Общ коментар

С указателите към функциите си се справил. Като цяло си разбрал основните неща и за шаблоните, научил си и синтаксиса им. Не ги използваш съвсем правилно обаче и въпреки че функциите работят при твоите тестове, някои от тях биха предизвикали грешки в друга ситуация. Може би не си разбрал добре някои от условията. За тестването идеята ти е много добра (въпреки че не е изпипана съвсем), бих казал, че това е начинът. Що се отнася до стила и подреждането на кода, единственото ми предложние е да оставяш малко повече пространство за пригледност.

Тестването

Много е добре, че за всяка задача си направил тестова функция, практично е също това, че избираш коя функция да се тества с конзолен вход и позволяваш това избиране повече от веднъж.

Не е добре обаче, че за тази цел правиш безкраен цикъл. Използването на цикли от типа for(;;) или while(true) е добре да се избягва, когато това е възможно (вярвай ми, по ООП няма да решаваме задачи, в които да ни е наложително). Когато все пак се пише такъв цикъл, то трябва някъде в тялото (извън switch или вложен цикъл) да се напише break, което ти не правиш.

По-добра main функция тук би била:

int main() {

int choice;

do {

cout << "Enter task number (choose 0 to quit the test): ";

cin >> choice;

switch (choice) {

case 1: task1(); break;

case 2: task2(); break;

case 3: task3(); break;

case 4: task4(); break;

case 5: task5(); break;

case 6: task6(); break;

case 7: task7(); break;

case 8: task8(); break;

}

} while (choice != 0);

return 0;

}

Въвеждането на специален избор за край на цикъла ти позволява хем да пускаш функциите неограничен брой пъти, хем да имаш „културно“ затваряща се програма.

Задачите

Грешките ти (по-точно, нещата в кода, които все пак работят, но не са хубави), са свързани с типа на аргументите в шаблонните функции.

1. На първите три задачи типа на arr трябва да е Т\*, а не Т. Така както си го написал, при трите извиквания на функцията input() типа Т ти е съответно int\*, char\* и double\*, т.е. извикването

input(a, n);

е еквивалентно на извикването

input<int\*>(a, n);

Причината тестът ти да работи е, че умният компилатор „разсъждава“ така:

Сашо ми подава функция input (int\* и int). Имам шаблонна функция input (T, int). Значи, замествам Т с int\*.

Това не е съвсем добре, тъй като, от човешка гледна точка, масивът е абстракция за

редица от int-ове и бихме искали да можем да напишем

input<int>(a, n);

което в твоя вариант не се компилира.

В общи линии, ето разликата между двата варианта:

|  |  |
| --- | --- |
| void input(T arr, int n)  Ако подадем указател за Т, или оставим компилатора да се сети какво е Т, всичко ще е наред.  Ако подадем неуказател за Т, компилаторът, ако имаме късмет, ще ни се скара. | void input(T\* arr, int n)  Компилаторът приема всякакъв тип за Т (е, трябва все пак да има cin).  Плюс това, може да искаме да имаме две функции с име input – една за указатели и една за други типове, тогава горната декларация би била полезна. |

Същото е и със задачи 2 и 3 – по-добре е типът да е Т\*. Това би ти спестило още

един проблем:

1. На задача 3 не ти трябва нов тип S в шаблона, тъй като има само един логичен тип за х – това Т, в случай, че масивът ти е T\*. Тук отново, на практика, разчиташ, че функцията няма да бъде викана с експлицитно посочване на типа. Ако се опитам да посоча типа в твоя вариант, ще трябва да посоча два типа, при положение, че ми трябва един. Но ако arr е от тип Т, друг вариант няма, така че все пак хубаво си намерил начин да се измъкнеш. ; )
2. Типът на f в задача 7 ти е казан – функция double –> double, така че не ти трябва шаблон, просто напиши:

void \_map(double arr[], int n, double(\*f) (double))

или, още по-добре:

void \_map(double arr[], int n, func f)

Стил

Когато нямаш строги изисквания за стил, както сега, принципно можеш да напишеш кода както на теб ти харесва (стига да не пишеш отявлени грозотии), но е добре да го направиш подреден и четим. В случая препоръката ми (по-скоро предложението, понеже така бих го написал аз) е да оставяш малко повече празно място, за да се чете по-лесно. Това го казвам за тестващите функции, другите си ги написал добре.

Не е лошо да се оставя празен ред отвреме-навреме, дори да е вътре във функция, особено ако тя може да бъде логически разделена. Ако трябва, остави повече място между нея и другите функции – ще изглежда по-добре, отколкото всичко да е наблъскано.

Бих оставил затварящата къдрава скоба на отделен ред само за нея, започвайки да пиша отдолу. А от дефинирането на всички променливи на един ред няма голяма полза. Спокойно можеш да ги дефинираш точно преди да ти потрябват. Ето как бих променил функциите task1 и task2:

void task1() {

int n;

cin >> n;

int a[10];

input(a, n);

for (int i = 0; i < n; i++) {

cout << a[i] << " ";

}

cout << endl;

char b[10];

input(b, n);

for (int i = 0; i < n; i++) {

cout << b[i] << " ";

}

cout << endl;

double c[10];

input(c, n);

for (int i = 0; i < n; i++) {

cout << c[i] << " ";

}

cout << endl;

}

void task2() {

int n;

cin >> n;

int a[10];

input(a, n);

cout << ordered(a, n);

cout << endl;

char b[10];

input(b, n);

cout << ordered(b, n);

cout << endl;

double c[10];

input(c, n);

cout << ordered(c, n);

cout << endl;

}