جلسه اول - مقدمه درس

کارایی برنامه:

در درس الگوریتم و ساختمان داده مورد بررسی قرار می‌گیرد. با موازی سازی تا حدی می توان کارایی را افزایش داد.

خوانایی برنامه (Readability)

کاربردپذیری

نگهداری برنامه (Maintenance)

امنیت برنامه

مقیاس‌پذیری (scalability) که در درس مهندسی اینترنت مورد بررسی است.

درستی عملکرد

کتابخانه‌های آماده

قابلیت تغییر برنامه

به عنوان مثال استفاده از global variable چندان توصیه نمی شود.

دستور Go to که در زبان های قدیمی تر موجود بود و کنترل اجرای برنامه را پیچیده می کرد.

تست نرم افزار را برای لحظه آخر نگه ندارید.

تاکید درس: نوشتن برنامه های خوب

Correctness: Testing, Debugging

Maintainability: Object orientation, coding style

Reusability: object orientation

از قطعات نرم‌افزار بتوانیم مجددا استفاده کنیم.

زبان اصلی: C++

JAVA هم کاربردی است.

زبان ها: C++، C#، Python، MATLAB، JAVA، R

C++ کارایی C (تاکید روی سرعت و استفاده بهینه از منابع) را با شی‌گرایی و انعطاف پذیری ترکیب کرده است.

کتاب:

Deitel: C++ How to program

Google Search

Stackoverflow.com

Cplusplus.com

Ramtung.ir/apnotes/html

جلسه دوم – مفاهیم مقدماتی زبان C++

سلام دنیا: یک برنامه‌ی بسیار ساده در زبان سی‌پلاس‌پلاس که صرفاً یک رشته را در خروجی می‌نویسد.

```
1. #include <iostream>
2. using namespace std;
3.
4. int main()
5. {
6.       cout << "Hello World\n";
7.       return 0;
8. }
```

Iostream هدرفایلی است که امکانات ورودی و خروجی C++ را در اختیار ما می‌گذارد.

قاعده قرار دادن h. قدیمی است و تقریبا منسوخ شده است.

Std مخفف استاندارد است و تمام امکانات استاندارد C++ در این فضای نام قرار دارد.

فضاهای نام برای پرهیز از تداخل بین کتابخانه‌های نرم افزاری مختلف مورد استفاده قرار می‌گیرند.

اگر خط دوم را ننویسم بایستی هربار بنویسیم std::cout

امصا تابع: نام تابع، نوع پارامتر ورودی، نوع پارامتر خروجی

در مورد خط چهارم فعلا بدانیم که امکان پاس کردن آرگومان به تابع main از طریق command line وجود دارد.

در C++ بلافاصله بعد از امضای تابع و در خط بعدی، آکلاد را باز می کنیم. این کار در جاوا درست بعد از امضای تابع و در همان خط انجام می گیرد.

در داخل بدنه‌ی تابع، یک تب (TAB) قرار می‌دهیم.

Cout مخفف console out است که اصطلاح اطلاق شده به همان ترمینال یا command line می باشد.

G++ کامپایلر C++ در لینوکس. فایل اجرایی در لینوکس پسوند out. و در ویندوز پسوند exe. دارد.

بازگرداندن صفر اطمینان از موفقیت آمیز بودن اجرای برنامه را تضمین می کند.

تایپ رشته – خواندن از ورودی: این مثال استفاده اولیه از تایپ رشته و نحوه‌ی خواندن از ورودی را نشان می‌دهد.

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
      string name;
      cout << "Please enter your name: ";
      cin >> name;
           cout << "Hello " << name << '\n';
           return 0;
      }
```

string هدرفایلی است که امکان کار روی رشته‌ها را به ما می‌دهد.

Name شی (object) از نوع string است.

تابع cin فقط کلمه اول از ورودی را می گیرد. کلمه دنباله ای از کاراکترها است که به فضای سفید محدود می‌شود.

فضای سفید: space، TAB، Enter

Cin کاراکترهای غیر فضای سفید را می‌خواند.

خواندن کل یک خط با تابع getline انجام می‌گیرد.

خواندن چند قلم از ورودی: در این مثال چند مقدار به دنبال هم از ورودی خوانده می‌شود. دقت کنید که اگر رشته‌ای از cin خوانده شود، یک کلمه از ورودی خوانده شده در آن متغیر قرار می‌گیرد. مثلاً اگر ورودی Gholam 29 در ورودی تایپ شود مقدار name برابر Gholam و مقدار age برابر ۲۹ خواهد بود.

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
	string name;
	int age;
	cout << "Please enter your name followed by your age: ";
		cin >> name >> age;
		cout << "Hello " << name << "!\n";
		cout << "Your age is " << age << endl;
	}
```

Endl ثابتی که در هدرفایل iostream تعریف شده است و با کاراکتر ‘\n’ مترادف است.

تایپ ده انگشتی

چنانچه return 0 را برنگردانید، C++ خود به صورت پیش‌فرض مقدار صفر را برمی‌گرداند.

البته این ویژگی چندان خوب نیست. لذا سعی کنید همواره بصورت explicit مقدار صفر را برگردانید و چنانچه تابع قرار است مقدار خروجی نداشته باشد، نوع خروجی را void تعریف کنید.

خواندن از ورودی در حلقه: این برنامه تعدادی کلمه را از ورودی می‌خواند و در صورتی که کلمه‌ای تکرار شود این موضوع را با نمایش پیغامی اطلاع می‌دهد.

خواندن ورودی تا آنجا ادامه می‌یابد که کاربر با ctrl-d (یا ctrl-z در ویندوز) خاتمه ورودی را مشخص کند.

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
	string previous = "";
	string current;
	while (cin >> current) {
		if (previous == current)
			cout << "repeated word: " << current << '\n';
		previous = current;
	}
	}
```

در برنامه بالا تا زمانی که اینتر نزنیم، C++ به خواندن از ورودی ادامه می‌هد و پس از آن پردازش خط بعد را آغاز می‌کند.

خواندن ورودی از فایل، پاس کردن به فایل اجرایی در لینوکس و نوشتن خروجی در فایل

a.out<masalan.txt>output.txt

نمایش مختویات فایل text در لینوکس

Cat output.txt

پاس کردن خروجی برنامه به تابع sort لینوکس (piping in linux)

a.out<masalan.txt|sort

از مشکلات آرایه این است که اندازه آن باید ثابت باشد. جهت حل این مشکل کتابخانه استاندارد C++ وکتورها را در اختیار ما قرار می دهد. تایپ vector جزئی از کتابخانه‌ی استاندارد سی‌پلاس‌پلاس است که کار با دنباله‌ای از عناصر را **بدون دغدغه‌های مدیریت حافظه** مهیا می‌کند.

محاسبه‌ی میانگین و میانه: در این برنامه تعدادی عدد اعشاری (که نماینده‌ی دما هستند) از ورودی گرفته می‌شود و میانگین و میانه‌ی آنها در خروجی نوشته می‌شود. تعداد این اعداد در زمان نوشتن برنامه نامعلوم است.

```
1. #include <iostream>
2. #include <vector>
3. #include <algorithm>
4. using namespace std;
5.
6. int main()
7. {
8.       vector<double> temps;
9.       double temp;
10.      while (cin >> temp)
11.            temps.push_back(temp);
12.
13.      double sum = 0;
14.      for (int i = 0; i < temps.size(); ++i)
15.            sum += temps[i];
16.
17.      cout << "Mean temperature: " << sum/temps.size() << endl;
18.      sort(temps.begin(), temps.end());
19.      cout << "Median temperature: " << temps[temps.size()/2];
20.      }
```

در سر فایل algorithm عده‌ای از توابع همچون sort که در برنامه بالا در خط ۱۸ مورد استفاده قرار گرفته، تعریف شده‌اند.

مقداردهی اولیه‌ی بردارها: این مثال نشان‌دهنده‌ی مقداردهی اولیه‌ی اندازه و عناصر بردار است.

```
1. #include <vector>
2. #include <string>
3. #include <iostream>
4. using namespace std;
5.
6. int main()
7. {
8.       vector<double> vec;
9.       // vec[0] = 10;
10.      vec.push_back(1.3);
11.      vec[0] = 12.4;
12.
13.      vector<int> v(6);
14.
15.      v[0] = 5; v[1] = 7;
16.      v[2] = 9; v[3] = 4;
17.      v[4] = 6; v[5] = 8;
18.
19.      vector<string> philosopher(4);
20.
21.      philosopher [0] = "Kant";
22.      philosopher [1] = "Plato";
23.      philosopher [2] = "Hume";
24.      philosopher [3] = "Kierkegaard";
25.
26.      //philosopher[2] = 99;
27.
28.      vector<double> vd(1000, 1.2);
29.      //vd[1000] = 4.7;
30.      }
```

در مثال بالا اگر خط ۹ را کامنت نکنیم با RUNTIME ERROR مواجه خواهیم شد. زیرا در لحظه‌ی تعریف، vector شامل صفر عنصر است و با هر pushback حافظه تخصیص می‌یابد.

Segmentation fault

خطای دسترسی غیرمجاز به حافظه در لینوکس است.

در خط ۶ یک بردار ساخته‌ایم که دارای ۶ عنصر integer است و مقدار اولیه‌ی هریک از این عناصر صفر است.

خط ۲۶ با خطای کامپایلر روبرو می‌گردد. چون ۹۹ رشته نیست. اما "۹۹" یا ’۹۹‘ رشته است.

در خط ۲۸، برداری ساخته‌ایم که ۱۰۰۰ عنصر اولیه آن با مقدار اعشاری ۱/۲ پر شده است. خط ۲۹ هم با خطای segmentation (دسترسی غیرمجاز به حافظه) روبرو است.

بردار دو بعدی:

```
Vector<vector<int>>twod(3);
Twod[0].push_back(12);
Twod[1].push_back(4);
Twod[2].push_back(65);
Cout<< Twod[1][0]<<endl;
Vector<vector<int>>twod(3,vector<int>(4));
```

وکتوری با ۳ عنصر اولیه بساز و هرکدام از عناصر اولیه‌ی آن را با یک بردار چهارتایی از نوع مقدار صحیح، مقدار اولیه بده.