Панчоян Сергей СКБ223_2. Отчёт по ЛР#6-7

Файлы:

1. main.cpp:
   В нём проверяются введённые ключи и вызывается основная функция.
2. funcs6.cpp (funcs7.cpp для ЛР#7):
   Файл, в котором находятся все необходимые функции, исполняемые программой.
3. funcs.hpp:
   Заголовочный файл. Используется в main.cpp. содержит некоторые функции из funcs6.cpp (funcs7.cpp для ЛР#7).
4. makefile:
   Файл make. Поддерживает команды make, clean, distclean, realclean и uninstall.
5. input.txt:
   Опциональный файл. Можно использовать как файл ввода при использовании соответствующего ключа.

Функционал:

1. При вызове .exe файла можно использовать 2 доступных ключа:
   1.1. “--fromfile”:
   Использует данные из файла. Если после ключа ввести полный путь к нужному файлу, то будет использован он, иначе будет использован файл по умолчанию input.txt.
   Стоит обратить внимание на то, что в пути файла не должно быть пробелов, иначе программа выдаст ошибку.
   Если не указать этот ключ, данные будет необходимо ввести непосредственно в консоль. Также при вводе необходимо только корректные пол, дату и тип, иначе программа выдаст ошибку.
   1.2. “--tofile”:
   Использует файл в качестве точки вывода данных. Если после ключа ввести полный путь к нужному файлу, то будет использован он, иначе будет создан и использован файл по умолчанию output.txt.
   Стоит обратить внимание на то, что если файл output.txt уже существует в папке с файлами, то всё его содержимое будет удалено и заменено на новые данные. Также, если в указанном пути будут присутствовать пробелы программа выдаст ошибку.

2. Вызов makefile допускает 5 основных команд:
   2.1. “make”:
   Эта команда транслирует файлы main.cpp и funcs.cpp в соответствующие объектные файлы и компанует из в main.exe. Если при вызове makefile не указать команду, то по умолчанию будет вызвана команда make.
   2.2. “clean” (будет работать только для ОС Windows):
   Эта команда удаляет созданные командой make объектные файлы.
   2.3. “distclean” (будет работать только для ОС Windows):
   Эта команда удаляет созданный командой make файл main.exe, а также созданный программой файл output.txt. Кроме того вызывает clean.
   2.4. “realclean” (будет работать только для ОС Windows):
   Эта команда удаляет файл makefile. Кроме того вызывает distclean. Будьте осторожны, после вызова этой команды восстановить удалённые файлы не получится!
   2.5. “uninstall” (будет работать только для ОС Windows):
   Эта команда удаляет исходные файлы. Кроме того вызывает realclean. Будьте осторожны, после вызова этой команды восстановить удалённые файлы не получится!

Функции (Для ЛР#6):

1. Файл main.cpp:
   1.1. main( ) В нём используются только функции описанные в файле funcs.cpp. Сама функция проверяет введённые ключи и передаёт их в качестве параметров в основную функцию.

```
1   int main (int argc, char **argv)
2   {
3     bool ff=false, tf=false, dff=false, dtf=false;
4     char *fdir="./input.txt", *tdir="./output.txt";
5     const char *key1 = "--tofile", *key2 = "--fromfile", *key = "--";
6
7     for (int i=1; i<argc; ++i)
8     {
9       if (comp(key,argv[i]))
10      {
11        if (comp(argv[i],key1))
12        {
13          if (tf) ERR
14          tf=true;
15          dtf=true; dff=false;
16        }
17        else if (comp(argv[i],key2))
```

```
18        {
19          if (ff) ERR
20          ff=true;
21          dff=true; dtf=false;
22        }
23        else
24        {
25          std::cerr << "\nWrong keys.\n";
26          return EXIT_FAILURE;
27        }
28      }
29      else if (dtf!=dff)
30      {
31        if (dtf)
32        {
33          tdir = argv[i];
34          dtf = false;
35        }
36        else if (dff)
37        {
38          fdir = argv[i];
39          dff = false;
40        }
41      }
42      else
43      {
44        std::cerr << "\nWrong keys.\n";
45        return EXIT_FAILURE;
46      }
47    }
48    func(tf, tdir, ff, fdir);
49
50    return EXIT_SUCCESS;
51  }
```

2. Файл funcs.cpp:

2.1. comp ( ):

Сравнивает 2 символьных массива на то, является ли первый массив началом второго. По совместительству, этой функцией можно проверить одинаковы ли два символьных массива.

```
1  bool comp (const char *a, const char *b)
2  {
3    for (int i=0; a[i]!='\0'; ++i)
4    {
5      if (a[i]!=b[i])
6      {
7        return false;
8      }
9    }
10   return true;
11 }
```

2.2.`Digit ( ):

Проверяет, является ли символ цифрой.

```
bool Digit (const char& a)
{
  if (a>='0'&&a<='9') return true;
  return false;
}
```

2.3. correct_date ( ):

Проверяет, правильно ли введена дата. Также приводит соответствующие переменные к необходимому виду.

```
void correct_date(string& sy, string& sm, string& sd)
{
  while (sy[0]==' ') sy.std::string::erase(0,1);
  while (sd[sd.length()-1]==' ') sd.std::string::erase(sd.length()-1,1);

  if (sy.length()>4||sm.length()>2||sd.length()>2) DATE_ERR

  for (int i=0;i<2;++i)
  {
    if (!(Digit(sy[2*i])&&Digit(sy[2*i+1])&&Digit(sm[i])&&Digit(sd[i]))) DATE_ERR
  }

  int y = stoi(sy), m = stoi(sm), d = stoi(sd);
  if (y>9999||y<0||m>12||m<1||d>31||d<1) DATE_ERR
  else if ((m==4||m==6||m==9||m==11)&&d>30) DATE_ERR
  else if (m==2&&(d>29||((d>28)&&!(((y%4)==0&&(y%100)!=0)||(y%400)==0))))
DATE_ERR

  while (sm.length()<2) sm = "0"+sm;
  while (sd.length()<2) sd = "0"+sd;
  while (sy.length()<4) sy = "0"+sy;
}
```

2.4. toi ( ):

Конвертирует символ в цифру.

```
int toi(char a) {return (int)(a-'0');}
```

2.5. class Studs:

Виртуальный класс-интерфейс. Содержит чисто виртуальные функции.

```
class Studs
{
  protected:
    string str;
    virtual const string sex(string&) =0;
    virtual const string rands(const string&, const string&, const string&) =0;
    virtual const string last(const string&) =0;
  public:
    Studs() : str(""){};
    ~Studs() {str="";};
    virtual const string& generate(string&, const string&, const string&, const
string&) =0;
    virtual void print(std::ostream&)=0;
};
```

2.6. class MIEM и class MGTUU:

Классы-наследники класса Studs. В них реализуются чисто виртуальные функции класса-родителя. (Дальнейшие функции представлены для класса MIEM)

2.6.1. sex ( ):

Проверяет, правильно ли введён пол, и в зависимости от пола возвращает необходимое значение в формате строки.

```
const string sex(string& a) override
  {
    while (a[0]==' ') a.std::string::erase(0,1);
    if (a=="man") return (string)"8";
    if (a=="woman") return (string)"4";
    cerr << "Wrong sex!\n";
    exit(EXIT_FAILURE);
  }
```

2.6.2. rands ( ):

Возвращает псевдослучайное число, уникальное для даты рождения.

```
const string rands (const string& sy, const string& sm, const string& sd)
override
  {
    int y=stoi(sy), m=stoi(sm), d=stoi(sd);
    string str = to_string(((((((y<<m)^y)&INT32_MAX)<<d)^y)&INT32_MAX)%100000);
    while (str.length()<5) str = "0" + str;
    return str;
  }
```

2.6.3. last ( ):

Возвращает цифру, подобранную так, чтобы сумма всех цифр номера, умноженных на их позицию, делилась на заранее заданное число. Для класса MIEM это число – **8**, а для MGTUU – **9**.

```
const string last (const string& str) override
  {
    int s=0, num;
    for (int i=0; i<str.length(); ++i)
    {
      s+=(toi(str[i])*(i+1));
    }
    s%=divider;
    for (num=0; num<10;++num)
    {
      if ((s+num*15)%divider==0) break;
    }
    return to_string(num);
  }
```

2.6.4. generate ( ):

Генерирует необходимый номер студенческого билета. Использует для этого вышеперечисленные функции.

```
const string& generate (string& s, const string& y, const string& m, const
string& d) override
    {
```

```
3
4         str = sex(s) + y + m + d + rands(y,m,d);
5         str = str + last(str);
6         return str;
7       }
```

2.6.5. print ( ):

Выводит в заданный поток номер билета.

```
1   void print(std::ostream& out) override
2       {
3         out << str << std::endl;
4       }
```

2.7. func ( ):

Основная функция.

На вход принимает строку формата <<sex> <yyyy.mm.dd>>, где <sex> – пол, а <yyyy.mm.dd> – дата, с разделителями в виде точек. Пробелы до, после и между элементами игнорируются, однако любые другие символы, а также пробелы внутри этих элементов вызовут ошибку.

Используя вышеперечисленные функции, создаёт и выводит 2 варианта номеров студенческого билета.

```
1   void func(bool tf, const char* tdir, bool ff, const char* fdir)
2   {
3     string sex, y, m, d;
4     if (ff)
5     {
6       std::ifstream fin (fdir);
7       if (!fin.is_open())
8       {
9         cerr << "Wrong input file directory!\n";
10        exit(EXIT_FAILURE);
11      }
12      getline (fin,sex,' ');
13      getline (fin,y,'.');
14      getline (fin,m,'.');
15      getline (fin,d);
16      fin.close();
17    }
18    else
19    {
20      getline (cin,sex,' ');
21      getline (cin,y,'.');
22      getline (cin,m,'.');
23      getline (cin,d);
24    }
25    correct_date(y,m,d);
26
27    MIEM SB_1;
28    MGTUU SB_2;
29
30    SB_1.generate(sex,y,m,d);
31    SB_2.generate(sex,y,m,d);
```

```
32
33   if (tf)
34   {
35     std::ofstream fout (tdir,std::ofstream::trunc);
36     if (!fout.is_open())
37     {
38       cerr << "Wrong output file directory!\n";
39       exit(EXIT_FAILURE);
40     }
41     SB_1.print(fout);
42     SB_2.print(fout);
43     fout.close();
44   }
45   else
46   {
47     SB_1.print(cout);
48     SB_2.print(cout);
49   }
50 }
```

Функции (Для ЛР#7):

Бóльшая часть кода осталась нетронутой.

Введённые изменения:

1. Добавлено:
   1.1. class template_pattern_generator:
   Шаблонный класс, призванный ускорить вызов необходимых функций.
   1.1.1. generator ( ):
   Возвращает указатель на объект того класса, который указан в его аргументе.

```
1  Studs* generator (const string& type)
2      {
3        if (type=="MIEM")
4        {
5          MIEM *sb;
6          return sb;
7        }
8        else if (type=="MGTUU")
9        {
10         MGTUU *sb;
11         return sb;
12       }
13     }
```

   1.2. correct_type ( ):
   Проверяет, правильно ли введён тип необходимого объекта, также приводит его к необходимому виду.

```
1  void correct_type(string& type)
2  {
```

```
3     while (type[0]==' ') type.std::string::erase(0,1);
4     while (type[type.length()-1]==' ') type.std::string::erase(type.length()-1,1);
5     if (type!="MIEM"&&type!="MGTUU")
6     {
7       cerr << "Wrong type!\n";
8       exit(EXIT_FAILURE);
9     }
10  }
```

2. Удалено:
   2.1. В class Studs, class MIEM и class MGTUU:
      2.1.1. print ( ):
         Удалено за ненадобностью.
3. Изменено:
   3.1. В func ( ):
      Теперь принимает на вход строку формата <<sex> <yyyy.mm.dd> <type>>, где <sex> – пол, <yyyy.mm.dd> – дата, с разделителями в виде точек, а <type> – тип студенческого билета, который должен быть выведен. Пробелы до, после и между элементами игнорируются, однако любые другие символы, а также пробелы внутри этих элементов вызовут ошибку.
      Используя вышеперечисленные функции создаёт и выводит только 1 номер билета, указанного типа.

```
1   void func(bool tf, const char* tdir, bool ff, const char* fdir)
2   {
3     string sex, y, m, d, type;
4     if (ff)
5     {
6       std::ifstream fin (fdir);
7       if (!fin.is_open())
8       {
9         cerr << "Wrong input file directory!\n";
10        exit(EXIT_FAILURE);
11      }
12      getline (fin,sex,' ');
13      getline (fin,y,'.');
14      getline (fin,m,'.');
15      getline (fin,d,' ');
16      getline (fin,type);
17      fin.close();
18    }
19    else
20    {
21      getline (cin,sex,' ');
22      getline (cin,y,'.');
23      getline (cin,m,'.');
24      getline (cin,d);
25      getline (cin,type);
26    }
27    correct_type(type);
28    correct_date(y,m,d);
```

```
29
30   template_pattern_generator templateGenerator;
31   string SB = templateGenerator.generator(type)->generate(sex,y,m,d);
32
33   if (tf)
34   {
35     std::ofstream fout (tdir,std::ofstream::trunc);
36     if (!fout.is_open())
37     {
38       cerr << "Wrong output file directory!\n";
39       exit(EXIT_FAILURE);
40     }
41     fout << SB << std::endl;
42     fout.close();
43   }
44   else
45   {
46     cout << SB << std::endl;
47   }
48 }
```