Започнат на	сряда, 18 август 2021, 11:32
Състояние	Завършен
Приключен на	сряда, 18 август 2021, 12:17
Изминало време	45 мин.

Въпрос **1**

Отговорен

От максимално 1,00

В C++ всяка променлива има точно определен по време на компилация размер

Изберете едно:

- Истина
- Лъжа

Въпрос **2**

Отговорен

От максимално 1,00

Посочете всички верни твърдения за функциите в C++:

Изберете едно или повече:

- a. дефинираните в тялото им променливи са видими само в тялото на функцията
- b. приемат строго типизирани параметри
- c. трябва да бъдат декларирани с прототип преди извикване
- d. връщат резултат от конкретен тип
- e. препоръчително е да се извикват от main()
- f. определят типа на аргументите си чак след извикване
- g. задължително трябва да върнат стойност

Въпрос **3**

Отговорен

От максимално 5,00

Напишете рекурсивна функция, която намира броя цифри на подадено като аргумент цяло число.

```
int countDigits(int num){
    if(num<0) num = -num;

    if(num<10) return 1;
    return 1+countDigits(num/10);
}
```

Въпрос **4**

Отговорен

От максимално 5,00

Открийте, обяснете и поправете грешките в следния код.
За удобство при посочване на грешките можете да използвате номерата на редовете.
Бъдете максимално конкретни при посочването на грешките, както и при обяснението в какво точно се състои грешката.

```
1:  #include <iostream>
2:  #include <cmath>
3:
4:  int MAX_SIZE = 100;
5:  int * readArray(int size) {
6:     assert(size < MAX_SIZE);
7:     int i;
8:     int arr[MAX_SIZE];
9:     for (i = 0; i < size; ++i) cin >> arr[i];
10:    return arr;
11: }
12:
13: void doubleNegative(int* arr, int* result, int size)
14: {
15:     size_t i = cnt = 0;
16:     while (i < size){
17:        if (arr[i]<0)
18:            cnt++;
19:        ++i;
20:     }
21:     result = new(nothrow) int[2*size];
22:     for (i = 0, cnt = 0; i<size; ++i)
23:         result[cnt]=arr[i];
24:           if (result[cnt] < 0) result[++cnt] = arr[i];
25: }
26:
27: int main()
28: {
29:     int* arr, res;
30:     arr = readArray(10);
31:     size_t size = 10;
32:     filterNegative(arr, res, size);
33:     for (--size; size>=0; --size)
34:         cout << res[size] << endl;
35:     delete arr, res;
36: }
```

4: използва се глобална променлива, което е прието за лоша практика

6: използва се assert без включването на библиотеката <cassert>

8: ще даде грешка понеже деклариране масив с неконстантен размер

не записваме стойност на всяка позиция от декларирания масив, понеже size<MAX_SIZE

функцията връща int* но когато излезем от функцията променливата arr ще се освободи и в крайна сметка ще върне указател към място в паметта, което не е наше, понеже програмата ще е освободила вече паметта на това място

15: трябва да access-нем типа чрез std::size_t

присвояваме на променливата cnt която не съществува стойност 0

21: трябва да бъде std::nothrow

23: въртим цикъла и записваме само в първия елемент

24: i вече е със стойност size и чрез arr[size] бъркаме извън масива

31: std::size_t

32: filterNegative - няма такава функция

35: delete[] а не delete понеже работим с масиви

също arr сочи някъде в паметта, където е освободена паметта за масива -> delete[] arr - undefined behaviour

Въпрос **5**

Отговорен

От максимално 1,00

Какъв тип е рекурсията в следния пример:

```
int fun(int x)
{
   if (x == 0) return 0;
   if (x % 2) x += fun(x-1);
   return fun (x);
}
```

Изберете едно или повече:

- a. Линейна
- b. Косвена
- c. Разклонена
- d. Пряка
- e. Опашкова

Въпрос **6**

Отговорен

От максимално 5,00

Какви са начините за подаване на параметри на функции?
Какви са начините за получаване на резултат от функция? Какви са ограниченията при връщане на резултат?

Начините за подаване на параметри на функции са два: подаване по копие или подаване по референция

Можем да получим резултат от функция, като върнем директно резултата, или като го запишем в променлива, която е подадена по референция в параметрите на функцията.

Ако връщаме резултат с return, то тогава можем да върнем само един елемент. Затова друг вариант да върнем повече стойности е чрез параметри, подадени по референция.

Въпрос **7**

Отговорен

От максимално 1,00

Изберете верните твърдения за двоичното търсене

- [x] a. Изисква данните да са подредени
- [x] b. Не може да намира всички елементи с дадена стойност
- [x] c. Служи за основа на много други алгоритми и структури от данни
- [ ] d. Не се прилага често
- [ ] e. Изисква допълнителна памет за да работи
- [ ] f. Не работи добре, ако елементът, който се търси не е наличен
- [ ] g. Работи само за масиви
- [x] h. Е изключително ефективен

Въпрос **8**

Отговорен

От максимално 1,00

Кое от посочените **не е** вярно:

Изберете едно

- (•) a. Елементите на масивите винаги се инициализират с 0 или еквивалент
- ( ) b. Елементите на масивите са последователни в паметта
- ( ) c. Масивите могат да са с елементи от произволен валиден тип
- ( ) d. Елементите на масивите са от един и същ тип
- ( ) e. След декларация не можем да променяме размера на масива

Въпрос **9**

Отговорен

От максимално 5,00

Опишете какво е преобразуване на типове.
Кога то е безопасно и кога не е - какви проблеми възникват?

Преобразуване на типове наричаме преобразуването на типа на дадена променлива към друг.

Също така има два вида преобразуване на типове: имплицитно(от компилатора) и експлицитно(от програмиста)

Например:

long long var = 999999999;

short var2 = var;

В този случай наблюдаваме имплицитно преобразуване на промелнивата var от long long -> short при което ще се получи загуба на информация, понеже капацитетът на var2 не може да побере стойността на var

Тоест, могат да възникнат проблеми ако преобразуваме от по-големи типове към по-малки. Разбира се, обратната посока е напълно безопасна, понеже така само разширяваме предишния ни тип.

Например:

short var1 = 42;

int var2 = var1;

long long var3 = var1;

Можем и експлицитно да пробразуваме типовете

Например:

int a = 5;

float b = (float)a/10;

Тук преобразуваме a към float и след това делим на 10 което е int. Ако не cast-нем a към float, тогава ще получим 5/10 което е 1/2, но компилатора ще запише като 0 защото работи с цели числа и след това като присвоява стойността към b ще преобразува във float.

Въпрос **10**

Отговорен

От максимално 5,00

Обяснете понятието адресна аритметика.
В кои случаи е приложима тя и кога не е ?

Адресна аритметика използваме основно, когато работим с масиви, понеже те са последователно подредени данни в паметта от един и същ тип. Т.е:

int arr[10]; - заделен е в паметта един блок от 10 цели числа в паметта

int* arrPtr = arr; - името на масива се преобразува към указател към първия елемент на масива;

cout << *arrPtr; - можем да изкараме стойността на мястото, където сочи arrPtr

сега правим

arrPtr = arrPtr + 1;

тоест, +1 означава, че към мястото, към което сочи arrPtr, в паметта добавяме един блок от типа на блока за масива, или с други думи ако примеме, че int е 4 байта, то местим напред в паметта 4 байта и така можем да access-нем следващия елемент в масива.

Можем и да използваме този метод при масиви от символи

```
unsigned len(const char* arr){
    unsigned count = 0;
    while(*(arr++)){
        count++;
    }
    return count;
}

char arr[] = "Hello!";
cout << len(arr);
```

По този начин лесно можем да минем през целия масив от символи, докато указателят не сочи нулевия байт и да изведем размера

За двумерни/многомерни масиви, заделени по този начин:

```
int** matrix = new int*[n];
for(unsigned i = 0; i < n; i++){
    matrix[i] = new int[m];
}
```

Пропускам валидност на кода с nothrow и проверки

Така например не можем да използваме адресната аритметика, понеже всеки елемент в matrix не знаем къде се намира заделената му памет (в общия случай не е последователна) и ако се опитаме да приложим тази техника ще бръкнем някъде в паметта където нямаме право.

◄ Анкета за оценка на курс, зимен семестър, 2020/2021

Отиди на ...