# Решения на Задачите за Структури, Обединения и Множества

## Задача 1: Дефиниране на Структура за Студент

```cpp  
#include <iostream>  
#include <vector>  
using namespace std;  
  
struct Student {  
 string name;  
 int age;  
 float grade;  
};  
  
void printHighGrades(const vector<Student>& students) {  
 for (const auto& student : students) {  
 if (student.grade > 5.0) {  
 cout << student.name << endl;  
 }  
 }  
}  
  
int main() {  
 vector<Student> students = {  
 {"Ivan", 20, 6.0},  
 {"Maria", 22, 4.5},  
 {"Petar", 21, 5.5}  
 };  
 printHighGrades(students);  
 return 0;  
}  
```

Примерен вход:

Примерен вход:  
- Студенти: Иван (6.0), Мария (4.5), Петър (5.5)

Примерен изход:

Примерен изход:  
Ivan  
Petar

## Задача 2: Използване на Обединение за Разходи

```cpp  
#include <iostream>  
#include <vector>  
using namespace std;  
  
union Expense {  
 float transportCost;  
 float foodCost;  
};  
  
struct ExpenseRecord {  
 Expense expense;  
 bool isTransport;  
};  
  
float calculateTotal(const vector<ExpenseRecord>& records, bool isTransport) {  
 float total = 0.0;  
 for (const auto& record : records) {  
 if (record.isTransport == isTransport) {  
 total += isTransport ? record.expense.transportCost : record.expense.foodCost;  
 }  
 }  
 return total;  
}  
  
int main() {  
 vector<ExpenseRecord> records = {  
 {{10.0}, true},  
 {{5.5}, false},  
 {{20.0}, true},  
 {{15.0}, false}  
 };  
 cout << "Total Transport: " << calculateTotal(records, true) << endl;  
 cout << "Total Food: " << calculateTotal(records, false) << endl;  
 return 0;  
}  
```

Примерен вход:

Примерен вход:  
- Разходи: Транспорт (10.0), Храна (5.5), Транспорт (20.0), Храна (15.0)

Примерен изход:

Примерен изход:  
Total Transport: 30  
Total Food: 20.5

## Задача 3: Операции с Множества

```cpp  
#include <iostream>  
#include <set>  
using namespace std;  
  
void printSet(const set<int>& s) {  
 for (int x : s) {  
 cout << x << " ";  
 }  
 cout << endl;  
}  
  
int main() {  
 set<int> even = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};  
 set<int> odd = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};  
  
 set<int> unionSet = even;  
 unionSet.insert(odd.begin(), odd.end());  
 cout << "Union: ";  
 printSet(unionSet);  
  
 set<int> intersectionSet;  
 for (int x : even) {  
 if (odd.count(x)) {  
 intersectionSet.insert(x);  
 }  
 }  
 cout << "Intersection: ";  
 printSet(intersectionSet);  
  
 set<int> symmetricDifferenceSet = unionSet;  
 for (int x : intersectionSet) {  
 symmetricDifferenceSet.erase(x);  
 }  
 cout << "Symmetric Difference: ";  
 printSet(symmetricDifferenceSet);  
  
 return 0;  
}  
```

Примерен вход:

Примерен вход:  
Четни числа от 2 до 20; нечетни числа от 1 до 19.

Примерен изход:

Примерен изход:  
Union: 1 2 3 ... 20  
Intersection:   
Symmetric Difference: 1 3 5 ... 20

## Задача 4: Комплексни Структури и Сортиране

```cpp  
#include <iostream>  
#include <vector>  
#include <algorithm>  
using namespace std;  
  
struct Employee {  
 string name;  
 string position;  
 float salary;  
};  
  
bool compareSalary(const Employee& a, const Employee& b) {  
 return a.salary < b.salary;  
}  
  
void printManagers(const vector<Employee>& employees) {  
 for (const auto& emp : employees) {  
 if (emp.position == "Manager") {  
 cout << emp.name << ": " << emp.salary << endl;  
 }  
 }  
}  
  
int main() {  
 vector<Employee> employees = {  
 {"Ivan", "Manager", 2500},  
 {"Maria", "Developer", 2000},  
 {"Petar", "Manager", 3000},  
 {"Georgi", "Designer", 1800}  
 };  
  
 sort(employees.begin(), employees.end(), compareSalary);  
 cout << "Managers sorted by salary:" << endl;  
 printManagers(employees);  
 return 0;  
}  
```

Примерен вход:

Примерен вход:  
Служители: Иван (Мениджър, 2500), Мария (Програмист, 2000), Петър (Мениджър, 3000), Георги (Дизайнер, 1800)

Примерен изход:

Примерен изход:  
Managers sorted by salary:  
Ivan: 2500  
Petar: 3000

## Задача 5: Управление на Списък със Студенти и Оценки

```cpp  
#include <iostream>  
#include <vector>  
#include <set>  
using namespace std;  
  
struct Student {  
 string name;  
 vector<float> grades;  
 float averageGrade;  
};  
  
float calculateAverage(const vector<float>& grades) {  
 float sum = 0;  
 for (float grade : grades) {  
 sum += grade;  
 }  
 return grades.empty() ? 0 : sum / grades.size();  
}  
  
void printTopStudents(vector<Student>& students) {  
 set<string> topStudents;  
 for (auto& student : students) {  
 student.averageGrade = calculateAverage(student.grades);  
 if (student.averageGrade > 5.5) {  
 topStudents.insert(student.name);  
 }  
 }  
 for (const auto& name : topStudents) {  
 cout << name << endl;  
 }  
}  
  
int main() {  
 vector<Student> students = {  
 {"Ivan", {6.0, 5.0, 5.5}},  
 {"Maria", {4.0, 5.5, 5.0}},  
 {"Petar", {6.0, 6.0, 5.5}}  
 };  
 printTopStudents(students);  
 return 0;  
}  
```

Примерен вход:

Примерен вход:  
Студенти: Иван (оценки: 6.0, 5.0, 5.5), Мария (4.0, 5.5, 5.0), Петър (6.0, 6.0, 5.5)

Примерен изход:

Примерен изход:  
Ivan  
Petar