МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВВГУ»)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

по дисциплине
«Информатика и программирование»

Студент
гр. БИН-25-2 _____ Шумейко И.А

Ассистент
Преподавателя _____ Водяницкий М.В

Владивосток 2025

Задание

Задание 1

Имеется список объектов Фонда с указанием уровня угрозы:

```
objects = [
    ("Containment Cell A", 4),
    ("Archive Vault", 1),
    ("Bio Lab Sector", 3),
    ("Observation Wing", 2)
]
```

Используя `sorted` и лямбда-выражение, отсортируйте объекты по возрастанию уровня угрозы.

Задание 2

Дан список сотрудников Фонда с количеством проведенных смен и стоимостью одной смены:

```
staff_shifts = [
    {"name": "Dr. Shaw", "shift_cost": 120, "shifts": 15},
    {"name": "Agent Torres", "shift_cost": 90, "shifts": 22},
    {"name": "Researcher Hall", "shift_cost": 150, "shifts": 10}
]
```

Используя `map` и лямбда-выражение, создайте список общей стоимости работы каждого сотрудника

Затем найдите максимальную стоимость с помощью «max».

Задание 3

Дан список персонала с уровнем допуска:

```
personnel = [
    {"name": "Dr. Klein", "clearance": 2},
    {"name": "Agent Brooks", "clearance": 4},
    {"name": "Technician Reed", "clearance": 1}
]
```

Используя `map` и лямбда-выражение, создайте новый список, где каждому сотруднику добавляется категория допуска:

1. "Restricted"` - уровень 1

2. "Confidential"` - уровни 2–3
3. "Top Secret"` - уровень 4 и выше

Результат должен быть списком словарей.

Задание 4

Дан список зон Фонда с указанием времени активности (в часах):

```
zones = [
    {"zone": "Sector-12", "active_from": 8, "active_to": 18},
    {"zone": "Deep Storage", "active_from": 0, "active_to": 24},
    {"zone": "Research Wing", "active_from": 9, "active_to": 17}
]
```

Используя `filter` и лямбда-выражение, выберите зоны, которые полностью работают в дневной период (с 8 до 18 включительно)

Задание 5

Фонд анализирует служебные отчеты. Некоторые отчеты содержат внешние ссылки, которые должны быть удалены перед архивированием

```
reports = [
    {"author": "Dr. Moss", "text": "Analysis completed. Reference: http://external-archive.net"},
    {"author": "Agent Lee", "text": "Incident resolved without escalation."},
    {"author": "Dr. Patel", "text": "Supplementary data available at https://secure-research.org"},
    {"author": "Supervisor Kane", "text": "No anomalies detected during inspection."},
    {"author": "Researcher Bloom", "text": "Extended observations uploaded to http://research-notes.lab"},
    {"author": "Agent Novak", "text": "Perimeter secured. No external interference observed."},
    {"author": "Dr. Hargreeve", "text": "Full containment log stored at https://internal-db.scp"},
    {"author": "Technician Moore", "text": "Routine maintenance completed successfully."},
    {"author": "Dr. Alvarez", "text": "Cross-reference materials: http://crosslink.foundation"},
    {"author": "Security Officer Tan", "text": "Shift completed without incidents."},
    {"author": "Analyst Wright", "text": "Statistical model published at https://analysis-hub.org"},
    {"author": "Dr. Kowalski", "text": "Behavioral deviations documented internally."},
    {"author": "Agent Fischer", "text": "Additional footage archived: http://video-storage.sec"},
    {"author": "Senior Researcher Hall", "text": "All test results verified and approved."},
    {"author": "Operations Lead Grant", "text": "Emergency protocol draft shared via https://ops-share.scp"}
]
```

Используя `filter` и лямбда-выражение:

1. Отберите отчеты, содержащие ссылки (`http` или `https`)
2. Преобразуйте их так, чтобы вместо ссылки отображалось `[ДАННЫЕ УДАЛЕНЫ]`

Задание 6

Дан список SCP-объектов с указанием их класса содержания:

```
scp_objects = [
    {"scp": "SCP-096", "class": "Euclid"},
    {"scp": "SCP-173", "class": "Euclid"},
    {"scp": "SCP-055", "class": "Keter"},
    {"scp": "SCP-999", "class": "Safe"},
    {"scp": "SCP-3001", "class": "Keter"}
]
```

Используя `filter` и лямбда-выражение, сформируйте список SCP-объектов, которые требуют усиленных мер содержания

К объектам с усиленными мерами относятся все SCP, класс которых не равен "Safe".

Результат должен быть списком словарей исходного формата

Задание 7

Дан список инцидентов с количеством задействованного персонала:

```
incidents = [
    {"id": 101, "staff": 4},
    {"id": 102, "staff": 12},
    {"id": 103, "staff": 7},
    {"id": 104, "staff": 20}
]
```

Используя `sorted` и лямбда-выражение:

1. Отсортируйте инциденты по количеству персонала
2. Оставьте только три наиболее ресурсоемких инцидента

Задание 8

Дан список протоколов безопасности и их уровней критичности:

```
protocols = [
    ("Lockdown", 5),
    ("Evacuation", 4),
    ("Data Wipe", 3),
    ("Routine Scan", 1)
]
```

Используя `map` и лямбда-выражение, создайте новый список строк вида:

"Protocol Lockdown - Criticality 5"

Задание 9

Имеется список смен охраны с указанием длительности (в часах):

Используя `filter` и лямбда-выражение, выберите только те смены, которые:

1. длятся не менее 8 часов
2. не превышают 12 часов

Задание 10

Дан список сотрудников с результатами психологической оценки (от 0 до 100):

```
evaluations = [
    {"name": "Agent Cole", "score": 78},
    {"name": "Dr. Weiss", "score": 92},
    {"name": "Technician Moore", "score": 61},
    {"name": "Researcher Lin", "score": 88}
]
```

Используя `max` и лямбда-выражение, определите сотрудника с наивысшей оценкой

Результатом должно быть имя сотрудника и его балл.

Содержание

1 Выполнение работы

1.1 Задание 1

Код для данного задания представлен на рисунке 1.

```
struct Object{
  std::string name;
  int danger = 0;
};
std::vector<Object> scp{
  {"Containment Cell A", 4},
  {"Archive Vault", 1},
  {"Bio Lab Sector", 3},
  {"Observation Wing", 2}
};

std::sort(scp.begin(),scp.end(),[](const auto& lhs, const auto& rhs){
  return lhs.danger < rhs.danger;
});
```

Рисунок 1 — код для задания 1

Для выполнения задания создадим класс Object, затем создадим вектор из таких объектов Object и используем алгоритм sort, с определенной лямбда функцией, которая принимает два аргумента и возвращает true, если левый аргумент строго меньше правого, иначе — false.

1.2 Задание 2

Код для данного задания представлен на рисунках 2 и 3.

```
struct Employee{
  std::string name; int shift_cost = 0; int shifts = 0;
  int GetCost()const{
    return shift_cost * shifts;
  }
  int GetOneCost()const{
    return shift_cost;
  }
  bool operator< (const Employee& other){
    return GetCost() < other.GetCost();
  }
};
```

Рисунок 2 — код для задания 2

На рисунке 2 Создаем класс Employee, который содержит стоимость смены и количество смен а так же имя. Далее определим метод GetCost, возвращающий общую стоимость, метод GetOneCost – стоимость одной смены и оператор меньше для сравнения двух структур.

```
  std::vector<Employee> employees{
    {"Dr. Shaw",120,15},
    {"Agent Torres",90,22},
    {"Researcher Hall",150,10}
  };

  std::vector<std::pair<Employee*,int>> shifts_map;

  for(auto& empl: employees){
    shifts_map.emplace_back(&empl,empl.GetCost());
  }

  auto max_shift = std::max_element(shifts_map.begin(),shifts_map.end(),[](const
auto& lhs, const auto& rhs){
    return lhs.second < rhs.second;
  });

  std::cout << max_shift->second;
```

Рисунок 3 - код для задания 2

Затем сохраним работников в векторе employees, затем создадим вектор shifts_map, который хранит указатели на объект Employee и соответственную общую стоимость. Затем, используя std::max_element, находим сотрудника с максимальной общей стоимостью работы.

1.3 Задание 3

Код для выполнения задания представлен на рисунке 4.

```
struct Personal{
  Personal(std::string n,int c):
  name(std::move(n)),clearence(std::move(c)){}

  std::string name;
  int clearence;
};

struct Personal_New:Personal{
  Personal_New(std::string n, int c, std::string cat):
  Personal(n,c),category(std::move(cat)){}

  std::string category;
};

    std::vector<Personal> personality{
     {"Dr. Klein",2},
     {"Agent Brooks",4},
     {"Technician Reed",1}
  };

  std::vector<Personal_New> new_personal_base;
  for(const auto& [name,clearence]: personality){
     std::string category;
     if(clearence == 1){
        category = "Restricted";
     }else if(clearence < 4){
        category = "Confidential";
     }else{
        category = "Top Secret";
     }
     new_personal_base.emplace_back(name,clearence,category);
  }
  for(const auto& personal: new_personal_base){
     std::cout << personal.name << ' ' << personal.clearence << ' ' <<
personal.category << '\n';
  }
```

Рисунок 4 — код для задания 3

Для выполнения задания создадим класс Personal, где хранится имя и уровень доступа, затем создадим класс Personal_New, который наследует данные и

добавляет категорию. Затем создадим вектор объектов Personal personality, а потом новый вектор с объектами Personal_New new_personal_base, затем проходимся циклом по вектору personality и создаем объекты Personal_New для нового вектора, так мы имеем новый вектор с новыми данными.

1.4 Задание 4

Код для выполнения задания представлен на рисунке 5.

```
struct Zone{
  std::string name;
  int active_from = 0;
  int active_to = 0;
};
    std::vector<Zone> zones{
     {"Sector-12",8,18},
     {"Deep Storage",0,24},
     {"Research",9,17}
  };
  auto daily_zones = zones |
       std::views::filter([](const Zone& zone){
         return zone.active_from >= 8 && zone.active_to <= 18;
       });
  for(const auto& zone: daily_zones){
    std::cout << zone.name << ' ';
  }
  std::cout << '\n';
```

Рисунок 5 — часть кода для задания 4

Для выполнения задания создадим структуру Zone, где будем хранить название и время начала работы и время конца работы. Далее создадим вектор с объектами Zone zone и создадим view daily_zones с использованием filter и лямбда функции, которая возвращает true, если время начала работы не меньше 8, а время конца — не меньше 18, которое будет содержать объекты, которые подходят под условие в лямбда функции.

1.5 Задание 5

Код для выполнения задания представлен на рисунке 6.

```
auto IsHttp = [](std::string_view str)->bool{
   return str.find("http") != std::string_view::npos;
};

auto ModificateStr = [](std::string_view str){
   auto pos = str.find("http");
   str.remove_suffix(str.size() - pos);
   return std::string(str) + "[ДАННЫЕ УДАЛЕНЫ]";
};
   std::vector<std::pair<std::string,std::string>> reports{
      {"Dr. Moss","Analysis completed. Reference: http://external-archive.net"},
      {"Agent Lee","Incident resolved without escalation."},
      {"Dr. Patel","Supplementary data available at https://secure-research.org"},
      {"Supervisor Kane","No anomalies detected during inspection."},
      {"Researcher Bloom","Extended observations uploaded to http://research-notes.lab"},
      {"Agent Novak","Perimeter secured. No external interference observed."},
      {"Dr. Hargreeve","Full containment log stored at https://internal-db.scp"},
      {"Technician Moore","Routine maintenance completed successfully."},
      {"Dr. Alvarez","Cross-reference materials: http://crosslink.foundation"},
      {"Security Officer Tan","Shift completed without incidents."},
      {"Analyst Wright","Statistical model published at https://analysis-hub.org"},
      {"Dr. Kowalski","Behavioral deviations documented internally."},
      {"Agent Fischer","Additional footage archived: http://video-storage.sec"},
      {"Senior Researcher Hall","All test results verified and approved."},
      {"Operations Lead Grant","Emergency protocol draft shared via https://ops-share.scp"}
   };

   auto mod_reports = reports | std::views::filter([](const auto& str_p){return IsHttp(str_p.second);})
                              | std::views::transform([](const auto& str_p)->std::pair<std::string_view,std::string_view>
                                                      {
                                                         return {str_p.first,ModificateStr(str_p.second)};
                                                      });

   for(const auto& report: mod_reports){
      std::cout << report.second << '\n';
   }
   std::cout << '\n';
```

Рисунок 6 — часть кода для задания 5

Для выполнения задания создадим две функции, одна проверяет есть ли строчка http или htttps в строке, вторая — находит вхождение строки с адресом и меняет его на строчку «[ДАННЫЕ УДАЛЕНЫ]». Затем создадим вектор reports где хранятся имена агентов и их репорты, содержащие адрес или нет. За-

тем при помощи filter и transform сохраняем в mod_reports репорты, которые содержат адреса с измененной частью с http и https.

1.6 Задание 6

Код для выполнения задания представлен на рисунке 7.

```
std::vector<std::pair<std::string,std::string>> scp_object{
   {"SCP-096","Euclid"},
   {"SCP-173","Euclid"},
   {"SCP-055","Keter"},
   {"SCP-999","Safe"},
   {"SCP-3001","Keter"}
};

auto not_safe_scp = scp_object | std::views::filter([](const auto& p){
                          return p.second != "Safe";
                       });

for(const auto& scp: not_safe_scp){std::cout << scp.first << ' ';}
std::cout << '\n';
```

Рисунок 7 — код для задания 6

Для выполнения задания создадим вектор scp_object с парами строк, где хранится код SCP и уровень их опасности, затем создадим view not_safe_scp, который ссылается на пары, уровень опасности которых не содержит строчку «Safe», используя filter с лямбда функцией. Функция возвращает true, если строчка не равна «Safe».

1.7 Задание 7

Код для выполнения задания представлен на рисунке 8.

```
std::vector<std::pair<int,int>> incidents{
  {101,4},
  {102,12},
  {103,7},
  {104,20}
};

std::ranges::sort(incidents,[](const auto& lhs, const auto& rhs){
  return lhs.second > rhs.second;
});

for(const auto& id: incidents | std::views::take(3)){
  std::cout << id.first << ' ';
}
std::cout << '\n';
```

Рисунок 8 — код для задания 7

Для выполнения задания создадим вектор incidents где содержатся пары id персонала и их количество. Затем используем функцию ranges::sort, которая при помощи лямбда функции сортирует пары. Функция возвращает true, если второй элемент одной пары меньше второго элемента другой. Затем создаем view, который содержит первые 3 элемента и выводим их.

1.8 Задание 8

Код для выполнения задания представлен на рисунке 9.

```
std::vector<std::pair<std::string,int>> protocols{
  {"Lockdown", 5},
  {"Evacuation", 4},
  {"Data Wipe", 3},
  {"Routine Scan", 1}
};

std::vector<std::string> strs;
for(const auto& [name,level] : protocols){
  strs.push_back(name + " - " + std::to_string(level));
}

for(const auto& str: strs){
  std::cout << str << '\n';
}
```

Рисунок 9 - код для задания 8

Для выполнения задания создаем вектор, который protocols содержит пару строчку и число. Строчка обозначает протоколы и уровень из критичности. Затем создаем другой вектор строчек strs, проходим циклом по protocols и добавляем в strs конкатенацию имени протокола и уровня критичности через тире.

1.9 Задание 9

Код для выполнения задания представлен на рисунке 10.

```
std::vector<int> shifts = {6, 12, 8, 24, 10, 4};

auto shifts_8_to_12 = shifts
| std::views::filter([](const auto& shift){
                        return shift >= 8 && shift < 12;
                      });
for(const auto& sh: shifts_8_to_12){
  std::cout << sh << ' ';
}
std::cout << '\n';
```

Рисунок 10 - код для задания 9

Для выполнения задания создаём вектор shifts, где содержатся время смен в часах. Затем создаём view shifts_8_to_12, которая ссылается только на те элементы, которые меньше 12 и не меньше 8. Наконец выводим числа.

1.10 Задание 10

Код для выполнения задания представлен на рисунке 11.

```
 std::vector<std::pair<std::string,int>> evaluations{
    {"Agent Cole",78},
    {"Dr. Weiss",92},
    {"Technician Moore",61},
    {"Researcher Lin",88}
  };

  auto max_evalutation =
std::ranges::max_element(evaluations,[](const auto& lhs,
const auto& rhs){
                     return lhs.second < rhs.second;
                   });
  std::cout << max_evalutation->first << ' ' <<
max_evalutation-> second<< '\n';
```

Рисунок 11 — код для задания 10

Для выполнения задания создаём вектор evaluations, который хранит пары со строчкой и числом. Строчка обозначает имя агента, число — его психологическую оценку. Далее используем max_element, находим элемент вектора evaluations с наибольшей оценкой, используя лямбда функцию, которая указывает, что сравнивать элементы нужно по числу. И выводим имя и оценку.