "size of vector before pop_back is:"
13.          << myvector.size() << endl;
14.     myvector.pop_back();
15.     cout << "siz of vector after pop_back is:"
16.          << myvector.size()<< endl;
17.     return 0;
18.  }
```

- **سوال:** عملکرد برنامه ی فوق را توضیح دهید!
- نکته: وکتور ها دارای برخی تابع دیگر نیز هستند!

Insert()	مقدار جدیدی را به یک وکتور اضافه کرد. تفاوت تابع `insert()` با تابع `push_back()` آن است که با استفاده از تابع `insert()` میتوان تعیین کرد که این عضو جدید در کجای وکتور قرار گیرد
erase()	میتوان یک سری از درایههای وکتور را پاک نمود
swap()	تعویض مقدار دو وکتور به کار میرود
clear()	تابع ()erase برای پاککردن وکتور مفید است؛ امّا اگر بخواهیم همه درایههای یک وکتور را با هم پاک کنیم، تابع مفیدتری هم وجود دارد و آن تابع ()clear است

at()	میتوان برای اشاره به خانهی خاصّی از وکتور استفاده کرد
back()	به آخرین درایه وکتور ارجاع میدهد
max_size()	بیشترین تعداد عضوی را که یک وکتور میتواند داشته باشد بر میگرداند
empty()	با استفاده از تابع()empty میتوان فهمید که آیا وکتور خالی است یا خیر

- **سوال:** کاربرد هر کدام از توابع فوق را نشان دهید!
- مثال: ارجاع وکتور ها به توابع:

```
1.  //File:CallAVector.cpp
2.  #include <iostream>
3.  #include <iomanip>
4.  #include <cmath>
5.  #include <vector>
6.  using namespace std;
7.
8.  const int SIZE = 24;
9.  const int MAX_PRINT = 24;
10.
11. void printArray(float a[][SIZE])
12. {
13.    if (SIZE <= MAX_PRINT)
14.    {
15.       for(int i = 0; i < SIZE; i = i + 1)
16.       {
17.          for(int j = 0; j < SIZE; j = j + 1)
18.          cout << setw(2) << static_cast<int>(a[i][j]) << " ";
19.          cout << "\n";
20.       }
21.    }
22.    else
23.       cout << "Array is too big to print.\n";
24.    return;
25. }
26.
27. void productArray(vector<float>& y,float O[][SIZE],vector<float> x)
28. {
29.    for(int i = 0; i < SIZE; i = i + 1)
30.       for(int j = 0; j < SIZE; j = j + 1)
31.           y[i] = y[i] + O[i][j] * x[j];
32.    return;
33. }
34.
35. int main(void)
36. {
37.    /* Initialize the D array */
38.    float D[SIZE][SIZE] = {0};
39.    for(int i = 0; i < SIZE - 1; i = i + 1)
40.    {
41.       D[i][i] = -1;
42.       D[i][i + 1] = 1;
43.    }
44.    // Initialize the I array
45.    float I[SIZE][SIZE] = {0};
46.    for(int i = 1; i < SIZE; i = i + 1)
47.       for(int j = 0; j <= i; j = j + 1)
48.          I[i][j] = 1;
49.
50.    // Print out D and I arrays
51.    cout << "The \"D\" array:\n"; printArray(D);
52.    cout << "The \"I\" array:\n"; printArray(I);
53.
54.    // Initialize the f and x vectors
55.    vector<float> f(SIZE), x(SIZE);
56.    for(int i = 0; i < SIZE; i = i + 1)
57.    {
58.       x[i] = -1 + 2.0*i/(SIZE - 1);
```

```
59.       f[i] = 1.0/(1.01 - x[i]);
60.    }
61.
62.    // Operate on f with D and normalize
63.    vector<float> d(SIZE);
64.    productArray(d,D,f);
65.    for(int i = 0; i < SIZE - 1; i = i + 1)
66.       d[i] = d[i]/(x[i+1] - x[i]);
67.
68.    // Operate on f with I and normalize
69.    vector<float> F(SIZE);
70.    productArray(F,I,f);
71.    for(int i = 1; i < SIZE; i = i + 1)
72.       F[i] = F[i]*(x[i] - x[i-1]);
73.
74.    // Print results
75.    cout << " x " << " f " << " d " << " F\n";
76.    for(int i = 0 ; i <= SIZE - 1 ; i = i + 1)
77.       cout << setiosflags(ios::fixed) << setw(11) << x[i] << " "
78.            << setw(11) << f[i] << " "
79.            << setw(11) << d[i] << " "
80.            << setw(11) << F[i] << "\n";
81.    return 0;
82. }
```

- **سوال:** اجرای مراحل مختلف برنامه ی بالا را توضیح دهید!
- **سوال:** با نوشتن برنامه ای ارسال call by value و call by reference متغیر به تابع را نشان دهید.

استراکچر ها:

نمونه ای از تعریف استراکچر ها:

```
// Define a structure called Student
struct Student
{
   string name;
   float gpa;
   int rank;
};
```

- برنامهی بالا یک استراکچر به نام student را ایجاد میکند. این استراکچر دربردارندهی نام دانشآموز (name)، معدل (gpa) و رتبهی وی (rank) میباشد. به صورت زیر هم میتوان چند متغیّر از نوع استراکچر student ایجاد نمود:

```
Student ali, maryam, group A[10];
```

- متغیرهای ali و maryam و آرایهی group_A از نوع student تعریف شدهاند. همانطور که ذکر شد راه دیگر تعریف متغیّرها از نوع استراکچر، تعریف آنها در انتهای بدنهی استراکچر است. بصورت معادل میتوانستیم متغیرها را اینگونه تعریف کنیم:

```
// Define a structure called Student
struct Student
{
   string name;
   float gpa;
   int rank;
} ali, maryam, group A[5];
```

- استراکچرها در همه جا میتوانند تعریف شوند، با این حال تعریف آنها خارج از تابع main یا خارج از هر تابع دیگری متداولتر است. در این حالت تمام قسمتهای برنامه به استراکچر مورد نظر دسترسی دارند.
- دسترسی به هر یک از اعضای استراکچر می تواند توسط عملگر نقطه «.» انجام گیرد. در واقع قبل از عملگر «.»، نام متغیر از نوع استراکچر مورد نظر و بعد از آن متغیری که در داخل استراکچر تعریف شدهاست، قرار میگیرد. برای مثال برای مقداردهی [3] به استراکچر معرّفی شده، میتوان اینگونه عمل کرد:

```
// Define a structure called Student
ali.name = "ali ali";
ali.gpa = 19.4;
ali.rank = 2;
```

- روش مستقیم دیگر برای مقداردهی به یک نوعداده از جنس استراکچر، در هنگام تعریف یا مقداردهی اوّلیه[4] است. مثلاً در همان مثال بالا داریم:

[3] Assignment

```
Student ali = { "ali ali", 19.4, 2 }
```

- با توجّه به آنچه ذکر شد میتوان با یک متغیّر تعریف شده از جنس یک استراکچر، مانند یک متغیّر معمولی رفتار کرد. استفاده از چنین ساختاری درست است:

```
Student s1, s2;
s1 = s2;
```

- مثال:

```
1.  //File: structureValueAssignment.cpp
2.  #include <iostream>
3.  using namespace std;
4.
5.  struct Student
6.  {
7.     string name;
8.     float gpa;
9.     int rank;
10. };
11. int main()
12. {
13.    //Declaration:
14.    Student ali;
15.
16.    //Assignment:
17.    ali.name = "ali ali";
18.    ali.gpa = 19.4;
19.    ali.rank = 2;
20.    cout << "ali.name: " << ali.name << "\nali.gpa: "
21.         << ali.gpa << "\nali.rank: " << ali.rank
22.         << endl << "============================\n";
23.
24.    //Assignment when declaration:
25.    Student maryam = {"maryam maryam", 19.4, 2};
26.    cout << "maryam.name: " << maryam.name << "\nmaryam.gpa: "
27.         << maryam.gpa << "\nmaryam.rank: " << maryam.rank
28.         << endl << "============================\n";
29.
30.     //Batch assignment:
31.     Student group_A[5] = ali;
32.     group_A[4] = maryam;
33.     for(int itr=0; itr<5; itr++)
34.     cout << "group_A[" << itr << "].name: "
35.          << group_A[itr].name << endl;
36.
37.     return 0;
38. }
```

- **سوال:** عملکرد قسمت های مختلف برنامه ی فوق را توضیح دهید.
- **مثال:** ارجاع به توابع:

```
1.  //File: structureFunCall.cpp
2.  #include<iostream>
3.  #include<string>
4.  using namespace std;
5.
6.  struct Student
7.  {
8.     string name;
9.     int section;
10.    float grade;
11. };
12. void switchStruct1(struct Student a, struct Student b)
13. {
14.    struct Student temp = a;
15.    a = b;
16.    b = temp; // Switch a and b
17.    return;
18. }
19. void switchStruct2(struct Student &a, struct Student &b)
20. {
21.    struct Student temp = a;
22.    a = b;
23.    b = temp; // Switch a and b
```

[4] Initialization

```
24.    return;
25. }
26. int main(void)
27. {
28.    struct Student Dan = {"Smith, Dan", 210, 79.8},
29.                   Jan = {"Brown, Jan", 201, 89.3};
30.    cout << "Before switching: " << Dan.name << endl;
31.    switchStruct1(Dan,Jan);
32.    cout << "After switching(Pass-by-Value): " << Dan.name << endl;
33.    switchStruct2(Dan,Jan);
34.    cout << "After switching(Pass-by-Reference): " << Dan.name
35.         << endl;
36.    return 0;
37. }
```

- **سوال:** در برنامه ی فوق تفاوت بین دو ارجاع متفاوت استراکچر به تابع را توضیح دهید.
- مثال: فیلم های مورد علاقه ی تدریسیار!

```
1.  //File structureFunValuePass.cpp
2.  #include <iostream>
3.  #include <string>
4.  #include <sstream>
5.  using namespace std;
6.
7.  struct movieT
8.  {
9.     string title;
10.    int year;
11. } mine, yours;
12.
13. void printmovie (movieT movie);
14.
15. int main ()
16. {
17.    string mystr;
18.    mine.title = "2001 A Space Odyssey";
19.    mine.year = 1968;
20.    cout << "Enter title: ";
21.    getline (cin,yours.title);
22.    cout << "Enter year: ";
23.    getline (cin,mystr);
24.    stringstream(mystr) >> yours.year;
25.    cout << "My favorite movie is:\n ";
26.    printmovie (mine);
27.    cout << "And yours is:\n ";
28.    printmovie (yours);
29.    return 0;
30. }
31. void printmovie (movieT movie)
32. {
33.    cout << movie.title;
34.    cout << " (" << movie.year << ")\n";
35. }
```

- **سوال:** عملکرد برنامه ی فوق را توضیح دهید.
- **مثال:** استراکچر های تودرتو

```
1.  //File: nestedStructures.cpp
2.  #include <iostream>
3.  using namespace std;
4.
5.  struct movieT
6.  {
7.     string title;
8.     int year;
9.  };
10. struct Student
11. {
12.    float gpa;
13.    int rank;
14.    movieT favorateMovie;
15. };
16. int main()
17. {
18.    Student maryam;
19.    maryam.gpa = 19.4;
```

```
20.     maryam.rank = 2;
21.     maryam.favorateMovie.title = "The lord of the rings";
22.     maryam.favorateMovie.year = 2005;
23.     return 0;
24. }
```

- **سوال:** با توجه به برنامه ی فوق عملکرد تو در توی دو استراکچر را توضیح دهید.
- **سوال:** آرایه ای از متغیری که توسط یک استراکچر ایجاد شده است را ایجاد کنید. سپس آن را توسط ورودی مقدار دهی کنید. و آن را به تابع ارجاع دهید!
- **سوال:** وکتوری از متغیری که توسط یک استراکچر ایجاد شده است را ایجاد کنید. سپس آن را توسط ورودی مقدار دهی کنید. و آن را به تابع ارجاع دهید!