گزارش سری دوم تمرین برنامه نویسی

زمستان ۹۶

نام و نام خانوادگی:

محمدرضیئی

شماره دانشجویی:

۹۴۲۳۰۵۲

تعداد سوالات: ۳

تاریخ تألیف: ۱۴اسفند۹۶

تمرین شماره ۱

الف : این کد درست است.

در خط اول ابتدا یک متغیر integer تعریف کرده و در خط بعدی باز هم متغیر integer ای را تعریف کرده ایم ولی این بار آدرسی را که دریافت می کند که همان آدرس a است را const تعریف کرده ایم، بنابراین نمی توان آدرس دیگری را در آن ذخیره کرد ولی می توان مقداری را که این پوینتر به آن اشاره می کند را تغییر داد و با ++(b*) می توان مقداری را که در a است را یکی اضافه کرد.

در خط بعد نیز مقداری را که در a است را یکی اضافه میکنیم و در نتیجه مقداری که در a حالا موجود است ۱۲ می باشد و زمانی که cout می کنیم a و b* هردو ۱۲ را در خروجی چاپ می کنند و b آدرس a است که در b ذخیره شده است.

ب : کد نوشته شده در این قسمت تا اولین cout مشکلی ندارد چون که در خط ۶ نوشته ایم cons int* در نتیجه نمی توان مقدار ذخیره شده در b* را که همان b است تغییر داد ولی آدرسی را که در b ذخیره شده است را تغییر داد پس با اروری مواجه نشده ایم ولی در خط بعدی به وسیله دستور int* const در آدرسی را که در پوینتر b ذخیره شده است را const کرده ایم و در نتیجه نباید آدرس دیگری را در آن ذخیره کنیم که در خط ۱۰ این کار را کرده است و آدرس d را در آن ذخیره کرده است

پ : در این کد در خط ۲ یه متغیر از نوع char* ایجاد کرده که ۸ بایتی بوده و یک string به نام Amir jahanshahi به آن داده ایم و در خط بعدی name را در p1 میریزیم که آن هم char* است که می توان که string است. خط ۵ قابل اجراست چون که a که یک خانه از حافظه به طول به یک بایت است و ما آدرس آن را به p1 داده داده ایم، ولی اگرچه اروری نمی گیرد ولی در خط بعد در هنگام cout کردن p* که همان a را برمی گرداند ولی (p+1)* به آدرس خانه بعدی اشار میکند و در نتیجه یه مقدار تصادفی مثلا در رایانه من ۵ را نمایش می دهد و همین طور در مورد (p+2)* که به آدرس دو خانه بعد اشاره می کند و اینبار رایانه من k^ را نمایش داد. خط ۸ ارور می دهد چون که p1 را const char* کرده ایم پس نمی توان مقداری را که این متغیر به آدرس آن اشاره می کند تغییر داد همچنین نوع آن char* است. خط ۱۰ نیز اجرا نمیشود و ارور می دهد چون که name از نوع const char* است و p2 از نوع char* و نمی تواند آن را خودش cast کند.

د : خط ۱ یک آرایه داینامیک یک بعدی به نام p1 تعریف کرده است که ده تا آرگومان ورودی دارد. خط ۲ یک آرایه استاتیک است که ده تا آرگومان ورودی دارد. در خط ۳ تابعی فراخوانی شده است که نام تابع به صورت پوینتری تعریف شده است. در خط چهارم در حقیقت آرایه ای از توابع را داریم که در واقع ده تا تابع داریم که هر کدام یک آرایه دوبعدی می گیرند.

- **تمرین شماره ۲**

ابتدا به بخشی از کد int main() نگاهی می کنیم.

```
const char* FileName{"Error_find.txt"};
std::ifstream File;
File.open(FileName, std::ios::in | std::ios::out);
std::string oneWord{};
std::vector<std::string> output;
while(File >> oneWord)
{
```

```
            std::vector<std::string> temp {check_find(oneWord)};
            output.insert( output.end(), temp.begin(),temp.end());
      }
```

در جا برای خواندن فایل متنی از کتابخانه fstream استفاده شده است. یک وکتور برای خروجی درنظر گرفته ام که این خروجی در واقع لغات غلط را جمع آوری میکند.

نکته اینجاست که میتوان برای خروجی که شرایط آن در سوال گرفته شده است، میتوان بدون جمع آوری فقط آن را نمایش داد. اما برای کارهای بیشتر تصمیم بر آن شد که این مقادیر را در وکتوری جمع آوری شود.

برای خواندن فایل از تابع زیر استفاده شده است.

```
while(File >> oneWord)
      {
            std::vector<std::string> temp {check_find(oneWord)};
            output.insert( output.end(), temp.begin(),temp.end());
      }
```

در اینجا check_find تابعی است که مقادیر غلط به صورت وکتوری خارج میشود. اینکار برای آن است که ممکن است بین دو کلمه فاصله ای نباشد و با علایمی جدا شده باشد. در این حالت از روش فوق استفاده کردم.

تابع check_find به صورت زیر تعریف میشود.

```
std::vector<std::string> check_find(const std::string& oneWord)
{
      const std::string separators{ " ,;:()[]{}.\"!?'\n\t" },
                                    vowels{"AaEeIiOoUu"}; // Word
```

جدا کننده های کلمات و حروف صدادار در یک استرینگ تعریف شده است.

```
      size_t start { oneWord.find_first_not_of(separators) }; //
First word start index
```

کلمات از پس از علایمی مانند ویرگول شروع میشوند.

```
      size_t end {};
      std::vector<std::string> Word, Out;
      while (start != std::string::npos) // Find the words
      {
            end = oneWord.find_first_of(separators, start + 1); //
Find end of word
            if (end == std::string::npos) // Found a separator?
                  end = oneWord.length(); // No, so set to last +
1Chapter 7  Working With StringS
            Word.push_back(oneWord.substr(start, end - start)); //
Store the word
```

پس از یک حلقه دقیقا لغت واقعی را بدون علامت در Word ذخیره میکند.

```
            start = oneWord.find_first_not_of(separators, end +
1); // Find 1st character of next word
      }
      start = oneWord.find_first_not_of(vowels) ;
      for(const auto& str : Word)
      {
```

```
          end = str.find_first_of(vowels, start + 1); // Find
end of word
          if (end == std::string::npos) // Found a separator?
               end = str.length(); // No, so set to last +
1Chapter 7
Working With StringS
```
حروف صدا دار را میابد و در یک یک آرایه ذخیره میکند.
```
          if ((end - start ) >= 5)
               if(!isCapital(str.substr(start,end)))
                    Out.push_back(str);
```
درصورتی که بیشتر از ۵ کاراکتر بیصدا پشت هم آیند چک میشود که آیا حروف بزرگ یا حروف کوچک هستند.
```
          start = str.find_first_not_of(vowels, end + 1); //
Find 1st character of next word
```
سراغ پنج حرف بعدی میرود.
```
     }
     return Out;
}
```

همانگونه که توضیح داده شد کد فوق به درستی عمل خواهد کرد.
نحوه ی عمکرد آن در زیر مشخص است.

تمرین سوم

در این سوال ساختمان داده ای به صورت زیر تعریف شده است.

```
class Queue
{
public:
	Queue(int);
	Queue(const char*);
	Queue();
	Queue(const Queue&);
	~Queue();
	int Size() const;
	double enQueue(double);
	double deQueue();
	void displayQueue() const;
	bool isEmpty() const;
	bool isFull() const;
	int inc(int) const;
private:
	double* data;
	int head;
	int tail;
	int size;
};
```

عبارت head اندیس مکانی است که داده ی آن مکان در()dequeue از صف خارج میشود و به همین صورت tail اندیس مکانی است که در ()enqueuer داده در آن مکان ریخته میشود و size نیز، اندازه ظرفیت صف را نشان میدهد.

تابع مهم inc به صورت زیر تعریف میشود تا حالت گردشی را به صف تشکیل شده از آرایه ی مارا بدهد.

```
int Queue::inc(int i) const {    return (( i + 1 ) % size); }
```

البته قبل از معرفی باقی بخش های سوال توجه داریم که در این سوال با این تابع بالا بخشی از داده را از دست میدهیم که هزینه ی آن برابر شدن شرط پر و خالی بودن میشود.
برای حل این مشکل دو راه استفاده میشود که روش اول بخاطر نداشتن خطا در مقدار ذخیره سازی برای حالتی که از این ساختمان داده به کرات استفاده کردیم، پیشنهاد می شود.

۱) در این روش size را یک عدد افزایش میدهیم. در این صورت شرایط پر و خالی بودن به صورتی که بعدا به آن اشاره میشود، تغییر خواهد کرد.

۲) در این روش head , tail حالت چرخشی ندارد و عملیات modگیری در لحظه ی استفاده از آن گرفته میشود.
و شرط خالی بودن آن صفر بودن اختلاف آن ها و شرط پر بودن آن به اندازه سایز اختلاف داشتن آن است.

با روش اول، توابع enqueuer , dequeuer به صورت زیر میشوند.

```
double Queue::enQueue(double input)
{
     if ( isFull() )
     {
          std::cout << "Cannot enQueue(" << input << ") because
dataStructure is full" << std::endl;
     }
     data[tail] = input;
     tail = inc(tail);
     return input;
}
double Queue::deQueue()
{
     if ( isEmpty() )
     {
          std::cout << "Cannot deQueue() because dataStructure
is  empty" << std::endl;
     }
     double tmp{data[head]};
     head = inc(head);
     return tmp;
}
```

در این حالت شرط پر و خالی بودن به صورت زیر میشود.

```
bool Queue::isEmpty() const
{
     return head == tail;
}
bool Queue::isFull() const
{
     return ((head-tail)%size == 1);
}
```


Q3 git:(master) ./main
12
20.5
14
-1
12
20.5
14
-1
15.5
-6
Deleted value = 12
Deleted value = 20.5
14
-1
15.5
-6
Q3 git:(master)

تمرین شماره ۴
این سوال نیاز به توضیح زیادی ندارد زیرا تابع را به صورت زیر تعریف کرده ایم.

```
<template <class T
void selectionSort(T arr[], int n)
}
    One by one move boundary of unsorted subarray //
        for (int i {}; i < n-1; i++)
        }
    Find the minimum element in unsorted array //
            ;int min_idx {i}
            for (int j {i+1}; j < n; j++)
        if (arr[j] < arr[min_idx])
                ;min_idx = j

    Swap the found minimum element with the first element //
            ;swap(arr[min_idx], arr[i])
        {
```

وساختار آن نیز مشخص است که در آن تابع swap همانطور که اسم آن گویاست به صورت زیر تعریف میشود.

```
<template <class T
void swap(T& xp, T& yp)
}
    ;T temp = xp
    ;xp = yp
    ;yp = temp
{
```

داده ی زیر به آن داده شده است و خروجی به صورت زیر در آمد.

```
;{۱۱ ,۲۲ ,۱۲ ,۲۵ ,۶۴}[]int arr
```

```
 Q4 git:(master)  ./main
Sorted array:
11 (T arr[], int n)
12
22move boundary of unsorted subarray
25 i < n-1; i++)
64
inimum element in unsorted array
 Q4 git:(master)
```

https://gitlab.com/MohammadRaziei/AP-HW2.git