Натпревари по информатика

Чести грешки при испраќање на решенија на системот за натпревари

- Во следната табела се наведени најчестите грешки кои, особено почетниците, ги прават за време на натпревари. Во овој документ се наведени само грешки на погрешно извршување на програмите (а не и печатни, и/или, синтаксички грешки кои се прават за време на пишување на код во некоја развојна околина - таквите грешки е на вас да ги избегнете, а дел се пријавуваат и од страна на компајлерот)

Опис и причина	***Пример код***

При користење на низи, една од најчестите грешки е декларирање на низа со помала големина од потребното. Оние што програмираат во C/C++, треба да запаметат дека "int arr[10]" декларира низа од 10 елементи, a[0] до a[9]. Секогаш, правете низи со големина (малку) повеќе од потребното, за да ги избегнете ваквите грешки (секогаш ќе има доволно меморија за тоа).	`int arr[100], i;`
	`int main () {`
	`  for (i=0; i <= 100; i++)`
	`  if (arr[i]==100) //грешка, нема arr[100]`
	`  arr[i] = -1; //i ќе добие вредност -1`
	`  //бидејќи 'i' е декларирана веднаш по arr`
	`  //нејзината мемориска локација ќе се наоѓа`
	`  //веднаш по онаа на низата arr (gnu gcc)`
	`}`

При користење на низи од знаци (стрингови) во C, една од најчестите грешки е мислењето дека за string чија големина е најмногу X, треба да декларираме низа од знаци со големина X, т.е. char[X]. Тоа е погрешно, бидејќи, во низата се чува уште еден знак кој означува каде е крајот на стрингот, т.е. "null character". Повторно, правете низи со големина (малку) повеќе од потребното, за да ги избегнете ваквите грешки.	`#include <string.h>`
	`#include <stdio.h>`
	`int main()  {`
	`  char s1[6], s2[2];`
	`  strcpy(s2, "0 ");  //s2="0 "`
	`  strcpy(s1, "POENI");  //s1="POENI"`
	`  printf("%s", s2); //печати "0 POENI", не "0 "`
	`  //бидејќи нема место за`
	`  return 0;  //null character во s2`
	`}  //а таков знак има во s1[5]`

Основните податочните типови (integer, longint, float, double, …) немаат доволно голем опсег за да ги претстават сите цели или реални броеви, па можно е да се случи програмата да отпечати погрешен резултат - ако одредена променлива е декларирана со погрешен податочен тип. Повеќе информации можете да најдете на следните две страници (ЗАДОЛЖИТЕЛНО ПРОЧИТАЈТЕ ГИ).	`var suma: integer;`
	`begin`
	`{  погрешен резултат, или Runtime Error  }`
	`{  ако е овозможено Range Checking  }`
	`suma := 500500; { integer има опсег до 32767 }`
	`writeln(suma); { печати -23788 }`
	`end.`

Бесконечна јамка и заборавање да се иницијализираат променливите е најчестата грешка за добивање на "Надминат временски лимит", заедно со испраќање на решение кое има поголема сложеност од потребното.
Условот за завршување на една јамка треба да е добро формиран, а променливите треба да се иницијализираат и да се декларирани со тип кој има доволно голем опсег.
	`var i, j: longint;`
	`  niza: array[1..10] of longint;`
	`begin`
	`  i:=0; //иницијализација - не заборавајте`
	`  for j:=1 to 10 do`
	`  niza[j] := 0; //важи и за низи!!!`
	`  repeat`
	`  i := i + 3;`
	`  until i = 10; //никогаш нема да се исполни`
	`end.`

Погрешен резултат за 1 е една од најчестите грешки за време на натпреварите (и најчест тип на приговори ☺). Внимавајте сите услови и формули да се добро формирани и напишани.	`if (Ocena[i] > 6) then //треба (Ocena[i] >= 6)`
	`  WriteLn('Studentot polozil!');`
	`Masi := Luge DIV 5; //погрешна формула`
	`//за 9 луѓе, ќе резервираме 1 маса (наместо 2)`

Грешките при работа со реални броеви се исто така чести – особено за почетниците во програмирање. Бидејќи постојат повеќе типови на грешки кои можат да се направат, имаме посветено цела една страница на овој дел (ЗАДОЛЖИТЕЛНО ПРОЧИТАЈТЕ).	`double r = 0.0;`
	`for (int i=0; i<10; i++)`
	`  r = r + 0.1;`
	`//r=0.9999999999999989, наместо r=1.0`
	`//бидејќи double има ограничена прецизност`
	`int z = r; //тука z е 0, наместо да биде 1`

Натпревари по информатика

Чести грешки при испраќање на решенија на системот за натпревари

- Мислењето дека со integer може да се престави било кој цел број Е ПОГРЕШНО!

Следнава програма нема да го отпечати точниот резултат, бидејќи сумата на броевите од 1 до 1000 е 500500 - број кој не може да се смести во променлива од тип integer (опсег до 32767).

```
program prva;
var i, suma: integer;

begin
   suma := 0;

  {  Runtime Error    }
  { ако е овозможено }
  {  Range Checking   }
   for i:=1 to 1000 do
       inc(suma, i);

   writeln(suma);
    { -23788 }
end.
```

Паскал – Основни податочни типови (прв дел)	
Тип	*Опсег*
byte	0 до 255
shortint	-128 до 127
smallint	-32768 до 32767
word	0 до 65535
integer	најчесто е smallint
longint	**-2147483648 до 2147483647 (-2^{31} до $2^{31}-1$)**
longword	0 до 4294967295
int64	**-9223372036854775808 до 9223372036854775807 ($2^{63}-1$)**
qword	0 до 18446744073709551615

C/C++ – Основни податочни типови (прв дел)	
Тип	*Опсег*
char	-128 до 127, или 0 до 255 (ако е unsigned char)
short	-32768 до 32767, или 0 до 65535
int	**-2147483648 до 2147483647 ($2^{31}-1$), или 0 до 4294967295**
long	-2147483648 до 2147483647 ($2^{31}-1$), или 0 до 4294967295
long long	**-9223372036854775808 до 9223372036854775807 ($2^{63}-1$)**

Како да ги избегнам овие грешки?

1. познавајте го опсегот (доволно е да се знае бројот на цифри на максималниот број) на позначајните податочните типови (особено longint и int64 во Паскал, и int и long long во C/C++)
2. пред да започнете да пишувате решение на одредена задача, пресметајте (приближно) до каде можат да се движат вредностите кои би ги сместиле во одредена променлива (ова не би требало да ви одземе многу време, а е чекор што го прават сите добри натпреварувачи), и искористете соодветен податочен тип кој има доволно голем опсег за да ги претстави тие вредности

Предлози:

1. избегнувајте да ги користите податочните типови кои имаат мал опсег
2. најчесто, користете ги типовите longint - во Паскал, и int - во C/C++ (ова се 32-битни податочни типови), освен кога има потреба од употреба на 64-битните типови – int64 и long long
3. Единствено во случаи кога ви се потребни големи низи од основните податочни типови, а меморискиот лимит е тесен е дозволено да ги користите останатите (помали) типови - и пред тоа, осигурајте се дека навистина може да се сместат потребните вредности во нив. Бидејќи задачите на натпреварите се, претежно, алгоритамски - ова не би требало да се случи - барем ние не знаеме за случај кога било потребно да се користи short, shortint или integer во Паскал. Исто така, бидејќи денешните архитектури на компјутерите се 32 и 64-битни, веројатно нема ниту да има никаков позитивен ефект (гледано временски и мемориски) од користењето на овие податочни типови

Натпревари по информатика

Чести грешки при испраќање на решенија на системот за натпревари

- Мислењето дека со float може да се престави било кој реален (децимален) број Е ПОГРЕШНО!

Следнава програма нема никогаш да заврши (ќе продолжи да печати 'x' до бесконечност) бидејќи ***'a'*** нема никогаш да биде (точно) еднакво на 1.0. Тоа е така бидејќи double има ограничена прецизност.

```
program vtora;
var a: double;

begin
   a := 0.0;

   while (a <> 1.0) do
   begin
      Write('x');
      a := a+0.1;
   end;

end.
```

Паскал – Основни податочни типови (втор дел)	
Тип	*Значајни цифри (цифри кои ги сметаме за точни)*
single	7-8 (првите 7 цифри од резултатот се значајни)
double	**15 до 16 значајни цифри**
real	најчесто е single
extended	19 до 20 значајни цифри
comp	крајно нестандарден податочен тип

C/C++ – Основни податочни типови (втор дел)	
Тип	*Значајни цифри (цифри кои ги сметаме за точни)*
float	7-8 (првите 7 цифри од резултатот се значајни)
double	**15 до 16 значајни цифри**
long double	зависно од архитектурата, најмалку 15-16

"Зошто кога внесувам X, програмата чита (X+0.0000001)" и "Зошто кога споредувам два броја кои се еднакви, програмата вели дека не се" се најчестите прашања од страна на почетниците поврзани со реалните броеви.

Како да ги избегнам овие грешки?

1. Кога сте во можност, избегнувајте да користите операции со реални броеви (користете ги целобројните податочни типови, бидејќи тие се егзактни – прецизни)
2. НЕ КОРИСТЕТЕ single (Паскал) и float (C/C++) - секогаш користете double, или евентуално, некој податочен тип со уште поголема прецизност, бидејќи прецизноста што ја нуди податочниот тип single/float (7 до 8 точни цифри) е премала за било каква практична примена

Предлози:

1. Користете ги функциите round(num) и floor(num+EPS), или (int)(num+EPS), кога сакате да претворите одреден реален број во цел (слични функции има и во Паскал). EPS е некоја мала вредност (на пример, 0.000000001) која ќе ни помогне да ги избегнеме овие, т.н. грешки при заокружување
2. Бидејќи броевите во променливите се чуваат заокружени до одредена децимала, не можеме да провериме дали два реални броеви се еднакви на истиот начин како кај целите броеви
   - можно е да постојат две променливи за кои мислиме дека ја чуваат истата вредност, а програмата да врати дека тие не се еднакви (поради ограничената прецизност) – види ја пример програмата дадена погоре (иако 'a' ќе добие вредност 1.0, операторот за споредба вели дека броевите се различни)
   - поради тоа, дали два реални броеви се еднакви или не, проверуваме на следниот начин

```
if (abs(A-B)<EPS) then Writeln('Ednakvi');  //fabs(A-B) во C/C++
```

каде EPS е некоја мала вредност (на пример, 0.000000001), која ни ја дава максималната можна разлика за да два реални броеви A и B ги сметаме за еднакви