برنامه‌سازی پیشرفته | تمرین کامپیوتری چهارم

زمان تحویل: ۰۱/۰۲/۱۴۰۴

**ایمیل دستیاران آموزشی مربوطه:**

زانکو کریمی ،امیرمرتضی یوسفی ،زانیار غزالی ،محمدمهدی نیک‌خواه

---

# مقدمه:

این تمرین با هدف آشنایی با مفاهیم تابع‌های لامبدا[1] و قالب‌ها [2] در برنامه‌نویسی طراحی شده است و شامل دو پرسش مستقل می‌باشد. توصیه می‌شود برای درک عمیق‌تر این مفاهیم، زمان مناسبی برای بررسی و پیاده‌سازی هر پرسش اختصاص دهید.

---

# تمرین‌ها:

## طراحی کامپایلر برای زبان D++:

در این تمرین، شما نقش یکی از توسعه‌دهندگان اصلی زبان برنامه‌نویسی جدیدی به نام D++ را بر عهده دارید. این زبان هنوز وارد فاز اجرایی نشده و تنها در مرحله طراحی اولیه است. فایل‌های این زبان با پسوند dpp ذخیره می‌شوند، و یکی از اولین قدم‌های توسعه آن، ساختن کامپایلر و تعریف مجموعه‌ای از توابع و عملگرهای پایه‌ای است که در هسته این زبان استفاده خواهند شد.

برای ساختن این کامپایلر اولین گام پیاده‌سازی اولیه این توابع پایه‌ای با استفاده از توابع لامبدا است، به گونه‌ای که بتوان آن‌ها را به راحتی در نسخه اولیه کامپایلر زبان به‌کار برد. تمامی توابع باید به صورت لامبدا طراحی شوند. به گونه ای که هر عملگر یا تابع عبارت لامبدا مربوط به خودش را داشته باشد.

---

[1] Lambda function

[2] Templates

برای دسترسی راحت تر به این عبارات لامبدا می توانید آن ها را در ساختار های Key و Value مثل unordered_map به صورت زیر استفاده کنید. برای گذاشتن تایپ لامبدا ها میتوان از تایپ function از کتابخانه <functional> استفاده کرد.

به این صورت که در ساختار اول کد زیر <function<double(double, double)> توابعی (و عبارات لامبدا) را شامل می شود که دو Double گرفته و یک Double به عنوان خروجی میدهند.

```
unordered_map<string, function<double(double, double)>> operations;
...
{...
{"+", [](double a, double b) { return a + b; }},
...
}
unordered_map<string, function<double(double)>> functions;
...
{...
{"tan", [](double a) { return tan(a); },
...
}
```

## ساختار زبان:

این زبان مثل زبان های پایتون و جاواسکریپت خط به خط خوانده و اجرا میشود. و نیازی به تعریف متغیر ها ندارد. هر متغیر مقادیر اعشاری را می تواند ذخیره کند. دستورات این زبان یکی از ساختار های زیر را می توانند داشته باشند:

```
<variable_name> = <single expresstion>
<variable_name> = <expresstion> <operator> <expresstion>
<function>
```

توجه شود که به جای <expresstion> میتواند متغیر ،مقدار ثابت و یا تابع قرار بگیرد. فاصله ها در این زبان اهمیت دارند و باید به صورت بالا نوشته شوند.

همانند بقیه زبان های برنامه نویسی در این زبان نیز اسم توابع و متغیر ها نباید با اعداد شروع شوند. و اسم متغیر ها نباید از اسامی توابع تعریف شده زبان باشند.

## توابع و عملگرها:

تمامی عملگرهای این زبان تو مقدار اعشاری گرفته و یک مقدار اعشاری را به عنوان خروجی می‌دهند. توابع یک ورودی اعشاری گرفته و یک خروجی اعشاری خواهند داد. نمونه کلی استفاده از توابع به صورت زیر است:

```
<function_name>(<single variable or constant>)
```

همانطور که گفته شد کد های زبان D++ در فایل هایی با پسوند dpp. نوشته می شوند پس این کامپایلر باید بتواند محتوای کد ها را از فایل نمونه main.dpp بخواند و خروجی های احتمالی را نمایش دهد.

در ادامه چند نمونه کد قابل قبول و غیر قابل قبول آمده است:

کد قابل قبول:

```
a = 3.4
b = 3
c = b
a = 3 + a
c = b + a
b = 3 * 12.5
c = b ^ 3
c = cos(b)
a = sin(0.3) + a
c = 3.9 + cos(a)
print(c)
print(3.162)
```

کد غیر قابل قبول:

```
a =3.4
b= 3
c=b
c = “f”
a = 3 + a + 4.3
c = b + a * 2
b = cos(3 * a)
```

در ادامه یک کد D++ و خروجی آن را اوردیم:

کد:

```
q = 2.3
s = 2 + q
d = s + cos(q)
f = 2 + sin(3.12)
g = q
a = b + c
r = cot(g)
r = print(10.1)
print(r)
print(f)
```

خروجی:

```
10.1
0
2.0544
```

## عملگرها و توابع تعریف شده:

توابع و عملگرهای شناخته شده برای کامپایلر D++ به شرح زیر است. توجه کنید کد شما باید به گونه نوشته شود که اضافه کردن عملگر و تابع جدید با همان ساختار گفته شده به راحتی و با کمترین تغییر در کد صورت گیرد.

| تابع | کارکرد | عمگر | کارکرد |
|---|---|---|---|
| “sin” | گرفتن سینوس از ارگومان (برحسب درجه) | “+” | جمع |
| “cos” | گرفتن کسینوس از ارگومان (برحسب درجه) | “-” | تفریق |
| “tan” | گرفتن تانژانت از ارگومان (برحسب درجه) | “*” | ضرب |
| “cot” | گرفتن کتانژات از ارگومان (برحسب درجه) | “/” | تقسیم |
| “print” | پرینت کردن آرگومان و دادن خروجی 0 | “^” | توان |

توجه کنید که کامپایر شما فرض میکند کد کاملا بدون باگ و صحیح است. و در صورت وجود ورودی که ساختار تعریف نشده دارد لزومی ندارد خروجی مشخصی بدهد.

برای پردازش ورودی های جداشده با فاصله می‌توانید از کلاس sstream به شکل زیر استفاده کنید:

```
line = “hay hello how are you”;
string a, b, c, d, e, f;
istringstream iss(line);
iss >> a >> b;
cout >> a >> “ “ >> b >> endl;

if(iss >> c >> d >> e){
     cout << c << “ “ << d << “ “ << e << endl;
}

if(iss >> f){
     cout << “dodo” << endl;
}
```

خروجی

```
hay hello
how are you
```

## تمرین دوم:

در این تمرین، هدف شما طراحی و پیاده‌سازی یک کلاس لیست پیوندی (Linked List) است که بتواند داده‌هایی از هر نوع (int، float، string، و ...) را ذخیره و مدیریت کند. این لیست باید با استفاده از قالب‌ها (template) به صورت عمومی طراحی شود تا با هر نوع داده‌ای کار کند.

کلاس LinkedList باید دارای یک کلاس داخلی (Nested Class) به نام Node باشد که هر گره (node) از لیست را نمایش دهد. عملیات پایه‌ای که روی لیست پیوندی انجام می‌شود (مثل اضافه کردن، حذف کردن، و چاپ عناصر) باید به درستی پیاده‌سازی شوند.

### بخش اول: طراحی لیست پیوندی عمومی

ابتدا یک کلاس عمومی به نام LinkedList طراحی کنید که قابلیت ذخیره و مدیریت عناصر از هر نوع داده‌ای را داشته باشد. این کلاس باید با استفاده از template پیاده‌سازی شود و شامل یک کلاس داخلی ( nested class) به نام Node باشد.

عملیات اصلی مانند افزودن، حذف، چاپ، شمارش و پاک‌سازی باید در این لیست پیاده‌سازی شوند.

- کلاس LinkedList باید یک کلاس template باشد.
- کلاس Node باید داخل کلاس LinkedList تعریف شود.
- هر گره باید شامل:
  - یک متغیر داده‌ای از نوع T
  - یک اشاره‌گر به گره بعدی باشد (Node* next)
- کلاس LinkedList باید شامل توابع زیر باشد:

## توابع مورد نیاز:

- افزودن عنصر به ابتدای لیست void push_front(const T& value)
- افزودن عنصر به انتهای لیست void push_back(const T& value)
- حذف عنصر اول لیست void pop_front()
- حذف عنصر آخر لیست void pop_back()
- بررسی خالی بودن لیست bool is_empty() const
- حذف تمام عناصر لیست void clear()
- چاپ تمام عناصر لیست به ترتیب void print() const
- حذف اولین عنصر با مقدار مشخص bool remove(const T& value)
- شمارش تعداد عناصر لیست int size() const
- دسترسی به عضو اول لیست T& front() const
- دسترسی به عضو آخر لیست T& back() const
- افزودن امکان جستجو در لیست bool contains(const &T)

## بخش دوم: طراحی پشته (Stack) با استفاده از LinkedList

در این قسمت می‌خواهیم یک stack طراحی کنیم. با جست‌وجو در اینترنت با ساختار داده stack آشنا شوید. یک کلاس عمومی به نام Stack طراحی کنید که از کلاس LinkedList به‌عنوان ساختار داده داخلی استفاده کند.

توابع مورد نیاز در Stack:

- افزودن عنصر به پشته void push(const T& value)
- حذف عنصر بالا void pop()
- مشاهده عنصر بالا T& top() const
- چاپ تمام عناصر لیست به ترتیب print()
- شمارش تعداد عناصر لیست int size() const
- بررسی خالی بودن لیست bool is_empty()

توجه شود که در کد از هیچ آرایه‌ای یا از STL استفاده نکنید. فقط از LinkedList استفاده شود.

## بخش سوم: طراحی صف (Queue) با استفاده از LinkedList

در این قسمت می‌خواهیم یک queue طراحی کنیم. با جست‌وجو در اینترنت با ساختار داده queue آشنا شوید. یک کلاس عمومی به نام Queue طراحی کنید که از کلاس LinkedList استفاده می‌کند.

توابع مورد نیاز در Queue:

- افزودن عنصر به انتهای صف void enqueue(const T& value)
- حذف عنصر از ابتدای صف void dequeue()
- مشاهده عنصر اول T& front() const const
- bool is_empty() const
- int size() const
- print()

نکات مهم برای همه‌ی بخش‌ها:

- از کلاس LinkedList که در بخش اول نوشته‌اید، در طراحی Stack و Queue استفاده کنید.
- تمام کلاس‌ها باید template باشند.
- از std::list, std::vector یا دیگر ساختارهای آماده STL استفاده نکنید.
- رعایت اصول مدیریت حافظه (عدم memory leak) الزامی است.
- کد خود را با چند نوع داده مختلف مانند int، string و struct تست کنید.

ورودی نمونه بعد از فراخوانی هر یک از کلاس های طراحی شده:

```
int main() {
    cout << "===== LinkedList Test =====\n";
    LinkedList< string> names;
    names.push_back("Ali");
    names.push_front("Zahra");
    names.push_back("Reza");
    names.print(); // “Zahra Ali Reza”
    cout << "Size: " << names.size() << endl; // “Size: 3”

    cout << "Contains 'Ali'? " <<
    (names.contains("Ali") ? "Yes" : "No") << endl;
    // “Contains 'Ali'? Yes”

    cout << "===== Stack Test =====\n";
    Stack<int> stack;
    stack.push(10);
    stack.push(20);
    stack.push(30);
    stack.print(); // “30 20 10”
    cout << "Top: " << stack.top() << endl; // “Top: 30”
    stack.pop();
    stack.print(); // ”20 10”

    cout << "===== Queue Test =====\n";
    Queue<string> queue;
    queue.enqueue("first");
    queue.enqueue("second");
    queue.enqueue("third");
    queue.print(); // “first second third”
    cout << "Front: " << queue.front() << endl; // ”Front: first”
    queue.dequeue();
    queue.print(); //“second third”

    return 0;
}
```

;

## نکات پایانی و نحوه تحویل:

- فایل تحویلی شما در ایلرن باید شامل دو فایل سی‌پلاس‌پلاس (.cpp) مستقل باشد که هر کدام مربوط به یکی از تمرین‌ها هستند. این دو فایل را در یک فایل زیپ (.zip) قرار دهید. فایل‌ها را با قالب روبه‌رو نام‌گذاری کنید:

```
<FirstName>_<LastName>_<StudentNumber>_CA4.zip
├── <FirstName>_<LastName>_<StudentNumber>_CA4_Q1.cpp
└── <FirstName>_<LastName>_<StudentNumber>_CA4_Q2.cpp
```

به طور مثال:

```
Amirreza_Akbari_810899000_CA4.zip
├── Amirreza_Akbari_810899000_CA4_Q1.cpp
└── Amirreza_Akbari_810899000_CA4_Q2.cpp
```

دقت داشته باشید که آزمون‌های خودکار، فایل‌های شما را بر اساس این قالب نام‌گذاری اجرا می‌کنند. هرگونه مغایرت بین شیوه نام‌گذاری شما و شیوه ذکرشده در بالا، به معنی تصحیح‌نشدن فایل‌های شما و ازدست‌دادن نمره خواهد بود.

- برنامه‌ی شما باید با مترجم g++ با استاندارد c++20 ترجمه و در زمان معقول برای ورودی‌های آزمون اجرا شود.
- رعایت اصول کدنویسی تمیز و منظم در طراحی برنامه اهمیت زیادی دارد. نام‌گذاری معنادار و منسجم (Consistent) برای متغیرها و توابع، تقسیم کد به بخش‌های منطقی، نوشتن توابع کوتاه، عدم وجود کد تکراری و رعایت دندانه‌گذاری (Indentation) باعث افزایش خوانایی کد شما و اطمینان از صحت عملکرد برنامه می‌شود.
- وبسایت‌هایی مانند stackoverflow و cplusplus می‌توانند در حل چالش‌های این تمرین به یاری شما بیایند.
- درستی برنامه‌ی شما از طریق آزمون‌های خودکار سنجیده می‌شود. پیشنهاد می‌کنیم با استفاده از ابزارهایی مانند diff خروجی برنامه‌ی خودتان را با خروجی‌هایی که به‌عنوان نمونه در اختیارتان قرار گرفته مطابقت دهید.
- در پایان توجه داشته باشید که نمره‌ی این تمرین تنها به اجرای صحیح برنامه و مطابقت خروجی‌های شما با آزمون‌های خودکار وابسته نیست؛ بلکه ارائه حضوری و میزان تسلط شما در کد تحویلی‌تان نیز در نمره‌ی نهایی تاثیر خواهد داشت.